Tietovaraston tietomallin suunnittelu

Sami Sorjonen

Helsinki 19.04.2013

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta – Fakultet – Faculty	Laitos – Institution – Department
Matemaattis-luonnontieteellinen tiedekunta	Tietojenkäsittelytieteen laitos

Tekijä – Författare – Author

Sami Sorjonen

Työn nimi – Arbetets titel – Title

Tietovaraston tietomallin suunnittelu

Oppiaine – Läroämne – Subject

Tietojenkäsittelytiede

Työn laji – Arbetets art – Level	Aika – Datum – Month and year	Sivumäärä – Sidoantal – Number of pages
Pro gradu -tutkielma	19.04.2013	94 sivua

Tiivistelmä – Referat – Abstract

Tietovarasto on järjestelmä, johon kerätään yrityksen liiketoimintatietoa eri lähteistä yhdenmukaisessa muodossa. Varastoon tallennettua tietoa käytetään päätöksenteon tukena yrityksen toiminnan kehittämisessä. Tietovarastossa säilytettävien tietojen tallentamiseen on olemassa erilaisia vaihtoehtoehtoisia tietomalleja. Mallit ja niihin liittyvät suunnittelumenetelmät ovat myös osittain sidoksissa erilaisiin tietovarastoinnissa käytettyihin arkkitehtuurivaihtoehtoihin ja tietovaraston toteuttamisessa käytettyihin menetelmiin. Käytetyt tietomallit, menetelmät ja arkkitehtuurit täydentävät osittain toisiaan, mutta ovat myös osittain toisensa poissulkevia vaihtoehtoja tai ristiriidassa toistensa kanssa. Osittain konfliktit johtuvat myös termistö- ja nimeämisyhteentörmäyksistä.

Tässä tutkielmassa vertaillaan tietovaraston vaihtoehtoisia tietomalleja ja suunnittelutapoja. Erityisesti keskitytään normalisoituun tietomalliin, moniulotteisiin tietomalleihin ja tietoholvimalliin (data vault). Tarkastelutasoksi otetaan pääasiassa tietojen käsitteellinen ja looginen mallinnus ja mallinnuksessa käytetyt tekniikat. Lisäksi tarkastellaan eri tietomallien yhteydessä käytettyjä arkkitehtuurivaihtoehtoja ja menetelmiä.

Tutkielmassa pyritään myös yhtenäistämään eri malleihin, arkkitehtuureihin ja suunnittelutapoihin liittyviä termejä ja nimeämiskäytäntöjä.

ACM Computing Classification System (CCS):
H.2.1 [Logical Design]
H.2.7 [Database Administration]

Avainsanat – Nyckelord – Keywords

tietovarasto, tietomallit, käsitesuunnittelu, looginen suunnittelu, tietoholvi, arkkitehtuuri

Säilytyspaikka – Förvaringställe – Where deposited

Kumpulan tiedekirjasto, sarjanumero C-

Muita tietoja – Övriga uppgifter – Additional information

Sisältö

1 Johdanto

Tietovarasto (data warehouse) on yrityksen toiminnan kehittämisessä ja liiketoiminnan johtamisessa käytettävä päätöksenteon tukijärjestelmä (decision support system, DSS). Se on aihekeskeinen, yhtenäistetty, pysyvä ja aikasidonnainen kokoelma tietoa [ISN08 s. 7][Inm02 s. 31-35]. *Aihekeskeinen* (subject oriented) tarkoittaa, että tietovarasto ei rajoitu yhden sovelluksen tai osaston näkökulmaan. Tietovarastossa voidaan esimerkiksi käsitellä usean järjestelmän asiakastietoja. *Yhtenäistetty* (integrated) tarkoittaa, että tietovarastoon kerätään tietoja useasta eri lähteestä, jotka on yhdenmukaistettu ja joita voidaan käsitellä ikään kuin ne tulisivat samasta lähteestä. *Pysyvä* (nonvolatile) tarkoittaa, että tietovarastoon kerran tuotua tietoa ei yleensä enää jälkeenpäin muuteta. Esimerkiksi asiakastiedoissa olevan asiakkaan osoitteen muuttuessa vanhaa osoitetta ei korvata uudella, vaan asiakkaalle tehdään uusi tietue, jossa näkyy uusi osoite, mutta myös vanha osoitetieto jää tietovarastoon. *Aikasidonnainen* (time variant) tarkoittaa, että tietovaraston tiedot ovat voimassa tietyllä aikavälillä. Esimerkiksi asiakkaan eri osoitetietoihin saattaa liittyä aikaleima, joka kertoo milloin osoite on ollut voimassa.

Tieto kerätään tietovarastoon erilaisista lähdejärjestelmistä, jotka ovat yleensä organisaation *operatiivisia järjestelmiä*, kuten tilaustenhallintajärjestelmä tai laskutusjärjestelmä, mutta kyseeseen voivat tulla myös yrityksen ulkopuoliset tiedot tai jäsentelemätön tieto, kuten taulukkolaskentaohjelmien tai tekstinkäsittelyohjelmien tiedostot. Tietovarastoinnissa tiedon käyttötapa poikkeaa operatiivisista järjestelmistä, mikä aiheutti varsinkin tietovarastoinnin alkuvaiheissa hämmennystä ja epätarkkuutta tiedon tallennukselle asetetuissa vaatimuksissa [CCS93]. Käsitteiden selventämiseksi otettiin käyttöön uusi termi *tosiaikainen tiedonjalostus* (online analytical processing, OLAP) kuvaamaan tietovaraston tiedon käyttötapaa erotukseksi perinteisten operatiivisten järjestelmien käyttötavasta, jota kutsutaan nimellä *tosiaikainen transaktioiden käsittely* (online transaction processing, OLTP) [CCS93].

Tietovarastojen käyttötavalla (OLAP) ja operatiivisten järjestelmien käyttötavalla (OLTP) on useita merkittäviä eroja [ChD97]. Operatiivisia järjestelmiä käytetään yrityksen päivittäisen operatiivisen toiminnan, kuten laskutuksen tai asiakaspalvelun, suorittamiseen. Tietovarastoa käyttää yleensä yrityksen johto yrityksen toiminnan kehittämisessä. Operatiivisen järjestelmän tietokanta on yleensä selvästi pienempi kuin tietovaraston tietokanta. Tietovaraston tietokanta voi olla kooltaan jopa useita teratavuja, sillä siihen kerätään ja arkistoidaan tietoa useista eri lähteistä pitkältä aikaväliltä. Operatiiviset järjestelmät suorittavat yleensä paljon lyhyitä, atomisia transaktioita ajankohtaiselle tiedolle ja tekevät ennalta määriteltyjä luku-, kirjoitus- ja poistooperaatioita muutamalle tietueelle kerrallaan. Tietovaraston tietoja taas käsitellään suurilla, monimutkaisilla, satunnaisilla lukuoperaatioilla, jotka kohdistuvat useisiin tauluihin ja jopa mil-

jooniin tietueisiin kerralla. Monimutkaisten lukuoperaatioiden yleisyyden vuoksi kyselyjen vasteaika on tietovarastoille erityisen tärkeää, operatiivisilla järjestelmillä taas on tärkeää suoritettujen transaktioiden määrä tietyllä aikavälillä (transaction throughput). Koska operatiivisia järjestelmiä käytetään päivittäisen liiketoiminnan suorittamiseen, ovat myös käytettävyys ja nopea virheistä toipuminen operatiivisissa järjestelmissä paljon tärkeämpiä kuin tietovarastoissa. Tosiaikaisen tiedonjalostuksen ja tosiaikaisen transaktioiden käsittelyn tyypillisiä ominaisuuksia ja niiden eroja on listattu tämän luvun lopussa taulukossa 1.

Tietovarastointiin läheisesti liittyvä käsite on *liiketoimintatiedon hallinta* (business intelligence, BI). Tietovarastossa oleva tieto on yrityksen liiketoimintatietoa, jota voidaan analysoida ja johon voidaan kohdistaa erilaisia näkökulmaa rajaavia kyselyitä. Tätä kutsutaan liiketoimintatiedon hallinnaksi. Liiketoimintatiedon hallinta on siis tietovarastoinnin osa-alue, vaikka jossain lähteissä termejä saatetaan käyttää toisin [KRT08 s. 10]. Liiketoimintatiedon hallintaan käytetään erilaisia sovelluksia ja työkaluja, joilla tietovarastossa olevaa tietoa voidaan esittää havainnollisessa muodossa, esimerkiksi raportteina ja kaavioina.

Useasta eri lähteestä kerättyjen tietojen yhdistäminen ja esittäminen yhtenäisesti tietovarastossa on osoittautunut monimutkaiseksi ongelmaksi. Erilaisia näkemyksiä on esitetty siitä, kuinka yhtenäistäminen tulisi toteuttaa, minne ja missä muodossa tiedot pitäisi tallentaa, miten ne tulisi mallintaa ja missä muodossa tiedot pitäisi tarjota liiketoimintatiedon hallintasovelluksille. Tietomallin valinnalla on erityisen tärkeä merkitys, sillä se vaikuttaa myös muihin ratkaisuihin, kuten arkkitehtuurin, liiketoimintatiedon hallintasovellusten ja kehittämismenetelmän valintaan.

Tietovaraston tietojen mallintamiseen on esitetty *normalisoitua tietomallia*, *moniulotteista tietomallia* ja *tietoholvimallia*. Tutkielmassa vertaillaan näitä kolmea tietomallia ja selvitetään niiden käyttötapoja. Tarkastelemme myös sulkevatko tietomallit toisensa pois vai voidaanko joitakin niistä käyttää yhdessä. Tietoholvimalli on mielenkiintoinen, mutta myös hieman monimutkainen vaihtoehto keskitetyn tietovaraston tietojen tallentamiseen. Tietoholvimallin sijasta yritystasoisen keskitetyn tietovaraston toteuttamiseen voidaan käyttää myös normalisoitua tietomallia. Moniulotteinen tietomalli taas soveltuu hyvin pienikokoisempien paikallisvarastojen toteuttamiseen. Kaikille tietomalleille on esitetty erilaisia käsitetason suunnittelutapoja, mikä osoittaa että käsitetason tietomalli ei ole näkökulmariippumaton. Käytännössä käsitemalli ei usein ole myöskään täysin toteutuksesta riippumaton, esimerkiksi sen vuoksi, että fyysisestä ja loogisesta suunnittelusta voi syntyä vaatimuksia myös käsitemallille.

Tietomallien vertailun lisäksi tutkielmassa selvitetään miten tietomallit liittyvät tietovaraston arkkitehtuuriratkaisun ja toteuttamisessa käytetyn kehittämismenetelmän valintaan. Tietovarastoissa käytetään viittä erilaista arkkitehtuuriratkaisua, joita ovat *itsenäiset paikallisvarastot*, *yhtenäistetyt paikallisvarastot*, *yritystason keskitetty tietovarasto*, *keskiö ja puolat* -*arkkitehtuu-*

ri sekä *liitosarkkitehtuuri*. Yhtenäistetyt paikallisvarastot, yritystason keskitetty tietovarasto sekä keskiö ja puolat -arkkitehtuuri on käytännössä parhaiten toimiviksi todettuja.

Tutkielmassa tarkastellaan myös kolmea tietovarastojen toteuttamiseen tarkoitettua kehittämismenetelmää, joita ovat *tietolähtöinen menetelmä*, *prosessikeskeinen menetelmä* ja *tietoholvimenetelmä*. Menetelmät koostuvat joukosta ohjeita ja toimintatapoja, joita voidaan käyttää tietovaraston toteuttamisessa. Tietolähtöinen menetelmä ja prosessikeskeinen menetelmä neuvovat yksityiskohtaisesti kuinka tietovarasto toteutetaan, alkaen suunnittelusta ja päättyen ylläpitoon. Sen sijaan tietoholvimenetelmä on hajanaisempi ja suppeampi joukko neuvoja, joilla pyritään ratkaisemaan tietovarastoinnissa esiintyviä yksittäisten osa-alueiden ongelmia, eikä sitä saatavilla olleiden lähteiden perusteella ainakaan vielä voi pitää hyvin määriteltynä kehittämismenetelmänä.

Tutkielman luvuissa 2-4 käsitellään yksityiskohtaisesti avulla kuinka normalisoitua tietomallia, moniulotteista tietomallia ja tietoholvimallia käytetään tietovaraston tietojen mallintamiseen. Yhtenäisten esimerkkien avulla kuvataan, kuinka erilaiset tietomallit suunnitellaan käsitetasolla, loogisella tasolla ja fyysisellä tasolla. Luvussa 4 johdetaan myös oma ehdotus tietoholvimallin kanssa käytettäväksi käsitemalliksi. Arkkitehtuureja ja niiden kanssa käytettäviä tietomalleja tutkitaan luvussa 5. Luvussa 6 esitellään kolme eri tietovarastojen toteuttamiseen kehitettyä menetelmää ja selvitetään mitä tietomalleja ja arkkitehtuureja niiden kanssa käytetään. Luvussa 7 vertaillaan eri tietomallien, arkkitehtuurien ja kehittämismenetelmien ominaisuuksia. Lisäksi johdetaan prosessi, jonka avulla voidaan valita tietovarastoinnin toteutuksessa käytettävä arkkitehtuuri, kehittämismenetelmä ja tietomalli. Luvussa 8 esitetään yhteenveto ja mahdolliset jatkotutkimuskohteet.

	OLTP	**OLAP**
Tarkoitus	Operatiivinen toiminta	Päätöksenteon tuki
Käyttäjät	Työntekijät	Johto
Tiedon päivitys	Välitön	Tietyin väliajoin
Tiedon ikä	Sekunneista kuukausiin	Päivistä kymmeniin vuosiin
Tietokannan koko	Sadoista megatavuista kymmeniin gigatavuihin	Sadoista gigatavuista useisiin teratavuihin
Operaatiot	Luku, kirjoitus ja poisto	Luku ja kirjoitus
Transaktiot	Pieniä muutamiin tietueisiin kohdistuvia toimenpiteitä	Suuria, erittäin suureen määrään tietueita kohdistuvia kyselyitä
Käsitetason kuvaustapa	ER-kaavio	ER-kaavio ja moniulotteinen malli
Loogisen tason tietomalli	Normalisoitu relaatiomalli	Normalisoitu relaatiomalli ja tähtimalli
Käytettävyyden mittarit	Transaktioiden läpimenoaika ja virheistä toipuminen	Kyselyjen vasteaika ja tiedon luotettavuus

Taulukko 1: Tosiaikaisen tiedonjalostuksen ja tosiaikaisen transaktioiden käsittelyn tyypillisiä piirteitä.

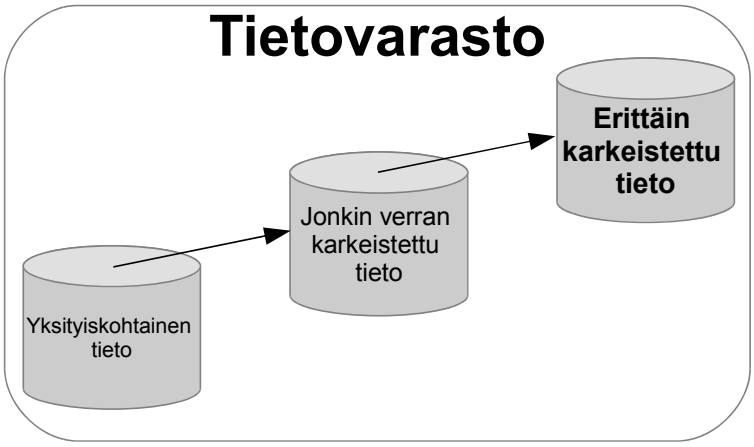
2 Normalisoitu tietomalli

Normalisoitua tietomallia käytetään yleensä operatiivisissa tietojärjestelmissä mutta sitä voidaan käyttää myös tietovarastoissa. *Normalisointi* on tekniikka, jonka avulla relaatiotietokannalle pyritään toteuttamaan eheä ja tehokas rakenne. Tämä on perinteinen tapa toteuttaa operatiivinen tietokanta, joka suorittaa ennalta määriteltyjä luku- ja kirjoitusoperaatioita yleensä pienelle joukolle tietueita. Tiedon eheys ja nopea virheistä toipuminen ovat tärkeitä operatiivisen järjestelmän ominaisuuksia [ChD97]. Normalisoinnin tarkoituksena on erityisesti pyrkiä välttämään tietojen toisteisuutta, koska se aiheuttaa ylimääräistä tallennustilan tarvetta, voi heikentää suorituskykyä ja saattaa johtaa *päivitysanomalioihin*. Päivitysanomaliaksi kutsutaan tilannetta, jossa tietoalkioon kohdistuva päivitysoperaatio aiheuttaa yllättäviä sivuvaikutuksia ja päivitystarpeita muihin tietoalkioihin. Päivitysanomaliat voidaan luokitella lisäysanomalioihin, poistoanomalioihin ja muutosanomalioihin. Normalisoinnilla voidaan karsia tietojen päällekkäisyyttä ja ehkäistä kaikkia päivitysanomalioita.

Normalisointi voidaan toteuttaa eri asteisesti. Korkeampiasteinen normaalimuoto toteuttaa kaikki alemman normaalimuodon vaatimukset ja määrittää niiden lisäksi tietokannalle lisärajoituksia. Ensimmäinen normaalimuoto (1NF) on vaatimuksiltaan heikoin ja kuudes normaalimuoto (6NF) tiukin. Korkeampiasteinen normaalimuoto ehkäisee päivitysanomalioita alempiasteisia tehokkaammin. Yleensä suositellaan käyttämään vähintään *kolmatta normaalimuotoa* (3NF) [CoB02 s. 386-387]. Kuuden normaalimuodon lisäksi käytössä on myös muita eri tavoin määriteltyjä normaalimuotoja, kuten esimerkiksi Boyce-Codd normaalimuoto (BCNF), joka edustaa lievää tiukennusta kolmanteen normaalimuotoon. Muita normaalimuotoja ovat esimerkiksi EKNF (Elementary Key Normal Form) ja DKNF (Domain/Key Normal Form). Tässä tutkielmassa normalisoinnilla tarkoitetaan vähintään kolmanteen normaalimuotoon normalisoitua tietokantaa.

Tietovarastoinnissa normaalimuodossa olevaa tietoa voidaan käyttää erityisesti yksityiskohtaista tietoa (atomic data, detailed data, raw data) sisältävässä tietokannassa, johon lähdejärjestelmistä saatu tieto on tallennettu niin tarkalla tasolla kuin se operatiivisista järjestelmistä saadaan tai katsotaan tarpeelliseksi tallentaa. Operatiivisista järjestelmistä tieto saadaan usein normaalimuodossa ja normaalimuoto voidaan ottaa lähtökohdaksi myös tietovaraston tietojen tallennuksessa [Inm02 s. 137]. Täydellinen normaalimuodon noudattaminen ei kuitenkaan tietovarastoissa ole yhtä tärkeää kuin operatiivisissa järjestelmissä, sillä tietovaraston tietoja päivitetään vain harvoin eikä päivitysanomalioista ole suurta vaaraa [Inm02 s. 106, 137].

Normalisoitu, yksityiskohtaista tietoa sisältävä tietokanta ei yleensä sellaisenaan sovellu palvelemaan tehokkaasti tietovaraston tosiaikaisessa tiedonjalostuksessa käytettyjä kyselyitä. Tosiaikaisessa tiedonjalostuksessa käytetyt operaatiot ovat yleensä luonteeltaan ennalta määrittelemättömiä ja monimutkaisia kyselyitä, jotka kohdistuvat suureen tietomassaan, useisiin tauluihin ja mahdollisesti miljooniin tietueisiin. Lisäksi ne suorittavat tyypillisesti paljon taulujen välisiä liitoksia ja raskaita tiedonjalostusoperaatioita kuten koostamista [ChD97]. Erityisesti isojen taulujen väliset liitokset voivat olla niin raskaita operaatioita, että ne voivat heikentää kohtuuttomasti kyselyjen vasteaikaa, joka on tosiaikaisessa tiedonjalostuksessa tärkeä suorituskyvyn mittari. Tämän vuoksi suuremmissa tietovarastoissa tarvitaan yksityiskohtaisen tiedon lisäksi eritasoisesti *karkeistettua tietoa* (kuva 1) [Inm02 s. 49-53]. Karkeisuustasot voidaan toteuttaa materialisoituina näkyminä, erillisinä tauluina, uusina tietokantainstansseina tai jopa omana tietokannan hallintajärjestelmänä. Karkeistamista voidaan käyttää esimerkiksi osastokohtaisen, rajatun aihealueen näkymän luomiseen [Inm02 s. 17]. Osastokohtainen tieto voi olla *epänormalisoitua* (denormalized) ja mallinnettu eri tavoin kuin tietovaraston yksityiskohtainen tieto. Karkeistettu ja rajattu näkymä sisältää vähemmän tauluja, jolloin kyselyissä ei tarvitse tehdä niin suurta määrää raskaita liitoksia. Suorituksen aikana tehtävien liitosten vähäisemmän määrän ansiosta tosiaikaisessa tiedonjalostuksessa käytettävät kyselyt ovat tehokkaampia.



Kuva 1: Tietovaraston eri karkeisuustasot.

2.1 Normalisoidun tietokannan suunnittelu

Tietovaraston tietokannan suunnittelussa voidaan käyttää suunnittelumenetelmää, joka jakaantuu kolmeen vaiheeseen: käsitesuunnitteluun, loogiseen suunnitteluun ja fyysiseen suunnitteluun. *Käsitesuunnittelussa* tietokannan tiedot mallinnetaan korkean tason näkökulmasta kaavioksi, joka on riippumaton toteutuksen yksityiskohdista, kuten laitteistosta, ohjelmistoista, sovellusohjelmista ja ohjelmointikielistä [CoB02 s. 281]. Käsitesuunnittelun jälkeen tapahtuvassa *loogisessa suunnittelussa* käsitesuunnittelun aikana syntynyttä tietomallia tarkennetaan ja siitä tuotetaan uusi tietomalli, jollaista tietokannan hallintajärjestelmä käyttää,

esimerkiksi relaatiotietokanta. Loogisen tason tietomalli on vielä riippumaton mistään nimetystä tietokannan hallintajärjestelmästä tai muista fyysisistä yksityiskohdista kuten tiedostorakenteista tai hakemistoista. *Fyysinen suunnittelu* on tietokannan suunnittelun viimeinen vaihe. Tässä vaiheessa päätetään toteutukseen käytettävä tietokannan hallintajärjestelmä ja suunnitellaan, kuinka looginen tietomalli toteutetaan valitulla järjestelmällä. Relaatiotietokantaa käytettäessä fyysinen toteutus sisältää esimerkiksi tietokantataulujen ja niihin liittyvien rajoitusten luonnin, hakemistojen luonnin, tallennusrakenteiden valitsemisen ja tietoturvasta huolehtimisen [CoB02 s. 281] [SKS11 s. 429-536].

Fyysisessä suunnittelussa vaaditaan toisenlaista osaamista kuin käsitesuunnittelussa ja loogisessa suunnittelussa, sillä siinä pyritään vastaamaan kysymykseen *kuinka tietokanta toteutetaan*, kun taas käsitesuunnittelussa ja loogisessa suunnittelussa pyritään vastaamaan kysymykseen *mitä tietokanta sisältää* [CoB02 s. 281-283]. Fyysistä suunnittelua edeltävää käsitesuunnittelua ja loogista suunnittelua tarvitaan siis sen vuoksi, että ensin on selvitettävä mitä tietoa tietokanta sisältää ennen kuin voidaan selvittää kuinka se toteutetaan. Tietomallin suunnittelu on luonteeltaan iteratiivista, sillä edellisten vaiheiden tietomallia joudutaan usein tarkentamaan tai korjaamaan seuraavassa vaiheessa tehtyjen havaintojen ja vaatimusten seurauksena [CoB02 s. 283]. Havainnot ja vaatimukset voivat liittyä esimerkiksi suorituskykyyn tai tiedon eheyteen.

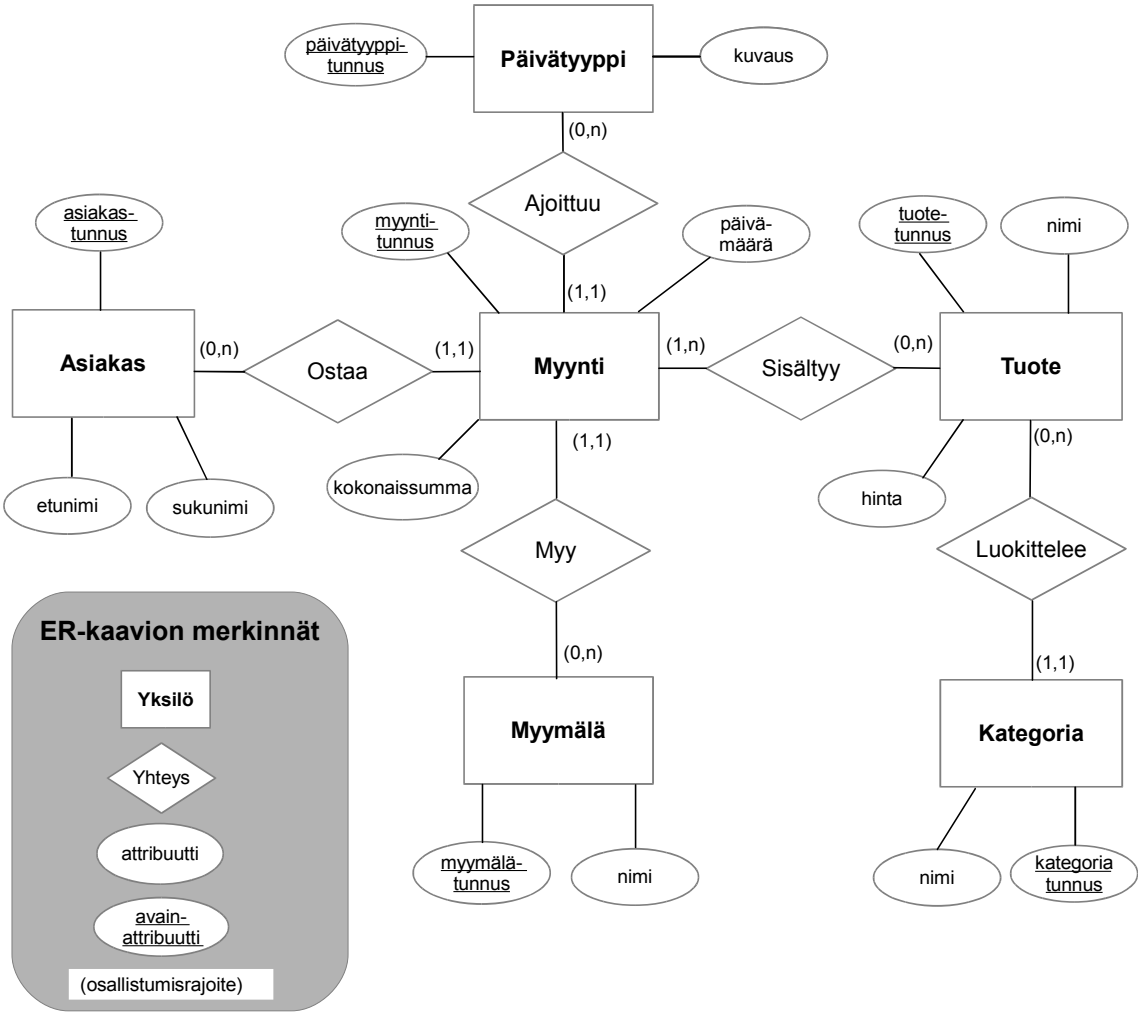
2.2 Käsitesuunnittelu

Tietokannan suunnittelu aloitetaan yleensä käsitesuunnittelulla, jossa mallinnetaan reaalimaailman kohteita ja niiden välisiä suhteita. Suunnittelun lähtökohtana voidaan käyttää vaatimusmäärittelyä. Jotta reaalimaailman kohteita voitaisiin käsitellä tehokkaasti ja tarkoituksenmukaisesti, tulee käsitetason tietomallin olla riittävän ilmaisuvoimainen, ymmärrettävä, mahdollisimman pienikokoinen ja yksikäsitteinen [BCN92 s. 29-30]. Yleensä tietomalli kuvataan kaaviona, joka sisältää sekä graafisia että sanallisia kuvauksia. Onnistumisen kannalta on tärkeää, että graafinen esitys on helposti luettava ja kuvaa kaikki tietomallissa olevat käsitteet.

Yleinen tietokannan käsitesuunnittelussa käytetty tietomalli on Chenin [Che76] kehittämä ER (Entity-Relationship) -malli. Alkuperäinen malli sisältää ainoastaan käsitteet *yksilötyyppi* (Entity), *yhteystyyppi* (relationship) ja *attribuutti* (attribute). Lisäksi yksilötyyppien ilmentymien määrää eri yhteyksissä voidaan kuvata yhteyteen liitetyillä *osallistumisrajoitteilla*. Jälkeenpäin mallia on laajennettu esimerkiksi yhdistelmäattribuutin ja yliluokan käsitteillä.

ER-mallin käsitteistä ja niiden merkityksestä vallitsee varsin selkeä yksimielisyys [BCN92 s. 330], mutta graafisen kaavion toteutukseen voidaan käyttää erilaisia merkintätapoja, joista ei ole olemassa yleistä standardia [SKS11 s. 305]. Chenin alkuperäisen merkintätavan ja siihen tehty-

jen laajennuksien lisäksi ER-kaavion tekemiseen voidaan käyttää esimerkiksi UML (Unified Modeling Language) -notaatiota tai Barkerin notaatiota [Bar90]. Kuvassa 2 on esitetty yhdellä tavalla kuvattu ER-malli tietovarastoon tallennetuista, kuvitteellisen *Market*-myyntijärjestelmän tiedoista. Samaa *Market*-myyntijärjestelmää käytetään jatkossa esimerkkinä myös muiden mallinnustapojen esittelyssä. Esimerkissä kannattaa huomata erityisesti monesta moneen yhteys *Myynti*- ja *Tuote*-yksilötyyppien välillä sekä *Myynti*-yksilötyypin attribuutti *kokonaissumma*, joka on valmiiksi laskettu kooste yhden myyntitapahtuman tuotteiden hinnoista. Kokonaissumma lasketaan *Tuote*-yksilön *hinta*-attribuutista sekä *Tuote*- ja *Myynti*-yksilöiden ilmentymien määrästä *Sisältyy*-yhteydessä.



Kuva 2: ER-kaavio *Market*-esimerkkimyyntijärjestelmän tiedoista tietovarastossa.

Käsitetason suunnittelussa voi helposti tapahtua virheitä, jotka johtuvat esimerkiksi yksilöiden välisten yhteyksien väärinymmärryksistä tai epätarkkuuksista [CoB02 s. 351-355][SKS11 s. 361-362]. Erityisesti suunnittelussa tulee pyrkiä välttämään tietojen toisteisuutta ja mallin epätäydellisyyttä. Käsitetason kaaviossa esiintyvät virheet tulevat esiin myöhemmissä suunnittelun, toteutuksen tai käyttöönoton vaiheissa, jolloin ne johtavat ongelmiin ja vaativat korjaavia toi-

menpiteitä, esimerkiksi iteraatiokierroksia. Tämän vuoksi on tärkeää, että käsitetason suunnittelu suoritetaan mahdollisimman huolellisesti ja osaavasti. Käsitekaavion suunnitteluun voidaan valita erilaisia strategioita, joita ovat osittava, kokoava, leviävä sekä paloitteleva strategia [BCN92 s. 66-76]. *Osittavassa strategiassa* (top-down) käsitetason kaavio aloitetaan kuvaamalla yksi yksilö, joka kuvaa kaikkea mallinnettavaa tietoa, minkä jälkeen kaaviota tarkennetaan jakamalla yksilöitä pienempiin osiin, kuvaamalla niiden ominaisuudet attribuuteilla ja liittämällä ne toisiinsa kuvaavilla yhteyksillä. *Kokoavassa strategiassa* (bottom-up) kerätään ensin kaikki mallinnettaviin tietoihin liittyvät attribuutit, ilman minkäänlaista rakennetta. Seuraavaksi attribuutit ryhmitellään yksilöiksi, jotka kuvataan eri abstraktiotasoilla ja lopuksi yksilöt liitetään toisiinsa yhteyksillä. *Leviävässä strategiassa* (inside-out) pyritään ensin löytämään tärkeimmät käsitteet, joihin vähitellen liitetään uusia käsitteitä ja näihin taas uusia käsitteitä, kunnes tietomalli on valmis. *Paloitteleva strategia* (mixed) jakaa ongelmakentän paloihin, jotka mallinnetaan itsenäisesti ja lopuksi yhdistetään yhdeksi tietomalliksi. Osittavaa strategiaa suositellaan käytettäväksi aina kun mahdollista [BCN92 s. 76]. Niissä tapauksissa, joissa se ei tunnu luontevalta, kannattaa valita ongelmakenttään parhaiten sopiva strategia. Esimerkiksi kokoava strategia sopii pienikokoisten tietokantojen suunnitteluun [CoB02 s. 279].

Tietovarastoinnissa tietomallin käsitesuunnittelu antaa tietovaraston toteutukselle hyvän lähtökohdan. Tietovarasto toteutetaan yleensä vaiheittain ja tietomalli sitoo eri vaiheet yhteen, jolloin lopputuloksena on yhtenäinen ja tiivis kokonaisuus [Inm02 s. 102]. Käsitetason kuvaus antaa tietovaraston sisältämistä tiedoista myös hyvän yleiskuvan ja toimii osana järjestelmän dokumentaatiota. Yhtenäisen tietomallin ansiosta eri toteutusvaiheiden työmäärä vähenee ja toteutus tehostuu.

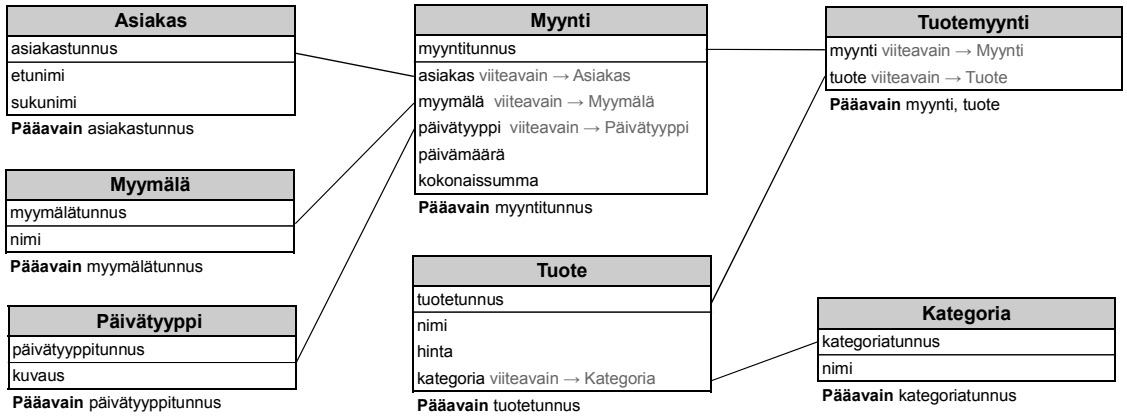
2.3 Looginen suunnittelu

Käsitesuunnittelun jälkeen toteutetaan looginen suunnittelu, jossa käsitetason tietomalli muunnetaan tietokannan hallintajärjestelmän käyttämään tietomalliin. Yleisin tietokannan hallintajärjestelmien käyttämä tietomalli on *relaatiomalli*, joka on käytännössä syrjäyttänyt sitä edeltäneet verkkomallin (network data model) ja hierarkkisen tietomallin (hierarchical data model) [SKS11 s. 9]. Uudempia vaihtoehtoja edustavat oliotietokannat ja XML-tietokannat, mutta niitä ei käsitellä tässä tutkielmassa. Loogisen tason relaatiomalli muodostetaan käsitetason kaaviosta kuvaamalla tieto relaatiotietokannan tauluina. Myös taulujen relaatioihin liittyvät *eheysrajoitteet* kuvataan loogisessa tietomallissa [CoB02 s. 442].

Loogisen suunnittelun aikana muodostettavaa relaatiomallia testataan ja validoidaan muun muassa käyttäjävaatimusten pohjalta [CoB02 s. 442]. Tässä vaiheessa pyritään myös huolehtimaan, että tietokanta ei sisällä turhaa toisteisuutta, vastaa kaikkeen tarpeisiin, sen sisältämä tieto

on eheää ja sitä voidaan käsitellä tehokkaasti. Jos käsitetason ER-mallinnus on tehty huolellisesti ja yksilöt sekä yhteydet tunnistettu oikein, käsitetason malli voidaan muuntaa relaatiomalliksi melko suoraviivaisesti ja lopputuloksena on normaalimuodossa oleva tietokanta. Jossain tapauksissa ER-kaavioon voi kuitenkin jäädä toisteisuutta aiheuttavia funktionaalisia riippuvuuksia [SKS11 s. 361]. *Funktionaalinen riippuvuus* tarkoittaa relaatioon kuuluvien attribuuttien välistä suhdetta, jossa yhden relaation attribuutin (tai attribuuttijoukon) arvoon liittyy täsmälleen yksi toisen attribuutin (tai attribuuttijoukon) arvo [CoB02 s. 379]. Tällainen funktionaalinen riippuvuus voi aiheuttaa tietokantaan tietojen toisteisuutta. Toisteisuus voi olla seurausta huonosta käsitetason suunnittelusta, mutta joissakin harvinaisemmissa tapauksissa ER-merkintätapa voi tuottaa kaavioita, joissa funktionaalisia riippuvuuksia on vaikea havaita [SKS11 s. 361]. Näin voi käydä esimerkiksi käytettäessä *moniasteisia yhteyksiä*, joihin liittyy enemmän kuin kaksi yksilötyyppiä. Tämä on estetty joissakin ER-merkintätavoissa kieltämällä muut kuin kaksiasteiset yhteydet [SKS11 s. 362]. Loogiseen relaatiomalliin jäänyttä toisteisuutta voidaan ehkäistä normaalistamisalgoritmeilla, joiden avulla relaation sisältämien funktionaalisten riippuvuuksien määrä minimoidaan. Normalisointiin voidaan käyttää esimerkiksi 3NF-normaalistamisalgoritmia, joka tuottaa kolmannen normaalimuodon mukaisen relaatiokaavion tai BCNF-normaalistamisalgoritmia, joka tuottaa Boyce-Codd normaalimuodossa olevan relaatiokaavion. Määritelmän mukaan kolmannessa normaalimuodossa olevan relaation jokaisella attribuutilla on tasan yksi arvo, kaikki attribuutit ovat funktionaalisesti riippuvia pääavaimesta (primary key) eikä mikään attribuutti ole epäsuorasti (transitiivisesti) funktionaalisessa riippuvuudessa pääavaimeen [CoB02 s. 397]. Normalisoinnin seurauksena relaatio, joka sisältää muita kuin avainriippuvuuksia, ositetaan häviöttömästi useammaksi relaatioksi, jotka täyttävät normaalimuodon vaatimukset.

Tietovaraston tietomallin loogisessa suunnittelussa kuvataan yksityiskohtaisemmin käsitesuunnittelun aikana toteutettu tietomalli (kuva 3). Attribuuttilistausta tarkennetaan, tiedot kuvataan relaatioina ja eheysrajoitteet ilmaistaan. Hyvin kuvattu looginen tietomalli on tärkeä dokumentti myös ylläpidon kannalta ja sen vuoksi sitä tulisi jatkuvasti ylläpitää [CoB02 s. 282]. Tarve normalisointialgoritmien käyttöön on tietovaraston tietomallin suunnittelussa kuitenkin käytännössä harvinainen, sillä yleensä tietovaraston tietoja päivitetään harvoin ja päivitysanomaliat eivät aiheuta paljon lisätyötä. Suorituskykysyistä tietovarasto sisältää yleensä aina myös jonkin verran koostettua tietoa ja toisteisuutta, eikä täydellisessä normaalimuodossa oleva tietokanta ole tietovarastoinnissa ehdoton tavoite [Inm02 s. 137]. Täysin normalisoitu tietokanta nimittäin sisältää yleensä suuren määrän tauluja, joiden väliset liitokset tekevät tosiaikaisessa tiedonjalostuksessa käytettävistä kyselyistä raskaita. Jos loogisen suunnittelun aikana kuitenkin huomataan tietomallissa epätoivottua toisteisuutta, käytännöllisempi vaihtoehto voi olla käsitemallin suora korjaaminen kuin normalisointialgoritmien käyttö [SKS11 s. 362].



Kuva 3: Looginen tietomalli *Market*-esimerkkimyyntijärjestelmän tiedoista.

2.4 Fyysinen suunnittelu

Tietokannan fyysisessä suunnittelussa päätetään, kuinka käsitetason ja loogisen tason tietomalli toteutetaan tietyllä tietokannan hallintajärjestelmällä. Fyysiseen suunnitteluun ja toteutukseen liittyy monia teknisiä yksityiskohtia, kuten tietokantaulujen luonti, tietojen tallennusrakenteen valitseminen sekä suorituskyvystä ja tietoturvasta huolehtiminen. Tietovaraston tietokannan fyysisessä suunnittelussa voidaan erityisesti suorituskyvyn parantamiseksi joutua käyttämään monipuolisia keinoja, jotka saattavat johtaa myös tietomallin muuttumiseen. Tällöin loogisen tason mallia korjataan uusien vaatimusten mukaisiksi [CoB02 s. 282]. Fyysisen suunnittelun aikana suunnittelijan on tiedettävä mitä tietokannan hallintajärjestelmää aiotaan käyttää ja tunnettava hyvin sen toiminta sekä sen tarjoamat ominaisuudet. Tietokantataulujen suunnitteluun kuuluu muun muassa taulujen nimien päättäminen sekä attribuuttien nimien ja ominaisuuksien suunnittelu. Tietokantataulut luodaan tietokannan hallintajärjestelmän mahdollistamalla tavalla, yleensä jollakin *tiedonkuvauskielellä* (Data Definition Language, DDL), joihin kuuluvat esimerkiksi SQL-kielen *create table* -lauseet. Tietojen pysyvään tallennukseen käytetään yleensä levyjärjestelmää ja erityisesti suorituskyvyn kannalta on tärkeää, että levylle sijoitettujen tietojen tietorakenne on optimaalinen. Suorituskykyä voidaan mitata esimerkiksi transaktioiden läpimenoajalla, yksittäisen transaktion vasteajalla ja käytetyllä levykapasiteetilla [CoB02 s. 484]. Tietokannan tiedolle ja sen käyttötavoille voidaan tehdä analyyseja, jotka auttavat suorituskyvyn varmistamisessa. Tiedostorakenne voidaan toteuttaa käyttäen hyväksi esimerkiksi kasarakennetta (heap), hajautusrakennetta (hash), B-puu-rakennetta, ISAM-hakemistoa tai klusteroituja tauluja [CoB02 s. 491-493].

Tiedostorakenteen lisäksi suorituskykyyn voidaan vaikuttaa relaatioille luoduilla *hakemistoilla* (index), joiden avulla tietokantaan voidaan kohdistaa nopeita hakuja tietyn attribuutin tai attribuuttijoukon perusteella. Hakemistojen suunnittelu on erityisen tärkeä fyysisen suunnittelun

osa, jolla on vaikutusta suorituskykyyn ja kyselyiden vasteaikaan. Jokaisella taululla on yleensä yksi taulun pääavaimeen perustuva *päähakemisto*, minkä lisäksi sille voidaan luoda muihin attribuutteihin perustuvia *oheishakemistoja*. Päähakemisto on yleensä *ryvästävä* eli monikot ovat levyllä hakemistoavaimen mukaisessa järjestyksessä ja *harva* eli hakemistossa ei tarvitse ole viitettä taulun jokaiseen monikkoon. Oheishakemistot taas ovat yleensä ryvästämättömiä ja tiheitä. Hakemistoavain voi perustua yhteen taulun attribuuttiin tai useampaan attribuuttiin, jolloin kyseessä on niin sanottu *lihava* tai *puolilihava* hakemisto. Puolilihava hakemisto sisältää useamman kuin yhden kyselyn ehto-osassa määritellyn attribuutin, lihava hakemisto sisältää kaikki kyselyn ehdossa käytetyt attribuutit. Yleisimmin päähakemistona käytetään B-puu-rakennetta. B-puun korkeus on yleensä muutama taso ja sen avulla *hakemistoavainta* vastaava monikko löytyy erittäin nopeasti. Muita hakemistoratkaisuja ovat hajautushakemisto, bittikarttahakemisto, projektiohakemisto, bittiviipalehakemisto ja liitoshakemisto. Kutakin hakemistoratkaisua voidaan käyttää parantamaan suorituskykyä erilaisissa tilanteissa. Hakemistojen suunnittelu ja valinta riippuu käytetystä tietokannan hallintajärjestelmästä, tietokannan rakenteesta ja yleisimmin käytetyistä kyselyistä. Erityisesti oheishakemistojen suunnittelu on monimutkainen tapauskohtainen ongelma, jonka ratkaisemiseen on vaikea antaa tarkkoja yleisiä ohjeita. Hakemistojen suunnittelu, luominen ja ylläpito jatkuukin yleensä koko tietovaraston elinkaaren ajan [KRT08 s. 344-350].

2.5 Tietovaraston tietomallin tuottaminen

Tietovarastossa käsitellään yleensä laajoja tietokokonaisuuksia, sillä se kokoaa yhteen tietoa kaikesta yrityksen toiminnasta. Tietovarasto on keskeinen komponentti *yritystason tiedonhallinnan* (corporate information factory, CIF) muodostavassa arkkitehtuurissa [Inm02 s. 388][ISN08 s. 12]. Yritystason tiedonhallinnan muodostavassa arkkitehtuurissa tietovarastoa ympäröivät sekä *lähdejärjestelmät*, jotka tuottavat tietoa tietovarastoon, että *järjestelmät jotka käyttävät hyväkseen tietovaraston yhtenäistettyä tietoa*. Lähdejärjestelminä voivat toimia operatiiviset järjestelmät, käytöstä poistuneet järjestelmät, yhtenäistettyä tietoa operatiivisille järjestelmille tarjoavat tilannekannat (operational data storage, ODS), laskentataulukot ja muut dokumentit sekä yrityksen ulkopuoliset järjestelmät.

2.5.1 Yritystason tietomalli

Koska tietovarasto voi sisältää tietoa kaikesta yrityksen toiminnasta, tarvitaan sen suunnittelussa myös *yritystason tietomallia* (corporate data model). Yritystason tietomalli kuvaa yrityksen tietotarpeet pääaiheittain ja se voidaan ottaa lähtökohdaksi tietovaraston suunnittelulle [Inm02 s. 278]. Tyypillisiä pääaiheita ovat esimerkiksi asiakas, tuote ja tilaus. Näkemykset siitä, kuinka

tarkka yritystason tietomallin tulisi olla, vaihtelevat. Inmon esittää, että kustakin pääaiheesta tulee olla tunnistettuna ainakin nimi, kuvaus, aliluokat, yhteydet, avaimet, attribuutit ja moniarvoiset attribuutit (repeating groups) [Inm02 s. 278]. Myöhemmin julkaistu [ISN08] arvioi, että yritystason loogisen tietomallin ei tulisi sisältää täyttä attribuuttilistausta vaan ainoastaan pääaiheet ja näiden väliset yhteydet, sillä yksityiskohtaisen yritystason loogisen tietomallin tuottaminen olisi liian aikaa vievää [ISN08 s. 292-293]. Jälkimmäistä näkökulmaa voidaan useissa tapauksissa pitää mallin laajuuden vuoksi tarkoituksenmukaisempana, sillä yritystason tietomallin tulee pääsääntöisesti antaa yleiskuva yrityksen operatiivisissa järjestelmissä ja dokumenteissa olevista tiedoista.

Yritystason tietomalli kuvaa suurelta osin samoja asioita kuin ne tietomallit, joilla yksittäisen operatiivisten järjestelmän tiedot on kuvattu. Operatiivisten järjestelmien käyttämä tietomalli on kuitenkin *sovelluskohtainen*, kun taas yritystason tietomalli on *aihekeskeinen* [ISN08 s. 161-164]. Sovelluskohtainen tietomalli on tehty yksittäisen sovelluksen vaatimuksista, mutta yritystason tietomallissa eri operatiivisten järjestelmien sisältämät tiedot on kuvattu yhtenäisesti ja aihekeskeisesti. Esimerkiksi asiakastietoja käsitellään laskutusjärjestelmässä vain laskutuksen kannalta, mutta yritystason tietomallissa asiakastiedot voi olla yhdistetty laskutusjärjestelmästä, tilausjärjestelmästä ja asiakaspalvelujärjestelmästä.

Yritystason tietomalli rakennetaan asteittain ja se vaatii jatkuvaa ylläpitoa [ISN08 s. 129-130]. Yritystason tietomallia ei yleensä pystytä luomaan pelkästään osittavilla, ylhäältä alaspäin tapahtuvilla mallinnusmenetelmillä, jotka perustuvat johdon ja asiantuntijoiden näkemyksiin [ISN08 s. 294-296]. Käytännössä yritykselle kerääntyy operatiivisiin järjestelmiin vuosien aikana dokumentoimattomia muutoksia ja järjestelmien todellinen tietosisältö täytyy selvittää [ISN08 s. 294-296]. Järjestelmien sisältämän tiedon selvittämisessä voidaan käyttää hyväksi erilaisia tiedon profilointityökaluja, joilla tietokannoista voidaan takaisinmallintaa tietoja yritystason tietomalliin. Yritystason tietomalli kannattaa siis muodostaa sekä ylhäältä alaspäin tapahtuvalla mallinnuksella että alhaalta ylöspäin tapahtuvalla takaisinmallinnuksella. Jos yritys on huolehtinut yritystasoisesta tietojen hallinnasta ja dokumentoinnista, niin yritystason tietomalli on jo olemassa, kun päätös yrityksen tietovaraston rakentamisesta tehdään. Jos näin ei ole, tai tietomalli ei ole riittävän huolellisesti kuvattu, niin tietovaraston kehitystyö kannattaa pysäyttää, kunnes riittävän laadukas yritystason tietomalli on saatavilla [Inm02 s. 358]. Tietovaraston kehityksen ensimmäinen vaihe on siis yritystason tietomallin analysointi. Vasta kun käytössä on riittävän laadukas yritystason tietomalli, voidaan sen pohjalta luoda *tietovaraston tietomalli* [Inm02 s. 90-91].

2.5.2 Tietovaraston tietomalli

Kun yritystason tietomallia sovelletaan tietovarastoon, siihen täytyy yleensä tehdä joukko muutoksia. Tällaisia muutoksia ovat puhtaasti operatiivisen tiedon poistaminen, aikakenttien lisääminen, johdetun tiedon lisääminen ja tiedon osittaminen sen mukaan kuinka usein tieto muuttuu [Inm02 s. 89-91].

Jotta tietojen mallintaminen ei jatkuisi loputtomiin, täytyy mallintajan, yrityksen johdon ja tietovaraston käyttäjien ensin päättää, kuinka laaja tietovaraston tietomallista tehdään ja mitä tietoja otetaan mukaan integraatioon [Inm02 s. 92]. Tietovarastoa ei yleensä kannata mallintaa ja toteuttaa kerralla valmiiksi, vaan jakaa yritystason tietomalli kokonaisuuksiin, jotka voidaan mallintaa ja toteuttaa tietovarastoon osa kerrallaan. Tietovaraston käyttäjillä on tärkeä rooli tietovaraston tietojen mallinnuksessa, jotta tietovarasto sisältäisi oikeat tiedot, oikein järjestettynä ja oikealla tavalla esitettynä. Koska tietomallin suunnittelu tapahtuu tietovaraston kehittämisen alkuvaiheessa, sitouttaa tietomallin suunnittelu loppukäyttäjät alusta asti tietovaraston kehittämiseen.

Tietovaraston tietomallinnukseen voidaan valita mikä tahansa tarkoitukseen sopiva lähestymistapa, esimerkiksi kolmivaiheinen suunnittelumenetelmä, johon kuuluu käsitesuunnittelu, looginen suunnittelu ja fyysinen suunnittelu. Inmon esittelee tietovaraston tietomallin suunnitteluun hieman erilaisen kolmivaiheisen suunnittelumenetelmän, joka perustuu korkean tason ER-mallinnukseen (high-level ERD), keskitason mallinnukseen (midlevel modeling) ja matalan tason fyysiseen suunnitteluun (low-level physical modeling) [Inm02 s. 92-101]. *Korkealla tasolla* mallinnetaan vain yksilöt ja niiden väliset yhteydet osallistumisrajoitteineen ja malli kuvataan *ER-kaavioina*. Attribuutteja ei kuvata. *Keskitasolla* korkean tason ER-malli tarkennetaan *DIS-kaavioksi* (data item set) tunnistamalla yksilötyyppien avaimet, attribuutit ja ominaisuudet. DIS-kaavio muodostuu tiedon pääryhmistä, toissijaisista ryhmistä, yhdistäjistä ja tiedon tyypeistä. Pääryhmän kuvaus sisältää yksilöiden avaimet ja attribuutit. Toissijainen ryhmä sisältää yksilötyyppien ne attribuutit, jotka voivat sisältää toisteista tietoa. Yhdistäjä liittää yksilöt toisiinsa ja osoittaa viiteavainten olemassaolon. Tiedon tyyppi osoittaa yksilön ilmentymien jakautumista eri tyyppeihin, esimerkiksi pankkitransaktio voi jakautua talletuksiin ja nostoihin. *Matalan tason* fyysisessä suunnittelussa DIS-kaavioista mallinnetaan joukko relaatiomallin mukaisia tauluja, jotka muodostavat mallinnusmenetelmän fyysisen tason. Ennen kuin tietokanta voidaan luoda, täytyy tietoa kuitenkin yleensä vielä järjestellä uudelleen suorituskyvyn näkökulmasta.

2.6 Suorituskyvyn parantaminen

Täydellisesti normalisoitu tietomalli ei yleensä sellaisenaan vielä sovi tietovaraston tietokannaksi, koska normalisointi hajottaa tiedon lukuisiksi relaatiotietokannan tauluiksi. Esimerkiksi toiminnanohjausjärjestelmien normalisoitu tietokanta saattaa koostua tuhansista tauluista [KRT08 s. 236]. Useista tauluista koostuva normalisoitu tietokanta minimoi tiedon toisteisuuden ja estää päivitysanomaliat, mutta tietokannan taulurakenne on vaikeasti ymmärrettävä ja paljon liitoksia sisältävien kyselyjen vasteaika kasvaa. Erityisesti tietovarastoissa tämä on ongelma, koska kyselyjen vasteaika on tietovaraston käytössä tärkeä suorituskykymittari ja kyselyt ovat usein monimutkaisia ja kohdistuvat suureen joukkoon tauluja. Riittävän vasteajan takaamiseksi niissä ratkaisuissa, joissa valitaan tietovaraston tietomalliksi normaalimuodossa oleva tieto, täytyy tietoa yleensä järjestellä tai täydentää toimenpiteillä, jotka saattavat sisältää jonkin verran epänormalisointia. Suorituskykyä parantavia menetelmiä ovat esimerkiksi taulujen yhdistäminen ja osittaminen, toisteisuuden salliminen, johdetun tiedon laskeminen, tietolohkojen käyttö, käyttöön perustuvien hakemistojen lisääminen ja viite-eheyksistä luopuminen [Inm02 s. 102-110] sekä materialisoitujen näkymien käyttö [ChD97].

Taulujen yhdistäminen voi olla tehokas tapa parantaa suorituskykyä, mutta johtaa yleensä tiedon toisteisuuteen ja epänormalisointiin. Jos esimerkiksi osoitetietoja sisältävä taulu yhdistetään asiakkaan perustiedot sisältävään tauluun, voivat tietyt asiakastietoja hakevat kyselyt nopeutua, mutta yhdistämisen seurauksena samat osoitetiedot (esimerkiksi postinumero) toistuvat usean eri asiakkaan kohdalla. Yhdistämisen jälkeen taulujen välistä liitosta ei kuitenkaan tarvitse tehdä kyselyn suoritusaikana, minkä vuoksi toimenpide voi parantaa kyselyiden vasteaikaa.

Yhdistämisen lisäksi kyselyiden vasteaikaa voidaan parantaa myös *osittamalla tauluja* pienemmiksi tauluiksi. Jotkin taulun tiedoista voivat olla sellaisia, että niitä kysytään erittäin harvoin ja ne voidaan osittaa omaksi tauluksi. Esimerkiksi tilitiedoissa voi olla kentät *saldo*, tilin a*vauspäivä* ja tilin *avauspaikka*. Näistä *saldoa* saatetaan kysellä usein, mutta *tilin avauspäivämäärää* ja *avauspaikkaa* ei kysellä juuri koskaan. Tällöin voi olla järkevää osittaa harvemmin kysellyt kentät *avauspäivä* ja *avauspaikka* omaksi tauluksi.

Toisteisuutta voidaan lisätä rajoitetusti kopioimalla yksittäisiä kenttiä taulusta toiseen. Esimerkiksi tuotteen nimi voitaisiin kopioida *Tuote*-taulusta *Tilaus*-tauluun. Tällöin tilaustietoja hakevissa kyselyissä, jotka tarvitsevat *Tuote*-taulusta ainoastaan tuotteen nimen, ei tarvitse tehdä liitosta taulujen *Tilaus* ja *Tuote* välillä. Liitokset voivat olla hyvin raskaita operaatioita, varsinkin suurten taulujen yhteydessä, joten niitä vähentämällä voidaan suorituskykyä parantaa merkittävästi. Tarkoituksellisen toisteisuuden lisäämisestä seuraa, että tietokanta ei ole enää täydellisesti normalisoitu, mikä saattaa johtaa päivitysanomalioihin. Tietovaraston tietoa kuiten-

kin päivitetään vain harvoin, joten päivitysanomaliat eivät yleensä ole tietovarastossa ongelma. Normalisoinnilla on kuitenkin päivitysanomalioiden estämisen lisäksi myös muita hyviä puolia, kuten tallennustilan säästäminen, joustavuus ja sopivuus tietomalleihin ja mallinnusmenetelmiin, joten epänormalisointia tulee käyttää harkitusti [Inm02 s. 137].

Toisteisuuden sallimisen sijaan parempi keino suorituskyvyn parantamiseen voi olla *materialisoitujen näkymien käyttäminen* [SKS11 s. 363]. Materialisoidut näkymät ovat etukäteen laskettuja kyselyjä, jotka tietokannan hallintajärjestelmä päivittää automaattisesti tietyin väliajoin, esimerkiksi aina kun näkymään liittyviä tauluja päivitetään. Esimerkiksi kahden tai useamman taulun liitos voidaan tallentaa materialisoiduksi näkymäksi, jolloin kyselyssä voidaan käyttää suoraan materialisoitua näkymää, eikä raskasta liitosta tarvitse tehdä kyselyn suoritusaikana. Materialisoitu näkymä voi sisältää myös muita SQL-operaatioita ja näkymien käyttäminen on yleisesti mahdollista nykyaikaisissa tietokannan hallintajärjestelmissä [SKS11 s. 363]. Koska raskaita SQL-operaatioita ei tarvitse tehdä kyselyn suoritusaikana, materialisoiduilla näkymillä voidaan merkittävästi parantaa kyselyiden vasteaikaa kunhan niiden valinta perustuu käyttäjien tekemiin kyselyihin.

Materialisoituja näkymiä hieman yksinkertaisempi tapa parantaa suorituskykyä on tallentaa tietokantaan *johdettuja tietoja*. Esimerkiksi kuukausittain tai vuosittain laskettavat palkkatulot voidaan laskea kerran jokaisen kuukauden ja vuoden lopussa, jolloin tieto on suoraan kyselyiden käytettävissä. Johdettujen tietojen tallentamiseen voidaan käyttää erillisiä summatauluja tai olemassa oleviin tauluihin tarkoitukseen varattuja kenttiä.

Tietolohkoja voidaan käyttää vähentämään levyhakujen määrää, ryhmittelemällä tietoa siten, että yhdellä levyhaulla voidaan yksittäisen sivun sijasta hakea joukko sivuja. Tietovarastoissa tällainen lohkoittainen tiedon tallentaminen ja lukeminen on toimiva tapa, koska tietovaraston tiedot ovat aikasidonnaisia ja lohkoja voidaan muodostaa ajankohdan perusteella. Esimerkiksi myyntitapahtumia voitaisiin tietovarastossa luontevasti lohkoa kuukausittaisiksi kokonaisuuksiksi. Niin sanotuissa sarakepohjaisissa tietokannoissa (column oriented database), jotka soveltuvat erityisen hyvin tietovarastojen tietokannan hallintajärjestelmiksi, lohkoittain tapahtuva tietojen luku on yleensä sisäänrakennettu ominaisuus [AMH08]. Lisäksi sarakepohjaisissa tietokannoissa ei tarvitse lukea levyltä kuin ne attribuutit, jotka ovat mukana kyselyssä ja tiedon tiivistäminen on mahdollista tehdä tehokkaammin [CDN11].

Käyttäjien käyttötarpeiden mukaan voidaan tietovaraston tietokantaan luoda *hakemistoja* parantamaan suorituskykyä. Tekniset haasteet ovat vastaavia kuin materialisoitujen näkymien käytössä: hakemistojen oikea valinta sekä niiden tehokas käyttö ja päivittäminen [ChD97]. Hakemistojen päivittäminen on tietovarastoinnissa kätevää, sillä tietovarastoon tuotu tieto joudutaan aina käsittelemään raskaan latausoperaation kautta ja latausoperaation yhteydessä

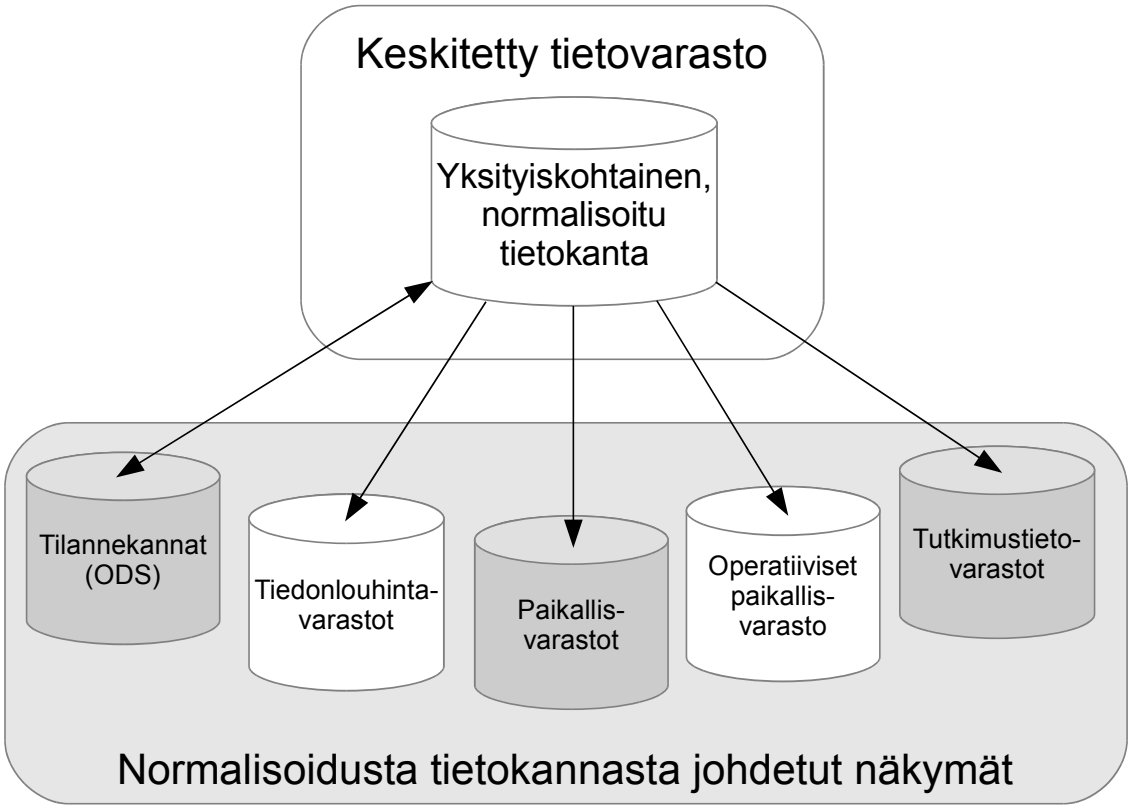
voidaan suhteellisen pienellä lisätyöllä suorittaa myös käyttöön perustuvien hakemistojen päivittäminen.

Myös operatiivisissa järjestelmissä esiintyvistä *viite-eheyksistä luopuminen* voi parantaa suorituskykyä. Tämä on tietovarastossa joskus mahdollista ja tarpeellista, koska tietovaraston tiedot ovat sekä muuttumattomia että aikasidonnaisia, mutta viite-eheyssäännöt sen sijaan saattavat muuttua ajan myötä. Operatiivisissa järjestelmissä määriteltyjä viite-eheyksiä voidaan tietovarastossa siis tarkoituksella rikkoa, esimerkiksi tietoa kopioimalla tai tietoa poistamalla.

2.7 Jalostetun tiedon tuottaminen normalisoidusta tietokannasta

Normaalimuodossa tai lähes normaalimuodossa oleva tietokanta ei yleensä ole ainoa tapa säilyttää liiketoimintatietoa tietovarastossa [ISN08 s. 290-292]. Jos tietovarasto ei ole kooltaan erityisen pieni, täytyy tietovaraston tietoa esittää eri karkeisuustasoilla [Inm02 s. 359]. Normaalimuodossa olevasta tietokannasta voidaan johtaa eri tavoin rajattuja ja muotoiltuja kokonaisuuksia eri käyttäjäryhmille. Normalisointiin perustuva *keskitetty tietovarasto* tarjoaa siis raakatietoa, joka sisältää lähdejärjestelmistä saadut tiedot tarkalla tasolla mutta kuitenkin yhdenmukaistettuna ja siitä johdetaan erilaisia näkymiä loppukäyttäjille. Käyttäjät käsittelevät tietoa erilaisilla työvälineillä, esimerkiksi raportointijärjestelmillä, moniulotteisen tiedon (ks. luku 3) esittämiseen tarkoitetuilla työvälineillä, tiedonlouhintasovelluksilla, johtamisen tukijärjestelmillä ja muilla erilaisilla kyselyvälineillä. Johdetut näkymät voivat olla normalisoidun tietokannan kanssa samaan tietokantaan eri tavoin toteutettuja lisäkokonaisuuksia, esimerkiksi materialisoituja näkymiä, summatauluja tai itsenäisiä tietokantakaavioita, mutta ne voivat olla myös erillisiä järjestelmiä. Yksityiskohtaista tietoa sisältävästä keskitetystä tietovarastosta voidaan johtaa paikallisvarastoja, tutkimustietovarastoja, tiedonlouhintavarastoja, tilannekantoja ja operatiivisia paikallisvarastoja (kuva 4).

Paikallisvarastot (data mart) ovat rajatun aihealueen tietovarastoja, jotka on suunniteltu rajatun käyttäjäryhmän tarpeisiin [HHK09 s. 188]. *Tutkimustietovarastot* (exploration warehouse) ovat usein väliaikaisia, raskaaseen tilastolliseen analyysiin tarkoitettuja varastoja, joita voidaan käyttää erilaisilla tilastointityövälineillä [ISN08 s. 13]. *Tiedonlouhintavarastot* (data mining warehouse) ovat varastoja, joissa tutkimustietovarastossa esiin tulleita hypoteeseja ja oletuksia voidaan testata [Inm07]. *Tilannekantojen* (operational data store, ODS) tarkoituksena on tarjota yhtenäistettyä tietoa operatiivisille järjestelmille [HHK09 s. 191] ja niitä voidaan toteuttaa useilla eri tavoilla. Tilannekanta voi tuottaa tietoa tietovarastoon, mutta tietovarastosta voidaan myös tuottaa tietoa tilannekantaan [Inm02 s. 143-144]. *Operatiiviset paikallisvarastot* (operational data marts) ovat väliaikaisia varastoja ja tilannekantoja pienempiä kokonaisuuksia, joissa säilytetään ainoastaan ajankohtaista tietoa [Imh01].



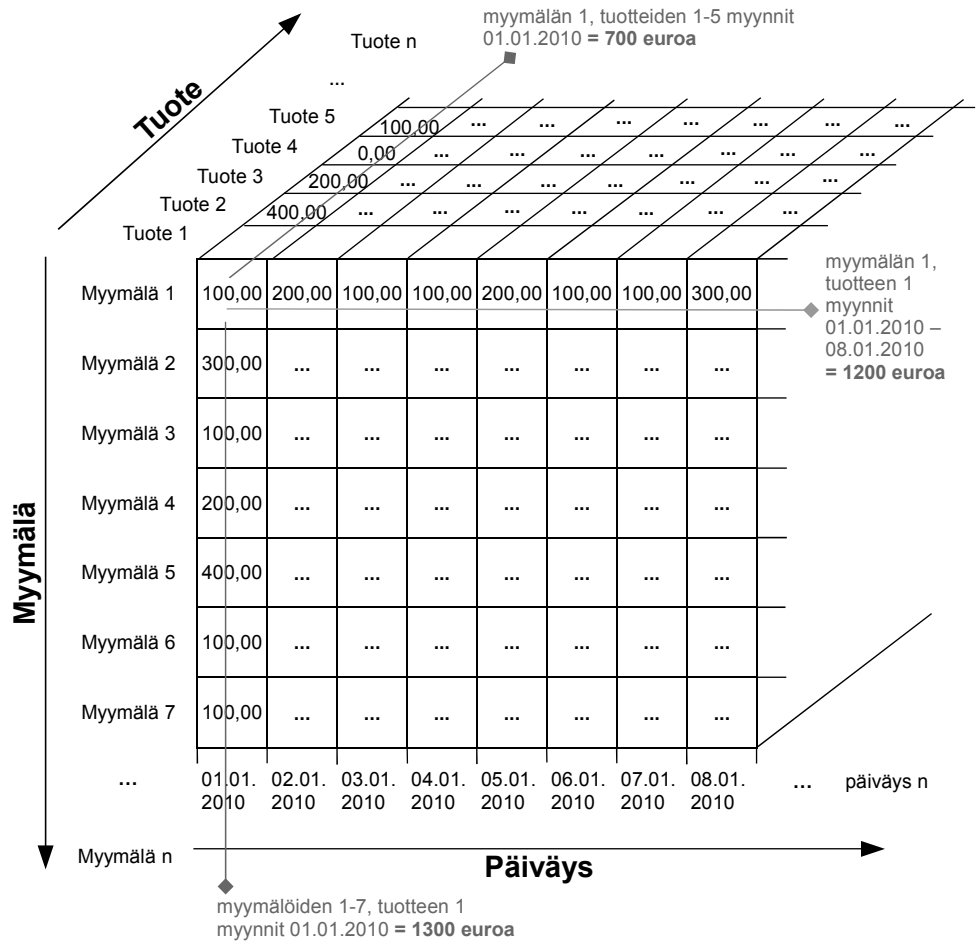
Kuva 4: Tietovaraston normalisoidusta tietokannasta johdettavat näkymät.

Normalisointiin perustuvasta keskitetystä tietovarastosta tuotetuilla muilla varastoilla on erilainen laajuus, tarkkuustaso, näkökulma, käyttäjäryhmä, elinkaari, tietomalli, tekninen toteutus ja ne voivat olla erillisiä järjestelmiä tai osakokonaisuuksia yksityiskohtaista tietoa sisältävässä tietovarastossa. Esimerkiksi paikallisvarastot ovat usein pitkäikäisiä ja osastokohtaisia, tähtimalliin (ks. luku 3) perustuvia, karkeistettua tietoa sisältäviä kokonaisuuksia, jota käyttävät liiketoiminta-analyytikot [ISN08 s. 152]. Tutkimustietovarastot taas ovat yleensä väliaikaisia, sisältävät yksityiskohtaista tietoa joka ei perustu tähtimalliin ja sitä käyttävät yleensä matemaatikot uusien havaintojen tekemiseen [ISN08 s. 152]. Jokainen varasto on kuitenkin suunniteltava ja toteutettava huolellisesti ja niille on hyvä suorittaa oma käsitetason, loogisen tason ja fyysisen tason tietomallinnus [ISN08 s. 290-291].

3 Moniulotteinen tietomalli

Käytännöllinen tapa esittää tietovarastoon tallennettua tietoa on moniulotteinen tietomalli, jota voidaan käyttää normalisoidun tietomallin täydentäjänä tai joidenkin näkökulmien mukaan jopa tietovaraston ainoana tietomallina [Kim97][KiR02][KRT08].

Moniulotteisessa tietomallissa pyritään kuvamaan liiketoimintaprosessien tuottamat numeeriset mitat, esimerkiksi myynnin määrät tai liikevaihto sekä mittoja kuvaavat tekijät kuten myyty tuote tai päiväys. Numeerisia mittoja kutsutaan *faktoiksi* ja niihin liittyviä tekijöitä *ulottuvuuksiksi* eli dimensioiksi. Faktat ovat liiketoiminnan analysoinnin kohteita ja niiden sisältämiä mitta-arvoja tarkastellaan ulottuvuuksien kautta. Esimerkiksi myyntitapahtumia (faktat) voidaan tarkastella tuotteittain (ulottuvuus) tai liikevaihtoa (faktat) voidaan tarkastella kuukausittain tai vuosittain (ulottuvuus). Kuvassa 5 on havainnollistettu *Market*-esimerkkimyyntijärjestelmän myyntejä kolmiulotteisena kuutiona, jossa ulottuvuudet ovat päiväys, tuote ja myymälä. Kuvassa on esitetty myös kolme ulottuvuuksien pohjalta laskettua esimerkkikoostetta.



Kuva 5: *Market*-esimerkkimyyntijärjestelmän myyntitietoja kolmiulotteisessa kuutiossa.

Kukin ulottuvuus voi sisältää hierarkkisia rakenteita. Esimerkiksi tuotteita voidaan ryhmitellä eri tuoteryhmiksi, jotka voivat edelleen jakautua alaryhmiksi. Tai päivämääriä voidaan kuvata puurakenteena, jossa laajemmat ajanjaksot osittuvat suppeammiksi, esimerkiksi vuodet jakautuvat vuosineljänneksiksi, neljännet kuukausiksi, kuukaudet viikoiksi ja viikot päiviksi. Hierarkian avulla tiedon tarkastelutasoa ulottuvuuden sisällä voidaan karkeistaa tai tarkentaa, esimerkiksi myyntejä voidaan tarkastella tuotteittain tai tuoteryhmittäin.

3.1 Moniulotteisen tiedon käsitesuunnittelu

Yleisesti hyväksytään ajatus, että tietovarastointiin kuuluu moniulotteinen tietomalli [ISN08] [KRT08][SBH99] sillä se kuvaa tiedon selkeästi ja havainnollisesti analysoinnin ja raportoinnin näkökulmasta. Operatiivisten järjestelmien käsitesuunnittelussa käytetty alkuperäinen ER-malli ei kuitenkaan sovellu sellaisenaan kovin hyvin moniulotteisen tiedon mallintamiseen [GMR98] [Kim97][SBH99]. ER-malli ei ole luonnollinen tapa kuvata moniulotteista tietoa, koska alkuperäisessä mallissa kaikki yksilötyypit ovat saman arvoisia eikä sillä pystytä esittämään ulottuvuuksien monimutkaista hierarkkista rakennetta [SBH99]. Loogisen tason moniulotteinen tietomalli on kyllä mahdollista toteuttaa ER-mallin pohjalta, mutta useiden ehdotusten mukaan tietovaraston tieto kannattaa mallintaa moniulotteisesti myös käsitetasolla [GMR98][HLV00] [Kam08][SBH99][TBC99]. Moniulotteinen käsitetason malli sisältää samat tiedot kuin ER-mallikin, mutta kuvaa tiedon ominaisuuksia monipuolisemmin ja toisista näkökulmista. Tilanne on hieman samanlainen kuin tilastotietokannoissa, joille on myös käytössä omia käsitetason graafisia kuvaustapoja ja niillä on mahdollista kuvata tietokannan tilastollisten taulujen merkitystä [SBH99].

Osa tutkijoista ja kehittäjistä on reagoinut ER-mallin rajoituksiin siten, että moniulotteisen tiedon käsitesuunnitteluun ei kiinnitetä huomiota lainkaan [KiR02] tai se on sotkettu loogiseen ja fyysiseen suunnitteluun [GMR98]. Vähäinen kiinnostus käsitesuunnitteluun voi johtua myös siitä, että kyselyjen vasteaikojen optimointi, joka tapahtuu pääasiassa loogisen suunnittelun ja fyysisen suunnittelun aikana, on tietovarastoinnissa poikkeuksellisen tärkeässä asemassa [GMR98]. Käsitesuunnittelun merkityksen hämärtyminen on kuitenkin johtanut sekaannuksiin ja jotkin lähteet yhdistävät käsitesuunnittelun ja loogisen suunnittelun [GMR98], toiset taas loogisen suunnittelun ja fyysisen suunnittelun [SBH99]. Osa lähteistä käyttää käsitesuunnittelusta termiä looginen suunnittelu [CaT98][HLV00]. Yleisesti hyväksytäänkin oikeastaan vain ajatus, että tietokannan fyysistä suunnittelua ja toteutusta tulee edeltää ainakin yksi korkeamman tason suunnitteluvaihe [HLV00].

Käsitesuunnittelu on kuitenkin tärkeä vaihe myös moniulotteisen tiedon hyväksikäytössä. Voidaan jopa väittää, että moniulotteisen tiedon mallintamisessa käsitesuunnittelu on erityisen

tärkeää [SBH99], esimerkiksi sen vuoksi, että moniulotteiselle tiedon tallennukselle on olemassa loogisella ja fyysisellä tasolla aidosti eri tavoin toteutettuja ratkaisuja, kuten ROLAP-, MOLAP- ja HOLAP-järjestelmät (ks. luvut 3.2 ja 3.3).

Koska alkuperäinen ER-malli on puutteellinen moniulotteisen tiedon mallintamiseen, on sen tilalle ehdotettu lukuisia uusia mallinnustapoja. Näistä kuitenkin vain harvat ovat aidosti käsitetason kuvaustapoja – toiset ovat loogisen tason malleja, toiset taas osittain loogisia ja osittain käsitteellisiä [Kam08]. Osa ratkaisuista ei tarjoa kunnollista graafista mallinnustapaa, vaan ne keskittyvät käsitemallin formaaliin muotoiluun. Jos graafista merkintätapaa ei oteta huomioon, on käsitteellisten mallien ilmaisuvoima yleensä kuitenkin hyvin saman vahvuinen [Gol09].

Käytännön tietokantasuunnittelussa selkeä ja ymmärrettävä graafinen merkintätapa on käsitesuunnittelussa tärkeä apuväline, esimerkiksi suunnittelijoiden ja käyttäjien välisen kommunikoinnin helpottamiseksi [Gol09]. Graafiset moniulotteisen tiedon mallinnusmenetelmät voidaan jakaa kolmeen ryhmään :

- ER-mallin laajennukset
- Oliopohjaiset mallinnusmenetelmät (esimerkiksi UML)
- Itsenäiset mallinnusmenetelmät

Kullakin lähtökohdalla on omat vahvuutensa: ER-malli on käytännössä testattu, yleisesti tunnettu ja laajennettuna se on osoittautunut riittävän joustavaksi ja ilmaisuvoimaiseksi myös moniulotteisen tiedon mallinnukseen. UML on erittäin ilmaisuvoimainen ja luonnollisesti laajennettava kieli, joka on standardoitu ja on yleistymässä oliopohjaisen sovellussuunnittelun suosion myötä. Itsenäiset mallit taas voivat ottaa moniulotteisen tiedon erityisluonteen huomioon ja voivat tarjota tehokkaan ja intuitiivisesti ymmärrettävän tavan kuvata tiedon moniulotteisuutta.

Moniulotteisen tiedon mallinnusmenetelmille on esitetty eri lähteissä kymmeniä erilaisia vaatimuksia. Esimerkiksi [Kam08] listaa 22 erilaista vaatimusta, jotka eri mallinnusmenetelmät täyttävät vaihtelevasti. Puhtaina käsitetason mallinnusmenetelminä [Kam08] pitää ulottuvuusfaktamallia (dimensional fact model, DFM) [GMR98], ME/R-mallia (multidimensional ER-model) [SBH99], starER-mallia [TBC99], DWPM-mallia (data warehouse process model) [HLV00] sekä itse esittelemäänsä CGMD (conceptual general multidimensional) -mallia [Kam08]. *Ulottuvuusfaktamalli* on ensimmäisiä moniulotteisen tiedon mallinnukseen tarkoitettuja itsenäisiä mallinnusmenetelmiä ja sitä pidetään edelleen tärkeänä vertailumenetelmänä muille malleille [RoA09]. Se on aito käsitetason malli ja siihen liittyy myös osittain automatisoitu menetelmä moniulotteisen mallin luomiseksi operatiivisten tietojärjestelmien ER-kaavioiden ja tietovaras-

ton käyttäjien vaatimusten pohjalta. *ME/R-malli* pyrkii laajentamaan ER-mallia mahdollisimman vähän, mutta kuitenkin riittävästi, jotta kaikki moniulotteisen tiedon peruskäsitteet voidaan ilmaista. Uusina elementteinä esitellään yksi uusi yksilötyyppi, jolla kuvataan ulottuvuuden tasoja sekä kaksi uutta yhteystyyppiä, joilla kuvataan faktayhteys ja karkeistus. *StarER-malli* pyrkii laajentamaan ER-mallia faktatyypillä ja joukolla uusia yhteystyyppejä, joilla tietovarastoissa yleisesti käytetty tähtimalli voidaan kuvata käsitetasolla. *DWPM* (Data Warehouse Process Model) on Hüsemannin ja kumppaneiden moniulotteisen käsitemallin tuottamiseen tarkoitettu menetelmä, jossa lähtökohtana on vaatimusmäärittely ja lopputuloksena moniulotteisessa normaalimuodossa oleva tietokantakaavio. *Moniulotteinen normaalimuoto* (MNF) on moniulotteiselle tiedolle määritelty normaalimuoto, jonka tarkoitus on estää koostamisessa tapahtuvia virheitä vastaavasti kuin perinteisten normaalimuotojen tarkoitus on estää tiedon päivittämisessä syntyviä virhetilanteita. *CGMD* on graafisen kuvaustavan sisältävä malli, joka on kehitetty formaalisti määritellyn GMD-tietomallin ja algebran pohjalta [FrK04]. GMD-malli puolestaan on yleistetty sitä edeltävän MD-tietomallin ajatuksista [CaT98].

Vain osittain käsitetason malleiksi [Kam08] katsoo MAC (multidimensional aggregation cube) -mallin [TKS01], EMDM-mallin (extended multidimensional datamodel) [JKP02][PJD01], OOCM-mallin (object-oriented conceptual model) [TPK01], GOLAP (general OLAP) -mallin [Pei03] sekä YAM^2 (yet another multidimensional model) -mallin [ASS06]. *MAC-malli* pyrkii kuvaamaan tietovaraston tiedot tosiaikaisessa tiedonjalostuksessa käytettyjen ulottuvuuksien, hierarkioiden, tasojen, mittojen ja kuutioiden käsitteiden avulla. *EMDM-malli* määrittelee algebran, jolla voidaan kuvata monimutkaista moniulotteista tietoa, kuten ulottuvuuksia, jotka sisältävät vaihtoehtoisia tai epätasapainoisia hierarkioita. Esimerkkitapauksena käytetään lääketieteellistä tietoa sisältävää tietovarastoa ja sen käytöstä tietomallille syntyviä vaatimuksia. *OOCM*-malli käyttää olioperustaisen UML-mallin laajennusmekanismeja, kuten lisätietomääreitä ja rajoitteita, moniulotteisuuden kuvaamiseen. *YAM^2-malli* on uudempi yritys luoda OOCM-mallia ilmaisukykyisempi, UML-kielen laajennusmekanismiin perustuva kuvaustapa. *GOLAP-mallilla* pyritään ratkaisemaan monimutkaisen tiedon, kuten geenitietojen tai osittain rakenteisen tiedon mallintamiseen liittyviä ongelmia.

Käsitetason mallin määritelmästä ja sen täyttävistä mallinnusmenetelmistä esiintyy selvästi tulkintaeroja: Esimerkiksi YAM^2-mallia sen tekijät pitävät itse täysin toteutusriippumattomana ja aitona käsitetason mallina [ASS06].

Moniulotteisille kuvaustavoille [Kam08] esittää muun muassa seuraavat vaatimukset:

1. Mallin on oltava toteutuksesta riippumaton.
2. Mallilla tulee kyetä esittämään kunkin ulottuvuuden eri tasot ja niistä syntyvät hierarkiat.
3. Yhdelle ulottuvuudelle tulee olla mahdollista esittää useita erilaisia hierarkioita.
4. Mallilla pitää voida esittää koosteita oikein.
5. Epätasapainoisten hierarkioiden, joissa polun pituus juuresta lehtitasolle vaihtelee, pitää olla mahdollisia.
6. Monesta moneen yhteyksien tulee olla mahdollisia faktojen, ulottuvuuksien ja ulottuvuushierarkioiden välillä.
7. Periytyminen (yleistys/erikoistapaus) pitää voida kuvata.
8. Ajan kuluessa tapahtuvat muutokset on voitava esittää.
9. Useiden kuutioiden esittäminen pitää olla mahdollista.
10. Mallilla täytyy voida kuvata käyttäjän määrittelemiä koostefunktioita.

Käsitemallien suhdetta vaatimuksiin kuten [Kam08] ne tulkitsee on esitelty taulukossa 2. Listan ulkopuolelle jää joukko muita moniulotteisen tiedon kuvaustapoja ja vaatimuksia, mutta se antaa yleiskuvan hyvin tunnetuista vaihtoehdoista ja niiden ominaisuuksista.

	Toteutus-riippuma-ton	Hierar-kioiden esitys	Useita hierarki-oita	Koostei-den esitys	Epätasai-set hier-arkiat	Monesta moneen yhteydet	Periyty-minen	Muutok-set ajassa	Useita kuutioita	Omat koosteet
DFM	X	X	X	O					O	
ME/R		X	X		O		O		O	
StarER		X	X	O		X	O			
DWPM		X	X	O	O		O			
CGMD	X	X	X	X	X	O	X		X	X
MAC		X	X		X	X				
EMDM		X	X	O	X	X		X		
OOCM		X	X	O	X	X	O		O	
GOLAP		X	X	O	O	X				
YAM²	X	X	X	X		X	O	O	X	X

Taulukko 2: Moniulotteisten mallien suhde vaatimuksiin (X = malli toteuttaa vaatimuksen, O = malli toteuttaa vaatimuksen osittain).

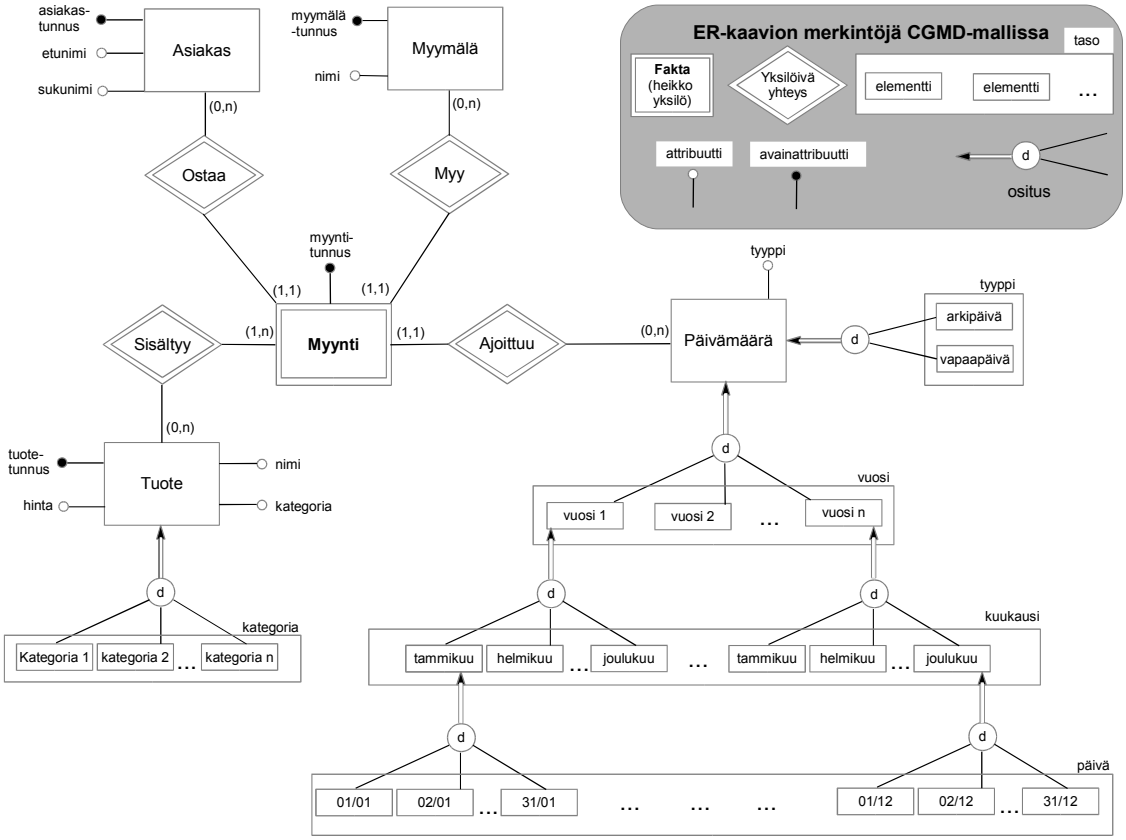
Mikään moniulotteisista käsitemalleista ei ole muodostunut standardiksi [Kam08]. Golfarellin, Maion ja Rizzin 1998 julkaisema ulottuvuusfaktamalli mainitaan ja esitellään lukuisissa lähteissä ja se on edelleenkin tärkeä vertailukohta muille malleille. Tässä luvussa esitellään lyhyesti Golfarellin, Maion ja Rizzin ulottuvuusfaktamallia uudempi, Kamblen vuonna 2008 julkaisema CGMD-malli [Kam08].

3.1.1 CGMD-malli

Kamblen *CGMD-malli* (conceptual general multidimensional model) on graafisen kuvaustavan sisältävä käsitetason mallinnusmenetelmä [Kam08]. CGMD-malli laajentaa ER-mallia ja sen avulla on mahdollista kuvata muun muassa tiedon moniulotteisuus, ulottuvuuksien hierarkiat ja erilaiset koosteet. Menetelmän tarkoituksena on luoda yleiskäyttöinen, tyylikäs ja tarkka malli, joka täyttäisi mahdollisimman suuren osan moniulotteisille tietomalleille asetetuista vaatimuksista. Sen avulla tietovaraston tiedon moniulotteiset rakenteet voidaan kuvailla abstraktilla, toteutuksesta riippumattomalla tavalla.

Alkuperäiseen ER-malliin kuuluvat *yksilötyypit*, *yhteystyypit* ja *attribuutit*. Yksilötyypit ja yhteystyypit merkitään CGMD-mallia esittelevässä artikkelissa samoin kuin muissakin ER-malleissa ja yhteydet voidaan merkitä osallistumisrajoitteilla. Attribuutit voidaan merkitä avoimilla ympyröillä ja yksilöivät attribuutit eli avaimet mustilla ympyröillä. Kuten jo aiemmin todettiin, ER-mallin symboleihin liittyvistä käsitteistä vallitsee laaja yksimielisyys, mutta merkintätavoista ei ole olemassa yleisesti hyväksyttyä standardia. Kamblen esittämät graafiset merkinnät voi siis nähdä vain yhtenä mahdollisena esittämistapana.

Alkuperäisen ER-mallin lisäksi CGMD-malli hyödyntää melko yleisesti käytössä olevia ER-mallin laajennuksia. Faktat esitetään *heikkoina yksilötyyppeinä* (kaksoisviivalla piirretty suorakulmio), joiden olemassaolo on riippuvainen ulottuvuuksista. Heikko yksilötyyppi (fakta) liitetään omistajayksilöihin (ulottuvuudet) *yksilöivien yhteyksien* (kaksoisviivalla piirretty vinoneliö) avulla. Moniulotteisessa tietomallissa esiintyviä *ulottuvuuksien hierarkioita* voidaan kuvata laajennetussa ER-mallissa käytössä olevan yleistämisen ja erikoistamisen avulla. CGMD-mallin esittelyssä yleistys kuvataan nuolella, jonka suunta on aliluokasta yliluokkaan. *Hierarkioiden eri tasot* merkitään sijoittamalla siihen kuuluvat yksilöt nimetyn suorakaiteen sisään. *Ositus* (disjoint) kuvaa yksilöjoukon jakautumista erillisiksi osajoukoiksi, esimerkiksi viikko osittuu erillisiksi viikonpäiviksi. Tämä kuvataan CGMD-mallissa hieman erikoisesti ympyröidyllä d-kirjaimella, josta osoittaa kaksiviivainen nuoli yliluokkaan ja yhteydet aliluokkiin. Kuvassa 6 on esimerkki *Market*-esimerkkimyyntijärjestelmän tiedoista moniulotteisella CGMD-käsitemallilla kuvattuna.

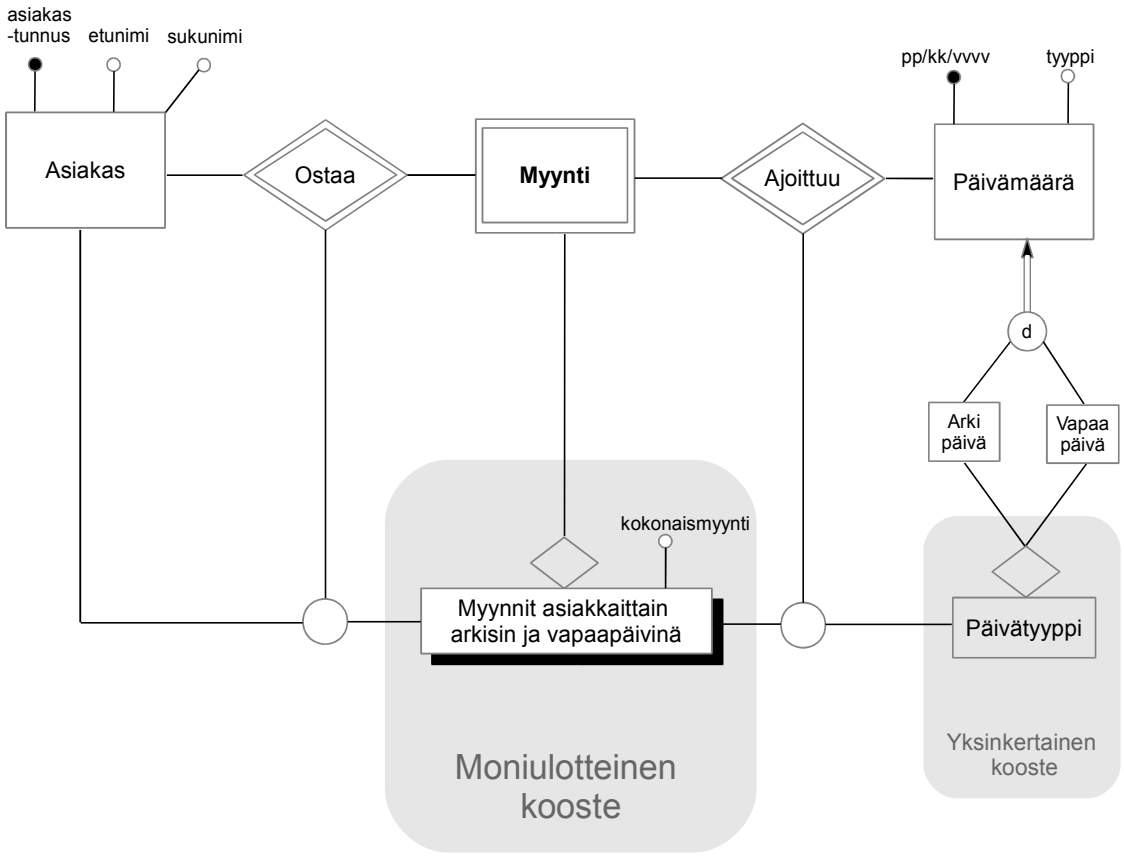


Kuva 6: Moniulotteinen CGMD-käsitemalli *Market*-esimerkkimyyntijärjestelmän tiedoista.

Täysin uutena ominaisuutena CGMD-malli lisää laajennettuun ER-kuvaukseen mahdollisuuden kuvata monipuolisesti myös koosteita, jotka voidaan jakaa yksinkertaisiin koosteisiin ja moniulotteisiin koosteisiin. *Yksinkertainen kooste* kokoaa yhteen joukon objekteja. Esimerkiksi yksilötyyppi-objektit *maanantai, tiistai... sunnuntai* voitaisiin koostaa yksinkertaiseksi koosteeksi nimeltä *viikonpäivä*. Yksinkertainen kooste kuvataan suorakulmiolla, johon on liitetty vinoneliö ja vinoneliöstä osoittavat yhteydet objekteihin, joista kooste muodostuu. *Moniulotteinen kooste* taas on faktataulun mitoista ulottuvuustaulujen avulla laskettu kooste, ja se kuvataan varjostettuna suorakulmiona, johon on liitetty vinoneliö. Vinoneliöstä kuvataan yhteys faktatauluun ja koosteesta kuvataan yhteydet ulottuvuuksiin ja faktataulun yksilöivään yhteyteen. Kuvassa 7 on esitetty yksinkertainen ja moniulotteinen kooste *Market*-esimerkkimyyntijärjestelmän myyntitiedoista CGMD-mallilla.

3.2 Moniulotteisen tiedon looginen suunnittelu

Normalisoituun tietomalliin pyrittäessä ER-mallin muuntaminen loogisen tason relaatiomalliksi on melko suoraviivaista, jos ER-malli on tehty huolellisesti ja oikein. Moniulotteisen tietomallin tapauksessa tilanne on monimutkaisempi. Ensinnäkin moniulotteinen tieto voidaan tallentaa re-



Kuva 7: Koosteiden esittäminen CGMD-mallilla.

laatiotietokantaan tai moniulotteisia taulukoita käyttävään tietokantaan (multidimensional OLAP, MOLAP). Toiseksi loogisen tason relaatiomalli voidaan toteuttaa monella eri tavalla, koska ulottuvuuksien hierarkiat voidaan esittää eri asteisesti epänormalisoituna.

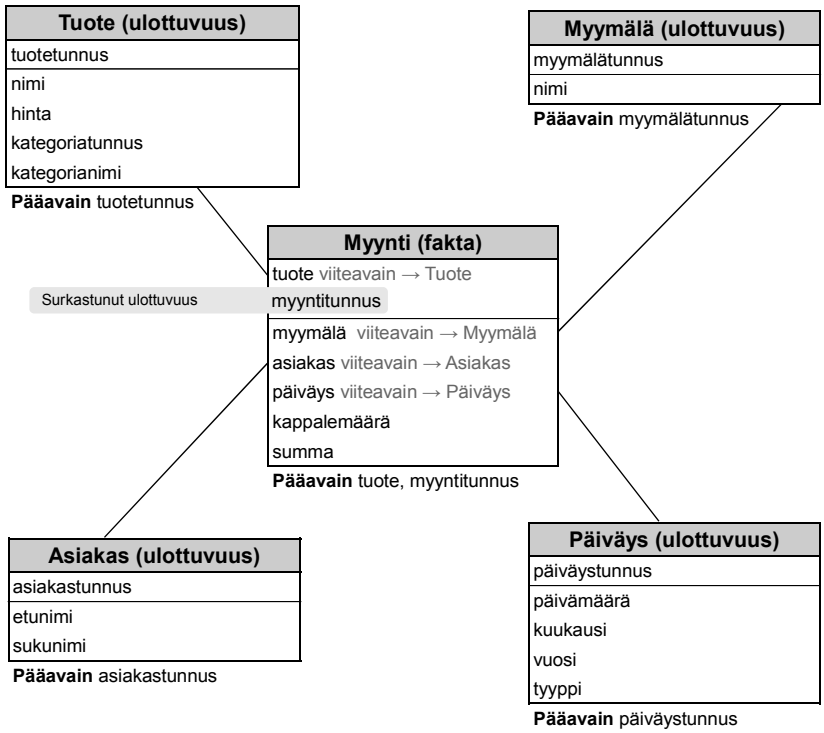
Moniulotteisia taulukoita käyttävissä *MOLAP-järjestelmissä* tieto tallennetaan rakenteisiin, joita kutsutaan *kuutioiksi*. MOLAP-järjestelmien käyttämät kuutiot voidaan toteuttaa esimerkiksi relaatiomallin avulla [VaS99] tai moniulotteisen käsitemallin pohjalta [CaT98]. Kuutioiden suunnittelussa looginen suunnittelu on kuitenkin yleensä osa fyysistä suunnittelua eikä jako käsitteelliseen, loogiseen ja fyysiseen suunnitteluun ole täysin samanlainen kuin perinteisten relaatiotietokantojen suunnittelussa [NNT00]. Raja eri suunnitteluvaiheiden välillä ei kuutioiden mallinnuksessa ole myöskään täysin selkeä. Lisäksi eri lähteet käyttävät sekaisin termejä käsitteellinen ja looginen suunnittelu [CaT98][HLV00]. Kuutioita voidaan mallintaa muodollisilla algebrallisilla menetelmillä [MVS03][VaS00], mutta käytännössä kuutioiden suunnittelu ja toteutus tehdään usein ohjelmistokohtaisilla työkaluilla, joissa esityskerroksen suunnittelulla voi olla merkittävä osa [MVS03]. Kuutiot voidaankin nähdä vain näkymänä alemman tallennuskerroksen tietoon [MVS03], joka on usein relaatiotietokanta [KRT08 s. 235, 349].

Moniulotteiselle relaatiomallille on määritelty omia normaalimuotoja kuten MNF (multidimensional normal form), GMNF (generalized multidimensional normal form) [LAW98] sekä ensimmäinen, toinen ja kolmas moniulotteinen normaalimuoto (1MNF, 2MNF ja 3MNF) [LeV03]. Moniulotteisten normaalimuotojen tarkoitus on ehkäistä koostamisessa tapahtuvia virheitä (koostamisanomalioita, aggregation anomalies) [HLV00], joita voi seurata ulottuvuuksien epänormalisoinnista ja siitä, että epänormalisoitujen ulottuvuuksien hierarkioiden esittämiseksi joudutaan joissakin relaation ilmentymissä attribuuteille käyttämään NULL-arvoja [LAW98]. Kolmas moniulotteinen normaalimuoto esimerkiksi estää kaikki ulottuvuuksien *käyttökelvottomat* NULL-arvot. Käyttökelvottomiksi arvoiksi luetaan sellaiset NULL-arvot, joissa relaation loogisen rakenteen vuoksi NULL-arvoa tarvitaan, koska relaation attribuutti on joillekin relaation ilmentymille tarpeeton. Esimerkiksi attribuutti *lajityyppi* on tarpeellinen CD-levyille, mutta tarpeeton ruohonleikkureille. Moniulotteisten normaalimuotojen käyttöön viitataan harvoin ja niiden saavuttamiseen voidaan suhtautua samoin kuin kolmanteen normaalimuotoon: jos tavoitteena todella on täysin normalisoitu tietokanta, on tietomalli on suunniteltava osaavasti ja huolellisesti käsitetasolta asti. Hyvän käsitesuunnittelun lopputuloksena on tarkoituksenmukaisen normalisointiasteen vaatimukset täyttävä tietokantakaavio [HLV00].

3.2.1 Tähtimalli

Yleinen moniulotteisen tietomallin looginen rakenne on niin sanottu *tähtimalli* (star schema, kuva 8), joka ei ole normalisoitu tietomalli [KRT08 s. 338]. Tähtimallin mukainen relaatiotietokanta koostuu yhdestä, tapahtumatyyppistä tietoa sisältävästä faktataulusta ja useasta tapahtumaa kuvaavasta ulottuvuustaulusta. Faktataulu on tähtimallin suurin taulu ja se on yleensä normaalimuodossa [KRT08 s. 236]. Ulottuvuustaulut sen sijaan eivät ole normaalimuodossa - ne ovat kooltaan pienempiä ja sisältävät yleensä toisteisuutta. Ulottuvuudet on tähtimallissa epänormalisoitu siten, että kaikki ulottuvuuteen liittyvä hierarkkisuus typistyy yhdeksi relaatioksi. Hierarkkisuus jää siis relaation sisäiseksi ominaisuudeksi, jota ei voi suoraan päätellä relaatioista ja niiden välisistä yhteyksistä. Tällöin relaatiossa voi myös esiintyä epäsuoria funktionaalisia riippuvuuksia, jolloin se rikkoo kolmannen normaalimuodon määritelmää, mutta ei toista normaalimuotoa [MoK03]. Käyttökelvottomia NULL-arvoja saatetaan myös tarvita, joten tähtimalli ei usein täytä moniulotteisen normaalimuodon (MNF) määritelmää. Ulottuvuuksien epänormalisoinnista seuraavaa toisteisuutta ei kuitenkaan yleensä pidetä ongelmana sillä ulottuvuustaulut sisältävät melko staattista tietoa ja ne ovat yleensä kooltaan kohtalaisen pieniä verrattuna koko tietokannan kokoon [KRT08 s. 267]. Kuvan 8 tähtimallia kuvaavassa esimerkissä kannattaa kiinnittää huomiota faktataulun *Myynti* attribuuttiin *kappalemäärä*, joka vaaditaan ilmoittamaan kuinka monta saman tuotteen ilmentymää liittyy yhteen myyntitapahtumaan. Attribuuttia tarvitaan, koska faktataulun ja ulottuvuustaulujen suhde on monen suhde

yhteen. Toinen vaihtoehto olisi liittää fakta- ja ulottuvuustaulut toisiinsa monesta moneen suhteen mahdollistavan *siltataulun* (bridge-table) avulla. Tilanne on esimerkki siitä, kuinka ainoastaan fakta- ja ulottuvuustauluista koostuva tähtimalli, ilman monesta moneen suhteita, rajoittaa tiedon karkeisuustasoa. Faktataulun attribuutti *summa* koostaa tässä tapauksessa myyntitapahtumaan sisältyvän *yhden* tuotteen hintojen summan, toisin kuin luvun 2.3 normalisoidussa tietomallissa, jossa attribuutti *kokonaissumma* koosti yhteen myyntitapahtuman *kaikkien* tuotteiden hintojen kokonaissumman. *Myynti*-taulun attribuutti myyntitunnus on niin sanottu *surkastunut ulottuvuus* (degenerate dimension), jonka kautta voidaan koostaa faktataulun tapahtumia, mutta siihen ei liity mitään kuvaavaa lisätietoa.



Kuva 8: Tähtimalli *Market*-esimerkkimyyntijärjestelmän tiedoista.

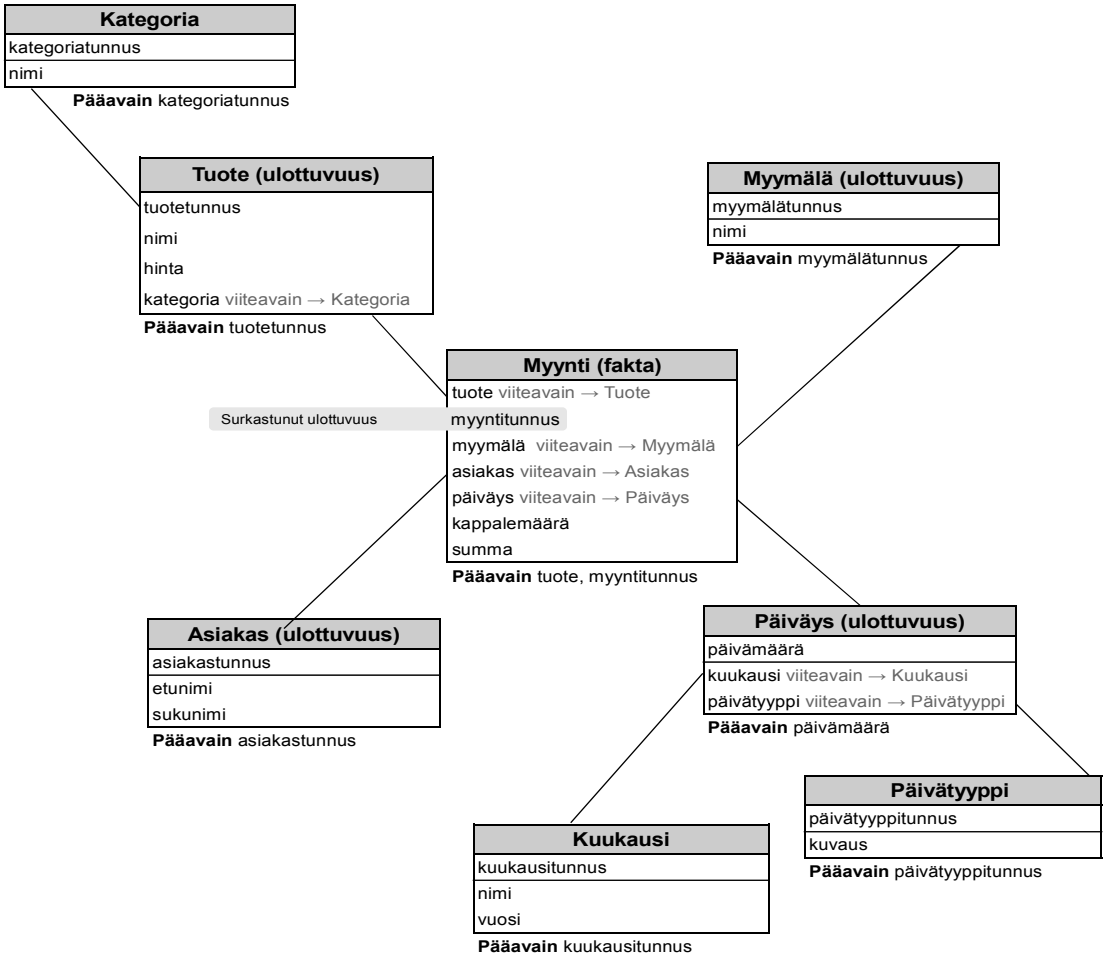
Tähtimallissa on vain kohtalaisen pieni joukko relaatioita ja niiden rooli on selkeä: *faktat* sisältävät jatkuvasti sisään vyöryviä tapahtumatietoja ja *ulottuvuudet* kuvaavat tapahtumiin liittyvää staattista tietoa. Ulottuvuudet määrittelevät yksikäsitteisesti yhden faktataulun rivin eli ne, tai osa niistä, muodostavat yhdessä faktataulun avaimen. Ulottuvuustaulujen avaimina käytetään yleensä tehokkuussyistä keinotekoisia avaimia (surrogate key) [ChD97].

Tähtimalli on kompromissi toisteisuuden välttämisen ja kyselyjen tehokkuuden välillä ja se on rakenteeltaan erittäin selkeä ja ymmärrettävä. Lisäksi monet tietovarastoissa käytettävät ohjelmistot on optimoitu erityisesti tähtimallille [KRT08 s. 237, 347]. Kimballin ja kumppaneiden mielestä tietovarasto voidaan perustaa lähes täysin tähtimallinnettuun tietoon [KRT08]. Vastakkaisten näkemysten mielestä tähtimalli soveltuu vain näkymäksi, joka on johdettu tietovaraston

normaalimuodossa olevasta tietokannasta [ISN08][Mar04]. Tähtimalli ei vastakkaisten näkemysten mielestä yksinään sovellu yhdenmukaistetun tietovaraston rakentamiseen, koska sillä ei voi kuvata tietojen välisiä monimutkaisia suhteita, se sisältää liikaa toisteista tietoa ja on näkökulmaltaan rajoittunut liiketoimintaprosessien kuvaamiseen [Inm02 s. 137-142][ISN08 s. 18-21] [Mar04].

3.2.2 Lumihiutalemalli

Loogisen tason moniulotteista tietomallia, jossa ulottuvuustaulut on normalisoitu ja ulottuvuustaulujen funktionaaliset riippuvuudet purettu osittamalla relaatiot pienemmiksi tauluiksi, kutsutaan *lumihiutalemalliksi* (snowflake schema, kuva 9). Esimerkiksi kuvassa 8 esitetyn tähtimallin *Päiväys*-taulu on jaettu kuvassa 9 tauluiksi *Päiväys, Kuukausi* ja *Päivätyyppi*. Lisäksi tuotteen kategoriatieto on erotettu omaksi taulukseen, jolloin *Tuote*-tauluun tallennetaan kategoriatiedon sisältävän merkkijonon sijasta kokonaislukuun mahtuva viite, mikä säästää tilaa ja vähentää toisteisuuden määrää tietokannassa.



Kuva 9: Lumihiutalemalli *Market*-esimerkkimyyntijärjestelmän tiedoista.

Jos ulottuvuudet on riittävästi normalisoitu lopputuloksena on kolmannen normaalimuodon vaatimukset täyttävä tietomalli [KiR02 s. 55]. Jos ulottuvuudet on normalisoitu vain osittain, voidaan mallia kutsua myös tähtihiutalemalliksi (starflake schema) [CoB02 s. 1081]. Termille lumihiutalemalli ei ole virallista määritelmää [LeL03], vaan puhutaan tähtimallin eri asteisesta *lumihiutaloittamisesta* (snowflaking) [KiR02 s. 55]. Lumihiutalemallille on kuitenkin määritelty erilliset normaalimuodot nimeltä lumihiutalenormaalimuoto (Snowflake Schema Normal Form, SSNF) ja laajennettu lumihiutalenormaalimuoto (Extended Snowflake Schema Normal Form, SSNF) [LeL03].

Lumihiutalemallin heikkoutena tähtimalliin verrattuna on vaikeampi ymmärrettävyys ja heikompi kyselyjen vasteaika. Lumihiutalemallissa suurempi taulujen määrä johtaa suurempaan määrään liitoksia kyselyn suorituksen aikana, mikä taas heikentää vasteaikoja kyselyissä. Lisäksi lumihiutalemalli estää bittikarttaindeksien käytön ja vaikeuttaa porautumisoperaatioita, joilla syvennetään tarkkuustasoa tiedon analysoinnissa [KiR02 s. 55-57]. Lumihiutalemallin vahvuudet tähtimalliin verrattuna ovat tilan säästö sekä helpompi ylläpidettävyys tietyissä tilanteissa. Esimerkiksi kategorian nimen vaihtuessa päivitys tarvitsee kuvan 9 esimerkissä tehdä vain yhdelle riville (jos päätetään muuttaa kategorian nimeä myös vanhoissa tapahtumissa). Joidenkin näkemyksien mukaan lumihiutalemallin vahvuudet eivät kuitenkaan riitä kompensoimaan sen heikkouksia suhteessa tähtimalliin, joten sitä tulisi käyttää vain perustelluista syistä [KiR02 s. 55-57]. Perusteltu syy voi esimerkiksi olla tilanne, jossa ulottuvuus sisältää tietoa, jota käsitellään vain harvoin. Silloin harvemmin käsitellyt tiedot kannattaa ehkä lumihiutaloittaa eli osittaa omaksi tauluksi.

3.2.3 Muita malleja

Harvinaisempia moniulotteisia loogisen tason tietomalleja edustavat laakamalli, porrasmalli ja tähtirypäsmalli [MoK00][RoA09]. *Laakamallissa* (flat schema) ei ole lainkaan ulottuvuustauluja vaan kaikki yksilöt on epänormalisoitu hierarkian alimman tason minimaalisiksi yksilöiksi. Minimaaliset yksilöt ovat niitä ja vain niitä faktoja, jotka eivät sisällä lainkaan yhdestä moneen yhteyksiä. Malli on äärimmäisen epänormalisoitu ja minimoi relaatioiden määrän hävittämättä kuitenkaan tietoa. Laakamalli sisältää runsaasti toisteisuutta ja funktionaalisia riippuvuuksia ja relaation attribuuttien määrä voi olla valtava. Relaatioiden vähäisestä määrästä seuraa, että laakamallista tehdyssä kyselyssä tarvitsee tehdä vain vähän liitoksia, mutta koosteoperaatioissa voi helposti tapahtua virheitä. *Porrasmallissa* (terraced schema) kaikki ulottuvuudet on epänormalisoitu faktoiksi, eli tapahtumatyyppistä tietoa sisältäväksi relaatioiksi. Se on siis vähemmän epänormalisoitu kuin laakamalli, mutta enemmän kuin tähtimalli. *Tähtirypäsmalli* (star cluster schema) on tähtihiutalemallin erikoistapaus, jossa ulottuvuusrelaatiot on epänormalisoitu vain

sille tasolle, jossa ne eivät liity useampaan ylemmän tason relaatioon. Toisin sanoen niitä relaatioita ei epänormalisoida, joilla on hierarkiassa useampi kuin yksi isä.

Moniulotteinen tietomalli voi sisältää myös useampia kuin yhden faktataulun, jolloin niistä voidaan käyttää nimiä *tähtikuvio* (constellation schema) tai *galaksi* [MoK00]. Tähtikuviossa faktarelaatiot ovat yhteydessä toisiinsa vain suoraan, galaksissa faktarelaatiot voivat liittyä toisiinsa myös yhteisten ulottuvuuksien kautta.

Tässä luvussa kuvatuilla loogisen tason malleilla esiintyy kirjallisuudessa paljon nimeämisyhteentörmäyksiä. Esimerkiksi [RoA09][MYB08] ja monet muut lukevat tässä luvussa esitellyt mallit loogisiksi, [DFS05] ja [NNH08] viittaavat niihin käsitemalleina ja [ISN08 s. 290] pitää tähtimallia ja lumihiutalemallia fyysisen tason malleina. Tässä luvussa esitellyistä malleista on kuitenkin selkeintä puhua loogisen tason malleina, sillä ne kuvaavat tiedon relaatioiksi mallinnettuna eivätkä siis ole toteutuksesta riippumattomia käsitemalleja. Niissä ei myöskään kuvata tiedon fyysisiä tallennusominaisuuksia, kuten hakemistoja tai tiedostorakenteita, joten ne eivät ole myöskään osa fyysistä suunnittelua.

3.3 Moniulotteisen tiedon fyysinen suunnittelu

Moniulotteinen tieto voidaan tallentaa *relaatiotietokantaan* (relational OLAP, ROLAP) tai *moniulotteisia taulukoita* käyttävään tietokannan hallintajärjestelmään (multidimensional OLAP, MOLAP). Niin sanotut HOLAP-palvelimet (hybrid OLAP) tallentavat osan tiedosta relaatiotietokantaan ja osan moniulotteiseen taulukkorakenteeseen. Tiedon jakaminen eri tallennusrakenteisiin voidaan HOLAP-palvelimissa tehdä eri perusteilla. Esimerkiksi tarkan tason tieto voidaan tallentaa relaatiorakenteseen ja koostettu tieto taulukkorakenteeseen. Valinta voidaan tehdä myös tiedon iän perusteella.

MOLAP-palvelinten tallennusrakenne tukee suoraan moniulotteisuutta, minkä ansiosta sillä on erinomaiset automaattiset indeksointiominaisuudet ja kyselyjen vasteajat [CDN11][KRT08 s. 149]. Moniulotteisen kuution suunnittelussa tärkeitä tavoitteita ovat kuution *tiiviys* (sparsity), *täydellisyys* (completeness) ja *summautuvuus* (summarizibility) [NNT00]. Tiiviys tarkoittaa, että kuution soluista mahdollisimman pieni osa on tyhjiä, täydellisyys, että kaikki operatiivisesta järjestelmästä saatu tieto voidaan esittää kuutiossa ja summautuvuus sitä, että koostaminen tapahtuu oikein.

Moniulotteista tietoa sisältävän relaatiotietokannan fyysisessä suunnittelussa käytetään samoja tekniikoita kuin normalisoitua tietoa sisältävän relaatiotietokannan suunnittelussa. Suorituskyvyn parantamiseksi käytetyt tekniikat vaativat kuitenkin moniulotteisen tietomallin huomioon ottamista erityisesti hakemistojen suunnittelussa [KRT08 s. 344-349]. Faktatauluille, joihin val-

taosa moniulotteisen tietokannan tiedosta on tallennettu, luodaan päähakemistoksi yleensä ryvästävä B-puu-hakemisto. Faktataulun päähakemiston avain perustuu yleensä useampaan attribuuttiin, jolloin kyseessä on niin sanottu lihava tai puolilihava hakemisto. Faktataulun avain muodostuu yleensä ulottuvuuksien viiteavaimista. Tietokannan hallintajärjestelmän ominaisuuksista ja käytetyimmistä kyselyistä riippuu millaisia oheishakemistoja faktatauluille kannattaa luoda. Oheishakemistoja luodaan yleensä kyselyissä eniten käytetyille attribuuteille. Jos käytetään lihavia tai puolilihavia hakemistoja, on attribuuttien järjestyksellä hakemistoa määriteltäessä merkitystä. Ulottuvuustauluissa kannattaa yleensä käyttää yhden sarakkeen keinoavainta päähakemistona ja luoda eniten käytetyille attribuuteille oheishakemistoiksi bittikarttahakemistoja. Bittikarttahakemistot ovat erityisen tehokkaita ulottuvuustaulujen indeksoinnissa, kun niitä luodaan attribuuteille joiden arvoalue on pieni, esimerkiksi asiakkaan sukupuoli, aviosääty tai jokin muu kaksiarvoinen attribuutti. Erityisen pienille ulottuvuustauluille ei välttämättä tarvitse luoda lainkaan oheishakemistoja.

Monet tietokannan hallintajärjestelmät osaavat optimoida tähtimallin mukaiseen tietoon tehtyjä kyselyitä muodostamalla ensin välitaulun ulottuvuustaulujen välisestä karteesisesta tulosta. Sen jälkeen tulosrivit saadaan tekemällä yksi liitos välitaulun ja faktataulun välillä, mikä voi olla kymmeniä kertoja nopeampaa kuin tehdä liitos faktataulun tietoihin jokaisen ulottuvuustaulun kanssa erikseen [KRT08 s. 347]. Pieniin ulottuvuustauluihin kohdistuvat ehdot voivat myös rajata lopputulokseen kuuluvia faktataulun rivejä erittäin tehokkaasti.

3.4 Moniulotteisen tietomallin erityispiirteitä

Moniulotteisen tiedon tuottamiseen operatiivisista järjestelmistä on esitetty monia eri menetelmiä, joihin liittyy usein oma mallinnustapa ja ohjeita moniulotteisen tietokannan vaiheittaiseen rakentamiseen operatiivisen järjestelmän tiedoista [RoA09]. Moniulotteisen tietomallin rakentaminen voidaan aloittaa mallinnettavan liiketoimintaprosessin ja halutun tarkkuustason valitsemisesta [KRT08 s. 246]. Tämän jälkeen pyritään tunnistamaan faktat ja ulottuvuudet, joista moniulotteinen tietomalli muodostuu. Faktojen tunnistaminen on yleensä melko suoraviivaista ja toimenpide voidaan jopa automatisoida osittain tai kokonaan. Automatisointi voidaan tehdä esimerkiksi muuntamalla käyttäjävaatimukset ensin SQL-lauseiksi [RoA06] tai muuntamalla operatiivisen järjestelmän tiedot XML-kaavioksi [VBR00]. Faktojen tunnistamisessa voidaan käyttää hyväksi myös operatiivisen järjestelmän käsitemallia, loogista mallia tai tutkimalla operatiivisen tietokannan osallistumisrajoitteita, taulujen päivitystiheyttä ja numeerista tietoa sisältäviä attribuutteja. Ongelmakentän syvällinen tuntemus ja erilaiset luovat keinot auttavat ulottuvuuksien ja faktojen tunnistamisessa. Oleellista on selvittää minkälaisia raportteja tietovaraston käyttäjät kaipaavat, mitä tietoja tarvitaan ja miten tiedot pitää käyttäjälle näyttää.

Yksityiskohtaista tapahtumatietoa sisältävä faktataulu sisältää yhden rivin jokaista tapahtumaa kohti. Yksityiskohtaisen faktataulun lisäksi samoja tietoja ylläpidetään yleensä myös karkeamman tason faktatauluissa, jotka sisältävät tapahtumista valmiiksi koostettua tietoa, niin sanottuja tilannekuvia (snapshot) [KRT08 s. 273-276]. Karkeamman tason faktataulu voi olla s*äännöllisin väliajoin koostettava tilannekuva* tai *kumuloituva tilannekuva*. Esimerkiksi pankkitilien saldoista on luontevaa tehdä erillinen faktataulu, jossa jokaisen tilitapahtuman sijasta esitetään tilannekuva kuukausittaisesta saldosta. Kumuloituvaa tilannekuvaa voitaisiin käyttää esimerkiksi tilausten eri vaiheiden seurantaan tilauspäivästä toimitukseen ja laskutuspäivään. Kumuloituva tilannekuva poikkeaa muista faktataulun tyypeistä siten, että faktataulun tietoa päivitetään. Säännöllisin väliajoin koostettavaa tilannekuvaa tai yksityiskohtaisen tapahtumatiedon faktataulua sen sijaan ei yleensä päivitetä. Tilannekuvan sisältämät faktataulut vaativat aina rinnalleen myös yksityiskohtaiset tapahtumatiedot sisältävän faktataulun jotta tietoa voidaan tarvittaessa tarkastella myös tarkemmalla tasolla.

Faktattomat faktat (factless facts) [KRT08 s. 280] ovat myös eräs faktataulujen erikoistapaus. Faktattomilla faktoilla kuvataan liiketoimintaprosesseja, joihin liittyy tapahtumia mutta ei lainkaan mittoja. Faktattomia faktatapahtumia halutaan kuitenkin seurata ulottuvuuksien kautta. Esimerkiksi työntekijöiden poissaoloja voitaisiin seurata faktattomalla faktataululla, josta voidaan tutkia työtekijöiden poissaoloja työntekijä-, päivämäärä- tai osastoulottuvuuksien kautta. Tällöin faktatauluun luodaan uusi rivi aina uuden poissaolotapahtuman yhteydessä. Riviin ei liity mitään mittatietoa, mutta siitä voidaan ulottuvuuksien avulla koostaa esimerkiksi yhden työntekijän poissaolot vuoden aikana.

Faktataulun tapahtumatietoon liittyy aina päiväys/aika [KRT08 s. 253-256], joka tallennetaan omaan ulottuvuustauluun. Päiväysulottuvuus sisältää yleensä paljon enemmän tietoa kuin pelkän päivämäärän, esimerkiksi viikonpäivän, kuukauden nimen, laskutuskauden, arkipäivä/vapaapäivä-tiedon ja niin edelleen. Vaikka päiväystaulun luonnollisena avaimena olisi mahdollista käyttää päivämäärämerkintää, on päivämäärällekin yleensä syytä luoda keinoavain muiden ulottuvuustaulujen tapaan. Keinoavain säästää tilaa faktataulussa, se tekee kyselyistä tehokkaampia ja päiväykseen liittyvien erikoistietojen käsittely on helpompaa. Päivämäärää tarkemman tarkkuustason aikaleimat tallennetaan yleensä faktatauluun, koska ne tekisivät ulottuvuustaulusta liian suurikokoisen.

Faktatauluun generoidaan jatkuvasti uusia rivejä, mutta ulottuvuustaulut pysyvät yleensä hyvin samanlaisina. Kuitenkin myös jotkin ulottuvuustaulujen attribuutit muuttuvat ajan kuluessa. Esimerkiksi operatiivisessa järjestelmässä tuotteen kuvaus voi muuttua. Miten tilanne pitäisi käsitellä tietovaraston ulottuvuustaulussa, jossa on jo tuotetietoja vanhalla kuvauksella? Tilanteen käsittelemiseen on kolme perustapaa: ylikirjoittaminen, uuden rivin lisäys ja uuden

sarakkeen lisäys. Lisäksi näistä on käytössä lukuisia erilaisia yhdistelmiä ja muunnelmia [KRT08 s. 257-262].

Attribuutin arvon ylikirjoittaminen on suoraviivaisin tapa hallita ulottuvuuksissa tapahtuvia hitaita muutoksia (slowly changing dimensions). Ylikirjoittamista voidaan käyttää, jos vanhoilla arvoilla ei ole enää liiketoiminnallista merkitystä. Ylikirjoittamista käytettäessä tietovaraston aikasidonnaisuus kuitenkin menetetään ja attribuutin pohjalta aikaisemmin lasketut koosteet eivät välttämättä ole yhteensopivia muutoksen jälkeisten tietojen kanssa.

Uuden rivin lisäys ulottuvuustauluun on yleisin tapa hallita ulottuvuuksissa tapahtuvia muutoksia [KRT08 s. 258]. Siinä ulottuvuusattribuutin arvossa tapahtuva muutos aiheuttaa uuden rivin lisäyksen ulottuvuustauluun, vanha rivi merkitään inaktiiviseksi ja uusi rivi aktiiviseksi. Jokaisen rivin aktiivisuusaika merkitään alkamis- ja päättymisaikaleimoin. Uuden rivin lisäys säilyttää tietovaraston tiedot historiallisesti oikein, mutta raportoinnissa on otettava huomioon, että vanhan rivin keinoavain on korvattu uudella, jolloin niihin perustuvat liitokset eivät välttämättä anna kaikissa tapauksissa haluttuja lopputuloksia ellei kyselyä muuteta.

Kolmas tapa hallita ulottuvuustauluissa tapahtuvia muutoksia on lisätä tauluun uusi attribuutti muutoksen tapahtuessa. Esimerkiksi *Tuote*-tauluun voitaisiin lisätä attribuutti a*lkuperäinen kategorianimi*, johon kopioidaan kaikki vanhat *kategorianimi*-attribuutin arvot kun jokin *kategorianimen* arvo vaihtuu. Uusia attribuutteja ei voi tauluun kuitenkaan lisätä loputtomasti, joten jossain vaiheessa vanhoja arvoja on pudotettava pois. Uuden attribuutin lisäys ulottuvuustauluun historioi siis tietovaraston aikasidonnaiset tiedot vain osittain.

Moniulotteisen tietomallin suunnittelussa osa tapahtumiin liittyvistä ominaisuuksista ei joskus kuulu luonnollisesti mihinkään ulottuvuuteen. Ne voivat olla esimerkiksi kaksiarvoisia lippuattribuutteja tai tekstikenttiä. Sekalaiset kuvaustiedot, joille ei löydy sopivaa paikkaa mistään ulottuvuustaulusta, voidaan kasata yhteen ja muodostaa niistä yksi *ylijäämäulottuvuus* (junk dimension) [KRT08 s. 263-265]. Sekalaisten tietojen poistaminen kokonaan, jättäminen faktatauluun tai jokaiselle attribuutille oman ulottuvuuden lisääminen ovat huonompia vaihtoehtoja [KRT08 s. 263].

4 Tietoholvimalli

Tietovaraston tietomalliksi on ehdotettu myös *tietoholvimallia* (data vault), joka tallentaa lähdejärjestelmien tiedot yksityiskohtaisella tasolla, aikasidonnaisesti, normaalimuodossa sekä pyrkii tukemaan eri näkökulmia tietovaraston tietoihin [Lin02]. Tietoholvimalli on suunniteltu erityisesti tietovarastointia varten, toisin kuin perinteinen normalisoitu tietomalli tai tähtimalli, joita on jälkikäteen sovellettu tietovarastointiin. Tietoholvimalli pyrkii ratkaisemaan ongelmia joita muiden tietomallien käyttö tietovarastoinnin yhteydessä aiheuttaa. Erityisesti kuvaustiedoissa tapahtuvat muutokset, tietomallin laajentaminen uusilla tiedoilla, tiedon aikasidonnaisuus sekä lähdejärjestelmistä tehtävien latausten rinnakkaistaminen on tietoholvimallissa pyritty ratkaisemaan mahdollisimman tehokkaasti ja täydellisesti. Tietomallin avulla on mahdollista tehdä useimmat rakenteelliset muutokset ainoastaan uusien relaatioiden lisäyksillä [JoB12].

Tietoholvimallissa kaikki lähdejärjestelmistä saatu tieto tallennetaan ja tiedon tallennusaika sekä lähdejärjestelmä merkitään. Muutoshistorian tallentamisesta voi olla hyötyä tiedon laadun tarkastamisessa tai palauttamisessa, jos esimerkiksi latausprosessissa tapahtuu virhe [Lin11a s. 25]. Lähdejärjestelmistä saadut tiedot jaetaan loogisella tasolla kolmeksi eri relaatiotyypiksi: keskiöiksi, linkeiksi ja satelliiteiksi [Lin11a]. *Keskiöt* (hub) kuvaavat liiketoiminta-avaimia, kuten asiakastunnuksia, tuotetunnuksia tai sopimustunnuksia. Keskiöiden lähin vastine ER-mallissa on yksilöiden avainattribuutit. *Linkit* (link) kuvaavat yhteyksiä liiketoiminta-avainten välillä. Kaikki yhteydet ovat monesta moneen yhteyksiä, toisin kuin ER-mallissa, jossa osallistumisrajoitteilla voidaan rajoittaa mallin joustavuutta ajan myötä tapahtuviin muutoksiin. Keskiöiden ja linkkien ominaisuuksia kuvataan *satelliiteilla* (satellite), joiden lähin vastine ER-mallissa on yksilöiden ja yhteyksien ominaisuuksia kuvaavat attribuutit.

Tietoholvimallin esitteli ensimmäisen kerran D. Linstedt vuonna 2000 [Lin02]. Malli perustuu hänen omiin vuosien mittaisiin käytännön kokemuksiinsa tietovarastoinnista sekä moniulotteisen tietomallin ja normalisoidun tietomallin rajoituksista. Tietoholvimalli on eräänlainen yhdistelmä normalisoidusta ja moniulotteisesta tietomallista. Monesta moneen suhteita luovat linkkitaulut perustuvat normaalimuotoon. Ajatus staattista tietoa tallentavista keskiöistä on peräisin ulottuvuuksien hallintatavasta, jossa kaikki ulottuvuuksissa tapahtuvat muutokset ylikirjoitetaan. Muuttuvaa tietoa sisältävät satelliitit taas on kehitetty tavasta hallita ulottuvuuksissa tapahtuvia muutoksia lisäämällä ulottuvuustauluun uusi rivi [Lin11b].

Tietoholvimallista ei löydy muutamaa mainintaa lukuun ottamatta juuri lainkaan viitteitä tai tutkimustuloksia tieteellisissä julkaisuissa. Sitä kuitenkin käytetään useissa organisaatioissa [Lin11c]. Linstedt itse on julkaissut kollegoidensa avustuksella aiheesta muutamia kirjoja

[Lin10d][Lin11a] ja hän esittelee ja mainostaa tietoholvimallia aktiivisesti Internetin eri sivustoilla sekä järjestää siihen liittyviä maksullisia koulutuksia. Pyrimme esittelemään ja arvioimaan tietoholvimallia saatavilla olevien lähteiden perusteella.

4.1 Keskiöt

Tietoholvimallissa *keskiöiksi* kutsutaan relaatioita, joihin tallennetaan niin sanottuja *liiketoiminta-avaimia* [Lin11a s. 33-47]. Liiketoiminta-avaimet ovat tietoa, jolla yritys yksilöi ja paikallistaa liiketoiminnan kohteita, kuten asiakkaita, tuotteita ja laskuja. Liiketoiminta kohdistuu liiketoiminta-avaimiin ja liiketoimintaprosessit käyttävät avaimia oman toiminnan ja prosessien välisen yhteistyön tukena. Liiketoiminta-avain on yleensä liiketoiminnan kohteen *luonnollinen avain*, esimerkiksi asiakastunnus tai laskun numero, ja sen tulisi olla yksilöllinen. Liiketoiminta-avain voi olla *keinoavain* ainoastaan tilanteessa, jossa toimihenkilöt käyttävät sitä omassa toiminnassaan. Esimerkiksi laskun numero voi olla järjestelmän luoma keinoavain, mutta asiakastunnukseksi sopii paremmin henkilötunnus.

Liiketoiminnan säännöllisesti kommunikoinnissaan käyttämät käsitteet ja työvälineet voivat antaa vihjeitä siitä, mitkä tunnisteet toimivat liiketoiminta-avaimina [Lin11a s. 36-37]. Liiketoiminta-avaimia käytetään liiketoimintaprosessissa tiedon välittämiseen sidosryhmien välillä ja avaimia voi löytää operatiivisten järjestelmien käyttöliittymistä, raporteista, tietomallikaavioista, XML-skeemoista ja tietokannan tiedoista. Liiketoiminta-avainten tulisi olla pysyvää tietoa, ne eivät saisi muuttua eikä niitä saisi käyttää uudelleen. Esimerkiksi asiakastunnus, joka on liiketoiminta-avain, on pysyvä, kun taas asiakkaan nimi ja osoite voivat muuttua. Käytännössä liiketoiminta-avaimet kuitenkin usein muuttuvat ja niitä käytetään uudelleen, erityisesti yrityksen eri osastojen välillä [Lin11a s. 37]. Avainten muuttuminen ja uudelleenkäyttö on aina epätoivottava tilanne ja voi aiheuttaa ajan kuluessa yritykselle huomattavaa ylimääräistä työtä ja lisäkustannuksia. Tietoholvimallin ajatus on, että liiketoiminta-avainten pitäisi olla muuttumattomia ja jos näin ei ole, yrityksen kannattaa muuttaa toimintatapoja, jotta avaimet pysyisivät muuttumattomina.

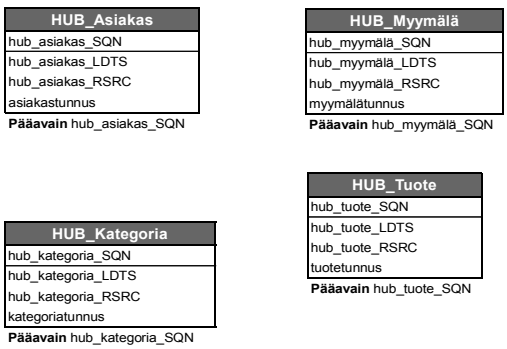
Liiketoiminta-avain voi olla moniosainen (smart key, intelligent key) [Lin11a s. 39-40]. Esimerkiksi tuotteen viivakoodin sisältämässä numerojaksossa voi olla ilmaistuna valmistusmaa, tehdas ja tuoteryhmä. Moniosainen liiketoiminta-avain sisältää siis muutakin tietoa, kuin pelkän yksilöllisen tunnisteen. Moniosaiset liiketoiminta-avaimet tallennetaan aina yhteen keskiöön. Kysely- tai tallennusteknisistä syistä avain voidaan pilkkoa keskiötaulussa eri attribuuteiksi tai avaimen sisältämät tiedot voidaan kopioida omiin keskiöihinsä. Pilkkomista ja kopioimista tulee kuitenkin tehdä erittäin harkiten, sillä se saattaa helposti johtaa vaikeasti hallittavaan tai rikkinäiseen tietoon [Lin11a s. 46]. Liiketoiminta-avaimet, joiden olemassaolo riippuu muista

avaimista, voidaan tallentaa samaan keskiöön, omaan keskiöönsä tai linkkitauluun [Lin11a s. 42-43]. Esimerkki muista liiketoiminta-avaimista riippuvasta avaimesta voisi olla laskun rivinumero. Jos avain mallinnetaan linkkitauluun, siitä käytetään nimitys surkastunut kenttä (degenerate field, ks. luku 4.2).

Keskiöt ovat itsenäisiä tietoyksiköitä eli ne ovat aina isätauluja (parent) muille tauluille eivätkä saa sisältää viiteavaimia [Lin11a s. 40]. Keskiö on kooltaan pieni, jotta mahdollisimman monta keskiötä voidaan lukea kerralla levylohkosta [Lin11a s. 46-47]. Liiketoiminta-avaimeen liittyvät ominaisuudet tallennetaan satelliittitauluun (ks. Luku 4.3) ja keskiöön tulisi aina liittyä vähintään yksi satelliittitaulu. Viiteavaimet osoittavat aina muuttuvia tietoja sisältävästä satelliittitaulusta keskiöön, jolloin keskiötauluihin ei tarvitse liiketoiminta-avainten ominaisuuksien muuttuessa tehdä mitään päivityksiä.

Keskiön pakolliset attribuutit ovat järjestelmän luoma *liiketoiminta-avain, latauspäivä* ja *lähdejärjestelmä* [Lin11a s. 35]. Mahdollisia lisäattribuutteja ovat järjestelmän luoma *keinoavain*, *salausavain* ja *aktiivisuuspäivä*. Keinovain voidaan luoda suorituskykysyistä tai jos liiketoiminta-avaimet eivät ole yksilöllisiä. Salausavain tarvitaan, jos liiketoiminta-avain on salatussa muodossa. Aktiivisuuspäivä kuvaa hetkeä, jolloin liiketoiminta-avain esiintyi viimeksi lähdejärjestelmistä tehdyissä latauksissa. Jos liiketoimintasäännöt sallivat, voidaan tietovarastoon tehtyjä kyselyitä nopeuttaa merkitsemällä poistetuiksi tai vanhentuneiksi ne liiketoiminta-avaimet, joissa aktiivisuuspäivä on sovittua vanhempi. Silloin niitä ei tarvitse ottaa huomioon kyselyissä [Lin11a s. 30-31]. Aktivisuuspäivän tarkastaminen ja ylläpito luonnollisesti hidastaa latauksia, eivätkä liiketoimintasäännöt välttämättä salli poistetuksi merkitsemistä, sillä raporteilla halutaan usein tarkastella nimenomaan historiallista tietoa.

Kuvassa 10 on esitetty *Market*-esimerkkijärjestelmän liiketoiminta-avaimet keskiöiksi mallinnettuina. Tietoholvimallin spesifikaatioon kuuluu myös joukko nimeämiskäytäntöjä, joita esimerkissä on pyritty noudattamaan. Keskiöt on nimetty etuliitteellä *HUB_,* keinoavaimet jälkiliitteellä *_SQN*, latauspäivä jälkiliitteellä *_LDTS* ja lähdejärjestelmä jälkiliitteellä *_RSRC*.



Kuva 10: *Market*-esimerkkimyyntijärjestelmän tietoholvimallinnetut keskiöt. Etu- ja jälkiliitteet: **HUB_** = Keskiö, **_SQN** = keinovain, **_LDTS** = latauspäivä, **_RSRC** = lähdejärjestelmä.

Keskiöiden mallintaminen ja liiketoiminta-avainten selvittäminen on periaatteessa melko yksinkertainen tehtävä, mutta vaatii yhteistyötä toimihenkilöiden kanssa [Lin11a s. 45]. Linstedt ehdottaa keskiöiden mallintamiseen viisivaiheista menetelmää :

1. Selvitä liiketoiminta-avaimet.
2. Tarkista liiketoiminta-avainten oikeellisuus.
3. Tarkista liiketoiminta-avainten yhteensopivuus eri järjestelmissä.
4. Mallinna keskiöt.
5. Tarkista keskiöiden oikeellisuus kaikkien lähdejärjestelmien suhteen.

Keskiöt ovat tietoholvin perusta ja niiden sisältämien liiketoiminta-avainten tulisi olla yksilöllisiä ja yhdenmukaisia. On kuitenkin harvinaista, että kaikki liiketoiminta-avaimet olisivat suuressa yrityksessä yhdenmukaisia [Lin11a s. 37]. Laajaa liiketoimintaa suorittava yritys voi esimerkiksi käsitellä asiakastietoja kymmenillä eri järjestelmillä, jotka voivat olla vuosia vanhoja tai vasta perustettuja. Ilman yritystasoista suunnittelua ja yksikköjen välistä *yhteisen tiedon* (master data) hallintaa asiakastunnukset eivät ole yhdenmukaisia. Yksi järjestelmä voi esimerkiksi käyttää asiakastunnuksena henkilötunnusta, toinen sähköpostiosoitetta, kolmas keinoavainta ja niin edelleen. Liiketoimintaprosessit eivät silloin ole yhteensopivia eikä yritysjohto voi tietää vastausta niinkään yksinkertaiseen kysymykseen kuin miten paljon yrityksellä täsmälleen on asiakkaita.

Liiketoiminta-avainten mallintaminen tietoholvin keskiöiksi ja niistä tapahtuva tiedonlouhinta tuo usein esiin eri syistä johtuvia avainten sisältämiä virheitä sekä avainten yhtenäistämistarpeita [Lin11a s. 43-44]. Keskiöt voivat toimia yhtenäistämisprosessin aloituspisteenä, kun tiedon laadun parantamiseen liittyvät toimenpiteet aloitetaan. Näillä toimenpiteillä voidaan parhaassa tapauksessa tehostaa merkittävästi yhtiön toimintaa ja saada huomattavia kustannussäästöjä [Lin11a s. 37, 44].

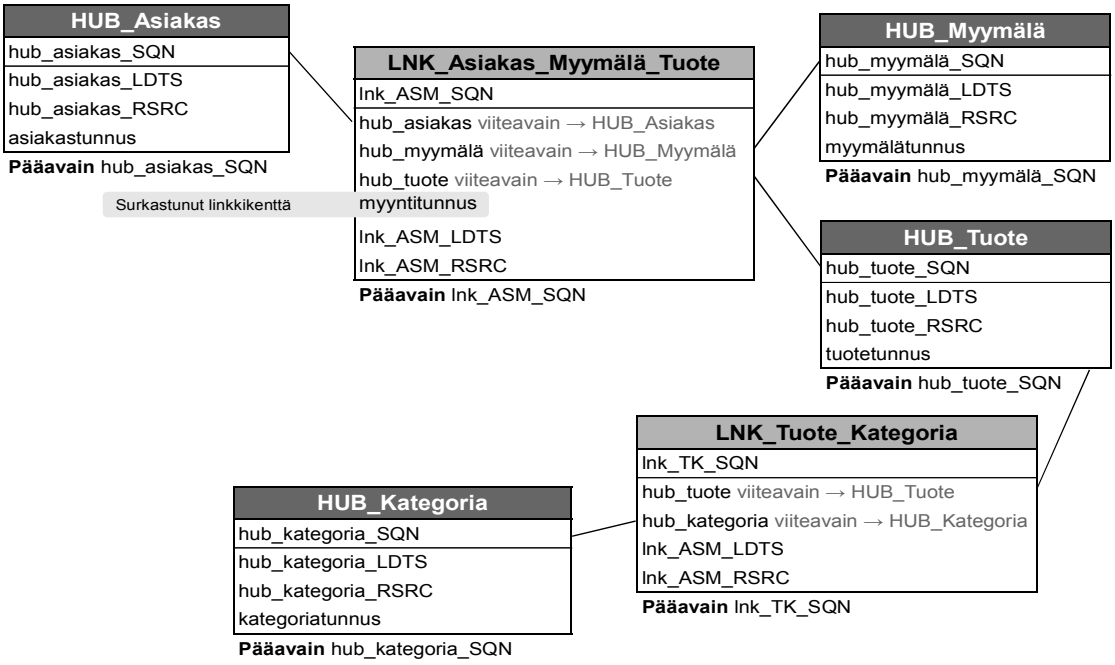
4.2 Linkit

Tietoholvimallissa yhteydet liiketoiminta-avaimia sisältävien keskiöiden välillä on toteutettu joustavasti *linkkien* avulla [Lin11a s. 48-74]. Kaikki tietomallissa tapahtuvat muutokset pyritään toteuttamaan ainoastaan lisäyksinä, mikä on yleensä mahdollista monesta moneen yhteyksiä luovien linkkitaulujen avulla.

Linkin pakollisiin attribuutteihin kuuluvat *keinovain*, *vähintään kaksi viiteavainta*, jotka viittaavat keskiöihin tai toisiin linkkeihin, *latauspäivä* ja *lähdejärjestelmä* [Lin11a s. 53-54]. Tietoholvimallin spesifikaatiossa keinoavain ilmoitetaan vapaaehtoiseksi [Lin10a]. Pakollisten

kenttien lisäksi linkki voi sisältää erilaisia valinnaisia attribuutteja kuten aktiivisuuspäivä, salausavain, luotettavuusarvo, voimakkuusarvo ja mahdollisia muita metatietoattribuutteja. *Luotettavuusarvoa* (confidence rating) ja *voimakkuusarvoa* (strength rating) voidaan käyttää tiedon louhinnassa riippuvuuksien ja toistuvien mallien etsimiseen [Lin11a s. 72-23]. Jos esimerkiksi kahdesta eri järjestelmästä ladatut liiketoiminta-avaimet, jotka on mallinnettu omiksi keskiöikseen kuvaavat täsmälleen samoilla ominaisuuksilla varustettua tuotetta, voidaan suurella todennäköisyydellä olettaa, että kyseessä on sama tuote ja muodostaa liiketoiminta-avainten välille linkki, jonka voimakkuusarvo on kohtalaisen suuri. Tiedonlouhintaa voi myös tehostaa se, että tietomallia voidaan helposti muokata, jolloin sillä on mahdollista heijastaa dynaamisesti tietoyksiköiden välillä esiintyviä muuttuvia suhteita ja riippuvuuksia [Lin11a s. 54-55]. Linkki voi myös sisältää liiketoiminta-avaimia, joiden olemassaolo riippuu täysin linkistä, jolloin siitä ei kannata tehdä erillistä keskiötä [Lin11a s. 60]. Tällaista linkin sisältämää liiketoiminta-avainta kutsutaan *surkastuneeksi linkkikentäksi* ja sitä vastaava käsite on tähtimallissa surkastunut ulottuvuus

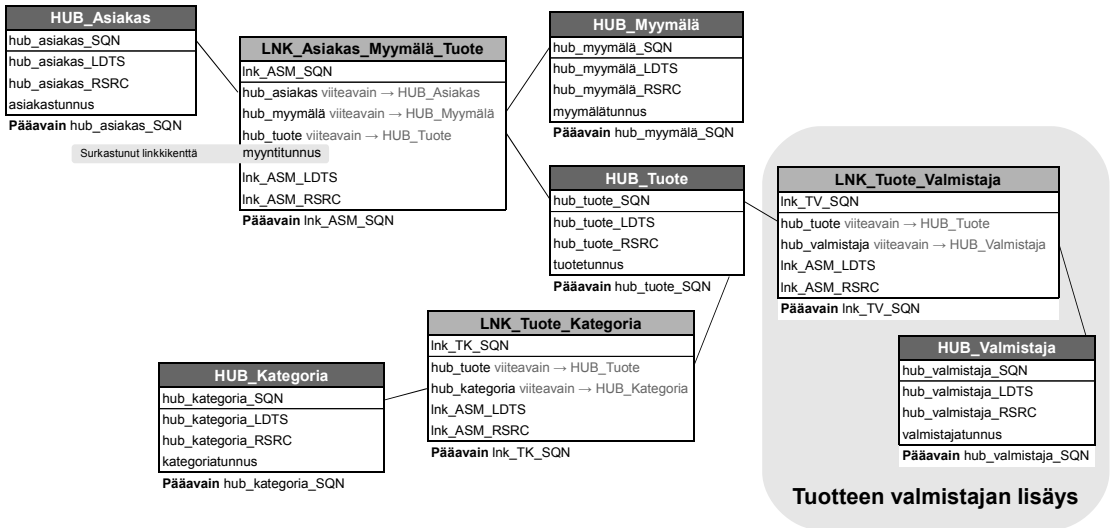
Kuvassa 11 on esitetty *Market*-esimerkkimyyntijärjestelmän tietoholvimallinnetut keskiöt, jotka on yhdistetty toisiinsa kahdella linkillä. Myyntitunnus on surkastunut linkkikenttä, jota ei ole mallinnettu liiketoiminta-avaimena, koska sen olemassaolo riippuu linkistä ja siihen liittyvä kuvaava tieto voidaan liittää satelliitilla (ks. luku 4.3) itse linkkiin.



Kuva 11: *Market*-esimerkkimyyntijärjestelmän tietoholvimallinnetut keskiöt ja linkit. Etu- ja jälkiliitteet: **LNK_** = Linkki, **HUB_** = Keskiö, **_SQN** = keinovain, **_LDTS** = latauspäivä, **_RSRC** = lähdejärjestelmä.

Linkkitaulu luo aina monesta moneen yhteyden keskiöiden liiketoiminta-avainten välille, joten osallistumisrajoitteet eivät rajoita liiketoimintasääntöjä, jotka voivat ajan kuluessa muuttua. Esimerkiksi kaikki seuraavat säännöt voidaan ilmaista kuvan 11 tietoholvimallin keskiöillä ja linkeillä ilman olemassa olevien relaatioiden rakenteen muuttamista: jokainen asiakas voi tehdä yhden ostotapahtuman (yhden suhde moneen), yhteen ostotapahtumaan voi liittyä useampi asiakas (monen suhde yhteen), asiakas voi tehdä useita ostotapahtumia ja yhteen ostotapahtumaan voi liittyä useita asiakkaita (monen suhde moneen).

Joustava yhteyksien muodostaminen linkkirelaatioiden avulla mahdollistaa tietomallin muuttamisen rikkomatta vanhaa tietoa. Monesta moneen yhteyksiä muodostavan linkkirakenteen avulla tietoholvimalliin voidaan vaivattomasti lisätä uusia liiketoiminta-avaimia (keskiöitä), sillä vanha tietomalli on uuden osajoukko ja sitä voidaan käyttää tietomallin laajennuksen jälkeen kuten ennenkin. Esimerkiksi kuvassa 12 on lisätty tietomalliin uusi liiketoiminta-avain *valmistajatunnus*, joka kuvaa tuotteen valmistajaa. On huomattava, että tähtimallissa ja lumihiutalemallissa tilanne ei ole sama, sillä yhteyksiä ei välttämättä muodosteta monesta moneen yhteyksiä luovan linkkirelaation avulla. Sekä tähti- että lumihiutalemallissa valmistajatiedon lisääminen *tuote*-ulottuvuuteen aiheuttaisi uuden attribuutin lisäämisen olemassa olevaan *tuote*-tauluun, jolloin vanhojen taulujen rakenne ei pysyisi tietomallin muuttuessa samana.



Kuva 12: Uuden keskiöksi mallinnetut liiketoiminta-avaimen lisäys tietoholvimallissa. Etu- ja jälkiliitteet: **LNK_** = Linkki, **HUB_** = Keskiö, **_SQN** = keinovain, **_LDTS** = latauspäivä, **_RSRC** = lähdejärjestelmä.

Myös tiedon karkeisuustasoa voidaan vaihtaa lisäämällä uusi linkkitaulu ja keskiö [Lin11a s. 53-54]. Esimerkiksi asiakkaan tekemiä ostotapahtumia myymälöissä voidaan ilmaista asiakkaan ja myymälän välisellä yhteydellä. Jos karkeisuustasoa halutaan muuttaa ja tarkastella asiakkaan ostotapahtumia myös tuotteittain, lisätään tietomalliin uusi keskiö ja vanhan linkin rinnalle voi-

daan lisätä uusi linkki. Tähtimallissa vastaava operaatio olisi uuden ulottuvuuden luominen ja uuden faktataulun lisääminen vanhan rinnalle. Vaihtoehtoisena tapana on muokata olemassa olevaa linkkitaulua (tähtimallissa faktataulua), mutta tätä ei tietoholvimallissa suositella tiedon luotettavuuden ja joustavuuden heikentymisen vuoksi [Lin11a s. 54].

Linkkien avulla voidaan kuvata monenlaisia yhteyksiä, kuten

- yleistä yhteyttä (relationship links), esimerkiksi työntekijän kuuluminen osastoon.
- hierarkiaa (hierarchical links), esimerkiksi yrityksen osastoista muodostuva organisaatiohierarkia.
- vastaavuutta (same-as links), esimerkiksi järjestelmistä ladattujen ja eri keskiöihin tallennettujen ja eri tavalla kirjoitettujen osastotunnusten vastaavuus.
- tapahtumatietoa (transactional links), esimerkiksi myymälän myyntitapahtumat.
- koostetietoa (computed and aggregated link), esimerkiksi myymälän myyntitapahtumien kokonaissumma kuukauden ajalta.
- heikkoarvoista yhteyttä (low value link), esimerkiksi toissijainen tavarantoimittaja.
- toisistaan riippumattomien elementtien välistä liiketoiminnallista yhteyttä (exploration links), esimerkiksi mainoskampanjan potentiaaliset asiakkaat.
- tiedonlouhinnan avulla löydettyä yhteyttä (dynamic links), esimerkiksi todennäköisyyttä sille, että kahden eri käyttäjätunnuksen omistaja on sama henkilö.

Kahden tai useamman linkin välisiä yhteyksiä ei suositella muuten kuin poikkeustapauksissa [Lin11a s. 61-64], sillä ne heikentävät latausoperaatioiden rinnakkaisuutta. Toisiinsa liitetyt linkit muodostavat hierarkkisen rakenteen, jossa linkeillä (lapset), on viittauksia muihin linkkeihin (isät), mistä seuraa että lapsilinkkejä ei voida ladata ennen kuin isälinkit on ladattu.

4.3 Satelliitit

Satelliitit sisältävät keskiöihin tallennettujen liiketoiminta-avainten ja linkkitauluihin tallennettujen yhteyksien kuvaavia ominaisuuksia [Lin11a s. 75-91]. Avainten ja yhteyksien ominaisuudet ovat usein luonteeltaan muuttuvia ja satelliitit kuvaavat ominaisuuksia nimenomaan tietyllä ajanhetkellä. Satelliitit sisältävät vastaavanlaista tietoa kuin tähtimallin ulottuvuuden attribuutit tai ER-mallin yksilöiden ja yhteyksien attribuutit, mutta tietoholvimallissa kuvaustiedot on eristetty avaimesta erilliseen relaatioon. Tällä pyritään hallitsemaan kuvaustiedoissa vähitellen tapahtuvat muutokset menettämättä kuitenkaan niiden historiaa. Koska keskiöistä tai linkeistä ei ole lainkaan viitettä satelliitteihin, voidaan keskiöihin ja linkkeihin

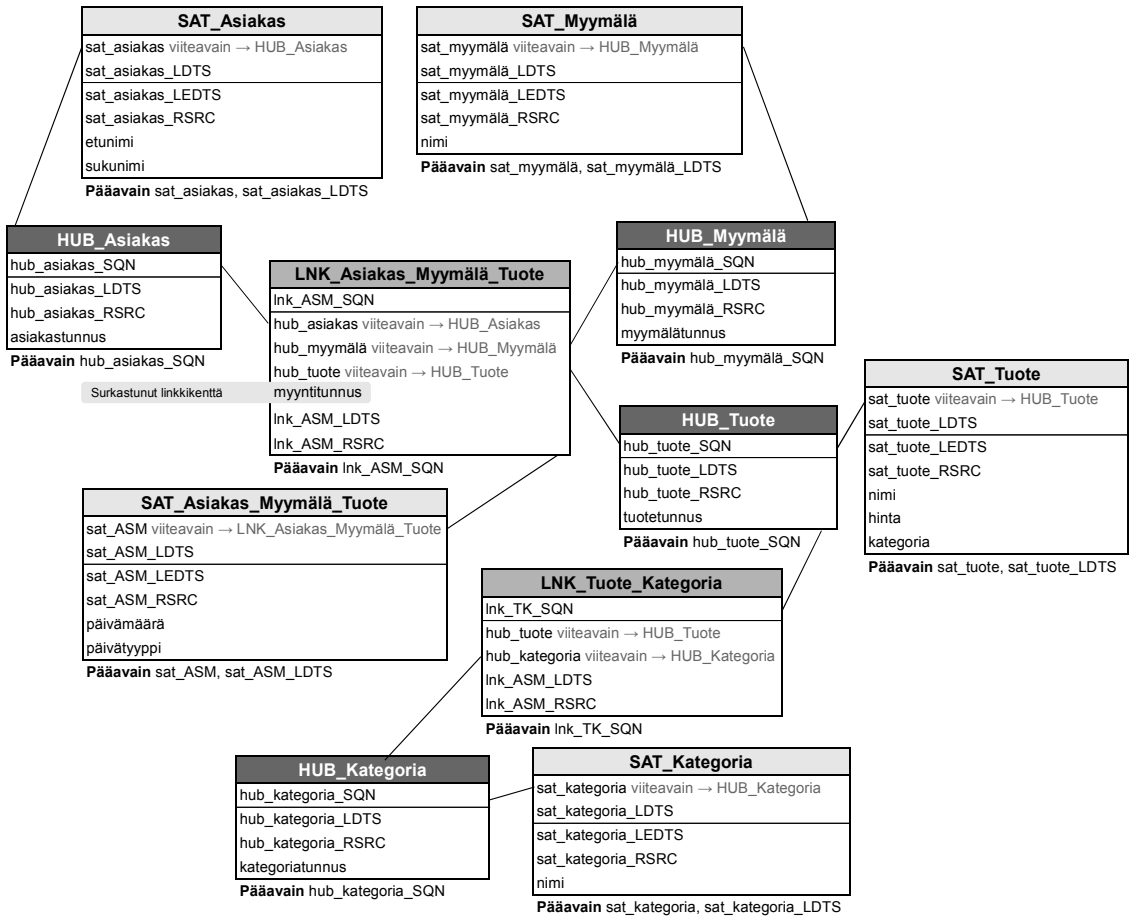
tarvittaessa liittää useita satelliitteja, mikä mahdollistaa kuvaavien tietojen jakamisen helposti erillisiin relaatioihin.

Kuvaustietoja voidaan jakaa erillisiin satelliitteihin kuvaustiietojen ominaisuuksien mukaan, esimerkiksi *tietotyyppien*, tiedon *muuttumistiheyden* tai *lähdejärjestelmän* perusteella [Lin11a s. 78-82]. Esimerkiksi sellaiset tietotyypit kuin vakiokokoiset elementit, vaihtelevankokoiset merkkijonot tai suuret tietotyypit voidaan tallentaa omiin satelliitteihinsa, jolloin niiden indeksointi ja pakkaus tehostuvat. Vastaavasti suorituskykyä voidaan tehostaa tallentamalla usein muuttuvat kuvaustiedot eri satelliitteihin kuin harvoin muuttuvat [Lin11a s. 80-81]. Usein muuttuvat attribuutit lisäävät relaation harvoin muuttuvien attribuuttien toisteisuuden määrää ja kasvattavat taulun kokoa. Harvoin muuttuvien attribuuttien ylimääräistä kopioimista voidaan välttää osittamalla harvemmin päivittyvät kuvaustiedot eri satelliitteihin kuin usein päivittyvät. Satelliittitietojen päivittymistiheys voi myös harventua, jolloin voidaan harkita satelliittien yhdistämistä [Lin11a s. 94-96]. Satelliitteja voidaan siis osittaa tai yhdistää dynaamisesti. Päätökset tulee tehdä tapauskohtaisesti esimerkiksi tiedon muuttumistiheyden, taulujen koon, käyttötavan ja käytettävän laitteiston suorituskyvyn perusteella. Myös eri lähdejärjestelmien kuvaustiedot suositellaan tallennettavaksi omiin satelliitteihin muun muassa selkeyden takia [Lin11a s. 81-84]. Satelliittia, joka sisältää esimerkiksi asiakastietoja kaikista operatiivisista järjestelmistä, kutsutaan *ylikuormitetuksi.* Ylikuormitettu satelliitti aiheuttaa helposti sekaannuksia tiedon tulkinnassa ja tiedon laadussa.

Kuvaustietojen pilkkominen suureen määrään satelliitteja voi kuitenkin vaikuttaa suorituskykyyn ja [Lin11a s. 78] antaa esimerkkejä laitekokoonpanoista, joilla ”kyselyt ja rinnakkaisuus toimivat melko hyvin” ja laitekokoonpanoista, joilla ”satelliittien normalisointi toimii suorituskykyperiaatteita vastaan”. Normalisoinnilla tarkoitetaan tässä yhteydessä kuvaustietojen vertikaalista ositusta eri satelliitteihin [Lin11a s. 79]. Mitään tarkempia testituloksia tai viitteitä tutkimuksiin ei kuitenkaan anneta, joten lausunto on kyseenalainen.

Satelliitin pakolliset attribuutit ovat *tasan yksi isätauluun viittaava avain*, *lähdejärjestelmä*, *latauspäivä* ja *päättymispäivä* [Lin11a s. 76-77]. Satelliitin isätaulu voi olla keskiö tai linkki eikä satelliitti saa sisältää viiteavaimia muihin tauluihin kuin isätauluun. Jos kuvaustietojen rakenne ei ole vakio, esimerkiksi puhelinnumeroita voi olla määrittelemätön määrä, voidaan niiden erottelemiseen käyttää *alinumerointia* (sub-sequence number) [Lin11a s. 89]. Päättymispäivä ei kuulu alkuperäiseen arkkitehtuuriin, mutta suorituskykysyistä se on tehty satelliiteille pakolliseksi [Lin11a s. 28][Lin10a]. Satelliitti voi viitata liiketoiminta-avaimeen, jota ei vielä ole olemassa. Linstedt ei kuitenkaan selitä tyhjentävästi miten tällainen tilanne käsitellään, vaan asia luvataan selittää henkilökohtaisissa koulutuksissa tai kirjassa, jota kutsutaan nimellä *Data Vault Implementation* [Lin11a]. Kirjaan ei ole mitään viitettä eikä ole mahdollista päätellä mistä

kirjasta on kysymys. Satelliitti voi sisältää myös tietueen *luontipäivän* ja *purkupäivän*, jos ne ovat saatavilla lähdejärjestelmästä [Lin11a s. 27-28]. Luontipäivä ilmoittaa milloin tietue kokonaisuudessaan luotiin lähdejärjestelmään ja purkupäivä, milloin se purettiin lähdejärjestelmästä tietovarastoon latausta varten. Kuvassa 13 on kuvattu *Market*-esimerkkimyyntijärjestelmän tietoja mallinnettuna tietoholvimallin keskiöiksi, linkeiksi ja satelliiteiksi. Satelliitin attribuutin päättymispäivä on spesifikaation mukaisesti merkitty jälkiliitteellä *_LEDTS*.

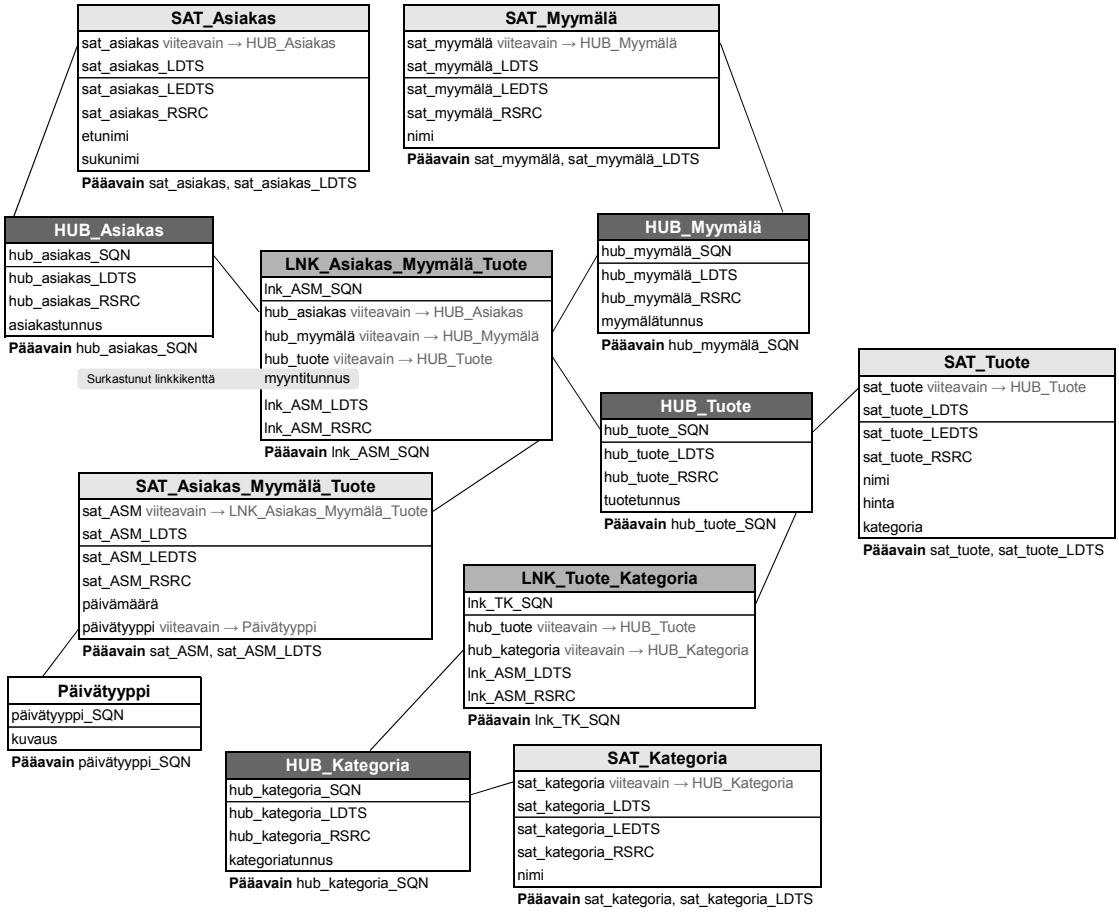


Kuva 13: Satelliittien lisäys *Market*-esimerkkimyyntijärjestelmän tietoholvimalliin. Etu- ja jälkiliitteet: **LNK_** = Linkki, **HUB_** = Keskiö, **_SQN** = keinovain, **_LDTS** = latauspäivä, **_LEDTS** = päättymispäivä, **_RSRC** = lähdejärjestelmä.

Satelliittien tarkoitus on tallentaa yksityiskohtaisesti liiketoimintaan liittyvät kuvaustiedot aikasidonnaisesti. Sen lisäksi satelliitteja voidaan käyttää joihinkin erikoistarkoituksiin, esimerkiksi liiketoiminta-avaimen tai linkin aktiivisuuden ilmoittamiseen (effectivity satellite), lataustapahtumien tietojen tallentamiseen (record tracking satellite), tiedoille suoritettujen operaatioiden tallentamiseen (status tracking satellite) tai liiketoimintasäännöillä käsitellyn tiedon tallentamiseen (computed satellite) [Lin11a s. 85-88].

4.4 Muut elementit

Viitetauluihin (reference tables, stand-alone tables, lookup tables) tallennetaan kuvaavaa tietoa, joihin viitataan tietovarastossa useissa eri yhteyksissä [Lin10a][Lin11a s. 102-106]. Tällaista tietoa ovat esimerkiksi erilaiset standardikuvaukset, koodistot, luokittelut ja lyhenteet, jotka voidaan tallentaa niille erikseen varattuihin itsenäisiin tauluihin. Viitetaulujen käyttö säästää tallennustilaa ja vähentää tietokannan toisteisuutta, sillä eri yhteyksissä käytettyä viitetietoa ei tarvitse kopioida useisiin paikkoihin vaan siihen voidaan tehdä useasta eri taulusta pelkkä viite. Viitetaulut voivat liiketoimintavaatimusten mukaan historioida tiedoissa tapahtuvat muutokset tai olla historioimatta. Jos viitetiedoissa tapahtuvia muutoksia ei tarvitse historioida, voidaan käyttää tavallista normaalimuodossa olevaa viitetaulua. Jos viitetiedoissa tapahtuvat muutokset pitää historioida, viitetietojen tallentamiseen voidaan käyttää keskiöitä ja satelliitteja. Kuvassa 14 on kuvattu *Market*-esimerkkimyyntijärjestelmän täydellinen tietoholvimalli viitetiedolla *Päivätyyppi* täydennettynä.



Kuva 14: *Market*-esimerkkimyyntijärjestelmä kuvattuna täydellisesti loogisen suunnittelutason tietoholvimallilla. Etu- ja jälkiliitteet: **LNK_** = Linkki, **HUB_** = Keskiö, **_SQN** = keinovain, **_LDTS** = latauspäivä, **_LEDTS** = päättymispäivä, **_RSRC** = lähdejärjestelmä.

Viitetiedot voivat muodostaa varsin hajanaisen joukon tietoja jotka kannattaa tallentaa samaan relaatioon. Nimissä tapahtuvien yhteentörmäyksien välttämiseksi voidaan viitetaulun tietoja ryhmitellä, antaa ryhmälle jokin nimi ja tehdä siitä yksi viitetaulun attribuutti.

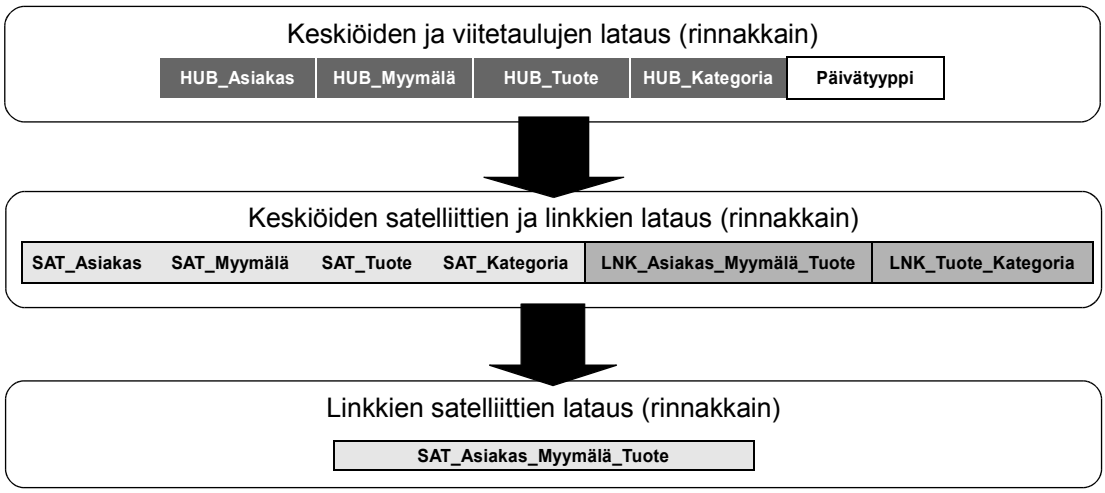
Muita tietoholvimallissa käytettäviä erikoistauluja ovat tilannekuvataulut ja siltataulut, joita voidaan tietoholvimallissa käyttää kyselyiden nopeuttamiseksi tai reaaliaikaisen tiedon kyselyiden viivästyttämiseen [Lin11a s. 97-101]. Taulujen tiedot luodaan automaattisesti esimerkiksi latausoperaation aikana mutta niille voidaan määritellä mikä tahansa päivitystahti sekunneista kuukausiin. Nämä taulut eivät ole välttämättömiä korkean suorituskyvyn takaavissa tietokannan hallintajärjestelmissä, kuten sarakepohjaisissa tietokantajärjestelmissä [Lin11a s. 97].

Tilannekuvataulut (point in time table, PIT) ovat satelliittien erikoistapaus. Tilannekuvataulut sisältävät täsmälleen yhteen isätauluun liittyvien satelliittien latauspäivät ja niiden avulla voidaan nopeuttaa satelliittien avulla tapahtuvia liitoksia, koska liitosehdossa voidaan käyttää yhtäläisyyttä [Lin11a s. 97-99]. Tarvittaessa mukaan voidaan liittää myös keskiön liiketoiminta-avain, lähdejärjestelmä ja päättymispäivä mutta yleensä tilannekuvataulu kannattaa pitää sekä sarakkeiden että rivien määrältä mahdollisimman pienenä. Sarakkeittain tapahtuva tiivistys ja vanhojen tietueiden poisto on myös suositeltavaa [Lin11a s. 99].

Siltataulut (bridge table) ovat linkkitaulujen erikoistapaus. Niillä yhdistetään useiden eri keskiöiden ja linkkien tietoja toisiinsa [Lin11a s. 99]. Ne ovat siis etukäteen laskettuja liitoksia keskiöiden ja linkkitaulujen pääavaimista, jotka eivät suoraan liity toisiinsa. Pääavainten lisäksi siltataulun tietueet voivat myös sisältää keskiöiden varsinaiset liiketoiminta-avaimet tai summatietoa. Siltataulun attribuuttien määrän kasvattaminen voi kuitenkin nopeasti heikentää siitä saatavaa suorituskyvyn parannusta, joten attribuuttien määrä kannattaa pitää pienenä [Lin11a s. 100].

4.5 Tietoholvimalli ja rinnakkaisprosessointi

Yksi tietoholvimallin ajatuksista on, että lähdejärjestelmistä tehtävät lataukset voidaan rinnakkaistaa tehokkaasti [Lin05]. Keskiöt ladataan ensin. Koska ne ovat täysin itsenäisiä kukin keskiö voidaan ladata rinnakkaissuorituksessa muista keskiöistä riippumatta. Seuraavana voidaan ladata linkit ja keskiöiden satelliitit rinnakkain. Jos suosituksia on noudatettu myös kaikki linkkien ja keskiöiden satelliittien lataukset voidaan rinnakkaistaa, sillä ne eivät viittaa toisiinsa. Lopuksi ladataan satelliitit, jotka viittaavat linkkeihin. Viitetaulujen lataamisesta ei anneta selkeitä ohjeita, mutta ne voidaan ladata esimerkiksi rinnakkaissuorituksessa keskiöiden kanssa. Kuvassa 15 on esitetty *Market*-esimerkkimyyntijärjestelmän taulujen lataus tietoholvirakenteeseen.



Kuva 15: *Market*-esimerkkimyyntijärjestelmän taulujen lataus tietoholvirakenteeseen.s

Linkkejä voidaan käyttää yhdistämään hajautetun järjestelmän fyysisesti eri paikoissa sijaitsevaa tietoa toisiinsa [Lin11a s. 52], mutta voidaan toimia myös toisin päin: hajauttaa keskiöitä ja linkkejä fyysisesti erillisiin prosessointiyksiköihin. Linstedt esittää, että tietoholvimalli tukisi erityisen tehokkaasti massiivista rinnakkaisprosessointia [Lin11a s. 12-16, 55-57]. Rinnakkaisprosessoinnin tehokkuus riippuu kuitenkin ennen kaikkea siitä, kuinka tehokkaasti suoritettava operaatio voidaan pilkkoa itsenäisiksi, rinnakkain suoritettaviksi osaoperaatioiksi. Lähdejärjestelmistä tehtävät *lataukset* voidaan tietoholvimallissa pilkkoa tehokkaasti itsenäisiksi operaatioiksi [Lin05]. Sen sijaan esimerkkiä tai koetulosta siitä, kuinka keskiöiden ja linkkien hajauttaminen nopeuttaisi *kyselyjen* vasteaikaa suhteessa muihin malleihin, ei anneta. Joustavan linkkirakenteen vuoksi tietoholvimalli sisältää suuren joukon tauluja, mikä lisää kyselyissä tarvittavien liitosten määrää ja suuri taulujen määrä yleensä nimenomaan heikentää kyselyiden vasteaikoja. Tosiaikaisessa tiedonjalostuksessa kyselyjen vasteaika on usein tärkeämpi suorituskyvyn mittari kuin latausten tehokkuus. Tietoholvimalli ei yleensä sovellukaan suoraan loppukäyttäjien ja sovellusten tietokantakyselyiden kohteeksi, minkä vuoksi sitä käytetään lähinnä keskitetyn tietovaraston yksityiskohtaisen tiedon tallentamiseen. Keskitetyn tietovaraston tietoholvimallinnetusta tiedosta voidaan johtaa erilaisia esityskerroksen näkymiä, esimerkiksi paikallisvarastoja [Lin04][Lin11a s. 21]. Tietovarastoinnissa tieholvimalli onkin vaihtoehto luvussa 2 esitellylle normalisoidulle tietomallille.

Tietoholvimallin rinnakkaisprosessointia tukevan rakenteen esittelyn yhteydessä [Lin11a s. 15-16] väittää, että ”tietoholvimalli noudattaa skaalautumattoman verkon topologiaa” ja ”mikä tahansa tietoholvimallin periaattein rakennettu fyysinen malli sisältää skaalautumattoman verkon matemaattiset ominaisuudet”. *Skaalautumattomassa* eli *mittakaavattomassa verkossa* solmujen yhteyksien todennäköisyysjakauma noudattaa potenssilakeja, jolloin suurimpaan osaan solmuja

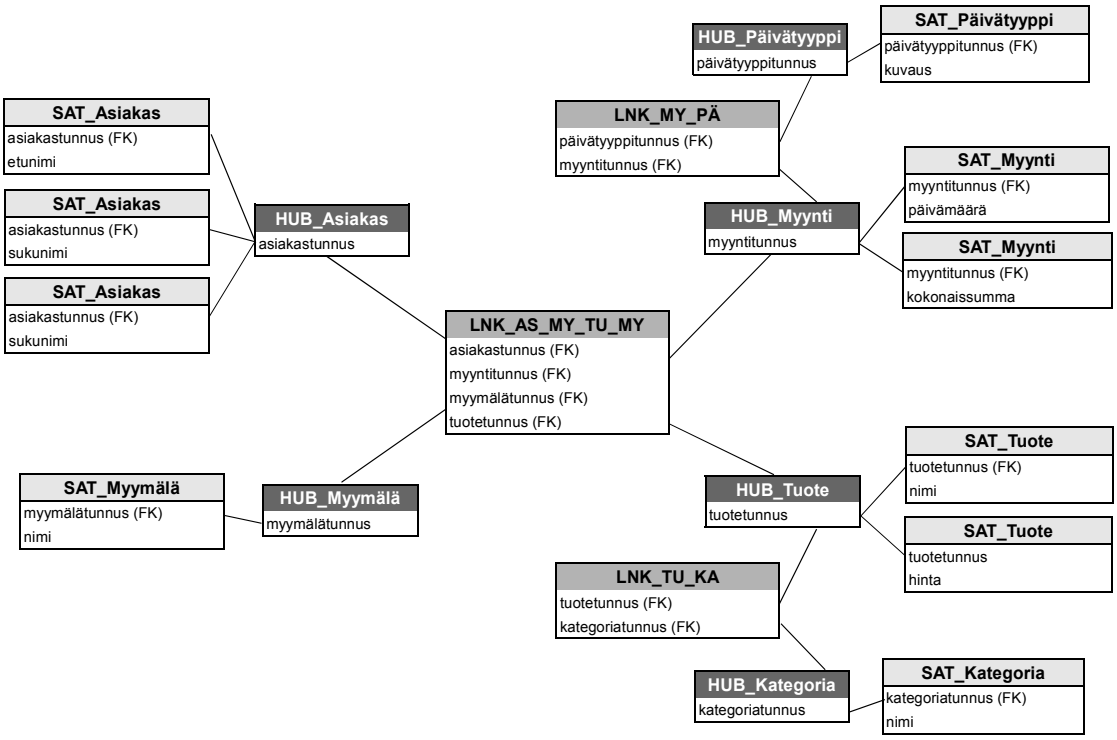
liittyy vain vähän yhteyksiä eli särmiä, kun taas joillakin harvoilla solmuilla on huomattavan paljon yhteyksiä [Bar02][HSK06]. Tietoholvimallissa, jossa solmut koostuvat keskiöiksi mallinnetuista liiketoiminta-avaimista ja särmät muodostuvat linkkirelaatioiden avulla esitetyistä liiketoiminta-avainten välisistä yhteyksistä, tieto mallinnetaan verkkona. Itse tiedon luonteesta kuitenkin riippuu muodostuuko niistä skaalautumaton verkko vai ei. Voisimme esimerkiksi kuvitella tietoholvimallin, jossa on ainoastaan keskiöt *Vuokra-asunto* ja *Vuokralainen.* Kaikki asunnot on vuokrattu tasan yhdelle vuokralaiselle ja jokainen vuokralainen vuokraa vain yhtä asuntoa. Jokaiseen vuokralaiseen ja vuokra-asuntoon liittyy siis täsmälleen yksi yhteys, jolloin kyseessä ei ole skaalautumaton verkko. Tietoholvimallinnus voi paljastaa (tai olla paljastamatta) tiedoista skaalautumattoman verkon, mutta niin voi tehdä myös ER-kaavioista johdettu tietomalli tai jopa tähtimalli. Toisin kuin Linstedt antaa ymmärtää, skaalautumattomuus ei siis ole erikoisesti tietovarastomallin ominaisuus vaan talletettavien tietojen ominaisuus. Linstedt ei myöskään esitä esimerkkiä tietoholvimallilla toteutetusta skaalautumattomasta verkosta tai tutkimustulosta siitä, kuinka tämä tehostaisi rinnakkaisprosessointia.

4.6 Tietoholvikäsitemalli

Linstedtin esittelemä tietoholvimalli on ennen kaikkea loogisen tason tietomalli, joka perustuu relaatioihin ja tiedon jakamiseen keskiö-, linkki- ja satelliittitauluihin. Jovanovic ja Bojocic pyrkivät laajentamaan loogista ja fyysistä tietoholvimallia myös käsitetasolle (conceptual data vault model, C-DV) [JoB12]. He nimittävät tietoholvimallia tietovaraston *latausalueen* (staging area) tai *pysyvän latausalueen* (persistent staging area) malliksi. Tässä tutkielmassa käytetään latausalueesta (staging area) kuitenkin yleisesti käytössä olevaa määritelmää, jonka mukaan kyseessä on alue, johon lähdejärjestelmien tiedot kopioidaan ennen tietovarastoon tuomista ilman muunnoksia ja historian tallentamista [HHK09 s. 25][Lin11a s. 19-20]. Tietoholviksi tai keskitetyksi tietovarastoksi nimitetään sellaista aluetta, jota [JoB12] nimittää latausalueeksi tai pysyväksi latausalueeksi ja joka voi toimia myös *mastertiedon* (master data) eli *yrityksen yhteisen tiedon päälähteenä* (system of record).

C-DV-mallin avulla pyritään esittämään ja analysoimaan tietovaatimuksia aikaisessa vaiheessa ja sitä on tarkoitus käyttää välimallina lähdejärjestelmien tietomallin ja loogisen tason tietoholvimallin välillä [JoB12]. C-DV-malli koostuu ainoastaan keskiöistä, linkeistä ja satelliiteista [JBK12]. Keskiöön liittyy liiketoiminta-avain. Linkki voi kuvata yleistystä, yhteyttä tai riippuvuutta. Satelliitti koostuu keskiön avaimeen liittyvistä attribuuteista. Jokainen satelliitti sisältää vain yhden kuvaustiedon, jolloin malli täyttää kuudennen normaalimuodon vaatimukset [JBK12]. Relaatio on kuudennessa normaalimuodossa, jos se ei sisällä lainkaan epätriviaaleja liitosriippuvuuksia eli sitä ei voida enää mitenkään osittaa useammaksi relaatioksi, jotka sisäl-

täisivät vähemmän attribuutteja kuin alkuperäinen [RRB10]. Loogista tietoholvimallia ei kuitenkaan suositella normalisoimaan erityisesti mihinkään asteeseen [Lin10c]. C-DV-mallissa myöskin linkkien väliset yhteydet ovat täysin sallittuja ja kaikkiin elementteihin liittyy joukko metatietoja, esimerkiksi lähdejärjestelmä ja ajankohta. C-DV-mallia esittelevässä artikkelissa on esitetty myös lyhyesti säännöt, joilla UML-tietomalli muuunetaan C-DV-malliksi [JoB12]. Kuvassa 16 on esitetty *Market*-esimerkkimyyntijärjestelmän tiedot C-DV mallinnettuna, ilman metatietoja.



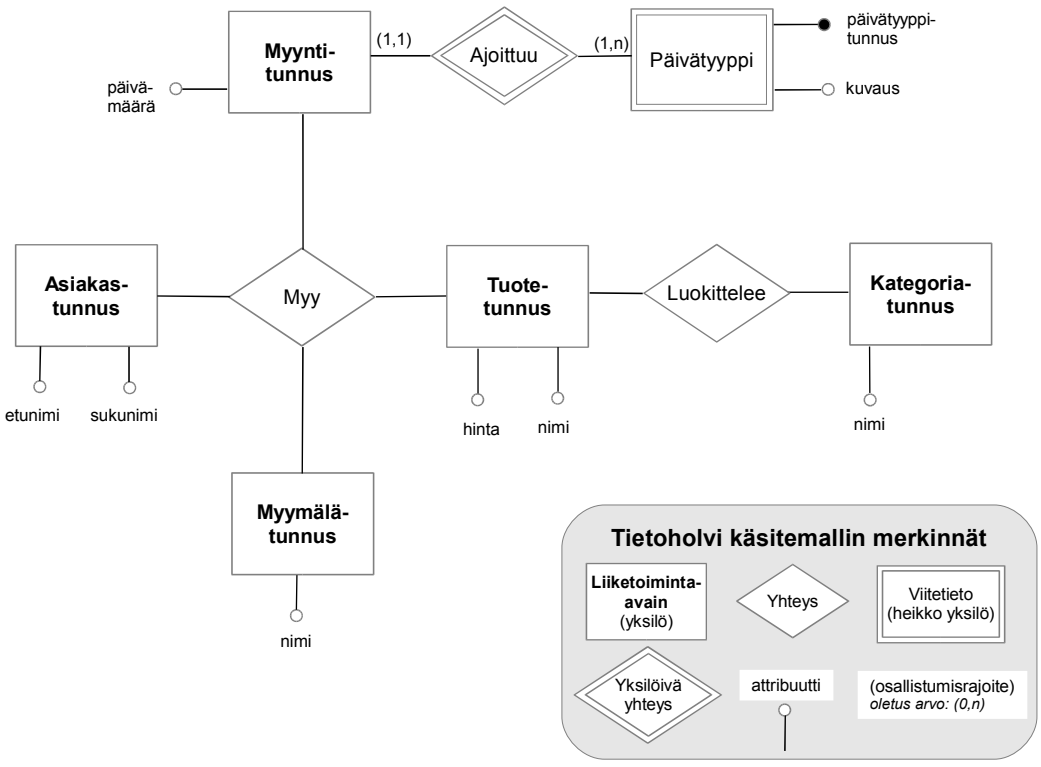
Kuva 16: *Market*-esimerkkimyyntijärjestelmä kuvattuna C-DV-mallilla.

Selvästikin C-DV-malli on rakenteeltaan hyvin samankaltainen kuin relaatioina kuvattu loogisen tason tietoholvimalli. Tämä johtuu siitä, että C-DV-mallissa käytetään loogisen tason käsitteitä *keskiö*, *linkki* ja *satelliitti*, jolloin voidaan kysyä, onko kyseessä aito toteutuksesta riippumaton käsitemalli. Keskiöt, linkit ja satelliitit ovat nimittäin alunperin relaatioiden kautta määriteltyjä tietoholvimallin käsitteitä ja niihin liittyy esimerkiksi sellaisia loogisen tason käsitteitä kuin taulu, keinovain ja viiteavain, joita esiintyy myös C-DV mallia kuvaavassa esimerkissä [JoB12]. Esimerkiksi ER-mallissa tai moniulotteisessa CGMD-mallissa vastaavanlaista käsitetason ja loogisen tason välistä riippuvuutta ei ole.

Voidaan tietysti väittää, että mikä tahansa käsitetason malli käyttää aina joitakin käsitteitä tiedon luokitteluun: esimerkiksi ER-malli käyttää yksilöiden, yhteyksien ja attribuuttien käsitteitä ja C-DV-malli käyttää keskiöiden, linkkien ja satelliittien käsitteitä. Toisin kuin ER-mallissa, C-DV-mallissa käytetyt käsitteet ovat kuitenkin täsmälleen samat kuin loogisen tason tietoholvi-

mallissa. Sen vuoksi on kyseenalaista, voidaanko puhua aidosta, toteutuksesta riippumattomasta käsitetason mallista. Jos C-DV mallia pidetään aitona käsitetason mallina, niin keskiöillä, linkeillä ja satelliiteilla pitäisi ajatella olevan eri merkitys käsitetasolla ja loogisella tasolla, toisin sanoen niitä ei pitäisi C-DV mallissa mieltää lainkaan relaatioiksi. Tämä on kuitenkin täydellisen vastaavuuden vuoksi vaikeaa, joten voidaan ajatella, että C-DV on vain karkeistettu versio loogisen tason tietoholvimallista. Se on siis eri tarkkuustason, mutta ei eri abstraktiotason kuvaus loogisesta tietoholvimallista.

Tässä tutkielmassa ehdotetaan loogisen tietoholvimallin kanssa käytettäväksi yksinkertaista käsitetason kuvaustapaa, joka käyttää seuraavia käsitteitä: *liiketoiminta-avain*, *liiketoiminta-avaimen ominaisuus*, *yhteys* ja *viitetieto*. Tällaiset käsitteet kuvaavat reaalimaailman objekteja toteutusriippumattomasti, jolloin kyse on aidosta käsitetason mallista. Käsitteet voidaan helposti esittää laajennetulla ER-kuvauksella, jossa liiketoiminta-avaimet esitetään yksilötyyppeinä, monesta moneen yhteydet yhteystyyppeinä, ominaisuudet attribuutteina ja viitetiedot heikkoina yksilötyyppeinä. Yhteyksien osallistumisrajoitteet ovat oletusarvoisesti monen suhde moneen, joten niitä ei tarvitse merkitä muuten kuin viitetietojen tapauksessa. Mahdolliset metatiedot, kuten *lähdejärjestelmä* voidaan haluttaessa kuvata liiketoiminta-avainten ja viitetietojen attribuutteina. Kuvassa 17 on esitetty *Market*-esimerkkimyyntijärjestelmän tiedot kuvattuna ehdotetulla tietoholvikäsitemallilla. Loogisen tason surkastunut linkkikenttä *myyntitunnus* on käsitetason kuvauksessa mallinnettu normaalina liiketoiminta-avaimena.



Kuva 17: *Market*-esimerkkimyyntijärjestelmä kuvattuna ehdotetulla tietoholvikäsitemallilla.

Loogisen tason kuvaus voidaan pääasiallisesti johtaa tästä käsitetason kuvauksesta muuntamalla liiketoiminta-avaimet keskiöiksi, yhteydet linkeiksi, attribuutit satelliiteiksi, viitetiedot erillistauluiksi ja heikot yhteydet viitteiksi erillistauluihin. Kuvan 17 liiketoiminta-avain M*yyntitunnus* voidaan loogisella tasolla mallintaa *Myy*-yhteydestä johdetun linkin surkastuneeksi kentäksi ja siihen liittyvä päivämäärä linkin satelliittitiedoksi.

4.7 Ankkurimalli

Tietoholvimallin kanssa hieman samantapainen tietomalli on ankkurimalli (anchor model). Kuten tietoholvimallissa, myös ankkurimallissa kaikki tietomallissa tapahtuvat muutokset pyritään tekemään ainoastaan lisäyksillä, jolloin ajan myötä tapahtuvat muutokset eivät riko jo olemassa olevaa toiminnallisuutta [RRB10]. Samoin ankkurimallissa historioidaan tarvittaessa kaikki muutokset, mikä mahdollistaa tiedon eri versioihin kohdistuvat kyselyt. Ankkurimallin eduiksi mainitaan mallinnuksen helppous, tietokannan hallinnan helpottuminen ja hyvä suorituskyky tietyissä tilanteissa [RRB10].

Ankkurimallissa tieto jaetaan *ankkureihin* (anchor), *solmuihin* (knot), *yhdistimiin* (tie) ja *attribuutteihin* (attribute). Yksilöt mallinnetaan ankkureina, ankkurien väliset yhteydet yhdistiminä, ankkureihin liittyvät ominaisuudet attribuutteina ja viitetiedot solmuina. Tapahtumatyyppiset tiedot, kuten myyntitapahtumat, mallinnetaan yleensä ankkureina. Tapahtumatiedot voidaan mallintaa myös yhdistiminä, jos ne ovat tähtimallin faktattomien faktojen kaltaisia, eli niihin ei liity lainkaan kuvaavaa tietoa. Attribuutit jaetaan staattisiin (static), historioitaviin (historized), solmuihin tallennettuihin (knotted static) ja historioitaviin solmuihin tallennettuihin (knotted historized).

Ankkurimalliin liittyy myös sitä varten kehitetty graafinen esitystapa, jota voidaan käyttää ankkurimallinnettavan tiedon käsitteelliseen suunnitteluun. Graafisesti kuvattu ankkurimalli voidaan yksinkertaisilla muunnossäännöillä muuntaa relaatioksi ja tietokantatauluiksi. Myös ER-kaaviolla tai ORM-mallilla (object role modeling) [Hal01] voidaan soveltaen esittää ankkurimallinnettua tietoa [RRB10].

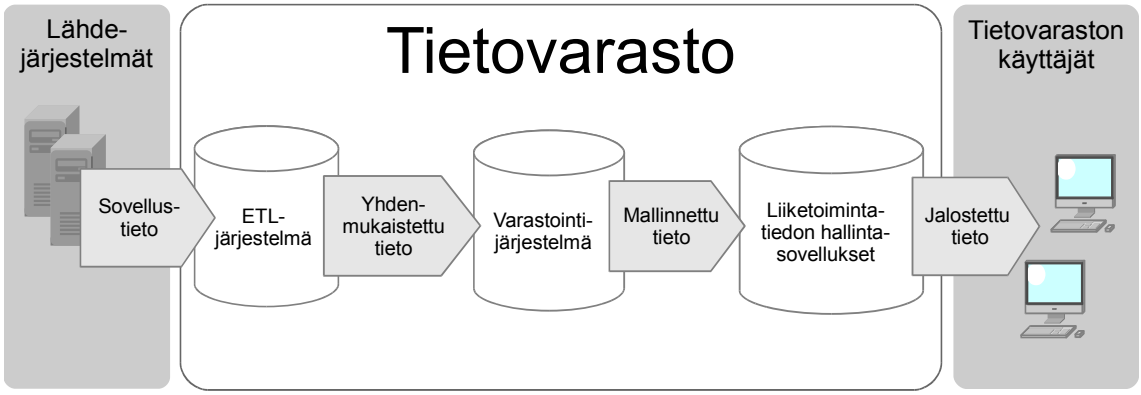
Tietoholvimallia ja ankkurimallia kutsutaan joskus *hypernormalisoiduiksi malleiksi* [Hug12 s. 264-298]. Ankkurimallin käyttö johtaa kuudennessa normaalimuodossa (6NF) oleviin relaatioihin ja erittäin suureen määrään tauluja. Ainoastaan yhdistiminä voi ankkurimallissa esiintyä relaatioita, jotka eivät täytä kuudennen normaalimuodon vaatimuksia. Tietoholvimalli ei ole täydellisessä kuudennessa normaalimuodossa, sillä satelliitit eivät välttämättä ole korkeasti normalisoituja [RRB10]. Keskiöiden voi kuitenkin ajatella olevan hyvin lähellä kuudetta normaalimuotoa keinoavaimen ja luonnollisen avaimen yksi yhteen suhteen vuoksi [Lin11b].

5 Tietovaraston arkkitehtuuri

Tietovaraston tieto kootaan useista eri lähdejärjestelmistä. Yleensä lähteet ovat operatiivisia järjestelmiä, joita käytetään organisaation päivittäisen liiketoiminnan, esimerkiksi tilausten tai laskujen hallintaan. Itse tietovarasto sisältää useita eri tehtäviin tarkoitettuja komponentteja, joiden avulla tietoa tuodaan tietovarastoon, tallennetaan ja jalostetaan päätöksentekoa tukevaksi tiedoksi loppukäyttäjille. Tietovaraston arkkitehtuuri kuvaa, minkälaisista komponenteista tietovarasto on toteutettu, mikä niiden tehtävä on ja miten ne toimivat suhteessa toisiinsa. Eri arkkitehtuuriratkaisuissa käytetään myös erilaisia tietomalleja.

5.1 Tietovaraston yleiset komponentit

Tiedon saavuttua operatiivisista järjestelmistä se kulkee tietovaraston eri komponenttien läpi ennen sen esittämistä tietovaraston käyttäjille. Tiedon kulku lähdejärjestelmistä tietovarastoon, tietovaraston sisällä ja lopulta tietovaraston loppukäyttäjille on esitetty kuvassa 18. Tietovarasto voidaan jakaa kolmeen yleiseen komponenttiin, jotka on tässä tutkielmassa nimetty *ETL-järjestelmäksi* (extract, transform, load), *varastointijärjestelmäksi* ja *liiketoimintatiedon hallintasovelluksiksi*.



Kuva 18: Tiedon kulku tietovaraston komponenttien välillä.

Tieto saapuu tietovarastoon yleensä ETL-järjestelmän kautta. ETL-järjestelmän tarkoituksena on *purkaa* (extract) tieto lähdejärjestelmistä, *muuntaa* (transform) se yhtenäiseen muotoon ja *ladata* (load) tietovarastoon. Tietojen lataus voi tapahtua reaaliaikaisesti, jolloin lähdejärjestelmän muutokset näkyvät lähes välittömästi tietovarastossa, tai tietyin väliajoin, esimerkiksi kerran vuorokaudessa tapahtuvana eräajona. *Purkamisen* yhteydessä tieto siirretään lähdejärjestelmästä tietovarastoon. Siirtäminen voidaan tehdä käyttäen joko veto- tai työntöstrategiaa ja purkamisessa voidaan käyttää hyväksi erillistä latausaluetta (staging area). *Muunnoksessa* eri lähteistä saatu

tieto yhdenmukaistetaan ja siistitään, sen eheys tarkistetaan ja se muokataan tietovarastolle sopivaan muotoon. *Latausvaiheessa* tieto tallennetaan tietovarastoon.

ETL-prosessista esiintyy myös muunnos, jossa lähdejärjestelmistä tulleet tiedot tallennetaan ensin sellaisenaan tietovarastoon ja vasta sen jälkeen tehdään tarvittavat muunnokset. Tällöin prosessi on nimeltään ELT (extract/load/transform). ELT-prosessia käyttämällä latausoperaatiot yksinkertaistuvat ja tehostuvat, niitä on helpompi laajentaa, eikä erillistä latauspalvelinta välttämättä tarvita [HHK09 s. 59][Lin11a s. 17]. Prosessista esiintyy myös muunlaisia muunnoksia, kuten ELTL tai TETL, sen mukaan missä järjestyksessä ja kuinka usein eri operaatiot suoritetaan [KRT08 s. 133]. Tässä tutkielmassa käytetään näistä kaikista muunnelmista yleistermiä ETL-prosessi.

ETL-prosessin jälkeen eri lähdejärjestelmistä saatu tieto tallennetaan tietovaraston varastointijärjestelmään yhdenmukaistetussa muodossa ja käyttökelpoisella tavalla mallinnettuna. Tällöin lähdejärjestelmistä saatu *sovellustieto* (application data) on muuttunut liiketoimintatiedoksi (corporate data) [ISN08 s. 216]. Tietovaraston liiketoimintatiedon avulla organisaatio pyrkii ymmärtämään paremmin liiketoimintaansa ja sen perusteella kehittämään omia toimintatapojaan. Varastointijärjestelmä voi olla toteutettu eri tavoin ja se voi jakautua eri tasoihin. Käytössä voi olla esimerkiksi keskitetty varasto, useita aihealueeltaan suppeampia paikallisvarastoja tai yhdistelmä keskitetystä varastosta ja paikallisvarastoista. Tieto voidaan myös tallentaa fyysisesti eri tekniikoilla: esimerkiksi relaatiotietokantaan, OLAP -kuutioon tai näihin molempiin. Yleensä tietoa tallennetaan varastointijärjestelmään myös monella eri karkeisuustasolla. Esimerkiksi myyntitietoa voidaan tallentaa yksittäisten myyntien tasolla, kuukausitasolla ja vuositasolla.

Varastointijärjestelmästä eheä ja yhtenäisesti mallinnettu tieto tarjotaan liiketoimintatiedon hallintasovelluksille, jotka ovat tietovaraston loppukäyttäjien työkaluja. Liiketoimintatiedon hallintasovellukset tarjoavat tietovaraston laajoihin tietoihin käyttäjien tarvitseman näkökulman. Liiketoimintatiedon hallintasovellukset ovat usein valmiita tuotteita, joiden avulla liiketoimintatieto näytetään loppukäyttäjälle jalostetussa muodossa, esimerkiksi tilastoina tai raportteina. Yleisiä tiedon jalostusoperaatioita ovat esimerkiksi koostaminen, ristiintaulukointi, karkeistus, porautuminen, vertailu ja viipalointi [ChD97]. *Koostamisessa* (aggregation) tiedosta esitetään jonkinlainen kooste, esimerkiksi keskiarvo tai summa. *Ristiintaulukoinnissa* (pivoting) tieto esitetään kahden ulottuvuuden avulla, esimerkiksi myyntitietoa voidaan esittää kaupungeittain ja tuotteittain. Tämä voidaan esittää kaksiulotteisessa taulukossa, jossa sarakeotsikkoina ovat kaupunkien nimet ja riviotsikkoina tuotenimet. *Karkeistuksessa* (roll-up) tiedon yksityiskohtaisuutta vähennetään, joko vähentämällä ulottuvuuksien määrää tai siirtymällä hierarkiassa ylemmälle tasolle [ChD97][JLV03]. Karkeistuksessa voitaisiin esimerkiksi siirtyä tarkastelemaan kaupungeittain ja vuosittain koostettua myyntiä pelkästään vuosittain tai siirtyä tarkastelemaan

kuukausittain koostettua myyntiä vuositasolla. *Porautuminen* (drill-down) on karkeistuksen vastakohta eli siinä tiedon yksityiskohtaisuutta lisätään. Esimerkiksi myyntitapahtumia voidaan porautua tarkastelemaan yksittäisen myyntitapahtuman tasolle asti. *Vertailussa* (comparing) verrataan kahta saman ulottuvuuden mukaan koostettua arvoa toisiinsa, esimerkiksi vuoden 2000 ja 2001 kokonaismyyntiä. *Viipaloinnissa* (slice and dice) tieto rajataan vain osaan ulottuvuuden arvoja, esimerkiksi tarkastellaan vain tietyn tuotteen myyntejä kaikissa kaupungeissa, tai tarkastellaan tietyn tuotteen myyntejä tietyssä kaupungissa, tiettynä vuotena. Muita jalostusoperaatioita ovat esimerkiksi järjestäminen ja arvojen johtaminen. Ennalta määriteltyjen jalostusoperaatioiden lisäksi liiketoiminnan hallintasovellusten on tarjottava loppukäyttäjälle mahdollisuus tehdä satunnaisia, ennalta määrittelemättömiä (ad hoc) kyselyitä tietovaraston tietoihin [ChD97] .

5.2 Arkkitehtuurit

Tietovarastointiympäristöissä käytetään muutamia vaihtoehtoisia arkkitehtuureja, jotka tarjoavat yleisiä suuntaviivoja tietovaraston toteuttamiseen. Arkkitehtuurivaihtoehdot on eri lähteissä luokiteltu, nimetty ja kuvattu hieman eri tavoin [AMO12][ArW06][ArW08][ArW10][KRT08 s. 248][SeS05][Sin10][WaA05]. Tässä tutkielmassa käytetään seuraavaa tietovarastojen arkkitehtuurien luokittelua ja nimeämistapaa:

- *Itsenäiset paikallisvarastot* (independent data marts)
- *Yhtenäistetyt paikallisvarastot* (data mart bus, data mart bus architecture with linked dimensional data marts, bus architecture, enterprise data warehouse bus architecture)
- *Yritystason keskitetty tietovarasto* (enterprise data warehouse, EDW, centralized architecture, centralized DW architecture, centralized data warehouse with no dependent data marts, centralized data warehouse)
- *Keskiö ja puolat* (hub and spoke, joskus myös corporate information factory tai dw2.0 [ArW10])
- *Liitosarkkitehtuuri* (federated architecture)

Yllä olevat arkkitehtuurit ovat referenssiarkkitehtuureja (reference architectures), jotka kuvaavat tietovaraston arkkitehtuuriratkaisun yleisellä tasolla, ja käytännön toteutuksessa esiintyy niistä erilaisia muunnelmia [ArW10]. Arkkitehtuurit luokittelevat tietovarastoratkaisut sen mukaan, mitä komponentteja tietovarastoratkaisu sisältää, miten komponentit on järjestetty suhteessa toisiinsa, miten eri lähteistä saatava tieto yhdistetään yritystasolla ja millaisia tietomalleja niissä yleensä käytetään. Referenssiarkkitehtuurin pohjalta toteutettavaan konkreettiseen arkkitehtuu-

riin liittyy monia yksityiskohtia, joihin referenssiarkkitehtuuri ei suoraan ota kantaa, esimerkiksi kuinka metadatan hallinta järjestetään, kuinka eri ikäistä tietoa hallitaan, miten ETL-järjestelmä toteutetaan ja miten strukturoitua ja strukturoimatonta tietoa käsitellään. Arkkitehtuurien kanssa yleisimmin käytetyt tietomallit on kuvattu taulukossa 3.

	Normalisoitu tietomalli	Moniulotteinen tietomalli	Tietoholvi-malli
Itsenäiset paikallisvarastot		X	
Yhtenäistetyt paikallisvarastot		X	
Yritystason keskitetty tietovarasto	X		
Keskiö ja puolat	X	X	X
Liitosarkkitehtuuri	Ei liity erityisesti tiettyyn tietomalliin		

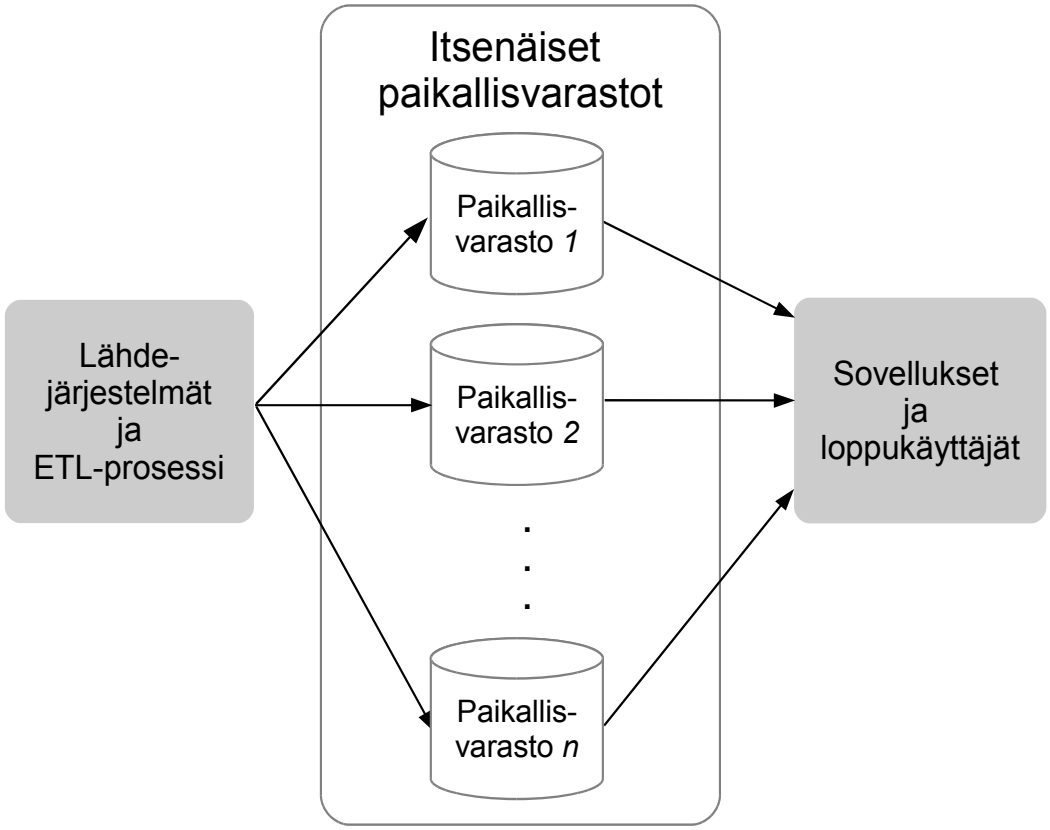
Taulukko 3: Arkkitehtuurien ja tietomallien suhde.

5.2.1 Itsenäiset paikallisvarastot

Itsenäisiä paikallisvarastoja käyttävässä arkkitehtuurissa (kuva 19) joukko pienehköjä, suppean aihealueen tietovarastoja rakennetaan toisistaan riippumatta ilman kokonaissuunnittelua. Kukin paikallisvarasto rakennetaan rajoitetun käyttäjäjoukon, esimerkiksi yrityksen yhden osaston tarpeisiin. Yksittäiset paikallisvarastot ovat nopeita rakentaa, mutta puuttuva kokonaissuunnittelu johtaa helposti ongelmiin tiedon tulkinnassa ja ylläpidossa, minkä vuoksi itsenäiset paikallisvarastot ei ole yleisesti suositeltu arkkitehtuuri [ArW06][ArW10]. Kyseessä on pikemminkin ratkaisu, joka syntyy sen seurauksena, että arkkitehtuurisuunnitelmaa ei ole. Tietomallina itsenäiset paikallisvarastot voivat käyttää paikallisvaraston tarkoituksiin parhaiten sopivaa mallia, esimerkiksi jotain moniulotteista mallia [ArW08].

Koska paikallisvaraston aihealue ja laajuus on etukäteen selvästi rajattu, itsenäinen paikallisvarasto on helppo, nopea ja edullinen rakentaa verrattuna yhtenäisesti suunniteltuun, koko yrityksen laajuiseen tietovarastoon. Tämän ansiosta tietovarastosta saadaan nopeasti tuloksia loppukäyttäjille eikä aikaa ja resursseja kulu väsyttävään ja raskaaseen projektiin, jossa myös riskit olisivat huomattavasti suuremmat.

Vaikka yksittäisen itsenäisen paikallisvaraston toteutuksen helppous saattaa tehdä siitä houkuttelevan vaihtoehdon, sisältyy ratkaisuun myös niin paljon heikkouksia, että yleensä ainakaan kookkaimpien yritysten ei kannata mallia käyttää. Suurin ongelma on *yhteisen totuuden* (single version of truth) puuttuminen. Yhteisellä totuudella tarkoitetaan, että eri lähteistä saatu tieto on yhtenäistä, se on käsitelty samoilla muunnossäännöillä, käyttää samoja tunnisteita kuvaamaan samoja asioita, laskee johdetut tiedot samalla tavalla eikä sisällä tietojen tarpeetonta toisteisuutta. Itsenäiset paikallisvarastot rakennetaan määritelmän mukaan toisistaan riippumatta, joten



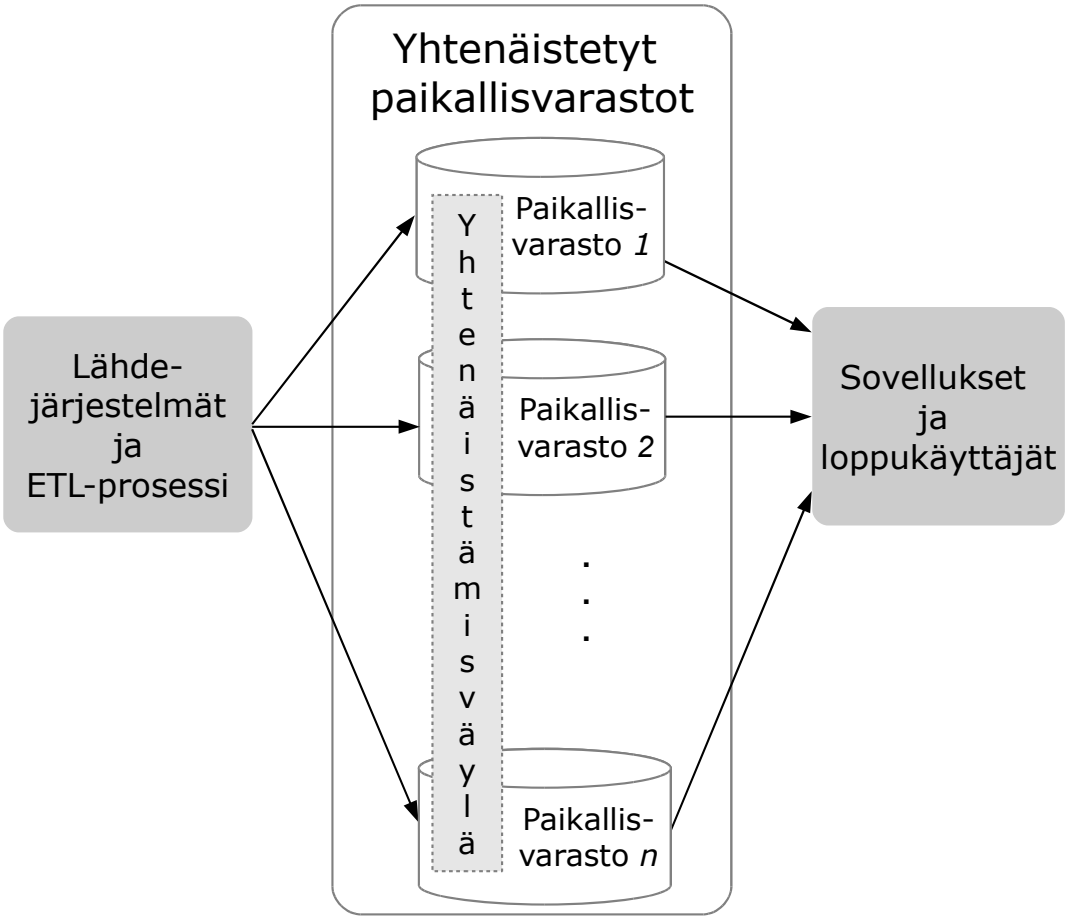
Kuva 19: Itsenäiset paikallisvarastot – arkkitehtuuri.

yhteistä totuutta ei voi syntyä. Koska liiketoimintatiedot ovat erillään toisistaan ja niiden kuvaus ja esitysmuoto on epäyhtenäinen on paikallisvarastojen välinen tietojen yhdistäminen ja analysointi vaikeaa. Paikallisvarastot sisältävät myös helposti päällekkäistä tietoa, mikä aiheuttaa päällekkäistä prosessointia sekä ylimääräisiä tallennus-, laitteisto- ja ylläpitokustannuksia. Tämän vuoksi ratkaisusta saattaakin tulla helpon ja edullisen sijasta vaikea ja kallis, jos paikallisvarastoja on useita ja niiden tietoja halutaan käsitellä yhtenäisesti.

5.2.2 Yhtenäistetyt paikallisvarastot

Yhtenäistetyissä paikallisvarastoissa (kuva 20) tietovaraston toteutus aloitetaan myös yksittäisestä paikallisvarastosta, kuten itsenäisten paikallisvarastojenkin tapauksessa. Ennen paikallisvarastojen toteutusta on kuitenkin tehty suunnittelutyötä eri paikallisvarastojen välisek-si tietojen yhdistämiseksi ja yrityslaajuisen yhtenäisen näkymän tarjoamiseksi. Yhtenäistetyissä paikallisvarastoissa sovelletaan siis *kokoavaa* (bottom-up) toteutusta ja *osittavaa* (top-down) suunnittelua.

Yhtenäistetyt paikallisvarastot käyttää moniulotteista tietomallia, yleensä tähtimallia, ja paikallisvarastojen yhtenäisyys pyritään saavuttamaan moniulotteisen tietomallin *yhdenmukaistetuilla*



Kuva 20: Yhtenäistetyt paikallisvarastot – arkkitehtuuri.

ulottuvuuksilla (conformed dimensions) [KRT08 s. 248-253]. Suunnitteluvaiheessa pyritään tunnistamaan paikallisvarastoille ja prosesseille yhteiset ulottuvuudet. Esimerkiksi tuotetiedot-ulottuvuus on oltava yhtenäinen laskutusprosessin paikallisvarastolle ja myynnin paikallisvarastolle. Eri paikallisvarastojen yhdenmukaistamista vaativat ulottuvuudet voivat olla identtiset, jolloin ne sisältävät täsmälleen samat attribuutit tai ulottuvuus voi olla toisen osajoukko, jolloin laajempi ulottuvuus on tarkemman tason tietoa ja sisältää suppeaa ulottuvuutta enemmän attribuutteja [KRT08 s. 244]. Kun paikallisvarastoille yhteiset ulottuvuudet ovat yhtenäisessä muodossa, ne muodostavat yritystasoisen *yhtenäistämisväylän* (enterprise data warehouse bus), jonka avulla eri paikallisvarastojen tietoja voidaan yhdistää toisiinsa.

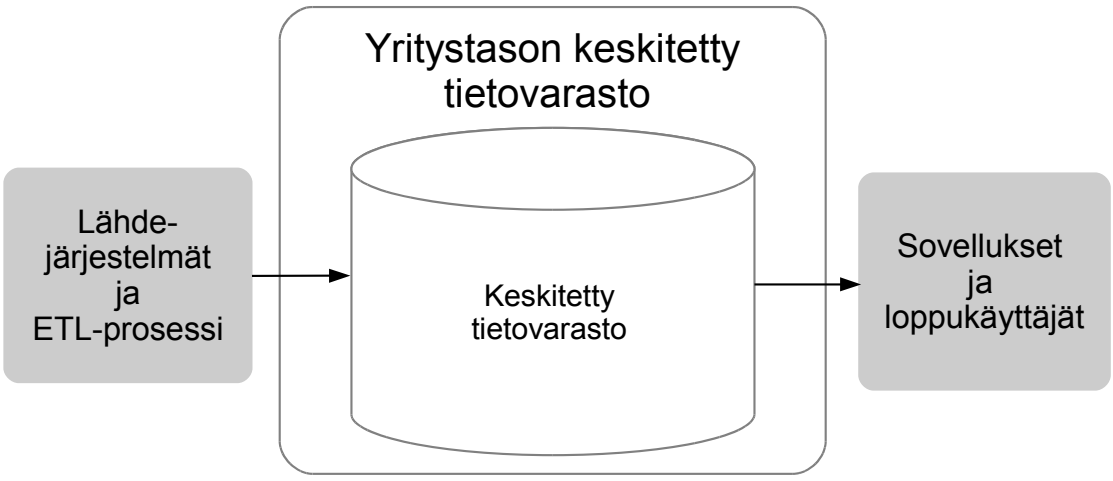
Kuten itsenäisissä paikallisvarastoissa, myös yhtenäistetyissä paikallisvarastoissa yksittäinen varasto on nopea, helppo ja edullinen rakentaa. Loppukäyttäjälle pystytään siis tarjoamaan nopeasti konkreettisia tuloksia, mutta yritystasoisen osittavan suunnittelun ja yhtenäistämisväylän ansiosta paikallisvarastojen tietoja voidaan myös yhdistää toisiinsa. Arkkitehtuuriratkaisu on yksinkertainen ja selkeä, mutta rajoittuu moniulotteisen tietomallin käyttöön[ArW08]. Tätä voidaan pitää rajoituksena, mutta esimerkiksi moniulotteinen tähtimalli on laajasti tunnettu ja eri sidosryhmät ymmärtävät sen helposti [KRT08 s. 233-286]. Lisäksi monet tietovarastoinnissa

käytettävät ETL-ohjelmistot, tietokannan hallintajärjestelmät ja liiketoimintatiedon hallintasovellukset osaavat käyttää tähtimallia tehokkaasti hyödykseen ja optimoida siihen kohdistuvia operaatioita [KRT08 s. 238].

Vaikka yhtenäistämisväylän avulla periaatteessa voidaan luoda yhtenäistetty näkymä yrityksen liiketoimintatietoon, rajoittuu yhtenäistäminen kuitenkin ainoastaan yhdenmukaistettuihin ulottuvuuksiin. Paikallisvarastojen välille saattaa edelleenkin syntyä päällekkäisyyttä ja epäyhtenäisyyttä yhdenmukaistettujen ulottuvuuksien ulkopuolelle. Yritystasoinen kokonaissuunnittelu vaatii myös työtä ja riskinä on, että se jää puutteelliseksi tai unohtuu kokonaan. Esimerkiksi ylläpitovaiheessa saattaa käydä niin, että paikallisvarastoon tuodaan uutta tietoa, mutta sen yhtenäistäminen unohtuu.

5.2.3 Yritystason keskitetty tietovarasto

Yritystason keskitetty tietovarasto (kuva 21) on suoraviivainen keino saavuttaa yhteinen totuus tietovarastossa. Tässä tietovarastojen arkkitehtuurivaihtoehdossa pidetään kaikki tietovaraston tieto yhdessä, yhdenmukaisessa ja *keskitetyssä tietovarastossa (*enterprise data warehouse, EDW, atomic/data warehouse, global data warehouse, primary data warehouse, corporate data warehouse, warehouse tai vain data warehouse) [ISN08 s. 12][JLV03 s. 3][Kim02][ChD97] [Inm02 s. 16]. Tavoitteena yritystason keskitetyssä tietovarastossa on välttää toisteisuudesta ja hajauttamisesta aiheutuva liiketoimintatiedon epäyhtenäisyys.



Kuva 21: Yritystason keskitetty tietovarasto -arkkitehtuuri.

Keskitetty tietovarasto tarjoaa liiketoimintatietoa yhdestä varastointijärjestelmästä useille eri käyttäjäryhmille kunkin käyttäjäryhmän tarvitsemassa muodossa. Tämän vuoksi tieto kannattaa tallentaa varastointijärjestelmään sellaisessa muodossa, joka taipuu mahdollisimman joustavasti erilaisiin näkökulmiin. Keskitetty järjestelmä mahdollistaa kaikille käyttäjäryhmille yhteisen totuuden, mutta vaarana on, että tietovaraston toteuttamisvaiheessa yritetään toteuttaa kaikkien

käyttäjäryhmien kaikki vaatimukset yhdellä kertaa, mikä johtaa suureen ja työlääseen projektiin, jossa mahdollinen epäonnistumisen riski on myös suuri [Inm02 s. 277]. Tätä voidaan välttää iteratiivisella kehityksellä, jossa tietovarastoa rakennetaan asteittain [Inm02 s. 102].

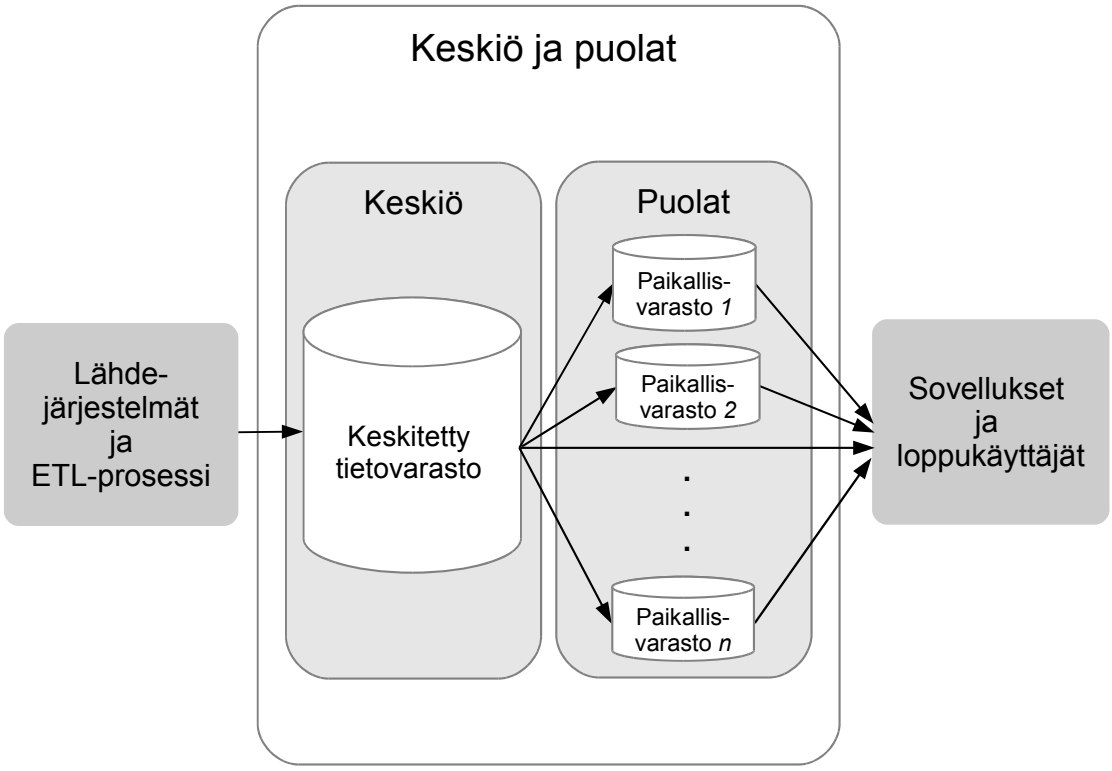
Keskitetyssä tietovarastossa säilytettävä yksityiskohtainen tieto on yleensä normalisoidussa muodossa, tarkemmin sanottuna kolmannessa normaalimuodossa [ArW10]. Vaatimus tiedon tallentamisesta kolmannessa normaalimuodossa lähtee ajatuksesta, että tietovarastot ovat tavallisia tietokantajärjestelmiä ja niiden tulee noudattaa perinteistä tietokantojen suunnittelutapaa, jota käytetään myös operatiivisissa järjestelmissä [Mar04]. Kolmas normaalimuoto ehkäisee tehokkaasti erilaiset anomaliat ja toisteisuuden sekä tukee tiedon pysymistä eheänä. Normalisoidun tietokannan etuna on myös, että se ei rajoitu mihinkään tiettyyn näkökulmaan [Inm02 s. 137].

Heikkoutena normalisoidussa tietomallissa on, että se sisältää yleensä paljon tauluja. Tämä aiheuttaa kyselyissä suuren määrän liitoksia, mikä erityisesti tietovarastoille tyypillisissä monimutkaisissa ja laajoissa, käyttäjän määrittelemissä satunnaisissa kyselyissä heikentää suorituskykyä. Suuri taulujen määrä tekee tietokannasta myös vaikeasti hahmottuvan ja vaikeasti ymmärrettävän. Monet liiketoimintatiedon hallintasovellukset on rakennettu ottamaan huomioon erityisesti moniulotteinen tietomalli ja tämä etu menetetään, jos tietovaraston tietokannan tiedot ovat ainoastaan kolmannessa normaalimuodossa [KRT08 s. 235-238]. Näiden syiden takia yritystason keskitetyssä tietovarastossakaan ei yleensä voida täysin välttää epänormalisointia, ja tietokanta sisältää normalisoidun tiedon lisäksi myös jonkin verran toisteisuutta ja koostettua tietoa [Inm02 s. 137]. Tietovarasto voi sisältää myös moniulotteisesti mallinnettua tietoa, joka on voitu toteuttaa esimerkiksi materialisoituina näkyminä ja joka johdetaan normalisoidusta tiedosta [Mar04]. Tämän vuoksi voidaan ajatella, että yritystason keskitetty tietovarasto toteuttaa luvussa 2.2.4 esitettävän keskiö ja puolat -arkkitehtuurin loogisella tasolla, mutta poikkeaa siitä fyysiseltä ratkaisultaan [WaA05].

5.2.4 Keskiö ja puolat -arkkitehtuuri

Keskiö ja puolat (hub and spoke) -arkkitehtuuri (kuva 22) on yhdistelmä yritystason keskitetystä tietovarastosta ja paikallisvarastoista. Keskiö ja puolat -arkkitehtuurin keskiöt eivät liity tietoholvimallin keskiö-käsitteeseen. Kuten yritystason keskitetyssä tietovarastossa, myös keskiö ja puolat -arkkitehtuurissa tietovaraston keskipisteenä toimii yksi keskitetty tietovarasto, joka sisältää lähdejärjestelmistä tuotettua tietoa yksityiskohtaisella tasolla sekä mahdollisesti jonkin verran koostettua tietoa [ArW06]. Keskitetyn tietovaraston lisäksi arkkitehtuurissa on kuitenkin myös paikallisvarastoja, jotka on johdettu keskitetystä tietovarastosta. Keskitetty tietovarasto muodostaa siis arkkitehtuurin *keskiön* ja siitä johdetut paikallisvarastot muodostavat arkkiteh-

tuurin *puolat*. Keskiö ja puolat -arkkitehtuurissa pyritään myös yhteiseen totuuteen, mutta toisin kuin yhtenäistetyissä paikallisvarastoissa, erillistä yhtenäistämisväylää ei tarvita, koska paikallisvarastojen tieto johdetaan kaikille yhteisestä keskitetystä tietovarastosta. Paikallisvarastot voidaan toteuttaa tähtimallilla tai jollain muulla sopivalla mallilla ja niiden tehtävä on palvella loppukäyttäjää jostain tiedon hyödyntämisen näkökulmasta. Paikallisvarastojen lisäksi loppukäyttäjällä on käytössään myös keskitetyn tietovaraston yksityiskohtainen raakatieto, johon he voivat halutessaan kohdistaa tiedonlouhintaoperaatioita tai satunnaisia (ad hoc) kyselyitä.



Kuva 22: Keskiö ja puolat -arkkitehtuuri.

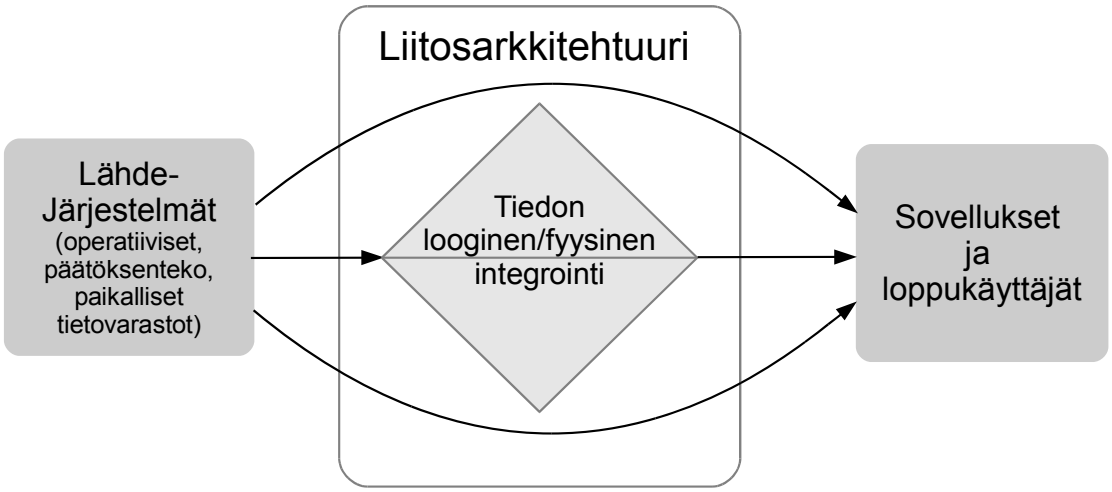
Keskiö ja puolat -arkkitehtuuri yhdistää paikallisvarastojen ja yritystason keskitetyn tietovaraston vahvuudet. Keskiökomponentti tarjoaa yhteisen totuuden ja sen sisältämät tiedot voidaan tallentaa normalisoidussa muodossa, mikä ehkäisee anomalioita, epäyhtenäisyyttä ja toisteisuutta. Keskiön tietomallin ei tarvitse rajoittua mihinkään näkökulmaan, mutta siitä johdettu paikallisvarasto voidaan toteuttaa mistä tahansa näkökulmasta. Näkökulman mukaan voidaan valita, mitä liiketoimintatietoa keskiöstä johdettuun paikallisvarastoon tuodaan sekä miten se tallennetaan ja esitetään. Kun keskitetty tietovarasto on käytettävissä, on siitä johdettu yksittäinen paikallisvarasto suoraviivaista toteuttaa.

Keskiö ja puolat -arkkitehtuurissa on kuitenkin toteutettava sekä keskitetty tietovarasto (keskiö) että paikallisvarastot (puolat) ja näitä on myös ylläpidettävä. Tämä tekee ratkaisusta yleensä kal-

liimman ja työläämmän toteuttaa kuin muista arkkitehtuurivaihtoehdoista [WaA05]. Arkkitehtuurissa on myös enemmän komponentteja kuin muissa arkkitehtuurivaihtoehdoissa, joten se on myös jossain määrin vaikeatajuisempi.

5.2.5 Liitosarkkitehtuuri

Liitosarkkitehtuuri (federated architecture, kuva 23) tulee kyseeseen tilanteessa, jossa organisaatio ja järjestelmät ovat voimakkaasti hajautuneet ja eriytyneet. Eri osastoilla on tällöin jo käytössään eri tavoin toteutettuja päätöksenteon tukijärjestelmiä. Nämä voivat olla esimerkiksi operatiivisten järjestelmien lisäkomponentteja tai osastokohtaisia paikallisvarastoja. Integrointityön vaativuuden tai yrityksen sisäisen rakenteen vuoksi tällaisessa tilanteessa voidaan päättää luopua yhtenäisen tietovarastoratkaisun rakentamisesta [ArW10]. Sen sijaan jätetään olemassa olevat päätöksenteon tukijärjestelmät entiselleen ja yhdistetään niiden tärkeimmät ja integrointia vaativat tiedot loogisesti tai fyysisesti toisiinsa, yrittämättä kuitenkaan tarjota kaikkia analysointitoimintoja keskitetysti [JiA04]. Yhdistämisessä voidaan käyttää hyväksi esimerkiksi yhteisiä avaimia, metatietoa ja hajautettuja kyselyitä.



Kuva 23: Liitosarkkitehtuuri.

Liitosarkkitehtuurissa voidaan käyttää hyväksi jo olemassa olevia päätöksenteon järjestelmiä ja parhaimmassa tapauksessa nämä myös täydentävät hyvin toisiaan. Arkkitehtuuriratkaisulla voidaan siis periaatteessa toteuttaa tietovarasto nopeasti ja helposti [JiA04]. Käytännön kokemukset osoittavat kuitenkin ratkaisun toteutuksen ja ylläpidon melko kalliiksi [WaA05]. Liitosarkkitehtuurin käyttämät lähdejärjestelmät voi olla toteutettu täysin erilaisilla tekniikoilla, mikä voi tehdä ratkaisusta teknisesti vaikean. Ongelmia voi aiheuttaa myös suorituskyky, koska loogisen tietovaraston toiminta on riippuvainen lähdejärjestelmien toiminnasta. Tietoa ei myöskään arkistoida tietovarastoon, joten historiatietoa on saatavilla vain sillä tasolla kuin se on talletettu lähdejärjestelmiin, eivätkä kyselyt tämän vuoksi myöskään palauta aina samaa loppu-

tulosta [ISN08 s. 16-17]. Esimerkiksi kuukausittaisen myynnin määrän hakeva kysely voi palauttaa joka minuutti eri tuloksen. Toisaalta ratkaisu on myös melko harvinainen, eikä sen yksityiskohtia ole määritelty yhtä hyvin kuin muissa arkkitehtuureissa [ArW06]. Esimerkkiratkaisuja ei siis välttämättä löydy tarpeeksi lopullisten johtopäätösten tekemiseen. Tähän mennessä saadut käytännön kokemukset eivät kuitenkaan vaikuta erityisen rohkaisevilta [WaA05].

6 Tietovarastoinnin kehittämismenetelmät

Arkkitehtuurien lisäksi tietovarastojen toteutukseen on esitetty erilaisia kehittämismenetelmiä (methodology). Joskus menetelmä saattaa liittyä johonkin tiettyyn arkkitehtuuriin, minkä vuoksi arkkitehtuurin ja menetelmän käsitteitä on saatettu käyttää tietovarastoinnin yhteydessä harhaanjohtavasti sekaisin [WaA05]. Esimerkiksi keskiö ja puolat -arkkitehtuuriin viitataan joskus osittavana (top-down) menetelmänä. Menetelmä ei kuitenkaan ole sama asia kuin arkkitehtuuri, vaan menetelmä on joukko tietovaraston toteuttamiseen liittyviä ohjeita ja toimintatapoja. Menetelmän ja arkkitehtuurin tulee kuitenkin olla yhteensopivia. Osa menetelmistä pyrkii ratkaisemaan vain jonkin tietyn osa-alueen ongelman, esimerkiksi lähdejärjestelmistä koottavien tietojen moniulotteiseen mallinnukseen on esitetty lukuisia erilaisia menetelmiä [RoA09].

Tietovarastojen käyttötarkoitus on toimia liiketoimintatiedon hallinnan ja päätöksenteon tukena. Päätöksenteon tukijärjestelmien historia juontaa informaatiojärjestelmien alkuaikoihin [Inm02 s. 2]. Laajemmin termin tietovarasto otti vuonna 1992 käyttöön W. H. Inmon [Inm92], jota on myös kutsuttu tietovarastoinnin isäksi. Inmonin lähestymistapa tietovaraston rakentamiseen oli luoda osittavaan (top down) suunnitteluun ja toteutukseen perustuva *tietolähtöinen menetelmä* (data driven). Menetelmän ajatuksena on toteuttaa yrityksen eri paikoissa sijaitseviin tietoihin yhtenäinen ja kattava näkymä, joka palvelee mahdollisimman monipuolisia käyttötarkoituksia.

Moniulotteinen tietomalli on osoittautunut tietovarastoissa erittäin käyttökelpoiseksi, mutta Inmonin kirjan ensimmäisessä versiossa siihen ei kiinnitetty lainkaan huomiota. Moniulotteisista tietomalleista tähtimalli on yleisin [KRT08 s. 338] ja sen merkitystä korosti R. Kimball vuonna 1996 [Kim96]. Kimball kannattaa tietolähtöisestä menetelmästä poikkeavaa menetelmää, joka on luonteeltaan prosessikeskeinen (process oriented). Prosessikeskeisessä menetelmässä toteutustapa on luonteeltaan kokoava (bottom-up), vastakohtana tietolähtöisen menetelmän osittavalle toteutukselle. Ajatuksena on tehdä tietovaraston rakentamisesta ketterämpää ja saada nopeammin tuloksia pienikokoisten, moniulotteiseen tähtimalliin perustuvien paikallisvarastojen (data mart) avulla.

Keskitettyä tietovarastoa käyttävä tietolähtöinen menetelmä (jatkossa vain *tietolähtöinen menetelmä*) ja tähtimallinnettuja paikallisvarastoja käyttävä prosessikeskeinen menetelmä (jatkossa *prosessikeskeinen menetelmä*) käyttävät erilaista arkkitehtuuriratkaisua ja molempien mentelmien ympärillä on kehittynyt oma ryhmittymänsä, joiden välillä on käyty paljon keskustelua menetelmien ja niissä käytettävien arkkitehtuurien ja tietomallien tarkoituksenmukaisuudesta [Bre04][Inm00][Inm02][ISN08][Juk06][Kim97][Kim02][KiR02][KRT08]. Ryhmien näkemykset ovat vuosien mittaan jonkin verran lähentyneet [ArW06]. Esimerkiksi Inmonin ryhmittymä

on omassa menetelmässään ottanut käyttöön moniulotteiseen tähtimalliin perustuvat paikallisvarastot (departmental level, data mart level, OLAP level, multidimensional DBMS level) kuvaamaan keskitetystä tietovarastosta johdettuja osastokohtaisia tietovarastoja [Inm02 s. 17]. Kimballin ryhmä taas on painottanut yrityksen yhteisen tiedon (master data) hallinnan [Tho07] ja paikallisvarastojen yhtenäisen suunnittelun tärkeyttä [KRT08]. Molemmissa menetelmissä painotetaan nopeiden tulosten saamista, mutta myös pitkäaikaisen suunnittelun tärkeyttä. Lähtökohtiensa erilaisuuden vuoksi jako menetelmien välillä on edelleen kuitenkin kohtalaisen selkeä.

Sekä tietolähtöisessä menetelmässä että prosessikeskeisessä menetelmässä on omat puutteensa ja käytännössä niistä esiintyy myös monia muunnelmia ja hybridejä. Haasteita aiheuttaa erityisesti tietomallissa tapahtuvat muutokset sekä tietojen historiointi. Näitä ongelmia on pyritty ratkomaan *tietoholvimenetelmässä* (data vault methodology), joka on D. Linstedtin vuonna 2000 julkaisemaa tietoholvimallia käyttävä menetelmä [Lin11a]. Pyrkimyksenä menetelmässä on yhdistää parhaat puolet normalisoidusta tietomallista ja tähtimallista tietoholvimalliin ja saavuttaa sen avulla mahdollisimman hyvä muutosjoustavuus, laajennettavuus, rinnakkaisprosessoinnin tuki sekä tietojen täydellinen historiointi.

Tässä luvussa luomme yleiskatsauksen tietolähtöiseen menetelmään, prosessikeskeiseen menetelmään ja tietoholvimenetelmään. Osittain menetelmät ovat päällekkäisiä tai täydentävät toisiaan, mutta joiltakin osin ne ovat myös ristiriidassa toistensa kanssa. Käsittelemme lyhyesti kunkin menetelmän käyttämiä lähtöoletuksia, käytettyjä tekniikoita sekä suositeltua tietomallia ja arkkitehtuuria. Taulukossa 4 näkyy koostettuna, mitä tietomalleja menetelmissä yleensä käytetään ja taulukossa 5 on kuvattu, mitä arkkitehtuuriratkaisuja menetelmien kanssa suositellaan käytettäväksi.

	Normalisoitu tietomalli	Moniulotteinen tietomalli	Tietoholvimalli
Tietolähtöinen menetelmä	X	X	
Prosessikeskeinen menetelmä		X	
Tietoholvimenetelmä		X	X

Taulukko 4: Menetelmien ja tietomallien suhde.

	Keskiö ja puolat	Yhtenäistetyt paikallisvarastot	Yritystason keskitetty tietovarasto
Tietolähtöinen menetelmä	X		X
Prosessikeskeinen menetelmä		X	
Tietoholvimenetelmä	X		

Taulukko 5: Menetelmien ja arkkitehtuurien suhde.

6.1 Tietolähtöinen menetelmä

Tietolähtöinen (data driven) tietovarastojen kehittämismenetelmä on esitelty teoksessa *Building the Data Warehouse* [Inm02] ja se pyrkii keskitettyyn tiedonhallintaan sekä tietovaraston rakentamiseen asteittaisesti, olemassa olevien tietojen ja sovellusten pohjalta. Ensimmäisessä vaiheessa operatiivisista järjestelmistä tuotetaan pohjatiedot, minkä jälkeen tietovarastoa kehitetään ja sen tietomäärää laajennetaan askel kerrallaan. Menetelmä on iteratiivinen ja se poikkeaa tyypillisestä operatiivisten järjestelmien kehittämismenetelmistä, jossa kehittäminen alkaa yleensä vaatimusten keräämisestä ja päätyy valmiiseen ohjelmakoodiin. Tietolähtöinen tietovaraston kehitysmenetelmä sen sijaan alkaa tiedon keräämisellä ja oikeastaan vasta päätyy vaatimuksiin [Inm02 s. 294]. Lähtökohtana on siis tieto, eivät vaatimukset. Vaatimusten keräystä projektin tai laajan tietovarastohankkeen alkuvaiheessa ei tule kokonaan jättää pois, mutta lähtöoletuksena pidetään kuitenkin sitä, että loppukäyttäjille näytetään ensin se, millaista tietoa tietovarastoon voidaan tuottaa ja vasta tämän jälkeen he voivat tietää, mitä he järjestelmältä todella haluavat [Inm02 s. 19]. Tietovaraston vaatimuksia ei siis tunneta kunnolla, ennen kuin tieto on ainakin osittain saatavilla.

Arkkitehtuuriratkaisuna tietolähtöinen menetelmä käyttää keskiö ja puolat -arkkitehtuuria. *Keskiön* (keskitetty tietovarasto) muodostaa tietovaraston osa, jota menetelmässä kutsutaan nimellä yksityiskohtainen tietovarasto (atomic/data warehouse) ja se pitää sisällään operatiivisista tietojärjestelmistä integroidun tiedon hienojakoisessa muodossa. *Puolat* ovat osastokohtaisia kokonaisuuksia ja ne sisältävät pääasiassa johdettua ja koostettua, näkökulmaltaan rajoitettua tietoa. Hajautettu, paikallisista tietovarastoista (local data warehouses) koostuva tietovarasto esitellään menetelmässä myös yhtenä arkkitehtuurivaihtoehtona, mutta tarvetta täysin hajautettuun ratkaisuun pidetään harvinaisena ja tässä yhteydessä varoitetaan erityisesti toisteisesta tiedosta tietovarastojen välillä [Inm02 s. 201-245]. Myös yrityksen ulkopuolisen tiedon käsittely tietovarastossa käsitellään ja teknisestä ympäristöstä annetaan yksityiskohtaisia ohjeita.

Keskitetyn tietovaraston tietomalliksi tietolähtöisessä menetelmässä suositellaan normalisoitua tietomallia. Sen eduiksi nimetään joustavuus, sopivuus yksityiskohtaisen tiedon mallinnukseen ja näkökulmariippumattomuus [Inm02 s. 137]. Epänormalisointia voidaan suorittaa soveltuvin osin, esimerkiksi tehokkuussyistä. Epänormalisointitekniikoina voidaan käyttää esimerkiksi taulujen yhdistämistä, pilkkomista tai kopiointia [Inm02 s. 102-110]. Jos tietovaraston tietomäärä on suuri, täytyy tietoa tallentaa myös eri karkeisuustasoilla [Inm02 s. 359]. Esimerkiksi myyntitietoja voidaan tallentaa yksittäisen myynnin tasolla ja kuukausitasolla. Moniulotteista tähtimallia voidaan käyttää keskitetystä tietovarastosta johdetuissa näkymissä, mutta menetelmässä varoitetaan perustamasta koko tietovarastoa tähtimallin varaan [Inm02 s. 137-143, 146].

Yhtenä lähtöoletuksena tietolähtöisessä menetelmässä on, että yleensä yritykselle kannattaa rakentaa keskitetty tietovarasto, joka tarjoaa yhteisen totuuden yrityksen liiketoimintatiedosta. Menetelmä on luonteeltaan osittava (top-down), koska siinä pyritään ensin luomaan *yritystason tietomalli* (corporate data model) ja keskitetty tietovarasto, josta voidaan tuottaa suppeamman osa-alueen tietokokonaisuuksia. Yritystason tietomalli esittää yrityksen operatiivisen tiedon korkean tason kaaviona, joka toimii lähtökohtana tietovaraston rakentamiselle. Yritystason tietomallia tarkennetaan pilkkomalla se *tietovaraston tietomallissa* aihekohtaisiin kokonaisuuksiin, joissa tiedot kuvataan tarkemmalla tasolla [Inm02 s. 278]. Yritystason tietomallissa tai tietovaraston tietomallissa ei kuitenkaan esitetä karkeistettua tai johdettua tietoa.

Menetelmässä varoitetaan yrittämästä rakentaa kaiken kattava tietovarasto kerralla, sillä tämä saattaa johtaa ylisuureen ja vaikeasti hallittavaan projektiin, jossa epäonnistuminen on todennäköistä [Inm02 s. 277]. Sen sijaan iteratiivisuutta ja spiraalimaista kehitystä, jonka avulla saadaan tuloksia kohtuullisessa ajassa, korostetaan [Inm02 s. 102, 293]. Kun uusi tietokokonaisuus on valmis tuotavaksi tietovarastoon, järjestetään katselmointi, johon osallistuu joukko tietovaraston käyttöön ja kehittämiseen liittyviä avainhenkilöitä [Inm02 s. 321-342]. Katselmointi toteutetaan käyttämällä avuksi menetelmässä esitettyä suunnittelun tarkistuslistaa.

Teoksen [Inm02] lopussa on esitelty yksityiskohtainen menetelmäkäsikirja, joka listaa tietovaraston toteuttamisen eri vaiheet ja niiden toteuttamisjärjestyksen. Menetelmä jakaantuu kolmeen vaiheeseen. Ensimmäinen vaihe on operatiivisten järjestelmien toteuttaminen, toinen keskitetyn tietovaraston tietomallin suunnittelu sekä tietovaraston toteuttaminen ja kolmas tietovaraston liiketoimintatiedon käyttö ja tiedon analysointi. Kyseessä on siis varsin laaja menetelmä, jonka avulla voidaan hallita kaikkia yrityksen tietoja. Jokainen kolmesta vaiheesta jakaantuu edelleen osatehtäviin, joille annetaan aikatauluarviot, tavoitteet, suoritusmäärät (kerran tai useammin), toimitettavat tulokset ja erityistä huomiota vaativat asiat. Ensimmäinen vaihe, operatiivisen järjestelmän toteutus, jakaantuu 21 osatehtävään, joista ensimmäinen on projektin aloittaminen ja viimeinen on toteutus. Tämä vaihe noudattaa melko tarkasti perinteistä vesiputousmallia. Toinen vaihe, keskitetyn tietovaraston toteuttaminen, alkaa tiedon analysoinnista ja päättyy keskitetyn tietovaraston tietokannan perustamiseen. Tietovaraston rakentamiseen ei tule käyttää vesiputousmallia vaan iteratiivista spiraalimallia [Inm02 s. 293]. Kolmannessa vaiheessa keskitetystä tietovarastosta tuotetaan rajatun alueen analysoitua liiketoimintatietoa ja raportteja eri osastoille. Tässä vaiheessa analysoidaan käyttäjien vaatimuksia ja toteutetaan niiden pohjalta heille sopivia raportteja ja analyyseja. Tietovaraston toteutus on siis alkanut tiedon analysoinnista (vaihe 2) ja päätynyt käyttäjien vaatimusten analysointiin ja niiden toteuttamiseen (vaihe 3).

Vuonna 2008 Inmon, Strauss ja Neushloss julkaisivat uuden teoksen nimeltä *DW2.0 The architecture for the Next Generation of Data Warehousing* [ISN08]. Teoksessa on paljon samoja ajatuksia kuin alkuperäisessä *Building the Data Warehouse* -teoksessa [Inm92], mutta myös monia uusia toimintatapoja ja näkökulmia, esimerkiksi tietovaraston eri sektorit eri ikäiselle tiedolle, spiraalimainen kehitysmetodi ja seitsemän tason toimintamalli (seven streams approach). Vastaavanlaista yksityiskohtaista menetelmäkäsikirjaa, joka löytyy teoksesta *Building the Data Warehouse*, uudessa teoksessa ei kuitenkaan enää ole.

6.2 Prosessikeskeinen menetelmä

Prosessikeskeisessä menetelmässä yrityksen tietovarasto rakennetaan tähtimallinnettujen paikallisvarastojen avulla. Moniulotteinen tähtimalli sopii hyvin kuvaamaan rajatun aihealueen ja sen prosesseihin liittyvän liiketoimintatiedon. Menetelmä hylkää keskitetyn tietovaraston ja normalisoidun tietomallin tarpeettomina [Kim02][KRT08], sillä yhtenäistettyjen paikallisvarastojen kautta toteutettua tietovarastoa pidetään käytännöllisempänä ja helpompana toteuttaa. Yritystasoista suunnittelua ei hylätä, mutta siihen ei käytetä keskitettyä tietovarastoa, vaan paikallisvarastojen välistä *yhtenäistämisväylää* (enterprise data warehouse bus) [KRT08 s. 248-249]. Tavoitteena on kokoavan (bottom-up) toteutuksen avulla saada loppukäyttäjille mahdollisimman helposti ja nopeasti käyttökelpoisia tuloksia.

Prosessikeskeisen menetelmän käyttämä arkkitehtuuriratkaisu on yhtenäistetyt paikallisvarastot (data mart). Termistä *data mart* Kimballin ryhmittymä on kuitenkin ilmoittanut luopuvansa, koska käsite on heidän mielestään ”kaapattu” tarkoittamaan itsenäisiä paikallisvarastoja, ilman kokonaissuunnittelua [KRT08 s. 248]. Tässä tutkielmassa käytetään englanninkielisestä data mart -termistä nimitystä paikallisvarasto. Tarvittaessa ilmausta tarkennetaan käyttämällä termiä i*tsenäinen paikallisvarasto* viittamaan itsenäisiin paikallisvarastoihin ja termiä *yhtenäistetty paikallisvarasto* viittamaan paikallisvarastoihin, jotka yhdessä muodostavat yritystasoisen yhtenäisen tietovaraston.

Prosessikeskeisessä menetelmässä tietovarasto rakennetaan alhaalta ylöspäin, paikallisvarasto kerrallaan, ottaen suunnittelussa kuitenkin huomioon myös kokonaisuus. Tällöin lopputuloksena on yritystason tietovarasto, aivan kuten tietolähtöisessäkin menetelmässä, mutta arkkitehtuuri ja toimintatavat vain ovat toisenlaiset. Prosessikeskeisen menetelmän arkkitehtuuri koostuu erillisistä paikallisvarastoista, mutta paikallisvarastojen yhtenäistämiseksi tarvitaan kokonaisnäkemykseen perustuvaa suunnittelua. Yhtenäistäminen saavutetaan niin sanotuilla *yhdenmukaistetuilla ulottuvuuksilla* (conformed dimensions). Yhdenmukaistetut ulottuvuudet muodostavat paikallisvarastojen läpi ulottuvan *yhtenäistämisväylän*, joka yhdistää eri osastojen ja liiketoimintaprosessien näkökulmat toisiinsa.

Tähtimalli on prosessikeskeisessä menetelmässä keskeisessä asemassa. Tähtimalli on helposti ymmärrettävä, jopa loppukäyttäjille, ja se mallintaa tiedon moniulotteisesti, mikä on OLAP-järjestelmille tärkeä vaatimus [CCS93]. Tähtimalli on myös tehokas tietokantakyselyissä, mikä seuraa muun muassa epänormalisoinnista, pienestä taulujen määrästä, ulottuvuustaulujen rajaavuudesta ja ohjelmistojen optimoinnista tähtimallille [KRT08 s. 237-238]. Tähtimallin kiistattomien etujen vuoksi se on laajassa käytössä ja sitä on tutkittu paljon. Prosessilähtöisessä menetelmässä tietomalli perustuu lähes pelkästään tähtimalliin. Ajatus kolmannessa normaalimuodossa olevasta keskitetystä tietovarastosta, joka ei saisi perustua moniulotteiseen tietomalliin ja josta paikallisvarastot pitäisi johtaa, on Kimballin ryhmittymän mielestä kuitenkin epäkäytännöllinen [KRT08 s. 10-11].

Prosessikeskeisen menetelmän toimintatapoja kuvattiin ensimmäisen kerran laajasti vuonna 1996 teoksessa *The data warehouse toolkit: practical techniques for building dimensional data warehouses* [Kim96]. Myöhemmin teosta on päivitetty ja Kimballin ryhmittymä on julkaissut myös useita kirjaa täydentäviä teoksia, jotka keskittyvät tietovaraston elinkaaren hallintaan, web-analytiikkaan ja ETL-prosessiin [KiC04][KiM00][KiR02][KRT08]. Kirjat ovat osittain päällekkäisiä ja yhdessä ne muodostavat laajan ja yksityiskohtaisesti kuvatun joukon suosituksia ja toimintatapoja, joita tässä tutkielmassa kutsutaan prosessikeskeiseksi menetelmäksi.

Menetelmää voidaan prosessikeskeisyyden lisäksi nimittää myös vaatimuslähtöiseksi [RoA09], sillä tietovaraston rakentaminen aloitetaan liiketoimintaprosessien tunnistamisella ja tietovarastolle asetettujen vaatimusten keräämisellä. Vaatimusten kartoitukseen annetaan tarkkoja ohjeita ja menetelmiä. Myös kaikkiin muihin tietovaraston rakentamisen vaiheisiin annetaan yksityiskohtaisia ja käytännöllisiä neuvoja. Esimerkit ovat tärkeässä asemassa ja ne kattavat useita liiketoiminnan osa-alueita kuten hankinta, varastointi, jälleenmyynti, tilausten hallinta, asiakashallinta, laskutus, henkilöstöhallinta, rahoitusala, terveydenhuolto ja vakuutusala. Tähtimallin merkitystä painotetaan ja sen mahdollisiin rajoituksiin liittyvä kritiikki torjutaan vasta-argumentein [KRT08 s. 282-286]. Myös MOLAP-palvelimien käyttämät kuutiorakenteet suositellaan tuotettavaksi tähtimallinnetusta relaatiotietokannasta [KRT08 s. 235, 349].

Prosessikeskeisen menetelmän toimintatavat ja parhaiksi havaitut käytännöt on kuvattu erittäin laajasti ja yksityiskohtaisesti Kimballin ryhmän teoksissa. Yksityiskohtia ei käydä tässä tutkielmassa tarkemmin läpi, mutta karkealla tasolla tietovaraston elinkaaren työvaiheet menetelmässä ovat projektin suunnittelu, liiketoimintavaatimusten määrittely, teknisten ratkaisujen suunnittelu ja toteutus, tiedon mallinnus ja fyysisen tallennuksen suunnittelu, ETL-prosessin suunnittelu ja toteutus, liiketoimintatiedon hallintasovellusten valinta, tietovaraston käyttöönotto ja tietovaraston ylläpito sekä jatkokehittäminen [KRT08].

6.3 Tietoholvimenetelmä

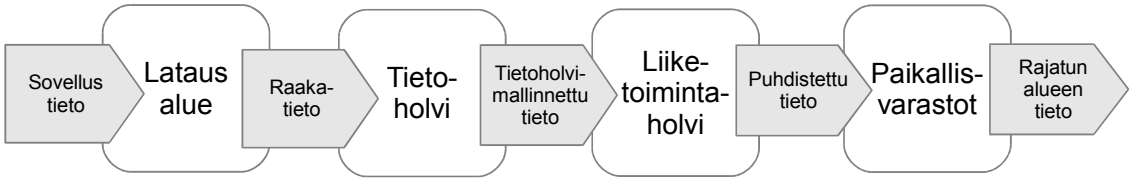
Kolmas tässä tutkielmassa esitelty tietovaraston kehittamismenetelmä on niin sanottu tietoholvimenetelmä (data vault methodology), jossa pyritään toteuttamaan tietovarasto, joka on mahdollisimman joustava ja laajennettava [Lin11a s. 9]. Nämä ominaisuudet saavutetaan ennen kaikkea menetelmää varten kehitellyllä tietomallilla, joka on erityisesti tietovarastointia varten suunniteltu tietomalli [Lin02]. Tietoholvimenetelmä noudattaa pääosin tietolähtöisen menetelmän periaatteita ja sen voikin nähdä tietolähtöisen menetelmän yhtenä toteutustapana tai täydennyksenä siihen. Tietoholvimenetelmää on esitelty muun muassa teoksessa [Lin11a].

Arkkitehtuuriratkaisuna tietoholvimenetelmä käyttää keskiö ja puolat -arkkitehtuuria [Lin11a s. 19-20]. Keskiössä sijaitsevan keskitetyn tietovaraston muodostaa tietoholvikomponentti (EDW, data vault), joka pitää sisällään operatiivisista järjestelmistä tuodun yksityiskohtaisen raakatiedon (raw data sets) tietoholvimallin mukaisessa muodossa [Lin11a s. 20]. Keskitetyn tietoholvin tiedosta tuotetaan paikallisvarastot, jotka voivat olla mallinnettu tähtimallin tai jonkin muun mallin mukaan [Lin11a s. 21]. Keskitetty tietoholvi jakaantuu käsitteellisesti eri osiin, jotka sisältävät tietoholvimallin mukaisen yksityiskohtaisen raakatiedon lisäksi esimerkiksi metatietoa, koostettua tietoa ja reaaliaikaista tietoa. Erityisen tärkeä on raakatiedon ja paikallisvarastojen väliin sijoittuva *liiketoimintaholvi* (business vault), jossa sijaitseva tieto on muunnettu, puhdistettu ja käsitelty liiketoimintasäännöin (esimerkiksi yhdellä asiakkaalla voi olla vain yksi tili). Liiketoimintaholvi voi olla joukko tauluja muiden tietoholvimenetelmällä mallinnettujen taulujen joukossa, tai se voi olla täysin erillinen komponentti [Lin11a s. 21].

Tietoholvimenetelmän käyttämä tietoholvimalli koostuu keskiöistä, satelliiteista ja linkeistä ja näistä komponenteista koostuva tietomalli on menetelmän ydin. Kaikkiin tietoihin liitetään useita metatietoja kuten aikaleimoja ja lähdejärjestelmä, minkä avulla saavutetaan täydellinen historiointi ja jäljitettävyys [Lin11a s. 27-32]. Tietojen välillä ei ole myöskään kardinaalisuusrajoitteita, jolloin tietomalli joustaa hyvin muuttuviin vaatimuksiin ja yksilöjoukoille voidaan helposti liittää uusia ominaisuuksia [Lin11a s. 9-11]. Tietomalli on suunniteltu myös suorituskykymielessä tehokkaaksi ja se sopii Linstedtin mukaan hyvin hajautettuun prosessointiin [Lin11a s. 55-56].

Tietoholvimenetelmän suosittelemassa ETL-prosessissa käytetään latausaluetta, jonne tieto ladataan täsmälleen sellaisena kuin se tulee lähdejärjestelmistä. Tietoa ei kuitenkaan arkistoida latausalueella. Muunnokset (transform), jotka perustuvat liiketoimintasääntöihin, pyritään menetelmässä tekemään mahdollisimman myöhään, vasta sen jälkeen kun tieto on muunnettu tietoholvimallin mukaiseen muotoon. Liiketoimintasääntöjen soveltamisen jälkeen tiedot sijoitetaan *liiketoimintaholvi*-nimiseen kerrokseen. Myöhään tapahtuvalla liiketoimintasääntöjen

soveltamisella pyritään yksinkertaistamaan lähdejärjestelmistä tapahtuva latausprosessi, joka helposti paisuu suurikokoiseksi ja virhealttiiksi. Latausoperaatiot kehotetaan tekemään pieniksi ja lähdejärjestelmäkohtaisiksi. Tällöin niihin ei tarvitse tehdä muutoksia lisättäessä uusia lähdejärjestelmiä ja latausprosessit voidaan rinnakkaistaa [Lin11a s. 16-18]. Muunnoksen sijoittamisesta myöhempään vaiheeseen seuraa kuitenkin se, että osa tietoholvin sisältämästä tiedosta on virheellistä, koska tietoa ei ole vielä puhdistettu ja käsitelty liiketoimintasäännöin. Tiedon kulku tietovarastossa ja sen eri muunnokset tietoholvimenetelmässä on esitetty kuvassa 24.



Kuva 24: Tiedon kulku ja käsittely tietoholvimenetelmässä.

Tietoholvimenetelmä ei ole yhtä laajasti ja yksityiskohtaisesti kuvattu kokonaisuus kuin tietolähtöinen tai prosessikeskeinen kehittämismenetelmä eikä Linstedt ainakaan vielä ole saatavilla olevissa lähteissä kuvannut tietoholvimenetelmää tyhjentävästi [Lin10b]. Tietolähtöinen ja prosessikeskeinen menetelmä esimerkiksi sisältävät yksityiskohtaisia ohjeita projektinhallintaan ja tietovaraston toteutusvaiheiden aikatauluttamiseen mutta tietoholvimenetelmästä tällaisia ohjeita ei löydy. Yksityiskohtaisesti kuvatun menetelmän sijasta tietoholvimenetelmä onkin hajanaisempi joukko parhaaksi katsottuja toimintatapoja, joita ovat esimerkiksi tietoholvimallin käyttö, ETL-prosessin yksinkertaistaminen ja liiketoimintasääntöihin perustuvien muunnosten tekeminen mahdollisimman myöhäisessä vaiheessa. Tietoholvimenetelmän käyttämä tietomalli on kuitenkin kohtalaisen selkeästi määritelty ja sen eduiksi perustellaan laajennettavuus, muokattavuus, skaalautuvuus, tietojen tarkka jäljitettävyys ja mahdollisuus rinnakkaista lähdejärjestelmistä tehtävät lataukset tehokkaasti [Lin11a s. 9-16, 24-25].

7 Tietomallien, arkkitehtuurien ja menetelmien vertailua

Tässä luvussa vertaillaan ensin tietovarastoissa käytettyjä käsitemalleja, toiseksi loogisen tason malleja ja kolmanneksi tietovaraston eri arkkitehtuurivaihtoehtoja. Tietomallien ja arkkitehtuurien erityiset vahvuudet, heikkoudet ja käyttötarkoitukset pyritään selvittämään sekä tuomaan samalla esiin tutkielmassa esiin tulleita yleisiä huomioita ja johtopäätöksiä. Lopuksi vertaillaan myös lyhyesti luvussa 6 esiteltyjä kehittämismenetelmiä.

7.1 Käsitemallit

Tutkielmassa havaittiin, että tietovaraston käsitetason malliksi on ehdotettu lukuisia erilaisia vaihtoehtoja, joita ovat esimerkiksi yksilö-yhteys- eli ER-malli, erilaiset moniulotteiset mallit kuten CGMD-malli sekä käsitetason tietoholvimalliksi tarkoitettu C-DV-malli. Vaikka käsitetason malleja onkin suuri joukko ei sen tarvitse tarkoittaa, että jotkut niistä olisivat jotenkin virheellisiä tai mallintaisivat tietoa jotenkin väärin. Samaa tietoa voidaan mallintaa käsitetasolla usealla eri tavalla. Käsitetason tietomalli on siis riippuvainen valitusta näkökulmasta. Valittu näkökulma määrittää minkälaisiin osakokonaisuuksiin tietovaraston tietoa jaetaan ja millaiseen luokitteluun jako perustuu. Eri mallit luokittelevat tietoa erilaisten käsitteiden avulla ja käsitteiden valinnasta riippuu kuinka monipuolisesti tietoon liittyvät ominaisuudet näkyvät mallissa. Esimerkiksi moniulotteiset käsitemallit kuvaavat tiedon hierarkkiset rakenteet usein tarkasti ja monipuolisesti kun taas alkuperäisessä ER-mallissa kaikki yksilöjoukot ovat saman arvoisia.

Toinen tutkielmassa esille tullut asia liittyy käsitemallien toteutusriippumattomuuteen. Tässä tutkielmassa määriteltiin käsitetason mallin nimenomaan toteusriippumattomaksi tavaksi kuvata tietosisältöä ja toteutusriippumattomasti mallinnettu tieto pitäisi olla mahdollista toteuttaa joko relaatiotietokantana tai mahdollisesti johonkin toiseen rakenteeseen ja kyselytekniikkaan perustuvina tiedostoina [BCN92 s. 6]. Käytännössä käsitetason tietomalli ei kuitenkaan usein ole täysin toteutuksesta riippumaton. Esimerkiksi normalisoitua tietokantaa suunniteltaessa käsitetason mallinnus vaatii intuitiivista näkemystä loogisen tason normalisointivaatimuksista [SKS11 s. 362]. Moniulotteista tietoa taas ei voida mallintaa tehokkaasti alkuperäisillä ER-kaavioilla [SBH99], minkä vuoksi sille on ehdotettu lukuisia muita käsitetason malleja (ks. luku 3.1). Käsitetason käsitteet on usein valittu siten, että ne sopivat johonkin loogisen tason malliin. Esimerkiksi ER-malli sopii hyvin relaatiotietokannan suunnitteluun. Lisäksi myöhemmistä suunnitteluvaiheista saattaa syntyä vaatimuksia käsitesuunnittelulle: fyysisestä toteutuksesta voi syntyä vaatimuksia loogiselle suunnittelulle ja loogisesta suunnittelusta voi edelleen syntyä vaatimuksia käsitetason suunnittelulle [CoB02 s. 281-283]. Esimerkiksi fyysisessä toteutuksessa

ilmenevien suorituskykyongelmien vuoksi voidaan loogisen tason mallia joutua muuttamaan. Samoin normalisointivaatimukset ja relaatioiden funktionaalisten riippuvuuksien havaitseminen loogisessa suunnittelussa voivat johtaa käsitetason mallin muuttamiseen [SKS11 s. 361-362].

Yhtenä osoituksena käsitetason ja loogisen tason riippuvuudesta voidaan pitää myös sitä, että käsitetason ja loogisen tason suunnittelun rajat sekoittuvat usein toisiinsa, joko tarkoituksella tai vahingossa [GMR98][KiR02][SBH99][CaT98][HLV00]. Tämä on kolmas tutkielmassa tehty huomio. Periaatteessa eri suunnitteluvaiheet on määritelty selkeästi: käsitetaso ei ole riippuvainen toteutuksesta, loogisessa suunnittelussa tarkennetaan tietomallia tietyn tyyppiselle tietokannan hallintajärjestelmälle ja fyysisessä suunnittelussa päätetään millä tietokannan hallintajärjestelmällä ja miten tietokanta toteutetaan [CoB02 s. 281-282]. Jotkin lähteet käyttävät käsitteitä kuitenkin eri tavoin ja eri suunnitteluvaiheisiin liittyvät termit menevät keskustelussa helposti ristiin toistensa kanssa.

Neljäs tutkielmassa esiin tullut seikka on, että käsitetason tietomalli voidaan kuvata erilaisilla karkeustasoilla. Esimerkiksi yritystason tietomallia suunniteltaessa voidaan mallintaa pelkät yksilöt ja yhteydet, mutta jättää attribuutit mallintamatta. Yleensä käsitesuunnittelussa kuitenkin mallinnetaan myös attribuutit. Myös erilaiset moniulotteiset käsitemallit kuvaavat tietovaraston tiedon ominaisuuksia vaihtelevalla tarkkuustasolla.

Lopuksi tutkielmassa havaittiin käsitemalleja käytettävän erilaisiin käyttötarkoituksiin. Eri lähteissä ja menetelmissä suhtaudutaan käsitetason tarpeellisuuteen eri tavoin. Käsitetason suunnitteluvaihe voidaan jättää kokonaan pois [KiR02] tai se voidaan sekoittaa loogiseen ja fyysiseen suunnitteluun [GMR98][HLV00]. Yleensä loogisesta ja fyysisestä suunnittelusta erillistä käsitemallinnusta pidetään kuitenkin tärkeänä osana tietokannan suunnittelua [BCN92][CoB02 s. 330-374][SKS11 s. 257-321] eivätkä tietovarastot ole tästä poikkeus [GMR98]. Käsitemallinnus toimii suunnitteluvaiheena, jossa selkeytetään ajatuksia ja saadaan yleiskuva ongelmakentästä. Käsitesuunnittelu voi myös helpottaa kommunikointia teknisten asiantuntijoiden ja tietovaraston käyttäjien välillä [Gol09]. Kaikkein ankarimmassa lähestymistavassa käsitemallinnuksessa pyritään täydelliseen tietomalliin, jonka pohjalta voidaan suoraviivaisesti toteuttaa looginen suunnittelu ja fyysinen toteutus. Tämän jälkeen käsitemallia myös ylläpidetään, jolloin se on jatkuvasti ajan tasalla ja heijastaa luotettavasti järjestelmän tietosisältöä.

Tässä luvussa arvioidaan vielä lyhyesti *ER-mallin*, moniulotteisen tietomallin ja tietoholvimallin kanssa käytettäväksi tarkoitetun *C-DV-mallin* ominaisuuksia. Useista moniulotteisista malleista valitaan edustajaksi luvussa 3.1.1 esitelty *CGMD-malli*. Kolmesta eri tietomallista arvioidaan niiden monipuolisuutta, selkeyttä, toteutusriippumaattomuutta, käyttötarkoitusta sekä sitä kuinka tunnettu se on.

Alkuperäiseen ER-malliin tai C-DV-malliin verrattuna CGMD-malli on malleista monipuolisin. Sillä voidaan kuvata samat asiat kuin laajennetulla ER-mallillakin, mutta sen lisäksi tiedon moniulotteisia rakenteita, hierarkioita ja erilaisia koosteita. Alkuperäinen ER-malli sisältää ainoastaan yksilön, yhteyden ja attribuutin käsitteet. ER-mallin yleisimmät laajennukset, kuten periytyminen ja ositus, lisäävät ER-mallin ilmaisuvoimaa huomattavasti, mutta CGMD-mallin elegantti tapa kuvata lisäksi yksinkertaisia ja moniulotteisia koosteita tekee siitä monipuolisimman mallinnustavan. C-DV-mallilla voidaan kuvata tietoa vain loogisen tietoholvimallin käsitteiden kautta.

ER-mallin saama suosio ja pitkä ikä osoittaa, että se on myös helppokäyttöinen ja selkeä. Alkuperäisessä muodossaan se yksinkertaistaa tietokannan käsitteellisen suunnittelun toteutettavaksi minimaalisin keinoin ja laajennuksilla sen ilmaisuvoimaa voidaan tarpeen mukaan vielä kasvattaa. Moniulotteinen CGMD-malli on erittäin monipuolinen, mutta samalla myös vaikeatajuisempi ja monimutkaisempi. Lisäksi se käyttää yleistykseen, ositukseeen ja koosteisiin täysin omia merkintätapoja, joista ainakin osa olisi voitu korvata jo yleisemmin käytössä olevilla [SKS11 s. 305-306] merkinnöillä. C-DV-malli ei ole erityisen monimutkainen, mutta vaatii tietoholvimallin käsitteiden tuntemista.

C-DV-malli ei ole toteutusriippumaton, sillä se käyttää loogisen tason käsitteitä ja on selkeästi tarkoitettu käytettäväksi loogisen ja fyysisen tietoholvimallin kanssa. Periaatteessa C-DV-mallista voitaisiin johtaa jokin muukin looginen malli kuin tietoholvimalli, mutta käytännössä tällainen menettelytapa olisi kaukaa haettu ja epäluonteva. ER-malli on periaatteessa toteutusriippumaton, mutta sen käsitteet sopivat erityisesti relaatiomalliin. Moniulotteinen CGMD-malli on kaikkein riippumattomin toteutuksesta, sillä se kuvaa myös sellaisia tiedon sisäisiä rakenteita, kuten hierarkioita ja tasoja, jotka eivät liity suoraan relaatiomalliin.

Jokaisella käsitetason mallilla on oma käyttötarkoituksensa. ER-mallia käytetään yleensä operatiivisten järjestelmien tietokannan suunnitteluun ja tietovarastoissa sitä voidaan käyttää normaalimuodossa olevan keskitetyn tietovaraston suunnitteluun. Moniulotteisen loogisen rakenteen paikallisvarastojen suunnitteluun ER-mallia pidetään liian rajoittuneena [SBH99]. Sen sijaan CGMD-malli on tarkoitettu erityisesti moniulotteisen tiedon mallinnukseen, mutta koska se on ER-mallin laajennus sitä voidaan periaatteessa käyttää kaikkeen mihin ER-malliakin. C-DV mallin käyttö rajoittuu tietoholvimallinnettuun keskitettyyn tietovarastoon.

ER-malli on kolmesta käsitemallista tunnetuin. Sitä käytetään laajasti operatiivisissa järjestelmissä, mutta myös tietovarastojen suunnittelussa [Inm02]. ER-mallia on testattu vuosia ja siitä löytyy myös paljon tieteellistä tutkimusta [Gol09]. Se mainitaan yleensä tietokannan suunnitteluun tarkoitetuissa oppaissa kuten [BCN92][CoB02][SKS11] ja lisäksi monet tietokannan suunnitteluvälineet hyödyntävät sitä [SKS11 s. 262-314]. Käsitteiden merkintätavat vaihtelevat

ja erilaisia laajennuksia käytetään, mutta ER-malli on kuitenkin niin hyvin tunnettu, että sillä voidaan sanoa olevan tietynlainen käytännön standardin asema. CGMD-malli tai mikään muu moniulotteinen tietomalli ei ole onnistunut saavuttamaan vastaavaa asemaa [Kam08], eikä niitä yleensä mainita tietokannan suunnittelun yleisteoksissa edes tietovarastoinnin yhteydessä. C-DV-malli on harvinainen ja tarkoitettu käytettäväksi ainoastaan loogisen tietoholvimallin kanssa. Rajoittuneisuutensa ja uutuutensa vuoksi C-DV-mallia voidaan ainakin toistaiseksi pitää marginaalisena mallinnustapana.

Taulukossa 6 näkyy koostettuna ER-mallin, moniulotteisen CGMD-mallin sekä C-DV-mallin tyypillisiä ominaisuuksia. Taulukossa on esitetty myös mallien erityisvahvuudet sekä mahdolliset heikkoudet.

ER-malli	• laajennuksien kautta monipuolinen • **helposti ymmärrettävä** • toteutuksesta riippumaton • sopii keskitetyn tietovaraston suunnitteluun • **hyvin tunnettu**
Moniulotteinen CGMD-malli	• **erittäin monipuolinen** • *hieman vaikeatajuinen* • **täysin toteutuksesta riippumaton** • sopii paikallisvarastojen suunnitteluun • ei kovin tunnettu
C-DV-malli	• *rajoittunut loogisen tietoholvimallin suunnitteluun* • kohtalaisen selkeä • *ei toteutuksesta riippumaton* • sopii tietoholvimallinnetun keskitetyn tietovaraston suunnitteluun. • *harvinainen*

*Taulukko 6: Käsitemallien tyypillisiä ominaisuuksia (***erityisvahvuudet lihavoitu***, heikkoudet kursivoitu ja vaalennettu).*

7.2 Loogiset mallit

Loogisen tason malleja ei tule sekoittaa käsitetason kuvaustapoihin. Esimerkiksi tähtimallin vertaaminen ER-malliin ei ole erityisen mielekästä, sillä tähtimalli ja ER-malli kuvaavat tietoa eri abstraktiotasolla. Saman abstraktiotason tietomalleja voidaan verrata toisiinsa, mutta vertailussa on syytä ottaa huomioon tietomallien käyttötarkoitus. Esimerkiksi tietoholvimallia ja normalisoitu relaatiomallia käytetään keskitetyn tietovaraston yksityiskohtaisen ja aikasidonnaisen tiedon tallennukseen. Moniulotteinen tähtimalli taas soveltuu suppeamman aihealueen ja mahdollisesti osittain karkeistettua tietoa sisältävän paikallisvaraston toteutukseen [Inm02 s. 137-143]. Arkkitehtuuriratkaisusta riippuu mitä tietomallia tietovarastossa käytetään ja onko käytössä yksi vai useampia tietomalleja.

Tässä luvussa arvioidaan kolmea loogisen tason tietomallia: *normalisoitua tietomallia*, *tietoholvimallia* ja moniulotteista tietomallia. Moniulotteisten tietomallien edustajaksi valitaan *tähtimalli*. Kolmea loogisen tason tietomallia arvioidaan seuraavilla kriteereillä: yleisyys, selkeys, laajennettavuus, aikasidonnaisuus, jäljitettävyys, latausten tehokkuus, kyselyiden tehokkuus ja käyttötarkoitus.

Tähtimallin käyttökelpoisuutta tietovaraston esityskerroksen tietomallina ei kyseenalaista oikeastaan mikään menetelmä. Jos tietojen yhtenäistämisestä pidetään huolta, voidaan tietovarasto toteuttaa kokonaan moniulotteisella tietomallilla [KiR02]. Tähtimallia käytetään itsenäisissä paikallisvarastoissa ja yhtenäistettyjä paikallisvarastoja käyttävissä arkkitehtuureissa, joiden osuus tietovarastototeutuksista on yhteensä noin 40 prosenttia [WaA05]. Lisäksi tähtimallia käytetään usein esityskerroksen tietomallina keskiö ja puolat -arkkitehtuurissa, jonka osuus on saman tutkimuksen mukaan 39 prosenttia. Voimme siis sanoa, että tähtimalli on tietovarastoinnissa käytetyistä tietomalleista tunnetuin ja yleisin [KRT08 s. 338]. Normalisoitu tietomalli on yleinen operatiivisissa järjestelmissä ja sen vuoksi myös se on hyvin tunnettu. Tietovarastoinnissa normalisoitua tietomallia käytetään kuitenkin lähinnä ETL-prosessissa latausalueen mallina [KiR02 s. 8-9] tai keskitetyn tietovaraston tietomallina [Inm02]. Sama koskee tietoholvimallia, johon liittyvää tutkimusta tai kirjallisuutta on varsin vähän. Se on ilmeisesti tietomalleista harvinaisin.

Tähtimallia pidetään usein tietomalleista selkeimpänä [KRT08 s. 237]. Yksinkertainen *tähti* sisältää vain yhden faktataulun ja siihen yhdestä moneen suhteella liitetyt ulottuvuustaulut. Faktojen ja ulottuvuuksien käsitteet on helppo selittää sekä teknisille asiantuntijoille että loppukäyttäjille. Tähtimallin ymmärrettävyyttä on verrattu normalisoituun tietomalliin muutamissa empiirisissä tutkimuksissa. Tähtimalli on tutkimuksissa osoittautunut selkeämmäksi ja helpommin ymmärrettäväksi malliksi kuin normalisoitu tietomalli [CSL06][VKS12][SCT11]. Tähtimalli on intuitiivisesti helpompi ymmärtää ja palauttaa mieleen kuin normalisointiin perustuva ER-kaavio [CSL06]. Myös satunnaisissa kyselyissä tähtimalli on usein käyttäjien kannalta ymmärrettävämpi kuin normalisoitu tietomalli. Käyttäjät pystyvät tähtimallin avulla tekemään satunnaisia kyselyitä nopeammin, saavat sen avulla tarkempia ja oikeampia tuloksia sekä hahmottavat kuvatun aihealueen paremmin [VKS12]. Asetelma kuitenkin muuttuu, kun mallinnetaan laajempaa ja monimutkaisempaa tietoa. Käytettäessä realistista ja monimutkaista tietomallia kohderyhmä ymmärsi tähtimallin ja normalisoidun mallin lähes yhtä tarkasti ja lähes samalla vaivalla [SCT11]. Jos tehtävänä on edelleen laajentaa tietomallia, normalisoitu tietomalli on jopa hieman tähtimallia tehokkaampi. Tietoholvimallin ymmärrettävyydestä ei löydy vertailevaa tieteellistä tutkimustietoa, mutta suuri taulujen määrä, monesta moneen yhteyksien käyttö, metatiedot ja pitkät nimeämiskäytännöt vaikeuttavat sen ymmärrettävyyttä.

Tietoholvimallissa on jo mallinnustapaa kehitettäessä pyritty ottamaan huomioon mallin laajennettavuus ja mallissa tapahtuvat muutokset. Linkkitaulujen ansiosta mallia voidaan yleensä laajentaa pelkillä lisäyksillä, jolloin olemassa olevia kyselyitä ei tarvitse muuttaa (ks. luku 4.2). Tietoholvimallia voidaan siis pitää helpoiten laajennettavana tietomallina. Normalisoidussa tietomallissa ja erityisesti tähtimallissa joudutaan useammin turvautumaan taulujen muuttamiseen tietomallia laajennettaessa, jolloin myös vanhat kyselyt voivat toimia virheellisesti.

Tietovaraston on aina käsiteltävä tietoa aikasidonnaisesti. Normalisoidussa tietomallissa tietoihin liitetään yleensä aikaleima. Koska uuteen aikaleimaan ei kuitenkaan liity uutta pääavainta, tulee kyselyistä ja erityisesti niissä tapahtuvista liitoksista astetta monimutkaisempia ja mahdollisesti merkittävästi hitaampia [KRT08 s. 286]. Tähtimallissa muutoksia tapahtuu yleensä vain ulottuvuustaulujen tiedoissa ja niitä voidaan hallita usealla eri tavalla. Osa tähtimallin kanssa käytetyistä muutosten hallinnan tekniikoista kuitenkin hävittää tietoa (ks. luku 3.3). Ulottuvuustauluissa tapahtuvia muutoksia voidaan tähtimallissa hallita myös menettämättä tietoa, mutta tämä aiheuttaa uuden rivin ja uuden keinoavaimen lisäämisen ulottuvuustauluun. Uudella keinoavaimella varustetun rivin lisääminen taas voi aiheuttaa tilanteen, jossa myös faktataulun viitettä on päivitettävä, sillä ulottuvuustaulu on isä faktataululle [Lin11b]. Tietoholvimallissa on varauduttu parhaiten ajan myötä tapahtuviin muutoksiin. Mallin spesifikaatioon [Lin10a] kuuluu aikaleiman lisääminen kaikkiin tietoihin ja kaikki muuttuvat tiedot on pyritty keskittämään satelliittitauluihin, jotka ovat lapsia keskiöille ja linkeille. Uuden rivin ja keinoavaimen lisääminen muuttuvia tietoja sisältävään satelliittitauluun ei siis aiheuta päivityksiä keskiöihin ja linkkeihin, koska niistä ei ole viitettä satelliittitauluun.

Tietoholvimallilla voidaan kätevästi jäljittää tiedon alkulähde ja tiedoissa ajan myötä tapahtuneet muutokset. Tietovarastoinnissa voi helposti syntyä tilanteita, joissa raporttien sisältämät tiedot ovat ristiriidassa toistensa kanssa. Tämän vuoksi tietoholvimallissa tallennetaan kaikki lähdejärjestelmistä saatu tieto ja lisäksi tietojen lähdejärjestelmä ja latausaika merkitään. Periaatteessa lähdejärjestelmien sisältämät tiedot tietyllä ajanhetkellä voidaan muodostaa uudelleen tietovaraston tiedoista [Lin11a s. 11] ja ristiriitatilanteissa voidaan selvittää, mistä virheeltä näyttävä tieto johtuu. Täydellinen jäljitettävyys voidaan toteuttaa muillakin malleilla, mutta tietoholvimallin spesifikaatiossa tämä on sisäänrakennettu ominaisuus.

Jos spesifikaatiota on noudatettu, tietoholvimalli mahdollistaa latausoperaatioiden rinnakkaisuuden parhaiten. Tietoholvimallin spesifikaatio määrää tarkasti viiteyhteyksien käyttötavan, minkä ansiosta keskiöiden, linkkien ja satelliittien lataukset voidaan järjestää siten, että rinnakkaisuutta voidaan käyttää mahdollisimman tehokkaasti hyödyksi (ks. luku 4.4). Myös tähtimallissa voidaan periaatteessa ladata ensin ulottuvuustaulujen tiedot rinnakkain ja sen jälkeen faktataulun tiedot. Tilanne on kuitenkin usein käytännössä hieman monimutkaisempi. Tietovarasto voi esi-

merkiksi sisältää useita toisiinsa liitettyjä tähtiä, ulottuvuustaulujen välillä voi olla viitteitä, faktataulun ja ulottuvuustaulujen välillä voi olla monesta moneen yhteyden luovia siltatauluja ja tähti voi olla osittain normalisoitu. Tämän vuoksi kaikista latauksista tulee erilaisia. Sama koskee normalisoitua tietomallia. Tietoholvimallin tapa jakaa tiedot keskiöihin, linkkeihin ja satelliitteihin luo valmiin kehyksen latausten rinnakkaistamiselle, jonka ansiosta kaikista latauksista tulee tehokkaita ja ne voidaan toteuttaa hyvin samaan tapaan.

Kyselyiden tehokkuus riippuu hyvin monesta asiasta, esimerkiksi tietokannan koosta, taulujen määrästä, käytettävästä tietokannan hallintajärjestelmästä, laitteistosta, rinnakkaisuuden määrästä, käytetyistä hakemistoista ja muista optimointikeinoista sekä tehdystä kyselystä, sen rajoittavuudesta ja siinä tarvittavien liitosten määrästä. Yhtä totuutta tietomallien suorituskykyä tukevista ominaisuuksista on siis vaikea antaa. Arvioimme suorituskykyä karkeasti tutkielmassa käytetyn *Market*-esimerkkimyyntijärjestelmään kohdistuvan koostekyselyn avulla. Arviointi perustuu kyselyssä tarvittavien taulujen ja niiden välillä tehtävien liitosten määrään. Alla on listattu luvun 3.1 kuvassa 7 esitetyn moniulotteisen koosteen laskemiseksi vaaditut kyselyt luvun 2.3 normalisoidusta tietomallista, luvun 3.2 tähtimallista ja luvun 4.4 tietoholvimallista. Kysely palauttaa kaikki myyntitapahtumat asiakkaittain arkisin ja vapaapäivinä.

Koostekysely normalisoidusta tietomallista

```
SELECT A.asiakastunnus, A.sukunimi, P.kuvaus, SUM (kokonaissumma)
FROM Asiakas A, Päivätyyppi P, Myynti M
WHERE A.asiakastunnus = M.asiakas AND M.päivätyyppi = P.päivätyyppitunnus
GROUP BY A.asiakastunnus, A.sukunimi, P.kuvaus
```

Koostekysely tähtimallista

```
SELECT P.tyyppi, A.asiakastunnus, A.sukunimi, SUM (summa)
FROM Asiakas A, Päiväys P, Myynti M
WHERE A.asiakastunnus = M.asiakas AND M.päiväys = P.päiväystunnus
GROUP BY P.tyyppi, A.asiakastunnus, A.sukunimi
```

Koostekysely tietoholvimallista

```
SELECT HA.asiakastunnus, SA.sukunimi, RP.päivätyyppi, SUM (ST.hinta)
FROM HUB_Asiakas HA, SAT_Asiakas SA, LNK_Asiakas_Myymälä_Tuote LM,
        SAT_Asiakas_Myymälä_Tuote SP, Päivätyyppi RP, HUB_Tuote HT,
        SAT_Tuote ST
WHERE SP.päivätyyppi = RP.päivätyyppi_SQN
          AND SP.sat_ASM = LM.lnk_ASM_SQN
          AND LM.hub_asiakas = HA.hub_asiakas_SQN
          AND SA.sat_asiakas = HA.hub_asiakas_SQN
          AND LM.hub_tuote = HA.hub_tuote_SQN
          AND ST.sat_tuote = HA.hub_tuote_SQN
GROUP BY RP.päivätyyppi, HA.asiakastunnus, SA.sukunimi
```

Selvästikin tietoholvimallissa käytetty kysely on monimutkaisin. Normalisoitu tietomalli ja tähtimalli joutuvat käymään läpi 3 taulua ja tekemään ainakin 2 taulujen välistä liitosoperaatiota. Tietoholvimallilla läpikäytävien taulujen määrä on 7 ja liitosten määrä 6. Esimerkin tietoholvimallissa ei ole myynnin kokonaissumman ilmoittavaa koostetietoa, minkä vuoksi kyselyssä tarvitaan myös *HUB_Tuote*- ja *SAT_Tuote*-tauluja. Jos tietoholvimallissa olisi ollut kokonaissumma esitettynä koostavalla linkillä (computed and aggregated link, ks. luku 4.2), *HUB_Tuote*- ja *SAT_Tuote*-tauluja ei olisi tietoholvikyselyssä tarvittu. Tällöin kyselyn läpikäymä taulujen määrä olisi ollut 5 ja liitosten määrä 4. Tämäkin on suhteellisesti selvästi suurempi määrä kuin normalisoidussa tietomallissa ja tähtimallissa. Yhteensä *Market*-esimerkkimyyntijärjestelmän tietoholvimallissa on 12 taulua, normalisoidussa tietomallissa 7 taulua ja tähtimallissa 5 taulua. Taulujen määrästä ei voi suoraan johtaa kyselyjen vasteaikoja kaikissa tapauksissa, mutta usein taulujen suuri määrä hidastaa kyselyitä liitosoperaatioiden raskauden vuoksi. Taulujen vähäisen määrän ja tähtimallille kehitettyjen optimointikeinojen ansiosta *t*ähtimallia pidetään usein tietovarastoinnissa käytetyistä tietomalleista suorituskykyisimpänä [Kim97][KiR02][KRT08]. Tässä kyselyssä tähtimallin ulottuvuuksien rajoittavuuteen perustuvasta optimoinnista (ks. luku 3.3) ei kuitenkaan ole hyötyä, sillä *asiakas*- ja *päivä*-tauluihin ei kyselyssä liity rajoittavaa ehtoa. Usein tällainen optimointi kuitenkin on mahdollista ja monet tietokannan hallintajärjestelmät osaavat hyödyntää sitä [KRT08 s. 347].

Tässä luvussa arvioiduista loogisen tason tietomalleista on syytä muistaa, että niiden käyttötarkoitus voi olla hieman erilainen eivätkä ne välttämättä ole toisiaan poissulkevia. Tähtimallia voidaan käyttää paikallisvarastojen tietomallina ja se sopii hyvin käytettäväksi esityskerroksen raportointivälineiden kanssa. Normalisoitu tietomalli ja tietoholvimalli ovat toisilleen vaihtoehtoisia tietomalleja, jotka sopivat yksityiskohtaisen tiedon tallentamiseen, tiedon

yhtenäistämiseen ja tiedonlouhintaan. Tähtimalli sopii hyvin käytettäväksi yhtenäistetyt paikallisvarastot -arkkitehtuurin kanssa. Normalisoitu tietomalli ja tietoholvimalli sopivat hyvin keskitetyn tietovaraston toteutukseen keskiö ja puolat -arkkitehtuurissa, jossa paikallisvarastot taas voidaan mallintaa tähtimallilla.

Taulukossa 7 esitetään koostettuna kolmen eri loogisen tietomallin tyypillisiä ominaisuuksia sekä niiden erityisiä vahvuuksia ja heikkouksia.

Normalisoitu tietomalli	• yleinen operatiivisissa järjestelmissä • melko helposti ymmärrettävä • melko helposti laajennettavissa • historiointi mahdollista toteuttaa • jäljitettävyys mahdollista toteuttaa • latausten rinnakkaistaminen tehtävä tapauskohtaisesti • kohtalaisen suuri määrä tauluja • sopii keskitetyn tietovaraston toteuttamiseen
Tähtimalli	• **yleinen tietovarastoissa** • **helposti ymmärrettävä** • laajentaminen voi vaatia lisätyötä • historiointi mahdollista toteuttaa • jäljitettävyys mahdollista toteuttaa • latausten rinnakkaistaminen tehtävä tapauskohtaisesti • **pieni määrä tauluja** • sopii paikallisvarastojen toteuttamiseen
Tietoholvimalli	• *jonkin verran käytetty tietovarastoissa* • *hieman vaikeatajuinen* • **helposti laajennettava** • **täydellinen historiointi** • **täydellinen jäljitettävyys** • **latausten rinnakkaistaminen helppoa** • *erittäin suuri määrä tauluja* • sopii keskitetyn tietovaraston toteuttamiseen

*Taulukko 7: Loogisten tietomallien tyypillisiä ominaisuuksia (***erityisvahvuudet lihavoitu***, heikkoudet kursivoitu ja vaalennettu).*

7.3 Arkkitehtuurit

Tutkielmassa esiteltiin viisi tietovarastojen arkkitehtuuriratkaisua: itsenäiset paikallisvarastot, yhtenäistetyt paikallisvarastot, yritystason keskitetty tietovarasto, keskiö ja puolat -arkkitehtuuri sekä liitosarkkitehtuuri (ks. luku 5). Tässä luvussa vertaillaan käytännön kokemuksia eri arkkitehtuurivaihtoehdoista, mutta vertailusta jätetään pois itsenäiset paikallisvarastot -arkkitehtuuri, koska sen käyttö on melko vähäistä [WaA05] eikä sitä yleensä suositella [ArW10]. Myös liitosarkkitehtuuri jätetään pois vertailusta, koska sitä käytetään erittäin vähän [WaA05] ja sen käytössä ilmenee perustavanlaatuisia suorituskykyyn, tiedon yhtenäistämiseen, historiointiin sekä teknisen ympäristön hallintaan liittyviä ongelmia [ISN08 s. 16-18].

Arvioimme *yhtenäistettyjä paikallisvarastoja*, *yritystason keskitettyä tietovarastoa* sekä *keskiö ja puolat* -arkkitehtuuria pääasiassa Watsonin ja Ariyachandran tekemän tutkimuksen [ArW06] [ArW08][WaA05] perusteella. Arkkitehtuureja ja niiden käyttöä arvioidaan seuraavilla kriteereillä: yleisyys, loppukäyttäjien saama hyöty, vaikutus liiketoimintaan, kehitysaika, kehityskustannukset, ylläpitokustannukset ja käyttötarkoitus.

Keskiö ja puolat -arkkitehtuuri on yleisin tietovarastojen arkkitehtuuriratkaisu. Sen osuus on tietovarastototeutuksissa noin 35-39 prosenttia kaikista viidestä arkkitehtuurivaihtoehdosta [AMO12][WaA05]. Toiseksi yleisin on yhtenäistetyt paikallisvarastot, jonka osuus on noin 23-27 prosenttia [AMO12][WaA05]. Kolmanneksi suosituin arkkitehtuuriratkaisu on yritystason keskitetty tietovarasto noin 17-20 prosentin osuudella.

Yksittäisten loppukäyttäjien tyytyväisyys tietovarastointijärjestelmään on paras tietovarastointijärjestelmissä, joissa arkkitehtuuriratkaisuna on yhtenäistetyt paikallisvarastot [WaA05]. Ero kahdella muulla arkkitehtuurilla toteutettuihin tietovarastoihin on hyvin pieni, mutta yhtenäistetyillä paikallisvarastoilla toteutettuja tietovarastoja käytetään aktiivisimmin, niiden käyttäjät löytävät tiedon helpoimmin, he ymmärtävät tiedon paremmin ja löytävät eniten uusia näkökulmia sekä saavat parhaiten apua päätöksentekoon.

Yhtenäistetyillä paikallisvarastoilla toteutetuilla tietovarastoilla on myös positiivista vaikutusta liiketoimintaan. Watsonin ja Ariyachandran tutkimuksen mukaan yhtenäistetyillä paikallisvarastoilla toteutettu tietovarasto muun muassa parantaa eniten liiketoimintaprosesseja sekä lisää eniten osastojen välistä kommunikointia [WaA05]. Yritystason keskitetty tietovarasto näyttäisi johtavan eniten liikevoiton kasvuun. Erot kolmen eri arkkitehtuurivaihtoehdon välillä ovat tällä osa-alueella kuitenkin lähes merkityksettömiä [ArW08].

Yritystason keskitetyn tietovaraston kehitysaika ensimmäisten tuloksien saamiseksi on lyhyin, keskimäärin 8,78 kuukautta [WaA05]. Ero yhtenäistettyjen paikallisvarastojen kehitysaikaan (keskimäärin 8, 93 kuukautta) on tutkimuksen mukaan pieni, mutta keskiö ja puolat -arkkitehtuuria käyttävän tietovaraston kehittämisaikaan (11,44 kuukautta) verrattuna selvästi suurempi.

Kuten kehitysaika, myös kehityskustannukset ensimmäisten tuloksien aikaansaamiseksi on pienin yhtenäistetyillä paikallisvarastoilla, keskimäärin lähes puolet pienemmät kuin keskiö ja puolat -arkkitehtuurilla toteutetuilla tietovarastoilla, mutta vain hieman pienemmät kuin yritystason keskitetyillä tietovarastoilla [WaA05].

Ylläpitokustannukset ovat keskimäärin pienimmät yritystason keskitetty tietovarasto -arkkitehtuurilla toteutetuissa ratkaisuissa ja toiseksi pienimmät yhtenäistetyillä paikallisvarastoilla toteutetuissa ratkaisuissa. Selvästi suurimmat ylläpitokustannukset ovat keskiö ja puolat -arkkitehtuuriratkaisua käyttävissä tietovarastoissa [WaA05].

Keskiö ja puolat -arkkitehtuurin suuret kehitys- ja ylläpitokustannukset voivat ainakin osittain johtua siitä, että sitä käytetään enemmän suuria tietomassoja sisältävien, koko yrityksen tasoisten tietojärjestelmien toteutuksissa [WaA05]. Yritystason keskitetyllä tietovarastolla toteutetaan myös paljon koko yrityksen laajuisia tietovarastoja, mutta ei yleensä tietomäärältään niin suuria kuin keskiö ja puolat -arkkitehtuuriratkaisulla. Yhtenäistettyjä paikallisvarastoja käytetään eniten yhden tai useamman osaston tietovarastoissa.

Taulukossa 8 esitetään koostettuna kolmella eri arkkitehtuurilla toteutettujen tietovarastojen tyypillisiä ominaisuuksia sekä arkkitehtuurivaihtoehtojen erityisvahvuuksia ja heikkouksia.

Yhtenäistetyt paikallisvarastot	• melko yleinen • **palvelee parhaiten käyttäjiä** • positiivinen vaikutus liiketoimintaan • l**yhyt kehitysaika** • **pienet kehityskustannukset** • melko pienet ylläpitokustannukset • käytetään yleensä hieman pienempien tietovarastojen toteutukseen
Yritystason keskitetty tietovarasto	• melko harvinainen • palvelee yleensä hyvin käyttäjiä • positiivinen vaikutus liiketoimintaan • **lyhyt kehitysaika** • **pienet kehityskustannukset** • **pienet ylläpitokustannukset** • käytetään yleensä koko yrityksen laajuisen tietovaraston toteutukseen
Keskiö ja puolat	• **yleisin** • palvelee yleensä hyvin käyttäjiä • positiivinen vaikutus liiketoimintaan • *melko pitkä kehitysaika* • *suuret kehityskustannukset* • *suuret ylläpitokustannukset* • käytetään yleensä kaikkein suurimpien ja laajimpien tietovarastojen toteutukseen

*Taulukko 8: Arkkitehtuuriratkaisujen tyypillisiä ominaisuuksia (***erityisvahvuudet lihavoitu***, heikkoudet kursivoitu ja vaalennettu).*

7.4 Menetelmät

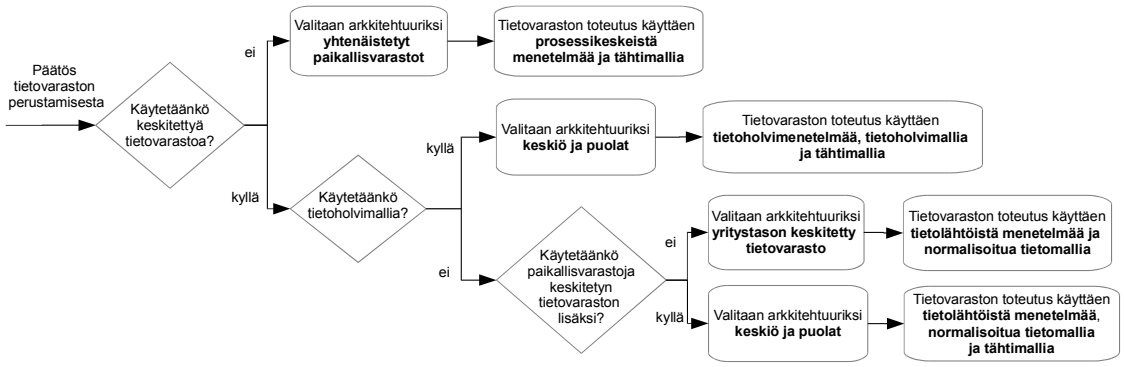
Luvussa 6 käsiteltiin myös kolmea tietovaraston kokonaisvaltaista kehittämismenetelmää, joita kutsuimme *tietolähtöiseksi menetelmäksi*, *prosessikeskeiseksi menetelmäksi* ja *tietoholvimenetelmäksi*. Menetelmät liittyvät tiettyihin arkkitehtuuriratkaisuihin ja niissä käytetään myös eri tietomalleja. Prosessikeskeisessä menetelmässä käytetään yhtenäistettyjä paikallisvarastoja ja

moniulotteista tietomallia. Tietolähtöisessä menetelmässä voidaan käyttää yritystason keskitettyä tietovarastoa tai keskiö ja puolat -arkkitehtuuria sekä normalisoitua tietomallia ja moniulotteista tietomallia. Tietoholvimenetelmässä käytetään keskiö ja puolat -arkkitehtuuria, tietoholvimallia ja mahdollisesti myös moniulotteista tietomallia.

Koska menetelmät ovat sidoksissa määrättyihin arkkitehtuureihin, on arkkitehtuuriratkaisu päätettävä ennen kuin jotain luvun 6 kehittämismenetelmistä voidaan käyttää. Jos arkkitehtuuriksi valitaan yhtenäistetyt paikallisvarastot, menetelmäksi sopii prosessikeskeinen menetelmä. Jos arkkitehtuuriksi taas valitaan keskiö ja puolat -arkkitehtuuri, voidaan valita tietolähtöisen menetelmän ja tietoholvimenetelmän välillä. Yritystason keskitetty tietovarasto voidaan toteuttaa käyttäen tietolähtöistä menetelmää.

Arkkitehtuuriratkaisua ja kehittämismenetelmää valittaessa kannattaa erityisesti ottaa huomioon menetelmien erilaiset näkemykset moniulotteisen tietomallin tarkoituksenmukaisuudesta tietovaraston toteutuksessa. Prosessikeskeisessä menetelmässä moniulotteista tietomallia pidetään riittävänä tietovarastolle. Tietolähtöisessä menetelmässä ja tietoholvimallissa moniulotteista tietomallia käytetään vain keskitetystä tietovarastosta johdettujen paikallisvarastojen tietomallina. Arkkitehtuuriratkaisua ja menetelmää valittaessa on mietittävä, onko moniulotteinen malli riittävä kaikenlaisen tietovarastossa säilytettävän tiedon tallentamiseen vai aiheutuuko sen käytöstä haitallisia rajoituksia näkökulmalle tai tiedon karkeisuustasolle, laadulle, yhtenäisyydelle tai ylläpidolle.

Toinen arkkitehtuurin ja menetelmän valintaan vaikuttava seikka on, kuinka tärkeinä pidetään täydellistä historiointia, jäljitettävyyttä ja tietomallin laajennettavuutta. Jos ne ovat erityisen tärkeitä, esimerkiksi lainsäädännöllisistä syistä, on tietoholvimalli vahva ehdokas. Muussa tapauksessa tietoholvimalli ei välttämättä tuo sellaisia hyötyjä, jotka riittäisivät kompensoimaan sen monimutkaisuudesta aiheutuvat heikkoudet. Kuvassa 25 on esitetty prosessikaavio arkkitehtuurin ja menetelmän valinnasta sekä valintojen vaikutuksesta käytettäviin tietomalleihin tyypillisissä ratkaisuvaihtoehdoissa.



Kuva 25: Prosessikaavio tietovaraston arkkitehtuurin, menetelmän ja tietomallin valitsemisesta.

Kolmeen kehittämismenetelmään kuuluu tietomalli- ja arkkitehtuurisuositusten lisäksi joukko yleisiä ohjeita ja parhaita käytäntöjä, jotka liittyvät esimerkiksi latausten hallintaan, tiedon eheyteen, metatiedon hallintaan, strukturoidun ja strukturoimattoman tiedon käsittelyyn ja projektinhallintaan. Tällaisiin tietovaraston toteutuksessa esille tuleviin haasteisiin löytyy menetelmistä paljon hyviä ohjeita ja niitä voi soveltuvin osin käyttää tietovaraston suunnittelussa ja toteutuksessa arkkitehtuuriratkaisusta ja menetelmästä riippumatta.

8 Yhteenveto

Tässä tutkielmassa käsiteltiin tietovaraston tietomalleja sekä niiden käyttötapoja erilaisissa arkkitehtuureissa ja kehittämismenetelmissä. Luvuissa 2-4 tutustuttiin *normalisoituun tietomalliin*, *moniulotteisiin tietomalleihin* ja *tietoholvimalliin*. Tutkielmassa selvitettiin erilaisten tietomallien käsitetason ja loogisen tason suunnittelua ja todettiin, että käsitetason malli ei ole näkökulmariippumaton eikä usein käytännössä myöskään täysin toteutuksesta riippumaton. Lisäksi tutkielmassa tutustuttiin tietomallien erilaisiin erityispiirteisiin ja havainnollistettiin niiden kaikkien suunnittelua tutkielmaa varten kehitetyn esimerkin avulla. Luvussa 4 johdettiin tietoholvimallin suunnitteluun tarkoitettu yksinkertainen käsitetason mallinnustapa ja esiteltiin hieman tietoholvimallia muistuttava ankkurimalli.

Luvussa 5 selvitettiin, millaisista osajärjestelmistä tietovarasto koostuu ja miten ne toimivat suhteessa toisiinsa. Erilaiset ratkaisuvaihtoehdot voidaan jakaa viiteen erilaiseen arkkitehtuuriin: *itsenäisiin paikallisvarastoihin*, *yhtenäistettyihin paikallisvarastoihin*, *yritystason keskitettyyn tietovarastoon*, *keskiö ja puolat -arkkitehtuuriin* sekä *liitosarkkitehtuuriin*. Tutkielmassa todettiin, että eri arkkitehtuurivaihtoehdoissa käytetään tietomalleja eri tavoin: esimerkiksi moniulotteinen tähtimalli sopii erityisesti paikallisvarastojen toteuttamiseen, mutta normalisoitua tietomallia ja tietoholvimallia käytetään keskitetyn tietovaraston mallina yritystason keskitetyssä tietovarastossa sekä keskiö ja puolat -arkkitehtuurissa.

Luvussa 6 tutustuttiin kolmeen erilaiseen tietovarastoinnin toteuttamisessa käytettyyn kehittämismenetelmään, joita ovat Inmonin *tietolähtöinen menetelmä*, Kimballin ja kumppaneiden moniulotteisen tietomallin merkitystä korostava *prosessikeskeinen menetelmä* sekä Linstedtin *tietoholvimenetelmä*. Tietolähtöisessä menetelmässä ja tietoholvimenetelmässä käytetään keskitettyä tietovarastoa, mutta prosessikeskeisessä menetelmässä tietovarasto toteutetaan ilman keskitettyä tietovarastoa yhtenäistettyjen paikallisvarastojen avulla. Tietoholvimenetelmä on uudempi kuin kaksi muuta menetelmää eikä sitä saatavilla olevien lähteiden perusteella voi ainakaan vielä pitää hyvin määriteltynä kehittämismenetelmänä.

Luvussa 7 vertailtiin tietomallien, arkkitehtuurien ja menetelmien vahvuuksia ja heikkouksia. Tutkielmassa johdettiin myös prosessikaavio, jota voidaan käyttää apuna tietovaraston toteutuksessa käytettävän arkkitehtuurin, menetelmän ja tietomallien valinnassa.

Keskitetyn tietovaraston toteuttamisessa tietoholvimalli on mielenkiintoinen vaihtoehto normalisoidulle tietoholvimallille. Tietoholvimallin vahvuuksia ovat muutosjoustavuus, historiointi, jäljitettävyys ja mahdollisuus rinnakkaistaa tehokkaasti lähdejärjestelmistä tehtävät lataukset. Tietoholvimallin looginen ja fyysinen toteutus sisältää kuitenkin paljon tietokantatauluja, mikä

tekee siitä monimutkaisemman ja mahdollisesti myös tehottomamman joissakin tilanteissa. Jos tietoholvimallin hyötyjä ei pidetä perusteltuina voidaan keskitetty tietovarasto toteuttaa myös normalisoidun tietomallin avulla. Tähtimalli taas sopii erityisen hyvin esitystason ja paikallisvarastojen toteutuksessa käytettäväksi tietomalliksi.

Arkkitehtuurivaihtoehdoista yhtenäistetyt paikallisvarastot, yritystason keskitetty tietovarasto sekä keskiö ja puolat -arkkitehtuuri ovat yleisimmät ja käytännössä parhaiksi koetut vaihtoehdot. Kaikilla kolmella arkkitehtuurivaihtoehdolla voidaan toteuttaa luotettavasti eri tyyppisiä tietovarastoja, mutta keskiö ja puolat -arkkitehtuuria käytetään eniten suurimpien tietovarastojen toteuttamiseen.

Tietolähtöinen menetelmä, prosessikeskeinen menetelmä ja tietoholvimenetelmä koostuvat joukosta käytännöllisiä ratkaisuohjeita, joita voidaan käyttää tietovaraston toteuttamiseen. Kehittämismenetelmät liittyvät tiettyihin arkkitehtuuriratkaisuihin ja tietomalleihin, mutta sisältävät myös paljon hyviä yleisiä ohjeita minkä tahansa tietovaraston toteuttamiseen.

Tietoholvimalli on mielenkiintoinen jatkotutkimuskohde. Siitä löytyy hyvin vähän tieteellisissä julkaisuissa esitettyä riippumatonta tutkimusta joten tietoholvimenetelmällä toteutetuista tietovarastointiratkaisuista saatuja käytännön kokemuksia ja mahdollisia konkreettisia hyötyjä olisi hyödyllistä kartoittaa. Tutkimuksessa voitaisiin esimerkiksi selvittää parantaako tietoholvimallissa käytetty historiointi ja jäljitettävyys tiedon laatua, saadaanko mallin laajennettavuudella kustannushyötyjä, helpottaako tietoholvimallin käyttö merkittävästi lähdejärjestelmistä tehtävien latausten hallintaa ja tehostaako tietoholvimallin käyttö tietovaraston hallintaa.

Lähteet

AMH08 Abadi, D. J., Madden, S. R., Hachem N., Column-stores vs. row-stores: how different are they really? *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Vancouver, 2008, sivut 967-980.

AMO12 Alsquor, M., Matouk, K., Owoc, M. L., A survey of data warehouse architectures: preliminary results. *Proceedings of the Federated Conference on Computer Science and Information Systems,* Wroclaw, 2012, sivut 1121-1126.

ASS06 Abelló, A., Samos, J., Saltor, F., YAM^2: a multidimensional conceptual model extending UML. *Information Systems,* 31(6), 2006, sivut 541-567.

ArW06 Ariyachandra, T., Watson, H., Technical opinion: Which data warehouse architecture is most successful? *Business Intelligence Journal,* 11(1), 2006, sivut 4-6.

ArW08 Ariyachandra, T., Watson, H., Technical opinion: which data warehouse architecture is best? *Communications of the ACM,* 51(10), 2008, sivut146-147.

ArW10 Ariyachandra, T., Watson, H., Key organizational factors in data warehouse architecture selection. *Decision Support Systems,* 9(2), 2010, sivut 200-212.

Bar02 Barabási, A-L., *Linked: The New Science Of Networks.* Perseus Books Group, New York, 2002.

Bar90 Barker, R., *CASE Method: Entity Relationship Modelling*. Addison-Wesley, Boston, 1990.

BCN92 Batini, C., Ceri, S., Navathe, S. B., *Conceptual Database Design An Entity-Relationship Approach*. The Benjamin/Cummings Publishing Company, Redwood City, 1992.

Bre04 Breslin, M., Data warehousing battle of the giants. *Business Intelligence Journal* 9(1), 2004, sivut 6-20.

CaT98 Cabibbo, L., Torlone, R., A logical approach to multidimensional databases. *Proceedings of the 6th International Conference on Extending Database Technology,* Valencia, 1998, sivut 183-197.

CCS93 Codd, E. F., Codd, S. B., Salley, C. T., Providing OLAP to user-analysts: an IT-mandate. Tekninen raportti, Hyperion Solutions, Sunnyvale, CA, 1993, http://www.minet.uni-jena.de/dbis/lehre/ss2005/sem_dwh/lit/Cod93.pdf. [10.04.2013].

CDN11 Chaudhuri, S., Dayal, U., Narasayya, V., An overview of business intelligence technology. *Communications of the ACM*, 54(8), 2011, sivut 88-98.

ChD97 Chaudhuri, S., Dayal, U., An overview of data warehousing and OLAP technology.

SIGMOD Record, 26(1), 1997, sivut 65-74.

Che76 Chen, P. S., The entity-relationship model: toward a unified view of data. *ACM Transactions on Database Systems (TODS) - Special issue: papers from the international conference on very large data bases*, 1(1), 1976, sivut 9-36.

CoB02 Connolly, T. M., Begg, C., *Database Systems: A Practical Approach to Design, Implementation, and Management, Third Edition*. Addison-Wesley, Harlow, 2002.

CSL06 Corral, K., Schuff, D., Louis R. D. St., The impact of alternative diagrams on the accuracy of recall: A comparison of star-schema diagrams and entity-relationship diagrams. *Decision Support Systems*, 42(1), 2006, sivut 450–468.

DFS05 Dori, D., Feldman R., Sturm, A., Transforming an Operational System Model to a Data Warehouse Model: A Survey of Techniques. *Proceedings of the International Conference on Software – Science, Technology and Engineering*, Herzelia, 2005, sivut 47-56.

FrK04 Franconi, E., Kamble, A., The GMD data model and algebra for multidimensional information. *Proceedings of the 16th International Conference, CAiSE 2004*, Riga, 2004, sivut 446-462.

Gol09 Golfarelli, M., From user requirements to conceptual design in data warehouse design - a survey. Teoksessa *Data Warehousing Design and Advanced Engineering Applications: Methods for Complex Construction,* IGI Global, 2009, sivut 1-16.

GMR98 Golfarelli, M., Maio, D., Rizzi, S., The dimensional fact model: a conceptual model for data warehouses. *International Journal of Cooperative Information Systems,* 7(2-3), 1998, sivut 215-247.

Hal01 Halpin, T. A., *nformation Modeling and Relational Databases: From conceptual analysis to logical design using ORM with ER and UML.* Morgan Kaufmann, New York, 2001.

HHK09 Hovi, A., Hervonen, H., Koistinen, H., *Tietovarastot ja Business Intelligence, Ensimmäinen Painos.* WSOY, Jyväskylä, 2009.

HLV00 Hüsemann, B., Lechtenborger, J., Vossen, G., Conceptual data warehouse modeling. *Proceedings of the Second Intl. Workshop on Design and Management of Data Warehouses,* Stockholm, 2000, sivu 6.

HSK06 Hein, 0, Schwind, M., König, W., Scale-free networks. *Wirtschaftsinformatik,* 48(4), 2006, sivut 267-275.

Hug12 Hughes, R., *Agile Data Warehousing Project Management Business Intelligence Systems Using Scrum.* Morgan Kaufmann, New York, 2012.

Inm00 Inmon, W. H., The problem with dimensional modeling. *Information Management,* 2000(5).

Imh01 Imhoff, C., Oper marts – an evolution in the operational data store. *Information Management,* 2001(7).

Inm02 Inmon, W. H., *Building the Data Warehouse, Third Edition*. John Wiley & Sons, New York, 2002.

Inm07 Inmon, W. H., Corporate information factory (CIF) overview. Inmon Consulting Services, 2007, http://www.inmoncif.com/library/cif/1 [10.04.2013].

Inm92 Inmon, W. H., *Building the Data Warehouse, First Edition*. John Wiley & Sons, New York, 1992.

ISN08 Inmon, W. H., Strauss, D., Neushloss, G., *DW 2.0 The Architecture for the Next Generation of Data Warehousing*. Morgan Kaufmann, New York, 2008.

JBK12 Jovanovic, V., Bojicic, I., Knowles, C., Pavlic, M., Persistent staging area models for data warehouses. *Issues in Information Systems*, 13(1), 2012, sivut 121-132.

JiA04 Jindal, R., Acharya, A., Federated data warehouse architecture. Tekninen raportti, 2004(5), http://hosteddocs.ittoolbox.com/Federated%20data%20Warehouse%20Architecture.pdf [10.04.2013].

JKP02 Jensen, C. S., Kligys, A., Pedersen, T. B., Timko I., Multidimensional data modeling for location-based services. *Proceedings of the 10th ACM International Symposium on Advances in Geographic Information Systems*, 2002, sivut 55-61.

JLV03 Jarke, M., Lenzerini, M., Vassiliou, Y., Vassiliadis, P., *Fundamentals of Data Warehouses, Second Edition.* Springer, Berlin, 2003.

JoB12 Jovanovic, V., Bojocic I., Conceptual data vault model. *Proceedings of the Southern Association for Information Systems Conference*, Atlanta, 2012, sivut 131-136.

Juk06 Jukic, N., Modeling strategies and alternatives for data warehousing projects. *Communications of the ACM* , 49(4), 2006, sivut 83-88.

Kam08 Kamble, A. S., A conceptual model for multidimensional data. *Proceedings of the Fifth Asia-Pacific Conference on Conceptual Modelling (APCCM 2008),* Wollongong, 2008, sivut 29-38.

KiC04 Kimball, R., Caserta, J., *The Data Warehouse ETL Toolkit Practical: Techniques for Extracting, Cleaning, Conforming, and Delivering Data, Second Edition*. Wiley Publishing, Indianapolis, 2004.

KiM00 Kimball, R., Merz, R., T*he Data Webhouse Toolkit: Building the Web-Enabled Data Warehouse*. JohnWiley & Sons, New York, 2000.

Kim02 Kimball, R., Design tip #34: you don't need an EDW. Kimball Group, 2002(2), http://www.kimballgroup.com/2002/02/28/design-tip-34-you-dont-need-an-edw [10.04.2013].

Kim96 Kimball, R., *The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses.* JohnWiley & Sons, New York, 1996.

Kim97 Kimball, R., A dimensional modeling manifesto. *DBMS - Special issue on data warehousing archive,* 10(9), 1997, sivut 58 - 70.

KiR02 Kimball, R., Ross, M., T*he Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, Second Edition*. JohnWiley & Sons, New York, 2002.

KRT08 Kimball, R., Ross, M., Thornthwaite, W., Mundy, J., Becker, B., *The Data Warehouse Lifecycle Toolkit: Practical techniques for Building Data Warehouse and Business Intelligence systems, Second Edition.* Wiley, Indianapolis, 2008.

LAW98 Lehner, W., Albrecht, J.,Wedekind, H., Normal forms for multidimensional databases. *Proceedings of the 10th International Conference on Scientific and Statistical Database Management,* Capri, 1998, sivut 63-72.

LeL03 Levene, M., Loizou, G., Why is the snowflake schema a good data warehouse design? *Information Systems,* 28(3), 2003, sivut 225-240.

LeV03 Lechtenbörger, J., Vossen, G., Multidimensional normal forms for data warehouse design. *Information Systems,* 28(5), 2003, sivut 415-434.

Lin02 Linstedt, D., Data vault series 1 - data vault overview. *The Data Administration Newsletter*, Robert S. Seiner, 2002(7), http://www.tdan.com/view-articles/5054 [10.04.2013].

Lin04 Linstedt, D., Data vault series 4 - Link Tables. *The Data Administration Newsletter*, Robert S. Seiner, 2004(1), http://www.tdan.com/view-articles/5172 [10.04.2013].

Lin05 Linstedt, D., Data vault series 5 - loading practices. *The Data Administration Newsletter*, Robert S. Seiner, 2002(7), http://www.tdan.com/view-articles/5285 [10.04.2013].

Lin10a Linstedt, D., DV modeling specification v1.0.9. 2010 (5), http://danlinstedt.com/datavaultcat/standards/dv-modeling-specification-v1-0-8 [10.04.2013].

Lin10b Linstedt, D., Data vault - what is it Exactly? 2010 (3), http://www.b-eye-network.com/blogs/linstedt/archives/2010/03/data_vault_-_wh.php [10.04.2013].

Lin10c Linstedt, D., Data vault basics 2010 (6), http://danlinstedt.com/about/data-vault-basics [10.04.2013].

Lin10d Linstedt, D., Graziano K., Hultgren H., *The New Business Supermodel: The business of Data Vault Data Modeling, Second Edition*. Lulu.com, 2010.

Lin11a Linstedt, D., *Super Charge Your Data Warehouse: Invaluable Data Modeling Rules to Implement Your Data Vault*. CreateSpace Publishing Platform, 2011.

Lin11b Linstedt, D., Hardcore table comparisons: dimensional and data vault. 2011 (08), http://danlinstedt.com/datavaultcat/hardcore-table-comparisons-dimensional-and-data-vault [10.04.2013].

Lin11c Linstedt, D., Data vault users. 2011, http://danlinstedt.com/about/dv-customers [10.04.2013].

Mar04 Martyn, T., Reconsidering multi-dimensional schemas. *SIGMOD Record*, 33(1), 2004, sivut 83-88.

MoK00 Moody, D. L., Kortink, A. R., From enterprise models to dimensional models: a methodology for data warehouse and data mart design *Proceedings of the Second Intl. Workshop on Design and Management of Data Warehouses,* Stockholm, 2000.

MoK03 Moody, D. L., Kortink, A. R., From ER models to dimensional models: bridging the gap between OLTP and OLAP design. *Business Intelligence Journal* , 8(3), 2003.

MVS03 Maniatis, A. S., Vassiliadis, P., Skiadopoulos, S., Vassiliou, Y., CPM: a cube presentation model for OLAP. *Proceedings of the Data Warehousing and Knowledge Discovery, 5th International Conference,* Prague, 2003, sivut 4-13.

MYB08 Mishra, D., Yazici, A., Basaran, B.P., A casestudy of data models in data warehousing. *Proceedings of the First International Conference on the Applications of Digital Information and Web Technologies,* Ostrava, 2008, sivut 314 - 319.

NNH08 Nazri, M. N. M., Noah, S. A. M., Hamid, Z., Automatic Data Warehouse Conceptual Design. *Proceedings of the International Symposium on Information Technology,* Kuala Lumpur, 2008, sivut 1 – 7.

NNT00 Niemi, T., Nummenmaa, J., Thanisch, P., Functional Dependencies in Controlling Sparsity of OLAP Cubes. *Proceedings of the Second International Conference on Data Warehousing and Knowledge Discovery,* London, 2000, sivut 199-209.

Pei03 Pei, J., A general model for online analytical processing of complex data. *Proceedings of the 22nd International Conference on Conceptual Modeling*, Chicago, 2003, sivut 321-334.

PJD01 Pedersen, T. B., Jensen, C. S., Dyreson, C. E., A foundation for capturing and querying complex multidimensional data. *Information Systems*, 26(5), 2001, sivut 384-423.

RoA06 Romero, O., Abelló, A., Multidimensional Design by Examples. *Proceedings of the 8th International Conference on Data Warehousing and Knowledge Discovery*, Krakow, 2006, sivut 85-94.

RoA09 Romero, O., Abelló, A., A survey of multidimensional modeling methodologies. *International Journal of Data Warehousing & Mining*, 5(2), 2009, sivut 1-23.

RRB10 Rönnbäck, L., Regardt, O., Bergholtz, M., Wohed, P., Anchor modeling — agile information modeling in evolving data environments. *Data & Knowledge Engineering*, 69(12), 2010, sivut 1229–1253.

SBH99 Sapia, C., Blaschka, M., Hofling, G., Dinter, B., Extending the E/R model for the multidimensional paradigm. *Proceedings of the ER '98 Workshops on Data Warehousing and Data Mining*, Singapore, 1999, sivut 105-116.

SCT11 Schuff, D., Corral, K., Turetken, O., Comparing the understandability of alternative data warehouse schemas: An empirical study. *Decision Support Systems*, 52(1), 2011, sivut 9–20.

SeS05 Sen, A., Sinha, A. P., A comparison of data warehousing design methodologies. *Communications of the ACM*, 48(3), 2005, sivut 79-84.

Sin10 Singh, J., Promulgation of contemporary testing techniques for effective engineering of data warehouses. Doctoral thesis, University college of engineering, Faculty of engineering & technology, Punjabi University, Patiala, India, 2010(7), http://shodhganga.inflibnet.ac.in/handle/10603/2099 [10.04.2013].

SKS11 Silberschatz, A., Korth, H. F, Sudarshan, S., *Database System Concepts, Third Edition*. McGraw-Hill, New York, 2011.

TBC99 Tryfona, N., Busborg, F., M., Christiansen, J. G. B., StarER: a conceptual model for data warehouse design. *Proceedings of the 2nd ACM International Workshop on Data warehousing and OLAP,* Kansas City, 1999, sivut 3-8.

Tho07 Thornthwaite, W., Kimball university: pick the right approach to MDM. *InformationWeek,* 2007(2), http://www.informationweek.com/software/business-intelligence/kimball-university-pick-the-right-approa/197004155 [10.04.2013].

TKS01 Tsois, A., Karayannidis, N., Sellis, T. K., MAC: Conceptual data modeling for OLAP. *Proceedings of the 3rd Intl. Workshop on Design and Management of Data Warehouses, Interlaken*, 2001, sivu 5.

TPK01 Trujillo, J., Palomar, M., Gómez, J., Designing data warehouses with OO conceptual Models. *Computer,* 34(12), 2001, sivut 66-75.

VaS00 Vassiliadis, P., Skiadopoulos, S., Modelling and optimisation issues for multidimensional databases. *Proceedings of the 12th International Conference on Advanced Information Systems Engineering,* Stockholm, 2000, sivut 482-497.

VaS99 Vassiliadis, P., Sellis, T., A survey of logical models for OLAP databases. *SIGMOD Record*, 28(4), 1999, sivut 64-69.

VBR00 Vrdoljak, B., Banek, M., Rizzi, S., Designing Web Warehouses from XML Schemas. *Proceedings of the 5th International Conference on Data Warehousing and Knowledge Discovery,* Prague, 2003, sivut 89-98.

VKS12 Vujoševic, D., Kovacevic, I., Suknovic, M., Lalic N., A comparison of the usability of performing ad hoc querying on dimensionally modeled data versus operationally modeled data. *Decision Support Systems*, 54(1), 2012, sivut 185–197.

WaA05 Watson, H. J., Ariyachandra, T., Factors in the selection decision and the success of architectures. Tekninen raportti, University of Georgia, Terry College of Business, 2005(7), http://www.terry.uga.edu/~hwatson/DW_Architecture_Report.pdf. [10.04.2013].