LAPPEENRANNAN TEKNILLINEN YLIOPISTO
Tietotekniikan osasto

Diplomityö

TEHDASMITTAUSTEN VARASTOINTI MONIULOTTEISELLA TIETOMALLILLA

Työn tarkastajat ovat Professori Kari Smolander ja Diplomi-insinööri Sampo Luukkainen
Työn ohjaaja on Professori Kari Smolander

Lappeenrannassa 10. joulukuuta 2007

Niko Sten
Panssarikatu 2 B 17
53850 Lappeenranta
Puhelin +358 44 040 2821

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
Teknillistaloudellinen tiedekunta
Tietotekniikan osasto

Niko Sten

Tehdasmittausten varastointi moniulotteisella tietomallilla

Diplomityö

2007

92 sivua, 23 kuvaa, 5 taulukkoa ja 5 liitettä

Tarkastajat: Professori Kari Smolander
Diplomi-insinööri Sampo Luukkainen

Hakusanat: tehdas, mittaus, tietokanta, tietovarasto, tähtimalli, moniulotteinen, tietomalli

Keywords: mill, measurement, database, data warehouse, star schema, multidimensional, data model

Tietovarastoissa moniulotteinen tietomalli on tehokkain tapa esittää tietoa päätöksentekijöille. Sen toimivuus on hyväksi havaittu monissa eri liiketoimintaympäristöissä. Tehdasympäristöissä on tuhansia mittalaitteita, joista jokainen mittaa uniikkia valmistusprosessiin liittyvää piirrettä. Tässä työssä kehitettiin tietovarasto tehdasmittausten varastointiin käyttäen moniulotteista tietomallia. Havaittiin, että moniulotteisella mallilla tehdasmittaukset voidaan tallentaa joustavalla tavalla ja esittää käyttäjälle mielekkäässä muodossa. Moniulotteinen malli antaa myös erinomaiset keinot tiedon ryhmittelyyn ja vertailuun. Sillä ei kuitenkaan saada vastaavanlaisia hyötyjä kuin klassisissa kaupanalan tietovarastointi esimerkeissä, koska eri mittaukset ovat keskenään hyvin erilaisia. Vaikka mittaukset eivät olekaan aina vertailtavissa tai summattavissa keskenään, saadaan ne moniulotteisella mallilla tallennettua ja luokiteltua loogisesti siten, että käyttäjän on helppo löytää tarvitsemansa tieto. Lisäksi yleisesti tunnettu ja paljon käytetty tietovaraston suunnittelumalli takaa sen, että markkinoilta on saatavissa työkaluja tietovaraston käyttöön. Tietokannan toteutus tehtiin vapaasti levitettävän MySQL-tiedonhallintajärjestelmän avulla. Sitä ei ole suunniteltu pääasiassa tietovarastokäyttöön, mutta halpa lisenssi ja hyvä skaalautuvuus tekevät siitä mielenkiintoisen vaihtoehdon. Sitä onkin käytetty luultua enemmän tietovarastoinnissa ja myös monien nimekkäiden organisaatioiden toimesta. Myös tässä työssä todettiin, että MySQL tarjoaa riittävät välineet tietovaraston kehittämiseen.

ABSTRACT

Lappeenranta University of Technology
Faculty of Technology Management
Department of Information Technology

Niko Sten

Mill measurement warehousing with multidimensional model

Thesis for the Degree of Master of Science in Technology

2007

92 pages, 23 pictures, 5 tables and 5 appendices

Examiners: Professor Kari Smolander
Master of Science Sampo Luukkainen

Keywords: mill, measurement, database, data warehouse, star schema, multidimensional, data model

In data warehouses multidimensional model is the most efficient way to represent data for decision makers. Its function has been approved in many different business environments. Mill environment contains thousands of measurement devices, every one measuring unique feature related in manufacturing process. In this work mill measurement warehouse is developed with use of multidimensional model. It seems that with multidimensional model it's possible to store mill measurements in very flexible manner and represented them for user in sensible form. Multidimensional model gives excellent means for grouping and comparing measurements. Benefits don't match the ones seen in sales transaction warehousing examples because every measurement is unique. Even though measurements are not always straight forwardly comparable or additive, they can be stored and categorized with multidimensional model so that, for user it is easy to find information that is needed. Also generally accepted and much used data warehouse design pattern secures that there are tools in the market for use of data warehouse. Implementation of data warehouse was made on freely distributed MySQL platform. MySQL is not designed mainly for data warehousing purposes, but its cost effective license and good scalability makes it interesting alternative. It has been used in data warehousing more than one would assume and also by quite big organizations. Also in this work it was noticed that MySQL gives adequate tools for data warehousing.

ALKUSANAT

Tämä diplomityö on tehty Stora Enson selluntutkimusyksikölle. Haluan kiittää työni tarkastajia Sampo Luukkaista ja Kari Smolanderia heidän antamastaan rakentavasta palautteesta. Osa työn aineksesta perustuu tietoon, joka kertyi useista kahvipöydässä ja työmatkalla käydyistä keskusteluista. Näistä keskusteluissa sain tietoa mm. sellunvalmistusprosessista, tehtaiden automaatiojärjestelmistä, organisaation työskentelytavoista ja tietotarpeista. Nämä tiedot ovat olleet edellytys työni onnistumiselle ja siksi haluankin kiittää kaikkia kanssani samassa kahvipöydässä ja autossa istuneita henkilöitä.

Erityiskiitokset haluan osoittaa vaimolleni Matleenalle, joka on hoitanut osuutensa arjessa ja siten mahdollistanut opiskeluni ja pojalleni Eelikselle, joka on tehnyt arjesta paljon hauskempaa.

Niko Sten 10.12.2007

SISÄLLYSLUETTELO

TERMIEN SELITYKSET

BI ***Business Intelligence***

Liiketoimintatiedon hallinta on systemaattista yrityksen suorittamaa liike-elämän tietojen hankintaa ja analysointia

CSV ***Comma Separated Value(s)***

Yksinkertaisen taulukkodatan tallennustapa, jossa tieto tallennetaan tekstitiedostoon pilkuilla erotettuna.

DBMS ***Database Management System***

Suom. tiedonhallintajärjestelmä. Järjestelmä jonka avulla tietokanta voidaan luoda ja jonka avulla tietokantaa voidaan ylläpitää ja käyttää.

DDL ***Data Definition Language***

Tarkoittaa kieltä, jolla voidaan määritellä tiedon loogisia rakenteita (tietokannan ulkoinen kuvaus).

DMQL ***Data Mining Query Language***

SQL kieleen perustuva OLAP operaatioita tukeva kyselykieli.

SDL ***Storage Definition Language***

Tarkoittaa kieltä, jolla voidaan määritellä tiedon tallennusrakenteita (tietokannan sisäinen kuvaus).

Dimensiotaulu Taulu jonka avulla faktataulun raakatietoa voidaan luokitella ja rajata.

ETL ***Extract, Transform, Load***

Tietovaraston taustaprosessit, jotka tuovat tiedon lähdetietokannasta kohdetietokantaan.

Faktataulu Taulu johon tallennetaan raaka tieto moniulotteisessa tietomallissa.

FLC ***Fuzzy Logic Controller***

Sumeaan logiikkaan perustuva säätöjärjestelmä.

FTP ***File Transfer Protocol***

Eräs tiedostonsiirtoprotokolla.

MOLAP ***Multidimensional OLAP***

OLAP-järjestelmä, joka on toteutettu moniulotteisella tietokannalla.

ODBC ***Open Database Connectivity***

Eräs rajapinta, jonka avulla sovellusohjelmat voivat käyttää tietokantaa.

OLTP ***Online Transactional Processing***

Toiminnallisessa tietokannassa käytettävä tiedonkäsittelytapa.

OLAP ***Online Analytical Processing***

Tietovarastoissa käytettävä tiedonkäsittelytapa.

Paikallisvarasto Tietovaraston osakokonaisuus. Vastaa englannin kielistä termiä Data mart.

Puskuripalvelin Termi, jota on käytetty tässä työssä kuvaamaan sitä palvelinta, jossa tietoa säilytetään väliaikaisesti kunnes se voidaan siirtää julkisesti saataville. Englannin kielessä käytetään yleensä termiä staging area. Staging area tarkoittaa yleensä paikkaa jossa laitteet kootaan eri osista.

RDBMS ***Relational Database Management System***

Tiedonhallintajärjestelmä, jonka looginen tiedon kuvaus tehdään relaatiomallin mukaisesti.

ROLAP ***Relational OLAP***

OLAP-järjestelmä, joka on toteutettu relaatiotietokannan ja ulkoisen OLAP-moduulin avulla.

SCD ***Slowly Changing Dimension***

Viittaa tekniikoihin, joilla hallitaan dimensioihin kohdistuvia muutoksia.

SQL ***Structured Query Language***

Standardoitu kyselykieli tietorakenteiden luomista (DDL), tallentamista (DSL) ja tiedon hakua varten.

UoD ***Universe of Discource***

Se tosimaailman osa, jota tietokanta kuvaa.

XML ***eXtensible Markup Language***

Merkkauskieli, jolla tiedon merkitys on kuvattavissa tiedon sekaan.

1 JOHDANTO

Yksi yritysten tärkeimmistä voimavaroista on heidän hallussaan oleva tieto. Tehdasympäristön tuhannet mittalaitteet tuottavat vuosittain suuren määrän raakaa tietoa, josta voidaan jalostaa tietämystä. Jalostus alkaa tiedon hakemisella ja rajaamisella joka on monimutkaista, jos erilaisia järjestelmiä on paljon. Tässä diplomityössä kehitettiin sellunvalmistusprosessin mittauksille tietokanta, joka piilottaa taakseen eri järjestelmiä siten, että käyttäjä pääsee kaikkeen mittaustietoon käsiksi yhden yhtenäisen lähteen kautta. Työn tilasi Stora Enson selluntutkimusyksikkö, jonka tavoitteena on kemiallisen sellun tuotannon ja kuidun hyödyntämisen tehostaminen.

1.1 Tarve tiedon varastointiin

Tieto on tutkimuksessa välttämätöntä ja yleensä tutkimus aloitetaankin tiedon keräämisellä. Sellutehtailla on kymmeniä tuhansia mittareita, jotka tuottavat jatkuvasti lisää tietoa. Tutkimusta varten on siis olemassa paljon materiaalia. Ongelmana on, kuinka siihen päästään käsiksi ja kuinka sitä hallitaan. Erilaisia järjestelmiä tiedon keräämiseen on paljon ja esimerkiksi jokaisella tehtaalla on omat työkalunsa tiedon keräämiseen, jotka voivat myös vaihdella sen mukaan, mitä tietoa halutaan. Tutkijat joutuvat usein itse selvittämään, mistä tutkimuksessa tarvittava tieto saadaan. Apuna tässä joudutaan käyttämään prosessikaavioita, joista selvitetään tarvittavien mittareiden positionimet. Lopulta, kun mittarien nimet tiedetään ja tieto saadaan jonkin järjestelmän kautta haettua, voi se olla hyvin puutteellista. Aina ei edes tiedetä haetun datan mittayksiköitä. Tutkijat joutuvat tekemään paljon työtä tämänkaltaisten asioiden selvittämiseen ja olemaan tiiviisti yhteydessä tehtaisiin, joissa mittareista yleensä tiedetään enemmän.

Tutkimuksen avulla voidaan kehittää tuotantoprosessin eri vaiheisiin uusia tehokkaampia menetelmiä joiden vaikutukset ovat yleensä pysyviä. Tuotantomäärät ovat niin suuria, että pienilläkin parannuksilla saadaan huomattavia hyötyjä. Jotta tutkijat voisivat keskittyä paremmin siihen työhön, jonka he osaavat parhaiten, täytyy heille tarjota helpompi pääsy siihen materiaaliin, jota he tutkimuksissaan tarvitsevat. Tiedon keräämisen ei välttämättä tarvitse olla hankalaa.

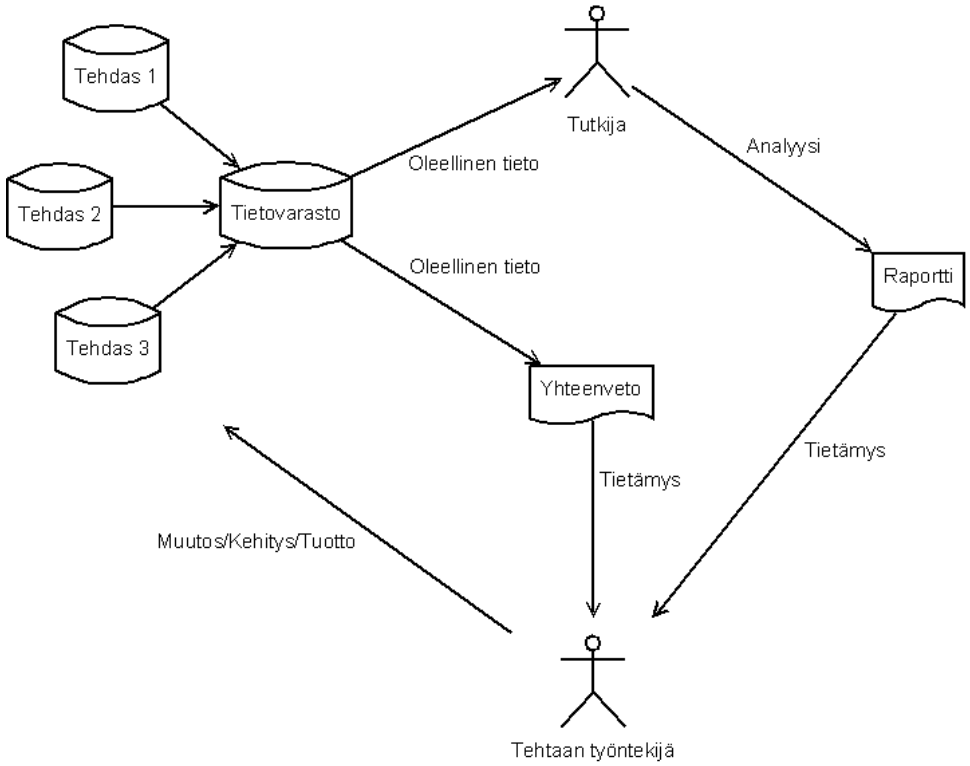
Tietovarasto piilottaa käyttäjiltä erilaiset tietolähteet ja esittää niiden sisältämän tiedon yhtenäisessä ja helposti selattavassa muodossa. Koska tietovarasto on tarkoitettu tiedon julkaisua varten, on tietokin tallennettu helposti ymmärrettävään muotoon. Lisäksi tietovaraston tiedoista on jo valmiiksi poistettu puutteelliset ja väärät tiedot. Tällöin tiedon käyttäjä pääsee suoraan keskittymään tiedon analysointiin.

Yhtenäisen mittausvaraston tarve on huomattu selluntutkimusyksikössä jo aiemmin ja sellainen on myös kehitetty. Se kehitettiin alun perin Sampo Luukkaisen ja Marko Harisen kehittämien uusien mittareiden datan julkaisuun, mutta myöhemmin sitä laajennettiin myös muiden tehdasmittausten varastointiin. Tietovarasto on koettu niin hyödylliseksi, että sitä on täytynyt laajentaa jatkuvasti yhä enemmän. Useat laajennukset ja parannukset ovat kuitenkin sekavoittaneet tietokannan rakennetta, jota ei ole alun perin suunniteltu käytettäväksi niin laajassa mittakaavassa. Tietokanta on koettu hyödylliseksi ja tietoa tulee jatkuvasti yhä nopeammin lisää. Jotta tietokanta voisi vastata myös tulevaisuuden vaatimuksiin, tarvittiin tutkimus, jossa pohditaan tiedon rakennetta uudelleen.

1.2 Työn kohde ja tavoite

Työn päätavoitteena on kehittää tietovarastoon yleinen tietorakenne, jota voidaan soveltaa kaikkien mittausten tallentamiseen. Sen täytyy skaalautua tukemaan uusia mittauksia ja kestää mittausympäristössä tapahtuvat muutokset. Järjestelmän avulla kaikki tiedon hakeminen pitäisi onnistua yhtenäisellä tavalla, jolloin myös tietokannan käytön pitäisi helpottua ja nopeutua. Tässä työssä tietokannan toteutus tehdään MySQL-tiedonhallintajärjestelmällä, joka on tällä hetkellä suosituin vapaan lähdekoodin relaatiotietokanta. Työssä arvioidaan myös MySQL-tietokannan ja moniulotteisen tietomallin soveltuvuutta tehtaiden prosessitiedon varastointiin.

Kuvassa 1 on esitetty tietovaraston käytön tavoitetilanne. Tiedot julkaistaan käyttäen tietovarastoa, jolloin jokaisen tehtaan tiedot ovat saatavilla yhden rajapinnan kautta. Tämä helpottaa tiedon hakemista, vertailua ja raporttien muodostamista. Mittaustiedoista kerättyä ja analysoitua tietämystä pyritään siirtämään tehtaan työntekijöille, jotka voivat hyödyntää uutta tietoa ja näin myös omalta osaltaan parantaa prosessia. Tietokannasta voidaan myös automaattisesti muodostaa yhteenvetoraportteja, joissa esiintyvät oleelliset tunnusluvut.



Kuva 1. Mittaustiedon hyödyntäminen ja tietämyksen välittäminen

1.3 Työn rakenne

Työn suoritus on jaettu kahteen suurempaan osakokonaisuuteen. Ensimmäiset kappaleet muodostavat teoreettisen osan, jossa perehdytään toimintaympäristöön sekä yleisesti tietokantoihin ja tietovarastoihin. Loppuosa työstä kuvaa tietovaraston kehityksen aina tietokannan suunnittelusta käyttöönottoon asti.

Teoreettisen osan alussa selvitetään millaista ympäristöä tietovarasto kuvaa ja mitä vaatimuksia ja rajauksia sillä on. Kohdeympäristö esitellään suoraan johdannon jälkeen, koska näin päästään johdantoa tarkemmin perille siitä, mitä tässä työssä käsitellään. Selvitetään mm. millainen organisaatio on kyseessä ja minkälaisia tehtaiden automaatiojärjestelmät ovat yleensä. Selvitetään myös käyttäjien tietotarpeet ja kuinka suuria tietomääriä täytyy varastoida ja hallinnoida. Lisäksi tutustutaan tämänhetkiseen tietovarastoon ja arvioidaan sen heikkouksia ja vahvuuksia. Ennen pureutumista tietovarastoihin esitellään yleisesti tietokantojen tärkeimpiä käsitteitä ja sitä kuinka tietokantoja yleensä suunnitellaan. Tämä antaa aineksia myöhemmille kappaleille ja kontrastia tietovaraston suunnittelumenetelmille ja kehitysprosessille.

Tämän jälkeen kerrotaan tarkasti tietovarastoista ja niiden kehittämisestä relaatiotietokannoissa. Teoreettisen osan päätteeksi kerrotaan MySQL:n soveltuvuudesta tiedon varastointiin.

Tietokannan toteutusvaihe on jaettu kahteen osaan: tietovaraston rakenteen suunnitteluun ja tietokannan toiminnan suunnitteluun. Rakenteellisessa suunnitteluvaiheessa suunnitellaan tietokantajärjestelmän arkkitehtuuri, käsitteellinen malli, looginen malli ja fyysinen malli. Arkkitehtuurisuunnittelussa tieto ja toiminnallisuus jaetaan loogisiin osakokonaisuuksiin. Käsitteellisessä suunnittelussa selvitetään käsitekaavioiden avulla tehdasympäristössä vallitsevat käsitteet ja niiden väliset suhteet. Käsitekaavioiden tietoa hyödynnetään loogisen tietomallin suunnittelussa, jossa käsitteelliset tietomallit denormalisoidaan tietomallin vaatimalla tavalla. Lopuksi loogisen mallin rakenteisiin valitaan sopivat MySQL-tallennusrakenteet.

Toiminnallinen suunnittelu näyttelee pienempää roolia tässä työssä ja se sisältää tietokannassa tarvittavien prosessien ja kyselyiden suunnittelun yleisellä tasolla. Toiminnallisuudesta toteutetaan vain tärkeimmät osat, jotta tietorakennetta voidaan testata. Suunnitteluvaiheiden jälkeen tietokanta toteutetaan ja siihen tuodaan tietoa vanhasta järjestelmästä, jonka jälkeen sitä testataan ja arvioidaan.

2 TEHTÄVÄN RAJAUS

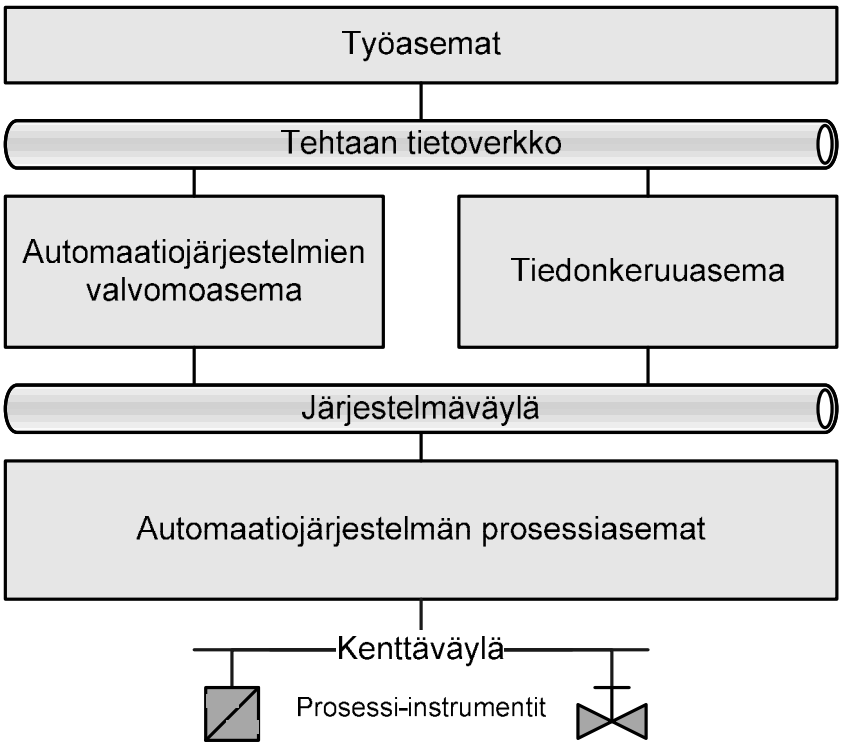
Tietovaraston kehitysprojektissa täytyy ymmärtää käyttäjäorganisaation liiketoimintamalli, jotta tietovarasto palvelisi organisaation päätöksentekijöitä. Käyttäjäorganisaatio on Stora Enson selluntutkimusyksikkö, joka tarvitsee tietovarastoa sellunvalmistusprosessin mittaustietojen tallentamiseen. Tässä kappaleessa on tarkemmin perehdytty organisaatioon, nykyiseen tietojärjestelmään ja selvitetty kuinka paljon tietoa järjestelmän tulee kyetä taltioimaan.

2.1 Käyttäjäorganisaatio ja sellutehtaan automaatiojärjestelmä

Stora Enso on paperin ja kartongin tuotannon markkinajohtaja. Sillä on tehtaita kaikilla mantereilla, joissa se valmistaa sellua, paperia, kartonkia ja muita puuvalmisteita (Stora Enso kotisivut, Mills). Stora Enson noin 15 sellutehdasta tuottavat yhteensä noin 5 – 6 miljoonaa kiloa sellua vuodessa (Stora Enso kotisivut, Pulp). Tehtailla tuotantoprosessia valvotaan ja ohjataan tehdasmittausten avulla. Yhdellä sellutehtaalla voi olla kymmeniä tuhansia mittalaitteita, jotka tuottavat jatkuvasti lisää tietoa. Tätä tietoa halutaan käyttää ohjauksen ja valvonnan lisäksi apuna myös tutkimuksessa. Tutkimuksen avulla kehitetään uusia menetelmiä sellun laadun parantamiseksi ja tuotantomäärien kasvattamiseksi. (KnowPulp, Prosessin hallinta)

Kuvassa 2 on esitetty tyypillinen sellutehtaan tietoverkon rakenne. Mittalaitteet on kytketty automaatiojärjestelmiin, jotka ohjaavat prosessia mittausten perusteella. Automaatiojärjestelmän toimintaa valvotaan valvomoasemalla, jossa voidaan puuttua myös tuotantoprosessiin. (KnowPulp, Prosessin hallinta)

Tiedonkeruuasema kerää mittaustietoa myöhempää analysointia varten. Se on paikallinen tietovarasto, jota käytetään pääasiassa vain tehtaalla. Tutkimuskäytössä tarvitaan eri tehtailta kerättyä mittaustietoa. Koska jokaisella tehtaalla on omat paikalliset tiedonkeruuasemansa, jotka ovat usein toteutettu myös erilaisilla järjestelmillä, joudutaan tiedonhaussa käyttämään useita eri työkaluja.



Kuva 2. Tehtaan tietoverkko

2.2 Nykyinen mittaustietovarasto

Tietoa on alettu keräämään yhteen selluntutkimusyksikön omaan tietokantaan, jotta tieto olisi helpommin löydettävissä. Tiedonhaku helpottuu, kun kaikki tieto saadaan yhden liittymän kautta. Nykyiseen mittaustietojärjestelmään kuuluu tietovarasto, johon kootaan eri tehtaiden selluvalmistusprosessista mitattua tietoa sekä käyttöliittymä, jonka avulla tietoa voidaan hakea tietovarastosta. Tietovarastoa hallitaan pääasiassa web-selaimen kautta joukolla PHP-sovelluksia sekä käyttäen phpMyAdmin-työkalua tietokannan suoraan hallintaan.

Tietokannassa mittaukset on luokiteltu tehdassijainnin perusteella tauluihin. Eri mittalaitteiden tuottamat mittaukset tallennetaan rinnakkain sarakkeisiin siten, että jokainen rivi edustaa yhtä ajanhetkeä. Yhdessä taulussa saattaa olla kymmenien tai jopa satojen mittareiden näytteet. Informaatio on taulun ulkopuolella toisessa taulussa, jossa on lueteltu kaikkien mittaustaulujen sarakkeiden ja niissä olevien mittausten yleiset ominaisuudet. Tällainen rakenne aiheuttaa ongelmia muutostilanteissa. Uuden mittaustarpeen ilmetessä mittauksille joudutaan luomaan joko uusia tauluja tai olemassa oleviin tauluihin uusia sarakkeita. Jos mittalaitteen ominaisuudet jotenkin muuttuvat, joudutaan sen näytteet tallentamaan uuteen paikkaan, koska sarakkeen

ominaisuuksien muuttaminen vaikuttaisi vanhoihin ja uusiin näytteisiin. Näin ei ole myöskään mahdollista liittää aikaan sidottua informaatiota mittauskohtaisesti, koska rivillä oleva informaatio jakautuu kaikkien näytteiden kesken. Rivikohtaisena informaationa on kerrottu muun muassa, miltä tehtaalta ja tuotantolinjalta näytteet ovat peräisin. Tämä on kuitenkin ristiriidassa sen kanssa, että mittauksista on jo ulkopuolisessa taulussa kerrottu nämä asiat. Vaikuttaakin siltä, että näillä kentillä on pyritty saamaan rakenteesta dynaamisempaa. Selvästikin tarvittaisiin mahdollisuus sellaiselle informaatiolle, joka voi vaihdella jokaisella näytteellä.

Taulujen ja sarakkeiden nimissä käytetään tehdaspaikan ja mittalaitteen liitäntäpisteen koodinimiä jotka ovat melko kryptisiä. Liitäntäpisteen koodinimeä sanotaan yleisesti positionimeksi. Tietyntyyppisen mittauksen löytäminen tietokannasta on hankalaa. Parhaimmat edellytykset löytää juuri oikea tietoa on heillä, jotka tuntevat positionimet ja niihin liitettyjen mittareiden toiminnan, esimerkiksi kokemuksensa perusteella. Mittareihin liitetty lisäinformaatio on myös erittäin puutteellista. Mittauksia ei voida hakea esimerkiksi niiden suureen tai yksikön perusteella, koska kaikkiin mittareihin ei ole liitetty yksikköä eikä yhteenkään mittariin suuretta. Jokaisesta mittauksesta tiedetään varmuudella vain positionimi ja milloin (aika) mitattiin. Ei tiedetä esimerkiksi miten (mittaustekniikka) ja mitä (mittauksen kohde) mitattiin. Yhteenkään mittariin ei ole liitetty sanallisesti tietoa siitä missä mitattiin. Tiedetään vain millä tehtaalla mittaus on tehty. Tarkempi sijainti täytyy päätellä positionimen ja prosessikaavion avulla. Tiedon epätäydellisyys on seurausta lähdetietokannoista saatavan tiedon epätäydellisyydestä.

Tiedonhakuun tarkoitettu käyttöliittymä on koettu käyttäjien keskuudessa hieman epäselväksi. Käyttöliittymän avulla osutaan helposti tyhjään. Tämä ongelma johtuu osittain tietolähteistä saatavien tietojen epätäydellisyydestä, mutta myös käyttöliittymässä on kehittämisen varaa. Käyttöliittymässä on esimerkiksi väliä optioiden valintajärjestyksellä, joka ei kuitenkaan selviä käyttäjälle intuitiivisesti käyttöliittymästä.

Tietokannassa on ollut melko vähän ongelmia nopeuden ja tallennuskapasiteetin suhteen. Tämä johtuu siitä, että mittaustaulun pääavain koostuu vain yhdestä kentästä. Näin ollen indeksi on pieni ja rivin löytäminen tehokasta. Lisäksi tämä tallennustapa on tilaa säästävä, koska yhdellä infosarakkeella yksilöidään monia näytteitä.

Suurimmat ongelmat tietokannassa ovat olleet sen ylläpidettävyydessä ja käytettävyydessä.

2.3 Tietolähteet

Jokainen tehdas on vastuussa omasta paikallisesta mittaustietokannastaan ja eri tehtailla onkin käytössään erilaisia tietovarastoja. Tehtaiden paikalliset tietovarastot toimivat selluntutkimusyksikön mittaustietovaraston lähdetietokantoina. Koska tietolähteet ovat tietovarastoja, ovat tiedot valmiiksi yksinkertaisessa muodossa ja siten myös helposti muokattavissa yhtenäiseen muotoon.

Lähdejärjestelmissä tieto ei kuitenkaan ole yhtenäistä ja joistain mittalaitteista saadaan hieman enemmän tietoa kuin toisista. Esimerkiksi joistain mittalaitteista tiedetään mittayksikkö, mutta ei kaikista. Lisäksi sama mittayksikkö saattaa olla kuvattu eri mittareille eritavoilla. Esimerkiksi celsius asteissa yksikkö voi olla ’C’, ’Cº’, ’degC’ tai se voi puuttua kokonaan. On helppo muuttaa olemassa olevat tiedot yhtenäisiksi, mutta puuttuvien tietojen löytämiseksi joudutaan tekemään paljon selvitystyötä. Tämän tyyppiset tiedot mittalaitteista ovat kuitenkin käytettävyyden kannalta ehdottoman tärkeitä ja niiden selvittämiseen kannattaa uhrata aikaa.

Mittauksiin on joissain tapauksissa liitetty myös lisätietokenttä, jossa on lyhyt kuvaus mittauksesta. Nykyisessä tietovarastossa käytetäänkin paljon juuri tätä kenttää tiedon hakemiseen. Näin käyttäjä voi hakea vapaasti sanahaulla mittareita. Tässä on kuitenkin se ongelma, että kaikkiin mittareihin ei ole liitetty lisätietoa. Sen lisäksi myöskään lisätietokentissä tieto ei ole yhtenäistä ja ne sisältävät paljon lyhenteitä. Tällöin haku lisätietokenttään ei välttämättä tuota toivottuja tuloksia.

Eri tehtaat tarjoavat erilaisia liityntöjä tietoon. Joiltain tehtailta tieto lähetetään FTP:n (File Transfer Protocol) välityksellä CSV-tiedostomuodossa (Comma Separated Value(s)) ja joihinkin tietokantoihin saadaan yhteys suoraan ODBC-linkin (Open Database Connectivity) avulla, jolloin tietoa voidaan hakea vapaammin. Kaikki tietolähteet eivät kuitenkaan ole tehdastietokantoja. Tehtailla on esimerkiksi tutkimusorganisaation valmistamia mittareita, jotka lähettävät datan suoraan tutkimusorganisaation tietovarastopalvelimelle. Kaikkea ei myöskään voida selvittää online-mittareilla ja monet mittaukset tehdäänkin ihmisten toimesta

projektiluonteisesti. Esimerkiksi joskus mittaukset joudutaan suorittamaan laboratoriossa.

2.4 Käyttäjät ja käyttötavat

Tietokantaa käyttävät pääasiassa tutkijat ja muut päätöksentekijät. Haut kohdistuvat erilaisille aikajänteille käyttötavasta riippuen. Tutkijat saattavat esimerkiksi tehdä tilastollista analyysiä vaikkapa vuoden mittausdatasta. Tietovarastoa käyttävät myös tehtaan työntekijät, joita yleensä kiinnostavat mittareiden viimeisimmät arvot. Tehtaan operatiivinen järjestelmä palvelee heitä useimpien mittausten osalta. Heille täytyykin tarjota pääsy vain niihin mittauksiin, jotka eivät tallennu tehdastietokantaan. Tämän vuoksi joitain mittauksia täytyy päivittää tietovarastoon tiheämmin tai kehittää erillinen operatiivinen järjestelmä näiden tietojen väliaikaiseen taltiointiin.

Tietovarastolla halutaan tarjota tutkijoille mahdollisuus tiedon nopeaan selaamiseen ja analysointiin. OLAP (OnLine Analytical Processing) mahdollistaa tiedon selaamisen lähes reaaliaikaisesti. Käyttäjän ei tarvitse hakea jatkuvasti tietoa uudestaan vaan hän voi esimerkiksi siirtyä tarkastelemaan jo haettua tietoa suoraan yksityiskohtaisemmalta tasolta. Tietovarasto suorittaa taustalla tarvittavan lisätiedon noutamisen. Näin myös eri tehtaiden, mittareiden ja ajanjaksojen vertailu on helpompaa.

Tietovarasto helpottaa myös tiedonlouhintaa *(engl. data mining)*, koska sen avulla louhinnassa käytettävä tietojoukko voidaan rajata helpommin. Tiedonlouhinta on tullut tietokoneiden suoritustehon kasvaessa yhä yleisemmäksi. Tiedon louhinnalla tarkoitetaan merkityksellisen tietämyksen etsimistä suuresta tietomäärästä. Sellunvalmistusprosessista voitaisiin esimerkiksi selvittää eri tuotantovaiheiden ominaisuuksien yhteisvaikutuksia sellunkokonaistuottoon tai laatuun. Louhinta viittaa siihen, että käytettävissä oleva tietomäärä on erittäin suuri, mutta siinä piilevä merkityksellinen osa on hyvin pieni. Tilanne on verrattavissa malmien tai mineraalien louhitaan maa-aineksesta. Tiedonlouhinnassa hakkujen sijaan käytetään tietokoneita ja hahmontunnistusmenetelmiä. (Han & Kamber 2006, s. 5-7)

Koska käyttäjäorganisaatio on monikansallinen yhtiö, täytyy myös pohtia kuinka hallitaan kieli-, kulttuuri- ja sijaintierojen aiheuttamat erot käyttäjien haluamassa tietosisällössä. Käyttäjät saattavat haluta nähdä mittauksissa paikallisen kellonajan ja

informaatiokentät omilla kielillään. Kieliasiat täytyy ottaa rakenteellisesti huomioon, mutta tietosisältöä ei tässä vaiheessa käännetä eri kielille. Ensimmäisessä vaiheessa tietokanta toteutetaan englannin kielellä, koska sitä ymmärretään laajimmin.

2.5 Tietovaraston kasvutarve ja tietomäärät

Organisaatioon kuuluu noin 15 sellutehdasta. Jokaisella tehtaalla voi olla kymmeniätuhansia mittareita. Näytteitä tulee vuosittain miljardeja. Eri mittareilla on erilaisia näytteenottotaajuuksia. Näytteiden kokonaismäärän kasvunopeus eli kokonaistaajuus voidaan laskea summaamalla kaikkien mittareiden näytteenottotaajuudet yhteen kuten yhtälössä 1.

$$f_{tot} = \sum_i f_i \tag{1}$$

Näytteiden kokonaismäärä tietyssä ajassa saadaan kertomalla kokonaistaajuus ajalla, kuten yhtälössä 2.

$$S = f_{tot} \cdot \Delta t \tag{2}$$

Todellinen tilatarve voitaisiin laskea näytteiden kokonaismäärän (S) ja yksittäisen näytteen vaatiman tilan (V_{sample}) tulona, kuten yhtälössä 3.

$$V_{tot} = S \cdot V_{sample} \tag{3}$$

Nykyisessä tietovarastossa mittausten tallennustapa on erittäin tilatehokas, koska yhdessä taulussa on monen mittarin tulokset rinnakkain ja informaatio on jaettu kaikkien mittareiden kesken. Koska mittareiden määrä vaihtelee tauluittain, on yhden näytteen vaatima tila nykyjärjestelmässä vaikea laskea.

Taulukossa 1 on laskettu kaavojen 1 ja 2 avulla, kuinka paljon näytteitä tulee vuosittain tietyllä mittareiden määrällä. Taulukon näytteen jaksonaika ei tarkoita oikeaa mittalaitteen jaksonaikaa vaan oikeastaan tiedon karkeustasoa, mutta tietomäärää laskettaessa sillä on vastaava merkitys. Eri tarkkuustason mittauksia käytetään hieman eri tarkoituksiin ja yleensä tarkempia mittauksia tarvitaan vain lyhyeltä ajanjaksolta. Mittareiden määrät on arvioitu hieman oletettua suuremmiksi sen perusteella, kuinka paljon niitä voitaisiin tarvita enintään lähitulevaisuudessa. Tehtailta olisi saatavilla huomattavasti enemmän tietoa, mutta kaiken tallentaminen ei tässä vaiheessa ole mahdollista. Tietovarasto onkin syytä suunnitella siten, että se

voidaan tulevaisuudessa laajentaa varastoimaan entistä suurempaa osaa kaikesta saatavilla olevasta tiedosta.

Taulukko 1. Näyttemäärän vuosittainen kasvu

Näytejaksonaika	*Mittareita*	*Näytettä / VK*	*Näytettä / KK*	*Näytettä / Vuosi*
1 sek	*100*	60 480 000	262 800 000	3 153 600 000
10 sek	*500*	30 240 000	131 400 000	1 576 800 000
1 min	*2000*	20 160 000	87 600 000	1 051 200 000
10 min	*4000*	4 032 000	17 520 000	210 240 000
1 h	*20 000*	3 360 000	14 600 000	175 200 000
	Yht. näytteitä	118 272 000	513 920 000	**6 167 040 000**

Järjestelmän täytyy kestää noin 6 miljardin näytteen vuosikasvu. Pienikin tavumääräinen ero yhden näytteen vaatimassa tilassa voi tuottaa gigatavujen muutoksen kokonaistilatarpeessa. Tiheämmän näytteenottotaajuuden mittareissa täytyy tarkemmin miettiä, mitä mittauksia tarvitaan, koska esimerkiksi jo yksikin sekuntitason mittari tuottaa noin 32 miljoonaa näytettä vuodessa. Näille myös varastointiajan täytyy olla lyhempi ja tiedot täytyy muuttaa korkeammalle tasolle nopeammin esimerkiksi summaamalla tai keskiarvottamalla. Toisaalta esimerkiksi tuntitason mittareista voidaan huolettomasti tallentaa kaikki, koska ne muodostavat vain hyvin pienen osan kaikista mittauksista.

2.6 Ongelman kuvaus lyhyesti

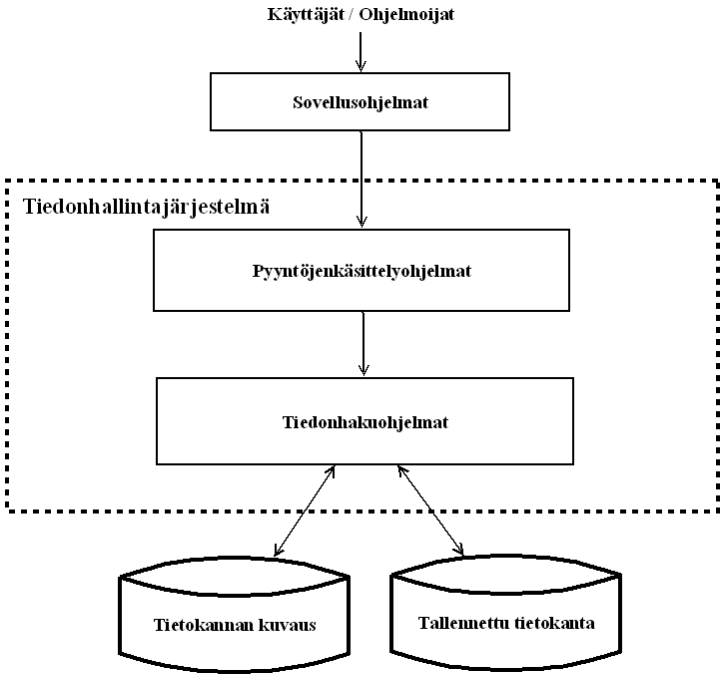
Stora Ensolla on noin 15 sellutehdasta, joista jokaisessa on tuhansia tai kymmeniä tuhansia mittalaitteita. Tietoa on paljon ja sen saantiin on käytössä useita eri järjestelmiä. Ongelmaa on ratkaistu kokoamalla tätä tietoa yhdelle palvelimelle, jotta tietoa voitaisiin tarkastella yhden järjestelmän kautta. Olemassa oleva tietovarasto tallentaa tietoa tehokkaasti, mutta ei tarjoa sitä käyttäjille yhtä helposti. Sen tietomalli ei myöskään ota kunnolla huomioon kohdeympäristössä tapahtuvia muutoksia. Lisäksi lähdejärjestelmistä saatavat mittareiden tiedot ovat epätäydellisiä. Kehitettävän tietovaraston tulisikin helpottaa tiedon esittämistä ja mukautua kohdeympäristön muutoksiin. Lisäksi sen tulisi olla helposti laajennettavissa. Tässä vaiheessa tietovaraston täytyy kyetä varastoimaan vähintään 6 miljardia näytettä vuosittain. Sitä käyttävät pääasiassa tutkijat ja jossain määrin myös tehtaan työntekijät. Kehitettävän tietovaraston tulee tukea molempia käyttäjäryhmiä.

3 TIETOKANTAJÄRJESTELMÄT JA TIETOVARASTOT

Tietokannoista on tullut yhä tärkeämpiä apuvälineitä. Ihmiset käyttävät tietokantoja huomaamattaan päivittäin. Esimerkiksi lentolipun varaus, jäsenetukortin käyttö tai tiedon haku internetistä hakukoneella aiheuttavat tapahtumia erilaisiin tietokantoihin. Tiedon määrä yhteiskunnassa kasvaa myös erittäin nopeasti ja ilman tietokantoja suuri osa tästä tiedosta häviäisi.

Tietokanta voidaan määritellä tietokokoelmaksi, joka kuvaa jotain tosimaailman osaa. Tosimaailman osaa kutsutaan englanninkielessä termeillä miniworld tai Universe of Discource (UoD). Muutokset tarkasteltavassa tosimaailman osassa heijastuvat tietokantaan. Tietokanta suunnitellaan ja rakennetaan tiettyä käyttötarkoitusta varten ja siihen tallennetaan sisäisesti merkityksellistä tietoa loogisella ja yhtenäisellä tavalla. Tietokantaan liittyy myös tietty joukko käyttäjiä ja käyttösovelluksia. (Elmasri & Navathe 2004, s. 4)

Kuvassa 3 on karkeasti esitetty tietokantajärjestelmän komponentit. Tietokantajärjestelmä koostuu tietokannasta ja tiedonhallintajärjestelmästä. Tietokantaan liittyy tiedon käyttöön tarkoitetut sovellukset, tallennettu tieto ja tietorakenteen looginen kuvaus.



Kuva 3. Tietokantajärjestelmä

Tiedonhallintajärjestelmä eli DBMS (Database Management System) koostuu ohjelmista joiden avulla käyttäjät voivat luoda tietokannan ja ylläpitää sitä. Tiedonhallintajärjestelmät luokitellaan useimmin tietomallin perusteella, mutta luokittelun voi tehdä myös monella muullakin tavalla, kuten käyttäjämäärän tai hinnan perusteella. (Elmasri & Navathe 2004, s. 43) Tietokannan käyttötapa on oleellinen kriteeri tiedonhallintajärjestelmän valinnassa. OLTP-järjestelmää (Online Transactional Processing) käytetään päivittäisiin toimintoihin. Se toimii nopeasti silloinkin, kun pyyntöjä tulee paljon. OLAP-järjestelmä (Online Analytical Processing) on tarkoitettu tiedon analysointiin ja sillä voidaan hakea suuria tietojoukkoja nopeasti tietovarastosta. (Han & Kamber 2006, s. 108-109)

3.1 Tietomallit ja relaatiomalli

Tietomalleilla pyritään piilottamaan käyttäjän kannalta epäoleelliset tiedot ja kuvaamaan tietoa erilaisilta abstraktiotasoilta. Tiedonhallintajärjestelmän arkkitehtuuri kuvataan usein kolmitasoisella mallilla, jossa jokainen taso on riippumaton muista tasoista. Matalimman tason kuvaukset tehdään sisäisellä tasolla käyttäen sisäisiä kaavioita. Sisäisellä tasolla tietomalli on lähellä tiedon fyysistä kuvausta. Kaavioissa viitataan tietueisiin eli kuvataan tiedon todelliset hakupolut. Käsitteellisellä tasolla tietokannan kuvaamiseen käytetään käsitekaavioita. Se piilottaa tiedon tallennuksen sisäiset yksityiskohdat ja esittää tietokannan kokonaisuudessaan yhtenäisesti. Ulkoisella tasolla käyttäjille kuvataan asiat käyttäjän kannalta mielekkäässä muodossa ja näkyvissä on ainoastaan käyttäjän kannalta oleellisia asioita. Käsitteellisen ja ulkoisen tason toteutuksessa käytetään loogista tietomallia. Loogisia tietomalleja ovat mm. relaatiomalli, hierarkkinen malli, verkkomalli ja oliomalli. (Elmasri & Navathe 2004, s. 26-32)

Tällä hetkellä yleisin looginen tietomalli on relaatiomalli. IBM:n tutkija E. F. Codd esitteli relaatiomallin vuonna 1970. Hänen tarkoituksenaan oli kehittää malli, jolla suurten tietomäärien hallinta olisi helpompaa ja joka olisi riippumattomampi fyysisestä toteutuksesta. (Codd 1970; Hernandez 2000, s. 11-12) Malli herätti välittömästi mielenkiintoa yksinkertaisuutensa ja matemaattisen perustansa ansiosta. Relaatiotietokannat perustuvat joukko-oppiin ja ensimmäisen kertaluvun predikaattilogiikkaan. (Elmasri & Navathe 2004, s. 125) 1970-luvulla

relaatiotietokantoja käytettiin keskustietokonejärjestelmissä. 1980-luvulla ne kehittyivät PC-järjestelmiin ja 1990-luvulla asiakas/palvelin-järjestelmiin. (Hernandez 2000, s. 17-19) Relaatiotietokannoista on tullut hallitseva tiedonhallintajärjestelmä ja nykyään relaatiotietokantoja löytyy aina henkilökohtaisista PC-koneista suuriin tietokantapalvelimiin asti (Elmasri & Navathe 2004, s. 21).

Relaatiotietokannat koostuvat relaatioista, jotka voidaan käsittää tietokannan tauluina. Relaatio käsite liittyy joukkoteoriaan, josta tietomallin nimikin on johdettu. Yleinen harhakäsitys on, että relaatiotietokannan nimitys olisi tullut siitä, että taulujen välille voidaan muodostaa suhteita. (Hernandez 2000, s. 12) Myös monissa muissa malleissa voidaan muodostaa suhteita eivätkä suhteet ole siksi relaatiotietokannalle uniikki piirre.

Taulut sisältävät rivejä, jotka ovat ominaisuuksiltaan samankaltaisia. Jokainen rivi edustaa tosiasiaa tai asioiden välistä suhdetta. Rivejä sanotaan monikoiksi *(engl. tuple)*. (Elmasri & Navathe 2004, s. 126-127) Myös monikko nimitys tulee matematiikasta. Matematiikassa monikoksi sanotaan sellaisten käsitteiden sekvenssiä, jotka yhdessä kuvaavat jotain asiaa. (Wikipedia, tuple) Esimerkiksi syntymäpäivä on vuoden, kuukauden ja päivän monikko. Syntymäpäivä ei kuitenkaan ole hyödyllinen tieto, jos ei tiedetä kenen syntymäpäivä on kyseessä. Taulun sarake kuvaa yhtä relaatioon liittyvää yhteistä ominaisuutta. Taulukossa 2 on esimerkkinä kuvattu henkilöiden syntymäpäivät relaatio.

Taulukko 2. Henkilöiden syntymäpäivät relaatio

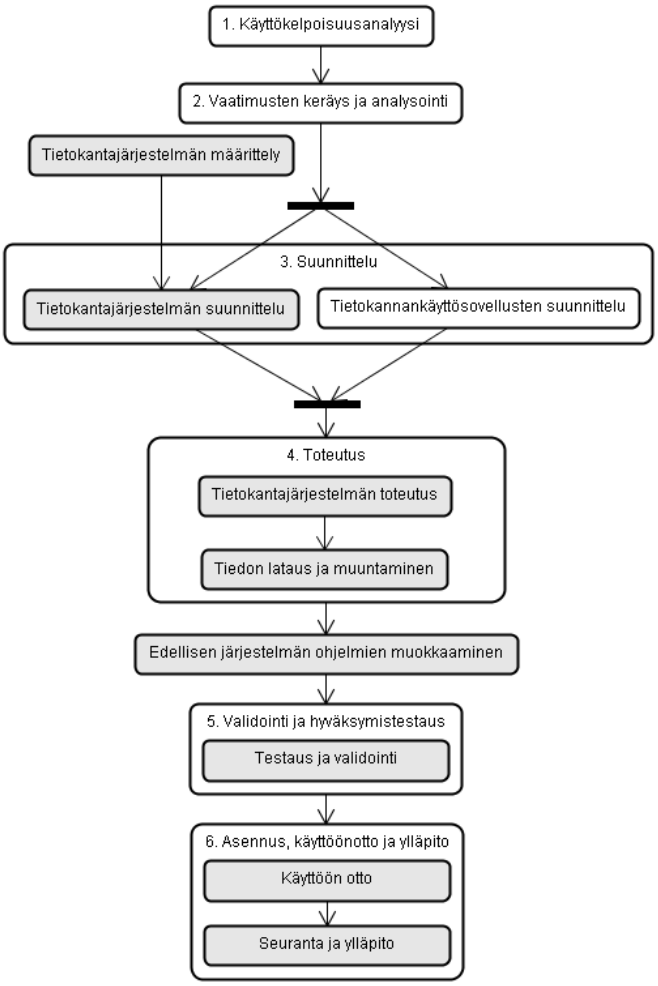
nimi	vuosi	kuukausi	päivä
Timo Tiedokas	1970	11	24
Tarmo Taidokas	1980	8	13
Into Tarmokas	1990	3	12

Tauluille voidaan myös määritellä erilaisia rajoitteita. Voidaan esimerkiksi määrätä tietty kenttä tai kentät relaation pääavaimeksi. Pääavain yksilöi rivin ja siten estää tiedon toistumisen. Viiteavaimen avulla voidaan kuvata tiedon riippuvuussuhdetta kahden taulun välillä. Esimerkin syntymäpäivätaulun nimikenttä voisi viitata henkilötaulun nimikenttään, jolloin sellaiselle henkilölle ei voisi muodostaa syntymäpäivää, jota ei ole olemassa. Viiteavaimilla voidaan siis varmistaa taulujen välinen viite-eheys. (Elmasri & Navathe, s. 126-139)

3.2 Tietokannan kehitys osana tietojärjestelmän kehitystä

Suuressa organisaatiossa tietokantajärjestelmä on tietojärjestelmän osa. Tietojärjestelmä käsittää kaikki ne komponentit, joita käytetään tiedon hallintaan, käyttöön ja levitykseen. Tietojärjestelmän komponentteja ovat tieto, tiedonhallintajärjestelmä, tietokoneet ja niiden tallennusmediat, ihmiset jotka käyttävät ja hallitsevat tietoa, ohjelmat joilla tietoa haetaan ja muokataan sekä näiden ohjelmien kehittäjät. Tietojärjestelmän kehityskaarta kutsutaan makroelinkaareksi, kun taas tietokantajärjestelmän kehityskaarta kutsutaan mikroelinkaareksi. (Elmasri & Navathe 2004, s. 364-365)

Kuvassa 4 on havainnollistettu tietojärjestelmän elinkaari, johon on liitetty myös tietokantajärjestelmän elinkaaren vaiheet niille kuuluviin kohtiin. Tietojärjestelmän kehitys alkaa käyttökelpoisuusanalyysillä, jossa selvitetään mm. taloudelliset kysymykset sekä tiedon ja prosessien kompleksisuudet. Tämän jälkeen eri käyttäjäryhmiltä selvitetään järjestelmän vaatimukset kuten tärkeimmät tietotarpeet, raportointimenetelmät ja sovellusten väliset yhteydet. Tietokantajärjestelmä ja tietokannan käyttöön tarkoitetut sovellukset suunnitellaan rinnakkain, koska muutokset toisessa prosessissa aiheuttaa muutoksia myös toiseen. Suunnittelua seuraa tietojärjestelmän toteutus. Tässä vaiheessa tietokantaan ladataan tieto ja tietokanta toteutetaan ja sitä testataan. Jos käytössä on vanha tietokantajärjestelmä, muokataan järjestelmän sovellukset uuden järjestelmän mukaisiksi. Tämän jälkeen varmistetaan, että tietojärjestelmä täyttää käyttäjien vaatimukset. Testaus suoritetaan teho- ja toiminnallisuusmäärittelyjen mukaan. Kun järjestelmän toimivuus on varmistettu, se asennetaan ja otetaan käyttöön, jonka jälkeen sitä voidaan käyttää ja ylläpitää. (Elmasri & Navathe 2004, s. 365-366)



Kuva 4. Tietojärjestelmän kehityskaari

3.3 Tietokannan suunnittelu ja toteutus

Teorey et al. mukaan tietokannan suunnittelun voi jakaa kolmeen vaiheeseen, jotka ovat vaatimusten analysointi, looginen suunnittelu ja fyysinen suunnittelu (Teorey et al. 2006, s. 3-8). Elmasri & Navathe jakavat suunnittelun viiteen vaiheeseen: vaatimusten keräys ja analysointi, käsitteellinen suunnittelu, tiedonhallintajärjestelmän valinta, looginen suunnittelu ja fyysinen suunnittelu (Elmasri & Navathe 2004, s. 367).

Molemmille vaihejaoille yhteistä on se, että ne alkavat vaatimusten analysoinnilla. Sitä voidaankin pitää koko kehitysprosessin kulmakivenä. Se mitä kaikissa seuraavissa vaiheissa tapahtuu, riippuu vaatimusten keräyksen ja analysoinnin onnistumisesta. Tietokannan suunnittelussa joudutaan yleensä tekemään kompromisseja esimerkiksi käytettävyyden, tehokkuuden ja tilankäytön suhteen. Vaatimusmäärittelystä selviää, mitkä ovat tietokannan tärkeimmät vaatimukset ja se

ohjaa suunnittelijoita myös kompromissitilanteissa. Tietokannan vaatimusmäärittely voi vaatia paljon aikaa, mutta sen onnistuminen on elintärkeää koko tietojärjestelmän onnistumisen kannalta. (Elmasri & Navathe 2004, s. 369-371)

Vaatimukset selvitetään tavallisimmin haastattelemalla asiakasorganisaation jäseniä ja perehtymällä organisaation dokumentaatioon ja olemassa olevaan tietojärjestelmään. Eri käyttäjillä on erilaisia tietotarpeita ja vaatimuksia. Vaatimusmäärittelyssä selvitetään mm. tietokannan sovellusalueet, käyttäjäryhmät ja heidän tietotarpeensa sekä käyttötapaukset. (Elmasri & Navathe 2004, s. 369-371; Hernandez 2000, s. 28)

Teorey et al. esittämän vaihejaon looginen suunnittelu jakautuu neljään vaiheeseen: käsitteellinen mallinnus, mallien integrointi, käsitteellisen mallin muuttaminen SQL-tauluiksi (Structured Query Language) ja taulujen normalisointi (Teorey et al. 2006, s. 3-8). Vastaavat vaiheet tehdään myös Elmasri & Navathen vaihejaon vaiheissa 2 - 4. Käsitteellinen tietokannan kuvaus tehdään yleensä käsitekaavioiden avulla. Suuressa projektissa voi olla monta suunnittelijaa, jonka takia suunnittelun päätyttyä kaaviot täytyy yhdistää ja niiden keskinäiset ristiriidat täytyy poistaa. Käsitteelliset mallit ovat riippumattomia tiedonhallintajärjestelmässä käytettävästä loogisesta tietomallista. Kun tiedonhallintajärjestelmä on valittu, muunnetaan käsitekaavioiden tietorakenteet tiedonhallintajärjestelmässä käytetyn loogisen mallin mukaisiksi. (Elmasri & Navathe 2004, s. 371-380)

Fyysinen suunnittelu kattaa tallennuspolkujen, indeksointien ym. fyysisten rakenteiden valinnat. Fyysisellä suunnittelulla viimeistellään tietokannan tehoon ja tilatarpeeseen liittyvät kysymykset. Tehoon ja tilatarpeeseen vaikuttaa myös tietokannan looginen suunnittelu ja fyysisellä suunnittelulla ei välttämättä voida pelastaa aiemmissa vaiheissa epäonnistunutta suunnittelua. (Elmasri & Navathe 2004, s. 383-384)

Lopuksi tietokanta toteutetaan. Ulkoisen mallin toteutuksessa käytetään tiedonhallintajärjestelmän DDL-kieltä (Data Definition Language) ja sisäisen mallin toteutuksessa SDL-kieliä (Storage Definition Language). Tämän jälkeen tietokanta populoidaan, eli siihen ladataan tietoa. Jos aiemmin on ollut käytössä tietokanta, niin siinä olevat tiedot täytyy muokata uuden tietomallin mukaisiksi ennen latausta. Myös toiminnalliset osat toteutetaan ja testataan tässä vaiheessa. Kun testaus on päättynyt

onnistuneesti, päättyy tietokannan kehitysvaihe, josta ylläpitovaihe alkaa. (Elmasri & Navathe 2004, s. 384-385)

3.4 Tietovarastot

Yksi yritysten tärkeimmistä voimavaroista on heidän hallussaan oleva tieto. Organisaation tietoja säilytetään yleensä kahdessa eri järjestelmässä. Toiminnallinen järjestelmä tukee organisaation päivittäistä toimintaa ja sen sisältö muuttuu jatkuvasti. Toiminnallista järjestelmää käyttävät tavallisesti asiakkaat ja työntekijät. Tietovarasto tukee organisaation pidemmän aikavälin toimintaa ja siellä säilytetään toiminnallisten järjestelmien tapahtumahistoriaa pitkältä ajanjaksolta. Tietovaraston sisältö ei muutu, mutta se voi lisääntyä tai vähentyä. Tietovarastot palvelevat päätöksentekijöitä ja siksi niitä kutsutaan myös päätöksentekojärjestelmiksi. Tietovarastossa tiedot on tallennettu yksinkertaisella mallilla ymmärrettävään ja yhtenäiseen muotoon. (Kimball et al. 1998, s. 9-11) Jotta yritys voisi saavuttaa kilpailullista etua, täytyy sen tietoja hallita, analysoida ja syöttää päätöksenteko prosessiin (Narasimhaiah 2003).

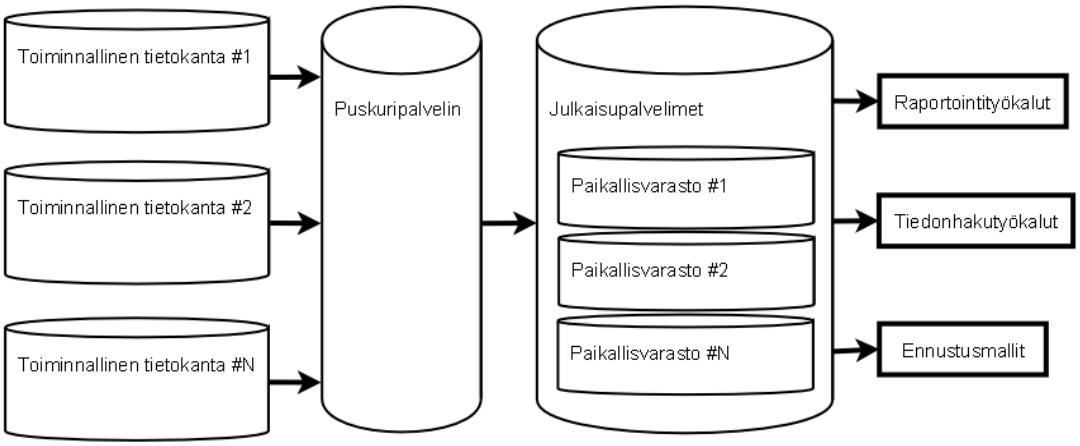
Tietovarasto on päätöksenteon perusta. Tiedon varastointi voidaan nähdä myös yhtenä tietämyksen selvitysprosessin (knowledge discovery) osana. Tietämyksen selvitysprosessiin kuuluu vaiheet: tiedon puhdistaminen, tiedon yhdistäminen, tiedon valinta, tiedon muokkaus, tiedon louhinta, hahmon tunnistus ja tietämyksen esitys. Tietovarastoon kerätään tietoa useista eri lähteistä. Kerätty tieto puhdistetaan eli väärät ja epäoleelliset asiat poistetaan. Lisäksi tietovaraston tieto on yhtenäisessä muodossa ja siitä on helppo rajata oleellisin tarkasteltava osa. Tietovarasto hoitaa siis tietämyksen selvitysprosessin kaksi ensimmäistä vaihetta ja auttaa kahdessa seuraavassa vaiheessa. Tällöin varsinaisen tiedonlouhinnan aloittaminen on helpompaa. (Han & Kamber 2006, s. 5-9)

Koska tietovarastoa käytetään liiketoiminnan ohjaamiseen, on tietovarastoprojektissa ensisijaisen tärkeää ottaa huomioon kohdeorganisaation liiketoimintamalli. Järjestelmä on hyödyllinen vain, jos käyttäjäorganisaatio omaksuu sen. Parhaimmassa tapauksessa tietovaraston kehittäjä onkin puoliksi tietokanta-asiantuntija ja puoliksi kauppatieteiden maisteri. (Kimball et al. 1998, s. 2)

3.4.1 Tietovaraston osat

Tietovaraston toimintaan tarvitaan kolme selvää osakokonaisuutta, jotka nähdään kuvassa 5. Tietolähteet ruokkivat tietovarastoa tiedolla, jonka puskuripalvelin muokkaa yhtenäiseen muotoon. Yhtenäiset tiedot tallennetaan julkaisupalvelimelle, josta käyttäjät pääsevät niihin käsiksi. (Kimbal et al. 1998, s. 13-28)

Tietovarasto koostuu yleensä monista samankaltaisista tietovarastoista. Yhtä loogista tietovaraston osaa sanotaan paikallisvarastoksi *(engl. data mart)*. Paikallisvarastoon säilöttäviä tietoja käytetään paljon keskenään ja usein sen vastuulla onkin yksi erillinen liiketoimintaprosessi. Tietovaraston kehitys on suuri ja aikaa vievä projekti. Se on helpompi suorittaa osissa paikallisvarastoittain, kuin suoraan tähtäämällä kaikenkattavaan järjestelmään. Yksi osakokonaisuus on nopea toteuttaa ja siten tietovarastosta saadaan nopeammin hyötyä jo joillekin käyttäjille. (Kimball et al. 1998, s. 168-169) Tietovaraston tehokkuus piilee siinä, että kaikissa paikallisvarastoissa tieto on tallennettu samalla periaatteella ja siksi tiedonhaku on aina lähes samanlainen riippumatta paikallisvarastosta. Paikallisvarastot voidaan piilottaa käyttäjiltä siten, että käyttäjät näkevät yhden yhtenäisen varaston jolla tietoa voidaan hakea mistä tahansa liiketoimintaprosessista ja mistä tahansa toimipaikasta. (Kimball et al. 1998, s. 18-19)



Kuva 5. Tietovarastoon liittyvät komponentit

Tieto haetaan varastoon monista lähdejärjestelmistä. Lähdejärjestelmät ovat toiminnallisia järjestelmiä ja niissä on tavallisesti vain vähän tietoa. Toiminnallisen järjestelmän vaatimuksina ovat ajantasaisuus ja saatavuus, jonka vuoksi tietomalli on yleensä optimoitu toiminnallisuutta varten. Tieto on myös jatkuvien muutosten alaista.

Selkein ero toiminnallisen järjestelmän ja tietovaraston välillä on se, että tieto menee toiminnalliseen järjestelmään sisään ja tietovarastosta tämän tiedon voi ottaa ulos. (Kimball et al. 1998, s. 9, 14) Toiminnalliset järjestelmät eivät ole osa tietovarastoa, koska tietovarastosta ei voida vaikuttaa niiden toimintaan (Kimball et al. 1998, s. 16-17).

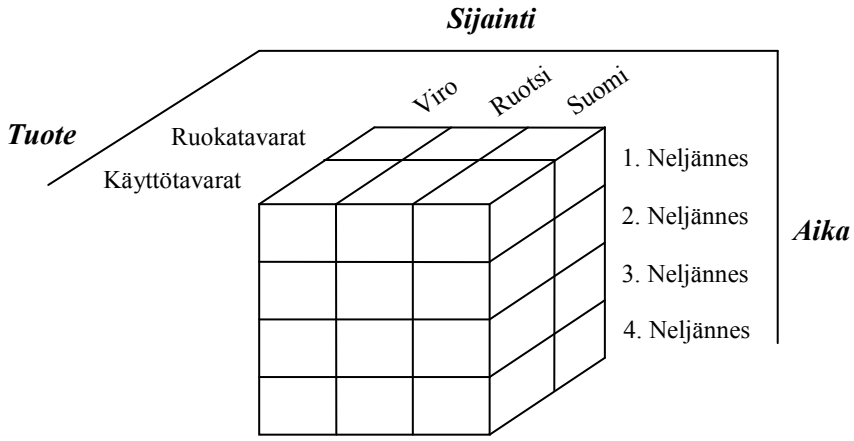
Ennen kuin toiminnallisesta järjestelmästä siirretään tietoa varastoon, täytyy sitä muokata tietovarastossa käytettävän tietomallin mukaiseksi. Tämän vuoksi tieto tuodaan aina ensin puskuripalvelimelle *(engl. staging area)*, jossa tietoa säilytetään muunnosoperaatioiden ajan. Puskuripalvelimella tietoa puhdistetaan, yhdistetään ja muokataan julkaisua varten. Näitä sanotaan yleensä ETL-toiminnoiksi (Extract Transform Load). Tavalliset käyttäjät eivät pääse kyselytyökaluilla puskuripalvelimen tietoihin käsiksi. Kun tieto on muunnettu oikeanlaiseksi, se voidaan siirtää puskuripalvelimelta julkaisupalvelimelle, joka on ainut käyttäjille näkyvä tietovaraston osa. (Kimball et al. 1998, s. 16)

Julkaisupalvelimilla tieto on yhtenäisessä muodossa eikä siellä olevaa tietoa tai sen tallennusmuotoa enää muuteta. Tiedon käyttöön tarkoitettuja työkaluja voi olla monenlaisia ja ne voivat olla eri käyttäjäryhmille suunnattuja. Yleensä tietovarastoissa on myös mahdollisuus tiedon analyyttiseen selaamiseen käyttäen OLAP-tekniikoita. (Kimball et al. 1998, s. 16-17)

3.4.2 Moniulotteisuus ja OLAP

Kimball et al. mukaan moniulotteinen tiedon mallinnus on tehokkain tapa tietovaraston kuvaamisen. Sen suunnittelu poikkeaa merkittävästi perinteisestä käsitteellisestä tietokannan suunnittelusta. (Kimball et al. 1998, s. 139) Moniulotteisen mallinnuksen perusajatus on se, että tieto sijaitsee kuutiossa. Yhdessä kuutiossa tiedot ovat aina samantyyppisiä. Esimerkiksi vähittäistavarakauppaketjulla voisi olla kuutiot myynnille ja markkinoinnille. Kuution särmiä sanotaan ulottuvuuksiksi ja ne edustavat yhden tyyppistä tiedon luokittelu- tai rajaustapaa. Myyntikuutiossa voisi olla aika-, sijainti- ja tuoteulottuvuudet kuten kuvassa 6. Yleensä kuutioissa on kuitenkin enemmän kuin kolme ulottuvuutta eli tieto on hyperkuutiossa. Haluttu tieto löytyy ulottuvuuksien rajaamista soluista. (Han & Kamber 2006, s. 110-123)

OLAP-työkalujen avulla eri käyttäjät voivat rajata suuresta tietomassasta juuri heidän kannaltaan merkityksellisen osan. Sillä voidaan tarkastella tietoa tehokkaasti eri karkeustasoilta ja eri näkökulmista. OLAP-työkalut perustuvat moniulotteiseen tietomalliin. (Han & Kamber 2006, s. 109-110). OLAP-järjestelmillä on saavutettu merkittäviä hyötyjä ja saatu jopa käännettyä tappiollinen liiketoiminta tuottavaksi (Narasimhaiah 2003).



Kuva 6. Tietokuutio

Relaatiomallin luoja E. F. Codd kehitti OLAP-termin yhdessä S. B. Coddin ja C. T. Salleyn kanssa vuonna 1993. He julkaisivat 12 sääntöä, joilla OLAP-työkalua voidaan arvioida. (Codd et al. 1993)

1. *Moniulotteinen käsitteellinen näkymä*

   Käyttäjä voi selata tietoa moniulotteisen näkymän kautta. Moniulotteinen näkymä tarjoaa intuitiivisemman tavan tiedon käsittelyyn ja mahdollistaa sellaisia tiedon analysointi- ja tarkastelumenetelmiä, jotka ovat mahdottomia esimerkiksi yksi- tai kaksiulotteisilla malleilla. Nämä ns. OLAP-operaatiot on esitelty kappaleessa 3.4.3.

2. *Läpinäkyvyys*

   OLAP-työkalun sijainti ei näy käyttäjälle eikä vaikuta sen käyttöön. Käyttäjän ei siis tarvitse olla tietoinen siitä, onko OLAP esimerkiksi osa sitä sovellusta, jonka avulla hän käsittelee tietoa tai onko se osa asiakas/palvelin-arkkitehtuuria.

3. *Saatavuus*

   Käyttäjän ei tarvitse tietää mistä ja minkälaisesta järjestelmästä OLAP-työkalun kautta saatava tieto on peräisin. OLAP-työkalun täytyy kuitenkin tietää tämä ja tuoda tieto näistä järjestelmistä käyttäjien saataville yhtenäiseen muotoon.

4. *Vakaa raportoinnin suorituskyky*

Ulottuvuuksien määrän ja tietokannan koon kasvaessa käyttäjän ei tulisi havaita merkittävää raportoinnin suorituskyvyn heikentymistä.

5. *Asiakas/palvelin -arkkitehtuuri*

Useimmin tieto sijaitsee keskustietokoneella, josta sitä haetaan erilaisilla asiakasohjelmilla. OLAP-työkalun tulee toimia asiakas/palvelin-arkkitehtuurissa ja tarjota OLAP-toimintojen suorittamiseen joustavat rajapinnat. Tällöin uusia asiakasohjelmia voidaan kehittää pienellä vaivalla.

6. *Yleinen moniulotteisuus*

Dimensioiden pitää olla yhdenmukaisia niin rakenteellisilta kuin toiminnallisiltakin valmiuksiltaan. Yleisiä dimensioiden toiminnallisia ominaisuuksia voidaan liittää erikseen mihin tahansa dimensioon koska dimensiot ovat yhdenmukaisia.

7. *Dynaaminen harvan matriisin hallinta*

OLAP-työkalun tulee pystyä tallentamaan erilaisia analyyttisiä malleja jokaiselle mallille optimaalisimmalla tavalla. Jokaiselle harvalle matriisille on olemassa vain ja ainoastaan yksi optimaalinen tallennusmuoto. Optimaalinen tallennusmuoto on tehokkain muistin käytön ja tiedon käsittelyn suhteen. Myös fyysisten tiedonsaantitekniikoiden tulee olla muutettavissa.

8. *Monen käyttäjän ominaisuudet*

OLAP-työkalulla voi olla useampia yhtäaikaisia käyttäjiä, jotka voivat käsitellä samaa tietomallia tai tietoa yhtäaikaisesti.

9. *Rajoittamattomat ulottuvuuksien väliset operaatiot*

Työkalussa tulee olla sisäänrakennettuja laskentakaavoja, joita käyttäjän ei tarvitse itse määritellä. Käyttäjällä tulee kuitenkin olla mahdollisuus määritellä myös itse eri ulottuvuuksien välisiä laskentakaavoja.

10. *Intuitiivinen käsittely*

Tietoa voidaan käsitellä suoraan käyttöliittymän graafisen esityksen avulla tietosolujen kautta. Käyttäjän ei tarvitse käyttää esimerkiksi valikoita tai muita monimutkaisia hakupolkuja yksinkertaisten tiedonkäsittelytoimenpiteiden

suorittamiseen. Näkymästä tulee selvitä suoraan olennaisten dimensioiden tiedonkäsittelyoperaatioissa tarvittavat tiedot.

11. Joustava raportointi

Tietoa on helpompi analysoida, kun vertailtavat rivit tai sarakkeet ovat lähellä toisiaan. Työkalun tulee sisältää rivien ja sarakkeiden ryhmittelytapoja monipuolisesti.

12. Rajoittamaton määrä ulottuvuuksia ja summaustasoja

Tutkimusten mukaan analyyttisessä mallissa voidaan tarvita jopa 19 ulottuvuutta. OLAP-työkalun tulisi tukea vähintään 15:sta ulottuvuutta, mutta mielellään yli 20:tä. Käyttäjät voivat määritellä dimensioihin summatasoja rajattomasti.

OLAP-työkalut voidaan jakaa kahteen pääluokkaan: MOLAP (Multidimensional Online Analytical Processing) ja ROLAP (Relational Online Analytical Processing). MOLAP-ympäristössä tiedonhallintajärjestelmä tukee moniulotteisia näkymiä, jossa tiedon looginen tallennus tehdään moniulotteisiin taulukoihin. MOLAP-järjestelmissä levytilan tarve voi olla pieni, kun tietokuutio on harva. Yleensä käytetään kaksitasoista järjestelmää, jossa tunnistetaan kuution tiheät ja harvat osat. Tiheät osat tallennetaan suoraan moniulotteisiin taulukoihin ja harvat osat pakataan. (Han & Kamber 2006, s. 135-136)

ROLAP-ympäristössä tieto on mallinnettu relaatiotietokannan tauluilla, esimerkiksi käyttäen ns. tähtimallia. Tiedonhallintajärjestelmästä puuttuvat OLAP-toiminnot on toteutettu erillisellä OLAP-moduulilla, joka on yhteydessä tiedonhallintajärjestelmään. ROLAP-järjestelmät tarjoavat yleensä MOLAP-järjestelmää paremman skaalautuvuuden. On myös olemassa HOLAP-järjestelmiä (Hybrid OLAP), jotka hyödyntävät ROLAP-järjestelmän skaalautuvuutta ja MOLAP-järjestelmän tehokasta aggregointia. Tällaisissa järjestelmissä erityyppisiin tilanteisiin voidaan aina valita tehokkain tallennusmuoto. (Han & Kamber 2006, s. 135-136)

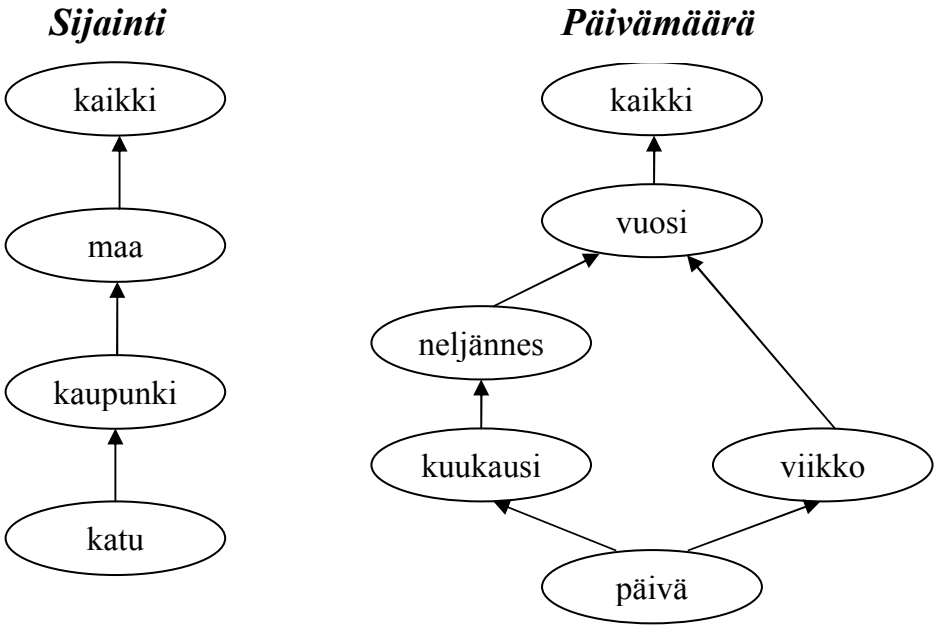
3.4.3 OLAP-operaatiot

OLAP-työkaluilla voidaan selata tietokuutioita OLAP-operaatioiden avulla. Keskeinen OLAP-operaatio on porautuminen *(engl. drill down)*. Porautumisen ajatuksena on se, että käyttäjä voi valita ulottuvuudesta haluamansa osan ja siirtyä tarkastelemaan tätä osaa yksityiskohtaisemmalta tasolta. (Hovi 1997, s. 60) Käyttäjä

voi esimerkiksi huomata, että ensimmäisellä vuosineljänneksellä myynti on ollut heikkoa ja sen perusteella porautua tarkastelemaan sitä kuukauden tarkkuudella. Tästä käyttäjä voi taas huomata, että jossain tietyssä liikkeessä myynti on ollut heikkoa helmikuun aikana ja porautua edelleen tämän liikkeen yksityiskohtaisempiin tietoihin. Porautumisella päästään siis helposti asian ytimeen.

Porautumisen vastakohta on karkeistaminen *(engl. roll up)*, jolla ulottuvuuksia joko vähennetään tai niiden käsitehierarkiassa siirrytään ylöspäin. OLAP-operaatioita varten jokaisella dimensiolla täytyy olla käsitehierarkia, jotta työkalu tietää, mikä on seuraava tarkempi ja karkeampi taso. Ulottuvuuteen voi kuulua myös useampia hierarkioita ja se voi jakautua useampaan haaraan. Käsitehierarkian ylin taso on ”kaikki”, joka tarkoittaa sitä, että ulottuvuutta ei käytetä hakuehdoissa. (Han & Kamber 2006, s. 121-126) Kuvassa 7 on esitetty käsitehierarkiat sijainnille ja päivämäärälle.

Porautumisen ja karkeistamisen ideana on tiedon karkeuden muutos. On myös olemassa sellaisia operaatioita, joilla tietoa voidaan pelkästään rajata tai esittää eritavalla. Tiedon rajaamiseen on olemassa mm. slice- ja dice-operaatiot. Slice-operaatiolla tietokuutiosta voidaan rajata yhden dimension suhteen osajoukko eli pienempi tietokuutio. Tämä tarkoittaa tietynkokoisen viipaleen leikkaamista kuutiosta. Dice-operaatio on slice-operaation kanssa vastaavanlainen muuten, mutta sillä rajaus voidaan tehdä useamman dimension suhteen. Yksi dice-operaatio on siksi mahdollista suorittaa tekemällä peräkkäisiä slice-operaatioita. Pivot-operaatiolla voidaan muuttaa tiedon esitystapaa ryhmittelemällä tietoa yhden sarakkeen sijaan useisiin sarakkeisiin rinnakkain. Pivot-operaatiolla vertailtavat tiedot saadaan lähelle toisiaan joka helpottaa tiedon tarkastelua. (Han & Kamber 2006, s. 123-126)



Kuva 7. Käsitehierarkiat sijainnille ja päivämäärälle

OLAP-operaatioita varten on kehitetty kyselykieliä, koska perinteiset SQL-kieliset kyselyt ovat OLAP-käytössä vaikeaselkoisia. Tällaisia kieliä ovat mm. SQL-kieleen perustuva DMQL (Data Mining Query Language) ja Microsoftin kehittämä MDX (Multidimensional Expressions). Näillä kielillä voidaan käsitellä suoraan tietokuutioita, perinteisten SQL-taulujen sijaan. (Burst & Forte 2006, s. 691; Han & Kamber 2006, s. 117)

3.4.4 Tietovaraston elinkaari

Ralph Kimball et al. ovat esittäneet tietovaraston kehitykseen elinkaaren, jossa otetaan huomioon nimenomaan tietovaraston tarpeet ja moniulotteinen mallinnus. Elinkaari noudattaa pääpiirteittäin tietokantajärjestelmän elinkaarta, jota on tarkennettu tietovarastoinnin näkökulmasta. Elinkaaressa on myös otettu huomioon projektinhallintaan liittyvät toiminnot rinnakkaisena koko elinkaaren läpi jatkuvana prosessina. Tietovaraston kehitys alkaa projektin suunnittelulla, josta se jatkuu liiketoiminnallisten vaatimusten määrittelyllä. Tästä elinkaari jakautuu kolmeen rinnakkaiseen osaan: teknisen arkkitehtuurin, tietomallin ja käyttäjäsovellusten kehittämiseen. Tietovarastot ovat usein hajautettuja ja sisältävät monenlaisia järjestelmiä. Sen vuoksi perinteisestä tietokannan elinkaaresta puuttuva arkkitehtuurisuunnittelu on lisätty tietovaraston elinkaareen. Siinä suunnitellaan myös tietovaraston infrastruktuuri ja metatieto. Tietomallin suunnittelu käsittää

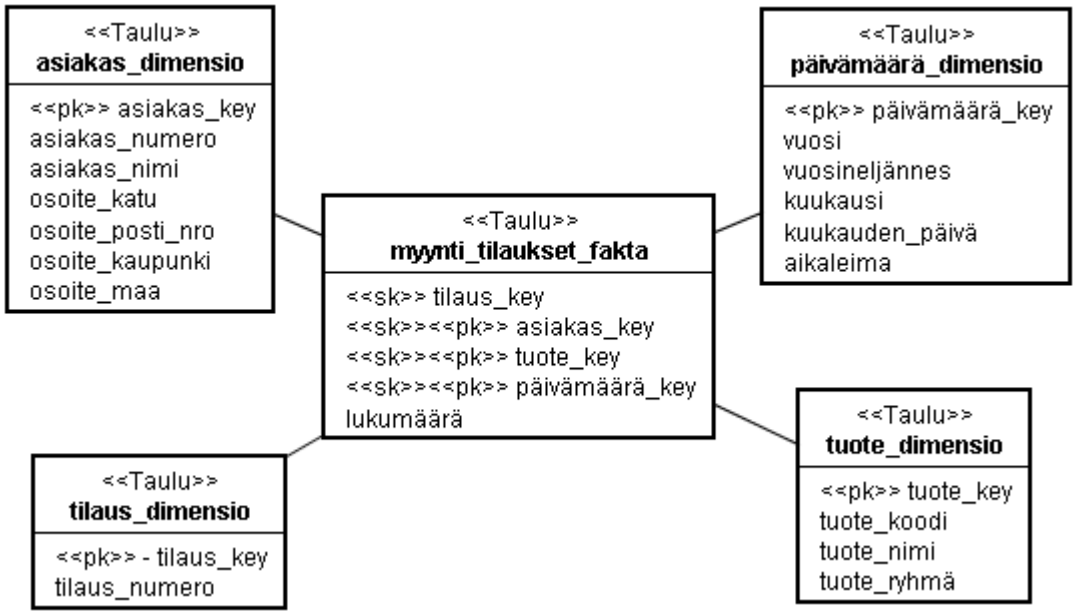
moniulotteisen mallin suunnittelun ja siinä käytettävien taulujen fyysisen suunnittelun. Lisäksi siinä suunnitellaan ja kehitetään tietovaraston puskuripalvelin. Käyttäjäsovellusten kehittäminen kattaa hakutyökalujen ja tiedon saantiin liittyvien sovellusten määrittelyn ja suunnittelun. Elinkaaren kolme haaraa päättyvät tietovaraston käyttöönottoon ja ylläpitoon. (Kimball et al. 1998, s. 33-38)

3.5 Moniulotteinen mallinnus käyttäen tähtimallia

Suuri osa tietovarastoista on rakennettu relaatiotietokannoilla. Relaatiotietokannassa moniulotteisen rakenteen voi muodostaa käyttäen ns. tähtimallia. Tähtimallissa tieto on jaettu kahdentyyppisiin tauluihin: faktatauluihin, joissa raaka tieto sijaitsee ja dimensiotauluihin, joiden avulla raakadata muuttuu informaatioksi. Dimensiotaulu vastaa tietokuution särmää, kun taas faktataulusta löytyy särmien rajaaman solun tieto. Faktataulussa tieto ei ole ihmiselle mielekkäästi ymmärrettävässä muodossa ja se sisältää paljon numeerisia kenttiä. Faktoja ovat faktataulun mittaussarakkeet, joita faktataulussa on tavallisesti yksi tai muutamia. Dimensiotaulussa kaikki tieto on erittäin ymmärrettävässä muodossa ja se sisältää paljon tekstikenttiä. Faktataulu on täysin normalisoitu (ei toistuvaa tietoa), kun taas dimensiotaulu on täysin denormalisoitu (paljon toistuvaa tietoa). Kun dimensiotaulut liitetään faktatauluun, muistuttaa näin muodostunut rakenne tähteä. (Kimball et al. 1998, s. 165-167)

3.5.1 Fakta- ja dimensiotaulujen liitos ja avaimet

Dimensiotaulut liitetään faktatauluihin käyttäen ns. sijaisavaimia *(engl. surrogate keys)*. Kuvassa 8 on esimerkki tähtimallista, jossa neljä dimensiotaulua on liitetty yhteen faktatauluun. Sijaisavain on dimensiotaulun pääavain ja sitä sanotaan sijaisavaimeksi sen vuoksi, että se korvaa toiminnallisessa järjestelmässä olevan vastaavan avaimen. Esimerkiksi asiakasnumero voi yksilöidä asiakkaan operationaalisessa järjestelmässä. Samaa avainta ei kuitenkaan voida käyttää suoraan dimensiossa pääavaimena, koska henkilön tietojen, kuten osoitteen muuttuessa dimensiossa täytyy olla viittaus sekä uuteen että vanhaan tietoon. Yleensä myös toiminnallisen järjestelmän avain halutaan ottaa dimensioon attribuutiksi, koska sitten dimensiotaulusta voidaan helpommin katsoa, minkälaisia muutoksia esimerkiksi tiettyyn asiakkaaseen on ajan saatossa kohdistunut. (Kimball et al. 1998, s. 191-193)



Kuva 8. Esimerkki tähtimallista

Faktataulut pyritään pitämään mahdollisimman kapeina ja toisaalta tauluista saattaa kasvaa monien miljoonien tai jopa miljardien rivien korkuisia. Pienikin rivikoon kasvu tekee taulusta heti paljon suuremman, koska taulussa on niin paljon rivejä. Tämän vuoksi faktataulun pääavain on yleensä sijaisavaimien yhdistelmä. Faktataulun tiedonjyvä saadaan yksilöityä muutaman sijaisavaimen avulla ja olisi turhaa luoda sarake pelkästään pääavainta varten. Kuvan 8 faktataulun pääavain koostuu kolmesta sijaisavaimesta. Näin faktataulu pysyy kapeana ja pääavaimena toimii indeksi, jota todennäköisesti tarvittaisiin muutenkin taulun nopean toiminnan takaamiseksi. (Kimball et al. 1998, s. 165-166; Hovi 1997, s. 72-76)

Faktataulun rivit kuvaavat tauluun liitettyjen dimensioiden kombinaatioita. Kombinaatioita on yleensä todella paljon joista yhdessä faktataulussa on vain murto-osa (se osa, joka on tai oli olemassa myös reaalimaailmassa). Tietokuutioajatteluna tämä tarkoittaa sitä, että kuution sisällä on paljon aukkoja eli faktataulu on harva. Jokin kuutio saattaisi kattaa esimerkiksi 20% kaikista kombinaatioista. (Hovi 1997, s. 76-77) Käyttäjän näkökulmasta tämä tarkoittaa sitä, että kun hän rajaa tietokuutiosta jonkin erittäin tarkasti määrätyn osan, voi olla, että tuloksena on tyhjä joukko. Toisaalta, jos käyttäjä porautuu tietoon, ei tällaista tilannetta pääse kovinkaan helposti syntymään.

Tähtimallin tiedonhaussa käytetään rajoitteina dimensioiden kenttiä. Dimensiot liitetään rajauksineen faktatauluun, jolloin saadaan rajauksien mukaiset faktatiedot.

Tähtimallilla suunniteltu rakenne on aina samankaltainen ja kyselyissä voidaankin tehdä jo suoraan oletuksia esimerkiksi taulujen läpikäyntijärjestyksistä liitoksissa. (Kimball et al. 1998, s. 147-148)

Dimensioihin liittyy käsitehierarkioita, joiden mukaan tieto luokittuu laajempiin tai yksityiskohtaisempiin osiin. Porautuminen tarkemmalle abstraktiotasolle tähtimallissa tapahtuu yksinkertaisesti lisäämällä hierarkiassa hienojakoisemmalla tasolla oleva kenttä SQL-kyselyn ryhmittelysääntöihin. Käytännössä siis haluttuihin tuloksiin lisätään vain tarkempia kenttiä (joiden mukaan ryhmitellään), jolloin myös tietojoukko tarkentuu. (Kimball et al. 1998, s. 167-170)

3.5.2 Summataulut, summautuvuus ja faktattomuus

Yleensä summatauluilla halutaan parantaa tietovaraston tehokkuutta, mutta niitä voidaan luoda myös muista syistä. Joskus tieto ei ole hienojakoisimmalla tasolla kuvaavaa ja siksi tiedon karkeistaminen myös helpottaa tiedon tarkastelua. Summataulut ovat faktatauluja joissa yksityiskohtaisemman tason tiedot on yhdistetty karkeammalle tasolle. Esimerkiksi faktataulusta, johon on tallennettu kaikki vuoden 2006 myyntitapahtumat, voitaisiin laskea kuukausittaiset, tuoteryhmäkohtaiset ja toimipistekohtaiset summataulut. Näin näiden karkeamman tason tietojen tarkastelu on nopeampaa. Summataulut vähentävät tietokannassa tehtävän laskennan määrää ja läpikäytävien rivien määrää, mutta toisaalta myös tietokannan koko kasvaa hieman. Summatauluja muodostettaessa onkin oleellista pohtia, minkä dimension suhteen ja mille abstraktiotasoille summatauluja kannattaa tehdä. Tämä riippuu siitä, kuinka käyttäjät käyttävät tietokantaa ja mitkä ovat yleisimpiä ryhmittelytasoja. Lisäksi täytyy miettiä, minkä dimensioiden suhteen kuutio on tiheä ja minkä dimensioiden suhteen harva, jolloin karkeistamisella saadaan mahdollisimman paljon hyötyä. (Kimball et al. 1998, 211-212, 544-545)

Summatauluille on myös olemassa se vaihtoehto, että myös summatut rivit laitetaan perustauluun, jolloin taulujen määrä ei kasva karkeistamisen takia. Silloin kuitenkin tarvitaan myös faktataulun jokaiselle riville tieto karkeusasteesta jokaisen dimension suhteen. Tämä on mahdollista tehdä dimensiotaulussa asettamalla esimerkiksi ”ALL” teksti niihin kenttiin, joilla ei ole merkitystä. Tällöin sekä fakta- että dimensiotauluihin tarvitaan enemmän rivejä. Eräs vaihtoehto on sellaisen dimension tekeminen, joka kuvaa tiedon karkeusastetta. Näin menetellen dimensiotaulut eivät

kasva, mutta faktataulussa tarvitaan yksi viittaus enemmän. Yleisesti ottaen näillä tavoilla kuitenkin menetetään ne edut, jotka saadaan summataulun pienuudesta. (Hovi 1997, s. 82-89; Han & Kamber 136-137)

Summatauluja varten voidaan tehdä osajoukkodimensioita. Esimerkiksi päivämäärädimensiossa on 365 riviä / vuosi, mutta kuukausitason osajoukossa tarvitaan ainoastaan 12 riviä / vuosi. Näin faktataulun lisäksi myös dimensiotaulut ovat huomattavasti pienempiä. Osajoukolle voidaan tehdä taulu ja muokata dimension latausfunktioita siten, että latausfunktiot päivittävät täydellisen dimension lisäksi myös sen osajoukkojen taulut. (Darmawikarta 2007, s. 155-158) Osajoukko käsite voi tässä yhteydessä johtaa harhaan ja kannattaakin huomata, että se tarkoittaa nimenomaan dimension sarakkeiden osajoukkoa. Osajoukkodimension rivit eivät ole alkuperäisen dimension rivien osajoukko vaan ennemminkin niiden yleistys. Esimerkiksi päivätason aikadimension jokainen rivi viittaa aina yhteen päivään ja kuukausidimension rivit yleistävät päivät kuukausiksi. Osajoukkodimensio vastaa karkeammalle tasolle ryhmiteltyä tarkkaa dimensiota.

Parhaita faktoja ovat numeeriset mitattavissa olevat arvot, joita voidaan summata minkä tahansa dimension suhteen. Esimerkiksi euromääräinen myynti on sellainen fakta, joka summautuu yleensä kaikkien dimensioiden suhteen. Toisaalta jotkin faktat on luonnollisempaa keskiarvottaa. Usein erilaiset fysikaaliset mittaukset kuten lämpötila tai teho täytyy keskiarvottaa, koska se antaa summaa paremman käsityksen. Lisäksi summaaminen voisi muuttaa käsitettä. Esimerkiksi teho integroituna ajan suhteen on energia, mutta eri laitteiden suhteen se on laitteiden kokonaisteho. Faktat joita ei voi ollenkaan aggregoida ovat kaikkein huonoimpia. Esimerkiksi tekstimuotoisia faktoja ei yleensä voida aggregoida, jollei niillä ole jotain matemaattista merkitystä. Esimerkiksi sään kuvausta ei varmastikaan voida aggregoida. Useimmissa tapauksissa tekstikentät tulisikin pyrkiä pitämään dimensioissa. (Kimball et al. 1998, s. 193-194)

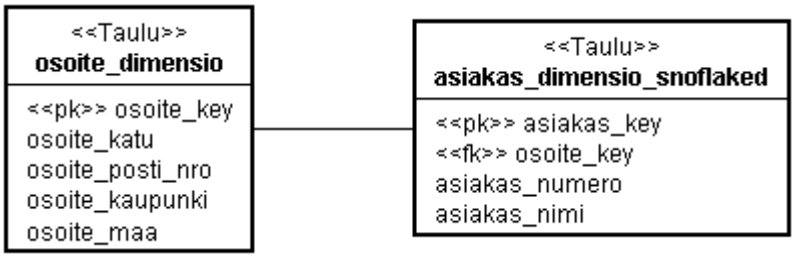
On myös olemassa sellaisia tekstikenttiä joilla on matemaattista merkitystä. Tällaisia ovat sumeat lingvistiset muuttujat, jotka saavat sanallisia arvoja, kuten ”melko vähän”, ”vähän”, ”normaali”, ”paljon” ja ”melko paljon”. Matemaattinen aggregointi näiden käsitteiden suhteen on mahdollista, tosin silloin täytyy käyttää jotain sumeaa aggregaatoria. Sumeassa logiikassa voi olla useita aggregaattoreita eri operaatiolle.

Esimerkiksi sumeiden käsitteiden yhdiste tai leikkaus voidaan laskea monella eri tavalla. Sumeita käsitteitä voidaan käyttää mittauksena monessakin eri tilanteessa. Esimerkiksi joissain tapauksissa ihminen toimii mittarina kuten asiakastyytyväisyyskyselyissä. Sumealla aggregaattorilla tässä tapauksessa voitaisiin selvittää asiakkaiden yleinen mielipide. Myös sumealla logiikalla toteutetut säätöjärjestelmät (Fuzzy Logic Controller (FLC)) voivat käsitellä mittausarvoja sumeina käsitteinä. Tällöin järjestelmä muuttaa mittarin antaman terävän arvon jollain sumentajalla sumeaksi käsitteeksi. (Mattila 2002, s. 150, 174-176)

Joistain faktatauluista puuttuvat faktat eli mittaussarakkeet kokonaan. Näitä tauluja sanotaan faktattomiksi faktatauluiksi *(engl. factless fact table)*. Tällaisissa faktoissa dimensioista ilmenevät jo kaikki tarvittavat asiat eikä faktasaraketta siksi tarvita. Yleisiä faktattomia faktoja ovat mm. tapahtumia kuvaavat ja jonkin asian esiintymistä kuvaavat faktat. Tapahtuman yhteydessä faktatauluun tulee rivi aina, kun tapahtuma esiintyy ja tapahtuma yleensä saadaan kuvattua dimensioilla. Tämän vuoksi faktasarakkeelle ei jää mitään virkaa. Jonkin asian esiintymisellä kuvataan, milloin jokin asia pitää paikkansa kuten milloin sisään ostettavan tavaran hinta on ollut suurempi kuin ulos myytävän. Tällaisessa faktassa ei tarvita mittausta, koska dimensiot yksin kuvaavat kaikkia niitä kombinaatioita, jolloin tarkasteltava asia on ollut tosi. (Kimball et al. 1998, s. 212-216)

3.5.3 Dimensioiden suunnittelunäkökulmia

Lumihiutaloimisella *(engl. snowflaking)* tarkoitetaan dimensiotaulun jakamista pienempiin tauluihin jotka liitetään viiteavaimella alkuperäiseen dimensiotauluun. Näin tähti jakautuu pienempiin osiin ja muistuttaa hieman lumihiutaletta. Esimerkiksi kuvan 8 asiakasdimension osoite voitaisiin erottaa toiseen tauluun kuten kuvassa 9. Huonona puolena tässä on se, että malli muuttuu monimutkaisemmaksi. Monimutkaisesta mallista tiedon hakeminen on hitaampaa ja myös hallinta ja mallin ymmärtäminen hankaloituu. Usein pilkotuilla dimensioilla pyritään säästämään levytilaa poistamalla dimensiossa toistuvia tekstikenttiä. Tilan säästäminen normalisoimalla dimensiota ei kuitenkaan ole perusteltua, koska dimensiotaulut muodostavat erittäin pienen osan koko tietovaraston vaatimasta tilasta. (Kimball et al. 1998, s. 170-173)



Kuva 9. Lumihiutaloitu dimensio

Dimensioilla voi olla yhdessä taulussa useita eri rooleja. Tällöin faktatauluissa tarvitaan samaan dimensiotauluun useita viittauksia. Esimerkiksi tilauksiin voisi hyvin liittyä useita aikadimensioita kuten tilaus_pvm, toivottu_toimitus_pvm, lähetys_pvm ja toimitus_pvm. Kaikki päivämäärät viittaavat yhteen dimensiotauluun. Yhteen dimensiotauluun ei voida liittää useita faktataulun viiteavaimia, koska SQL tulkitsee silloin, että esimerkiksi kaikkien päivämäärien tulisi olla yhtä suuria. Tämä ongelma voidaan ratkaista tekemällä jokaiselle erilaiselle viittaukselle näkymä, joka viittaa alkuperäiseen dimensioon. Näkymässä pitää myös huomioida, että sen sarakkeiden nimissä on käytettävä erilaisia nimiä. Muutoin raporttiin saattaa päätyä sarakkeita, joilla on eri merkitys, mutta sama nimi. Näkymien avulla käyttäjä näkee useita eri ulottuvuuksia, vaikka taustalla käytetään vain yhtä ulottuvuutta. (Kimball et al. 1998, s. 223-226)

Dimensio voidaan poistaa, jos se kuvaa jossain määrin samaa asiaa kuin faktakin. Esimerkiksi kuvassa 8 tilausdimensio on tehty säilyttämään operationaalisen järjestelmän tilausnumeroa ja koska faktataulu kuvaa nimenomaan tilauksia, voidaan tilausdimensio poistaa. Tilausnumero laitetaan suoraan faktatauluun muiden dimensioviittausten rinnalle. Tällaista viittausta sanotaan degeneroiduksi dimensioksi. (Kimball et al. 1998, s. 188)

Kun kaikki dimensiot on suunniteltu, huomataan yleensä, että jäljelle jäi käsitteitä, jotka eivät sovi mihinkään dimensioon. Yleensä nämä ovat epämääräisiä on/off-lippuja ja vapaatekstimuotoisia kommentteja. Tässä vaiheessa kannattaa tutkia näitä kenttiä lisää ja yrittää vielä löytää niistä jokin yhteinen tekijä. Jos kentät kuitenkin ovat epämääräisiä eivätkä liity toisiinsa mitenkään, kannattaa ne koota yhteen tai useampaan niin sanottuun rojudimensioon *(engl. junk dimension)*. Näin nämä kentät saadaan käyttöön ilman faktataulujen paisumista. (Kimball et al. 1998, s. 189-191)

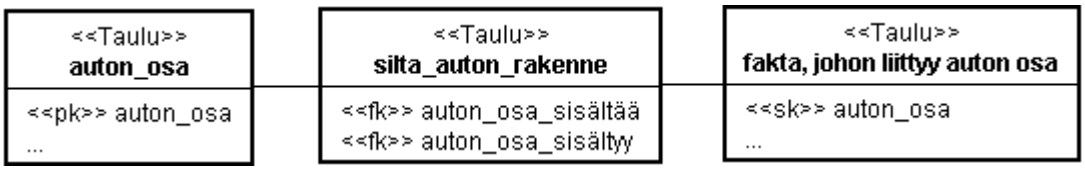
3.5.4 Siltataulut tähtimallissa

Joskus fakta- ja dimensiotaulun välille joudutaan tekemään siltataulu. Siltataulu monimutkaistaa rakennetta ja hidastaa kyselyitä, mutta joissain tapauksissa niiden käyttö on välttämätöntä. Siltataulu joudutaan tekemään silloin, kun faktataulun ja dimensiotaulun välillä on monesta moneen liitos tai silloin, kun halutaan kuvata dimension rivien välisiä hierarkioita.

Monesta moneen liitos faktan ja dimension välillä syntyy silloin, kun yhteen faktataulun riviin tarvitaan monta viittausta samaan dimensioon. Esimerkiksi kuvan 10 faktataulun yksi rivi kuvaa yhtä katsastustapahtumaa, mutta yhdellä katsastuskerralla voidaan havaita useita vikoja. Sen vuoksi tarvitaan kaksi taulua: vikadimensiotaulu ja vikaryhmä-siltataulu. Fakta liitetään ryhmittelevään tauluun, josta voidaan hakea kaikki ryhmän jäsenet, tässä tapauksessa viat. (Kimball et al. 1998, s. 218-222)

Kuva 10. Siltataulu monesta moneen liitoksessa

Hierarkia syntyy rivien välille silloin, kun hierarkiaa ei saada kuvattua dimension sarakkeiden käsitehierarkialla mielekkäästi. Hierarkia voi syntyä esimerkiksi auton osia listaavaan dimensioon. Auton konkreettista rakennehierarkiaa ei saada kuvattua käsitehierarkialla, koska todellisuudessa hierarkia on hyvin epämääräinen ja erittäin syvä. Tämän takia hierarkia tehdään rivien kesken käyttäen siltataulua kuten kuvassa 11. Faktataulu liitetään siltatauluun, joka taas liitetään dimensiotauluun. Vaikka siltataulu olisikin olemassa, ei sitä ole kuitenkaan pakko käyttää. Sellaisissa hauissa, joissa hierarkiaa ei tarvita, liitetään fakta suoraan dimensiotauluun. (Kimball et al. 1998, s. 226-231)



Kuva 11. Siltataulu dimension hierarkian kuvaamiseen

3.5.5 Dimensiomuutosten hallinta

Dimensiomuutokset voidaan jakaa karkeasti kahteen luokkaan. Virheellisen tiedon korjaamiseen ja tosimaailman muutosten heijastamiin muutoksiin. Virheellinen tieto voidaan korjata päivittämällä virheellisiin riveihin oikea tieto. Tosimaailman muutokset hallitaan yleensä muodostamalla dimensiotauluun uudelle tilanteelle uusi rivi. Siten voidaan viitata uuteen ja vanhaan tietoon. Dimension muuttamisessa oleellinen kysymys onkin siis halutaanko dimensiomuutoksilla muokata myös historiaa vai ei. Tämä täytyy arvioida jokaisessa tilanteessa erikseen. Dimension muutosten hallinnassa puhutaan usein kolmesta SCD-hallintatavasta (Slowly Changing Dimension). SCD 1 päivittää jokaisen muutoksen yhteydessä rivin (historia muuttuu). SCD 2 luo jokaisen muutoksen yhteydessä uuden rivin (historia ei muutu). SCD 3:ssa dimensiotauluun luodaan lisäsarakkeita, joissa säilytetään arvojen edellistä tilannetta (rajallinen historian säilyvyys). SCD 3:lla hallittavassa laitedimensiossa voisi olla esimerkiksi sarakkeet sijainti ja edellinen_sijainti. SCD 3 on harvoin käytetty menetelmä. Sitä voidaan soveltaa silloin, kun tietovarastolla on tiukat tilarajoitteet ja osittainen historian menettäminen on hyväksyttävää. Käytettäessä SCD 2 hallintamenetelmää ulottuvuustaulussa on hyvä olla sarakkeet rivin voimaantulopäivää ja vanhentumispäivää varten. Näitä sarakkeita voidaan hyödyntää esimerkiksi silloin, kun tietoa tuodaan lähdejärjestelmistä tietovarastoon. Tiedonmuuntovaiheessa voidaan valita päivämäärien avulla oikea sijaisavain jokaiselle mittaukselle. (Darmawikarta 2007, s. 25-26; Kimball et al. 1998, s. 180-188)

3.5.6 Metatieto tietovarastossa

Metatieto on tietoa tiedosta. Metatietoa ovat esimerkiksi tiedonhallintajärjestelmän taulu- ja sarakekatalogit. Tietovarastossa täytyy pitää kirjaa tietovaraston tietokuutioista, faktatauluista, dimensiotauluista, taulujen välisistä yhteyksistä, dimensiohierarkioista ja tietolähteistä ja näiden kaikista ominaisuuksista. Metatiedolla kuvataan siis tiedon rakenteet ja yhteydet ja sen avulla tiedolle voidaan antaa uusia merkityksiä. Esimerkiksi tietyt taulut ovat dimensiotauluja ja tietyt faktatauluja. Nämä yhdistettynä voidaan puhua myös tietokuutioista. Tietovarastoissa metatieto jaetaan yleensä backroom-metatietoon, joka sisältää kaiken taustaprosesseihin liittyvän tiedon ja frontroom-metatietoon, jolla kuvataan julkaisupalvelimen tauluja ja hakutyökalun

tarvitsemia tietoja. Metatiedolla tiedon hallinta saadaan joustavammaksi ja kyselyn muodostaminen helpommaksi. (Kimball et al. 1998, s. 435-436). Metatietoa voidaan säilyttää suoraan tiedonhallintajärjestelmässä tai esimerkiksi XML-tiedostoissa (eXtensible Markup Language). Esimerkiksi Pehtaho Mondrian käyttää XML-tiedostoja kuutiomäärittelyjen tekemiseen.

Tiedostojärjestelmässä tieto tallennetaan yksiulotteisesti kiintolevylle tiettyyn muistipaikkaan. Relaatiotietokannassa tieto tallennetaan kaksiulotteisesti relaatioihin eli tauluihin, joissa on sarakkeita ja rivejä. Myös relaatio on tallennettu fyysisesti yksiulotteiseen tiedostojärjestelmään, mutta käyttäjälle se näytetään kaksiulotteisena. Tietovarastoissa tieto on yleensä tallennettu kuutioihin, joissa on n ulottuvuutta, joilla on m karkeustasoa. Jotta n ulottuvuutta karkeustasoineen voidaan saavuttaa, tarvitaan metatietoa jolla kuvataan moniulotteiset rakenteet niin, että ne voidaan oikeasti tallentaa kaksiulotteisesti relaatioihin.

3.5.7 Suunnittelu väyläarkkitehtuurin avulla

Kimball et al. mukaan moniulotteinen tietovarasto kannattaa kehittää käyttäen niin sanottua väyläarkkitehtuuria. Sen perimmäinen tarkoitus on tietovarastoprojektin pilkkominen pienempiin osiin. Väyläarkkitehtuuri pakottaa nimeämään paikallisvarastot ja niihin liittyvät dimensiot. Suunnittelu aloitetaan muodostamalla matriisi, jonka riveille tulevat kaikki paikallisvarastot ja sarakkeisiin kaikki dimensiot. Tämän jälkeen paikallisvarastojen ja dimensioiden risteyskohdat merkitään niiltä kohdilta, joissa dimensioita tarvitaan paikallisvarastossa. Väyläarkkitehtuurin matriisista nähdään myös dimension tärkeys. Jos dimensio liittyy todella moneen paikallisvarastoon, kannattaa sen suunnitteluun kiinnittää erityisen paljon huomiota. (Kimball et al. 1998, s. 266-272) Väyläarkkitehtuurin nimi on johdettu siitä tavasta, jolla paikallisvarastot liittyvät dimensioihin kuten esimerkiksi laitteet liittyvät tietokoneen yleiseen väylään. Dimensiot yhdessä muodostavat väylän jossa jokainen johdin kuljettaa signaalia (dimensio), joka nähdään kaikkialla tietovarastossa. Laitteet (paikallisvarastot) voivat liittyä mihin tahansa väylän johtimiin, jolloin liitettyjen johdinten signaalit (dimensiot) ovat kaikkien laitteen resurssien (faktataulut) saatavilla. (Kimball et al. 1998, s. 346)

Faktataulujen suunnittelu tehdään neljässä vaiheessa. Ensin valitaan paikallisvarasto, johon faktataulu kuuluu. Tämän jälkeen määritellään tiedonjyvä eli se, mitä yksi

faktataulun rivi kuvaa. Viimeisenä valitaan tarvittavat dimensiot ja faktat eli faktataulun sarakkeet. (Kimball et al. 1998, s. 272-276)

3.6 MySQL RDBMS tietovarastona

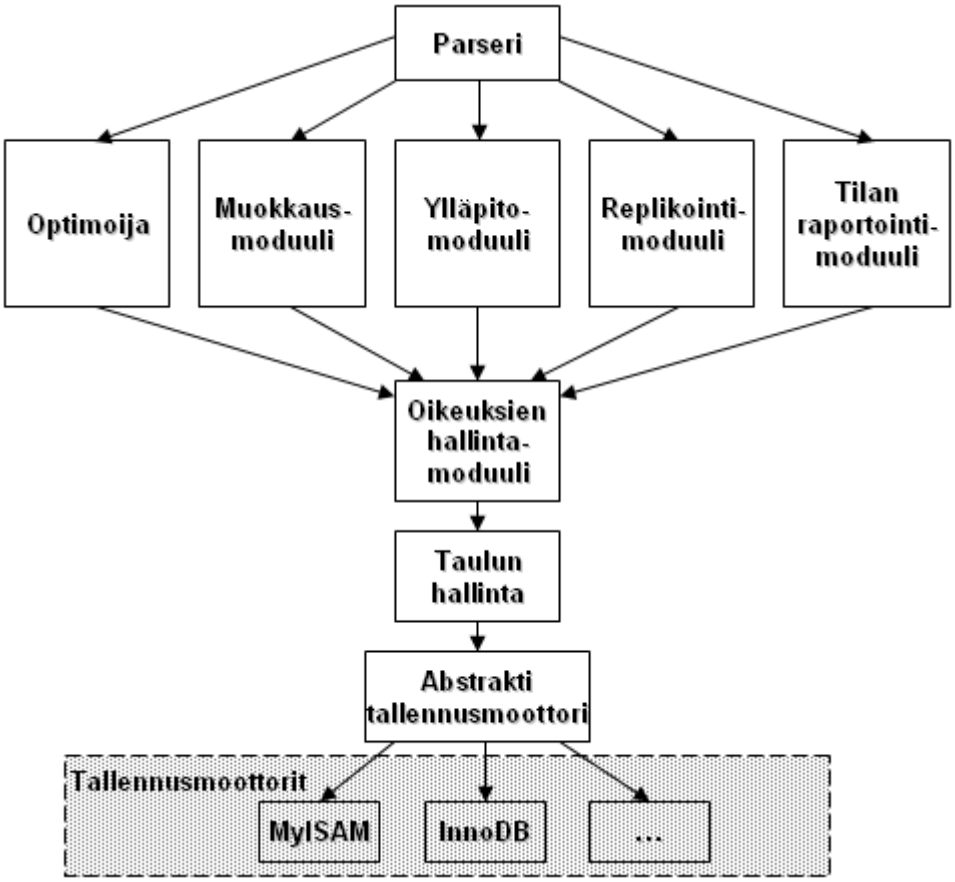
Tällä hetkellä suosituimpia ovat tiedonhallintajärjestelmät, joissa tiedon looginen tallennus tehdään relaatiomallin mukaisesti. Usein myös tietovarastot on rakennettu relaatiotietokantoihin. MySQL AB on perustettu 1995 ja se omistaa oikeudet suurimpaan osaan MySQL-lähdekoodista. MySQL:ää mainostetaan maailman suosituimpana vapaan lähdekoodin tiedonhallintajärjestelmänä. (MySQL 2007, s. 4)

MySQL-tietokantaan on saatavilla sekä kaupallisia että vapaan lähdekoodin OLAP-työkaluja. Kaupallisia työkaluja ovat mm. EBM-Solutionsin OLAP4All, Tableau Softwaren OLAP Analysis for MySQL ja Visual Softin Visual SQL-Designer. Vapaan lähdekoodin OLAP-työkaluista löytyi vain yksi kehittyneempi vaihtoehto: Pentaho Mondrian. Se on suosituin OpenSource BI-työkalu *(engl. Business Intelligence)*, joka tarjoaa OLAP-toiminnot käytettäväksi myös MySQL-tietokannassa. Mondrian antaa tietokannan huolehtia tiedon tallennuksesta, SQL-kyselyiden suorituksesta ja aggregoinneista. Sen avulla relaatiotietokannan tauluista voidaan määritellä tietokuutioita käyttäen XML-metatietoa. Tietokuution määrittelyä varten on myös olemassa työkalu, jolla määrittelyn voi tehdä kirjoittamatta XML-merkkauskieltä. Mondrianin kautta voidaan suorittaa MDX-kielisiä lausekkeita, jotka Mondrian kääntää SQL-kielelle ja suorittaa SQL-tietokannassa. (DeSouza et al. 2006)

3.6.1 Tallennusmoottorit MySQL-arkkitehtuurissa

Mielenkiintoisinta MySQL-arkkitehtuurissa ovat tallennusmoottorit, jotka on jätetty myös tietokannan ylläpitäjälle näkyviin. Etuna on se, että jokaisessa tallennustilanteessa voidaan valita parhaiten soveltuva tallennusmuoto. Tallennusmoottorit vastaavat taulun fyysisestä tallennuksesta. Niitä on mahdollista tehdä myös itse ja viimeisimmän MySQL-version myötä niiden liittäminen palvelimeen on yhä helpompaa, koska tallennusmoottori voidaan tehdä ns. liitännäisenä. (Schumacher 2004) Tiedon tallentamiseen ei ole olemassa sellaista menetelmää, joka toimii mille tahansa tiedolle optimaalisesti käyttötilanteesta riippumatta. Siksi on hyvä, että tallennusmoottori voidaan valita jokaiselle taululle erikseen.

MySQL-tietokannan käyttäjä voi antaa palvelimelle kahden tyyppisiä pyyntöjä: komentoja ja kyselyjä. Komennot voidaan suorittaa välittömästi komennon käsittelijän toimesta, mutta kyselyt ohjataan parserille, joka selvittää kyselyn rakenteen. Kyselynä voidaan pitää mitä tahansa MySQL:n ymmärtämää SQL-lauseketta. Kyselyn eteneminen parserista eteenpäin on esitetty kuvassa 12. SELECT-kyselyt ohjataan parseroinnin jälkeen optimoijalle, joka optimoi kyselyn suoritustavan. INSERT, UPDATE, DELETE ja muut taulua muokkaavat kyselyt annetaan taulunmuokkausmoduulin käsiteltäväksi. MySQL sisältää myös taulun ylläpitoa, replikointia ja tilatietojen kyselyjä varten omat moduulinsa. Se moduuli, joka suorittaa kyselyn, lähettää ensin listan kyselyyn liittyvistä tauluista oikeuksienhallintamoduulille, joka tarkastaa onko käyttäjällä riittävät oikeudet kyselyn suorittamiseen. Jos oikeudet riittävät, lähetetään tauluista lista edelleen taulunhallintamoduulille, joka suorittaa tarvittavat taulujen lukitsemiset. Tämän jälkeen kyselyä suorittava moduuli voi lähettää suoraan pyyntöjä tallennusmoottorille. Konkreettiset tallennusmoottorit periytyvät abstraktista tallennusmoottorista ja toteuttavat yleiset tallennusmoottoreissa tarvittavat metodit. Näin kyselyä suorittavan moduulin ei tarvitse tietää, minkä tallennusmoottorin kanssa se on tekemisissä. (Pachev 2007, s. 4-8)



Kuva 12. MySQL kyselyn eteneminen tallennusmoottoreille

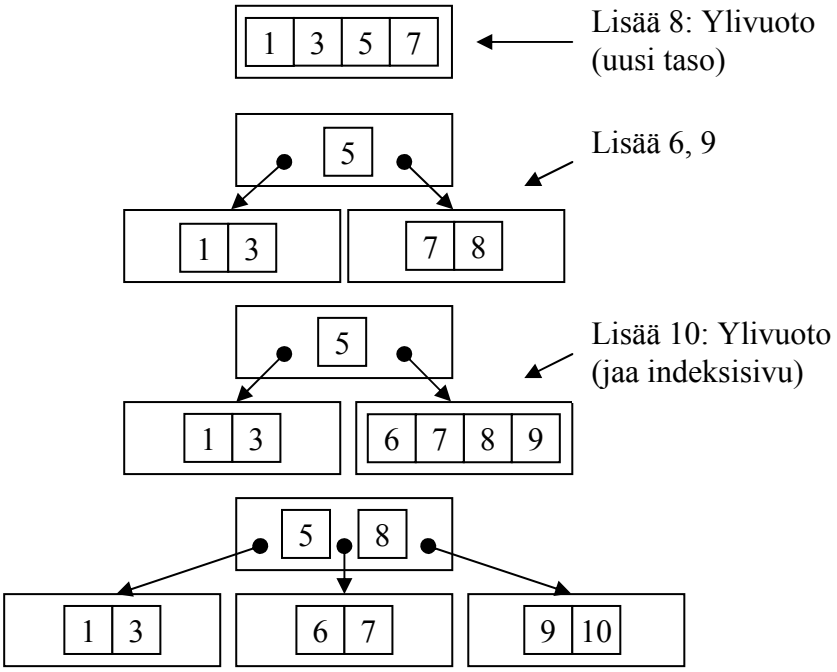
3.6.2 MySQL-taulujen indeksointimenetelmät

Indeksien tarkoitus on tarjota nopea hakupolku haettavaan tietoon. Sitä voidaan verrata esimerkiksi kirjan hakemistoon, jossa on sivuviittaukset eri termeille. Sisällysluettelo on muutaman sivun mittainen, kun taas kirja voi olla satojen tai tuhansien rivien mittainen. Tietyn termin löytäminen selaamalla kirjaa on hidasta, mutta hakemiston avulla tieto löytyy nopeasti. Vastaava tilanne on myös tietokannan tauluissa, joissa tiedon läpikäynti rivi riviltä on hidasta. (Elmasri & Navathe 2004, s. 456) Tässä kappaleessa on selostettu sellaisten yleisten indeksointimenetelmien toiminta, joita hyödynnetään myös MySQL-tauluissa. Lisäksi on kerrottu bittikarttaindeksoinnin eduista moniulotteisen tiedon indeksoinnissa.

Indeksitietueet *(engl. index record tai index entry)* koostuvat avain-osoite-pareista, jotka tallennetaan indeksisivuille *(engl. index page tai block)* avaimen mukaan yleensä nousevaan järjestykseen. Indeksisivu vastaa yleensä fyysisen tallennusmedian tallennuslohkoa. Avain vastaa indeksoitavan kentän arvoa ja osoitin osoittaa avainta vastaavalle datasivulle *(engl. data page)* eli todelliseen taulun riviin. (Elmasri & Navathe 2004, s. 457)

B-puuindeksit *(engl. B-tree)* ovat suosittuja ja myös monissa MySQL-tauluissa käytetään niitä. Sitä käytetään mm. MyISAM-, MEMORY-, NDB CLUSTER- ja InnoDB-tauluissa. Puurakenteen avulla tietty avainarvo on mahdollista löytää nopeasti seuraamalla puun osoitinhaaroja. Kuvassa 13 on näytetty B-puun lisäysoperaatiot lukujonolle 1, 3, 5, 7, 8, 6, 9 ja 10 sellaiseen puuhun, jonka indeksisivulle mahtuu neljä indeksitietuetta. B-puun täyttö aloitetaan juurisivulta. Kun juurisivu on täynnä ja siihen yritetään lisätä uutta arvoa, luodaan kaksi uutta indeksisivua, jotka tulevat juurisivun lapsiksi (kuvassa 13 arvon 8 lisäys). Juuritietueeseen jää ainoastaan keskimmäinen tietue ja viittaukset molempiin lapsitietueisiin. Kun muu kuin juurisivu tulee täyteen, jaetaan se kahtia sillä tasolla, jolla tietueet ovat (kuvassa 13 arvon 10 lisäys). Solun keskimmäinen arvo tallennetaan jaettavan sivun vanhempitietueeseen. Kun vanhempitietueet tulevat täyteen, jaetaan ne taas kahtia ja niin edelleen. Juurisivun jako aiheuttaa aina uuden juuren muodostumisen. (Elmasri & Navathe 2004, s. 469-473)

Kun tietueita poistetaan, tullaan väistämättä tilanteeseen, jossa indeksisivu on alle 50-prosenttisesti täysi. Tässä tilanteessa kyseinen indeksisivu yhdistetään sen naapurustoon (viereiset oksat). Indeksisivujen täyttö- ja poistotavoista johtuen jokainen indeksisivu on aina 50-100-prosenttisesti täysi. Simuloimalla on myös osoitettu, että satunnaiset lisäys- ja poisto-operaatiot johtavat siihen, että indeksisivut ovat keskimäärin noin 69-prosenttisesti täysiä. (Elmasri & Navathe 2004, s. 473)



Kuva 13. B-puun tietueiden lisäys

Yleisin B-puun variaatio on niin sanottu B+-puu. Perinteisen B-puun jokaisesta tietueesta osoitetaan datasivuille. B+-puussa datasivuille osoitetaan ainoastaan puun lehdistä eli alimman tason sivuilta. Koska B+-puussa tarvitaan dataosoittimia ainoastaan lehtitasolla, jää muille indeksisivuille enemmän tilaa avaimille ja puuosoittimille. Tämä johtaa matalampaan ja nopeampaan puurakenteeseen. Yleensä myös lehtisivut ovat linkitetty toisiinsa, jolloin kahden avainarvon välille jäävät datasivut voidaan hakea tehokkaasti. (Elmasri & Navathe 2004, s. 474-475) MySQL:ssä käytetään B+-puuindeksointia ainoastaan InnoDB-tauluissa.

B-puuindeksoinnit toimivat hyvin sellaisien kenttien indeksoinnissa, joissa on suuri kardinaliteetti eli paljon eri arvoja suhteessa rivien kokonaismäärään. Tietovarastoissa on paljon sellaista tietoa, jolla on matala kardinaliteetti eli paljon toistuvaa tietoa. Tällaisen tiedon indeksointiin on olemassa tehokkaampia keinoja.

Bittikarttaindeksit toimivat hyvin matalan kardinaliteetin kentille. Niissä jokaiselle avainsarakkeessa esiintyvälle arvolle muodostetaan oma bittivektori, joka on taulun korkeuden mittainen. Vektorissa on arvo ’1’ kaikilla riveillä, joissa avainarvo esiintyy. Muissa kohdissa arvo on ’0’. Bittikarttaindeksien tehokkuus perustuu siihen, että yksi vektori vie erittäin vähän tilaa ja on nopea käydä läpi. (Hovi 1997, s. 112) Jos taulussa on miljoona riviä, tarvitaan yhteen vektoriin miljoona bittiä, joka tarkoittaa noin 125 kilotavua ja pakattuna vielä vähemmän. Lisäksi vektorien välillä

on mahdollista tehdä bittitason operaatioita, joista prosessorit suoriutuvat erittäin nopeasti. MySQL:n tallennusmoottorit eivät kuitenkaan tue bittikarttaindeksointia.

Koska bittikarttaindeksoinnissa jokaiselle eri arvolle joudutaan tekemään oma vektorinsa, voitaisiin ajatella, että mitä enemmän sarakkeessa on mahdollisia arvoja niin sitä enemmän tarvitaan vektoreita ja tallennustilaa. Pakatuilla bittikarttaindekseillä tallennustarve ei kuitenkaan ole aivan näin massiivinen. K. Stockinger ja kumppanit ovat kehittäneet FastBit-nimisen bittikarttaindeksien pakkausmenetelmän. FastBit-pakatun bittikarttaindeksin hakunopeutta on verrattu MySQL-taulujen hakunopeuksiin. Kyseisessä tutkimuksessa käytettiin dataa, joka on koottu Enronin työntekijöiden sähköpostilogeista. Näissä tapauksissa FastBit osottautui 10-1000 kertaa MySQL-tauluja nopeammaksi. Merkittävää oli, että haettaessa moniulotteisesta datasta, haussa esiintyvien dimensioiden määrän kasvaessa kasvaa myös FastBit-pakatun indeksin ylivertaisuus. Nimenomaan moniulotteista tietoa haettaessa päästiin 1000-kertaiseen nopeuteen. (Stockinger et al. 2006) Tutkimuksesta ei selvinnyt tarkasti, minkälaisilla rakenteilla testeissä käytetty tieto on tallennettu. Näitä nopeuseroja ei siksi välttämättä voida rinnastaa vastaavaksi jokaiseen moniulotteisella mallilla MySQL-ympäristössä kehitettyyn tietovarastoon, mutta se antaa kuitenkin viitteitä siitä, että bittikarttaindeksit todella toimivat hyvin moniulotteisissa malleissa.

Moniulotteisessa mallissa dimensiot ovat täysin denormalisoituja, jonka takia niissä on paljon toistuvaa tietoa. Tämän vuoksi nimenomaan dimensiotauluissa hyödyttäisiin bittikarttaindekseistä. B-puu-indekseillä joudutaan tekemään monen sarakkeen indeksejä, jotka toimivat ainoastaan silloin, kun haussa esiintyy vähintään ensimmäinen monisarakkeisen indeksin kenttä. Bittikarttaindekseillä voidaan indeksoida jokainen sarake erikseen, jolloin haut toimivat hyvin kaikilla ennalta arvaamattomillakin hakuehdoilla.

MySQL:ssä MEMORY- ja InnoDB-tauluissa hyödynnetään HASH-indeksiä. HASH-indeksi perustuu sellaiseen HASH-funktioon, joka jakaa avaimet indeksisivuille tasaisesti, mutta satunnaisesti. Oikea indeksisivu saadaan haettua levyltä nopeasti, koska HASH-funktion ja hakuavaimen avulla voidaan laskea suoraan oikea indeksisivun osoite. Indeksisivulle tulee useiden eri avainarvojen tietueita, jonka takia indeksisivu joudutaan käymään aina kokonaan läpi. HASH-indeksi toimii parhaimmin

silloin, kun taulusta haetaan yksittäisiä arvoja. Jos haetaan arvojoukkoa tiettyjen rajojen sisältä, joudutaan tekemään paljon hakuja HASH-funktion avulla. Tässä tilanteissa B-puu päihittää HASH-indeksin. (Elmasri & Navathe, s. 485, 434)

3.6.3 MySQL-tallennusmoottorit tietovarastoissa

Tietovarastossa tarvitaan erilaisia tallennusmuotoja mm. eri-ikäisen datan tallentamiseen, taustaprosesseihin ja metatiedon tallentamiseen. Taulun tallennusmoottorin vaihtaminen on myös mahdollista yksinkertaisen ALTER TABLE -komennon avulla, tosin muutoksessa voi hävitä sellaisia tietoja, joita tallennusmoottori ei tue. Mistä tahansa taulusta voidaan tehdä väliaikainen käyttäen TEMPORARY-sanaa CREATE-sanan perässä. Väliaikainen taulu tuhotaan silloin, kun tietokantayhteys päättyy tai palvelin ajetaan alas. Liitteessä 1. on taulukko, johon on koottu yleisimpien tallennusmoottoreiden ominaisuuksia. (Lenz 2004)

Yleisimmin käytettävät tallennusmoottorit ovat InnoDB ja MyISAM. InnoDB tukee viiteavaimia ja transaktioita ja soveltuu hyvin OLTP-järjestelmään. Vastaavasti MyISAM-taulun lukeminen on erittäin nopeaa ja sen käyttö soveltuu paremmin käyttötarkoituksiin, joissa tiedon nopea saatavuus on tärkeintä. MyISAM-taulu vie InnoDB-taulua vähemmän levy- ja muistikapasiteettia. MyISAM-tauluissa käytetään taulukohtaisia lukituksia, kun taas InnoDB-taulussa voidaan lukita yksittäisiä rivejä, joka mahdollistaa transaktiotyyppisen käytön. MyISAM-taulun voi myös pakata. Pakattuun MyISAM-tauluun ei voi kirjoittaa. (Lenz 2004)

MyISAM-tauluissa käytetään perinteistä B-puuindeksiä ja InnoDB-taulussa B+-puuindeksiä. InnoDB-taulussa on aina oltava pääavain, koska se käyttää ryvästettyä indeksiä, jossa jokainen puun lehti sisältää myös rivin tiedot. InnoDB-taulu osaa myös automaattisesti arvioida hyödyttäisiinkö haussa HASH-indeksistä. Jos hyödytään, niin se luo automaattisesti sellaisen muistiin jo olemassa olevien B-puuindeksien perusteella. MyISAM-tauluissa sen sijaan käytetään yksinkertaista B-puuindeksointia, jossa jokainen lehti sisältää ainoastaan viittauksen taulun rivi-informaatioon. Näin ollen MyISAM-taulussa ei välttämättä tarvita pääavainta. MyISAM-tauluissa ovat mahdollisia myös r-tree indeksi maantieteellisen datan indeksointiin ja fulltext indeksi tekstikenttien tehokasta hakua varten. (MySQL 2007, s. 815; Pachev 2007, s. 194-204)

Koska InnoDB-tauluilla voidaan varmistaa viite-eheys, soveltuu se hyvin esimerkiksi metatiedon ja muun tietovaraston hallinnassa käytettävän muuttuvan tiedon tallentamiseen. MyISAM taasen soveltuu parhaimmin datataulujen tallennukseen, koska se on nopeudessa ja tilatehokkuudessa ylivoimainen.

MERGE tallennusmoottoria voidaan käyttää sellaisten MyISAM-taulujen yhdistämiseen joiden sarakemäärittelyt ovat identtisiä. MERGE-taulua voidaan käyttää hyväksi silloin, kun data on jaettu esimerkiksi ajan perusteella identtisiin tauluihin ja kaikkeen dataan halutaan päästä käsiksi yksinkertaisesti yhden taulun kautta. (Lenz 2004) Tietovarastossa tieto on monesti pilkottu ajan perusteella eri tauluihin, jotta yhden taulun koko pysyisi pienenä ja siten helpommin hallittavana. Esimerkiksi faktataulut tuotanto_2006 ja tuotanto_2007 voitaisiin yhdistää MERGE-taululla, jolloin molempien taulujen tiedot ovat saatavilla yhden taulun kautta.

MEMORY-tallennusmoottorissa taulun sisältö pidetään jatkuvasti tietokoneen keskusmuistissa. Tätä tallennusmoottoria voidaan käyttää tiedon väliaikaisena tallennuspaikkana tai usein tarvittavien pienten taulujen tallennukseen. Se tukee B-puuindeksien lisäksi HASH-indeksointia, joka mahdollistaa nopeat täsmähaut. Taulu on pysyvä, mutta sen sisältö haihtuu palvelimen uudelleenkäynnistyksen yhteydessä. (Lenz 2004) Hyötyä tietovarastossa MEMORY-taulusta voidaan saada esimerkiksi silloin, kun tietoa tuodaan lähdetietokannasta tietovarastoon. Tällä voidaan luoda nopeita lookup-tauluja, joista dimensioiden sijaisavaimet saadaan selvitettyä nopeasti lähdetiedolle. Sitä voidaan käyttää myös OLAP-moduulin optimoinnissa. Esimerkiksi silloin kun käyttäjä suorittaa pivot-komennon eli ryhmittelee tietoa rinnakkain, ei tieto itsessään muutu. Siksi riittää, että olemassa olevan tiedon esitysmuotoa muutetaan. Se saadaan suoritettua helposti ja nopeasti, kun viimeisin haettu tieto on valmiiksi keskusmuistissa. Dimensiotaulut ovat yleensä melko pieniä ja niitä luetaan paljon. Siksi tämä voi soveltua myös niiden tallentamiseen.

FEDERATED-tallennusmoottoria voidaan käyttää viitteenä tauluun, joka fyysisesti sijaitsee toisella MySQL-palvelimella. (MySQL 2007, s. 834) Tietovarastossa sitä voidaan käyttää tiedon tuontiin silloin, kun lähdetietokantana on MySQL-tietokanta. Näin tieto näyttää paikalliselta, vaikka se sijaitseekin etäpalvelimella. Kaikki tiedonmuunnosoperaatiot suoritetaan suoraan tietovaraston puskuritietokannassa ilman erikseen suoritettavaa tiedonpoimintavaihetta. Tällaisesta käyttötavasta voi olla

etua myös siksi, koska tiedon luku tehdä eri järjestelmässä kuin kirjoitus. Levy-I/O -toiminnot (Input / Output) eivät näin pääse muodostumaan pullonkaulaksi.

CSV-tallennusmoottori (Comma Separated Value(s)) on ns. “flatfile” tallennusmoottori. CSV-tiedostoissa kaikki tiedot on tallennettu selkokielisesti pilkulla erotettuna. Yksi CSV-moottorilla luotu taulun rivi vastaa myös tiedostossa yhtä tekstiriviä. (MySQL 2007, s. 838) Usein tiedot tuodaan lähdetietokannoista tietovaraston saataville esimerkiksi ftp:llä CSV-muodossa. CSV-moottorin avulla tiedosto saadaan helposti tiedonhallintajärjestelmän saataville, kuten mikä tahansa muu taulu. Ensin täytyy luoda taulu, joka vastaa sarakemäärittelyiltään tietokantaan tuotavaa CSV-tiedostoa. Tämän jälkeen tiedostojärjestelmässä korvataan kyseisen taulun datatiedosto tuotavalla CSV-tiedostolla. Tiedot ovat välittömästi tämän jälkeen tiedonhallintajärjestelmän saatavilla. (Schumacher 2007)

ARCHIVE-tallennusmoottori tallentaa taulun häviöttömällä zlib-pakkausalgoritmilla erittäin pieneen tilaan. (MySQL 2007, s. 831-832) Tiedon hakeminen on kuitenkin hidasta, koska taulua ei voida indeksoida ja pakkauksen purkaminen vaatii suoritintehoa. ARCHIVE-moottorin tauluun on kuitenkin nopeampaa lisätä tietoa kuin MyISAM-moottorin tauluun. Lisäksi se on nopeampi silloin, kun indeksejä ei voida hyödyntää ja taulua joudutaan käymään läpi järjestelmällisesti rivi riviltä. ARCHIVE-tauluun on mahdollista tehdä vain INSERT- ja SELECT-kyselyjä, jonka takia tallennetut rivit eivät varmasti muutu. (Schumacher 2007) ARCHIVE-tauluja kannattaa käyttää tietovarastossa vanhimpien historiatietojen tallentamiseen. Tarvittaessa ARCHIVE-tallennusmuodosta tiedon muuttaminen takaisin tehokkaaseen MyISAM-tallennusmuotoon on helposti tehtävissä muutamalla SQL lauseella.

MySQL:n mukana tulevien tallennusmoottoreiden lisäksi saatavilla on kolmansien osapuolten valmistamia kaupallisia tallennusmoottoreita. NitroEDB on Nitrosecurityn kehittämä tallennusmoottori, joka pystyy käsittelemään nopeasti erittäin suuria tietomääriä ja soveltuu siksi hyvin tietovarastokäyttöön. Se tarjoaa markkinoiden parhaan volyymin ja tehon suhteessa hintaan. (Computerworld 2006) BrightHousen kehittämän InfoBright-tallennusmoottorin sanotaan tarjoavan markkinoiden parhaan tiedonpakkaussuhteen (10:1) (Infobright 2007). Siinä tiedon tallennus tehdään sarakekohtaisesti, jolloin haku voi toimia erittäin nopeasti, koska hauissa ei tarvitse lukea kokonaisia rivejä. InfoBrightin pitäisi olla pakkaukseltaan ARCHIVE-moottoria

tehokkaampi ja silti hakuajoiltaan nopea. Siksi se soveltuu hyvin suurien historiatietomäärien tallentamiseen. (Schumacher 2007)

3.6.4 Osiointi

Osioinnin perusidea on jakaa taulu osiin niin, että kyselyssä ei välttämättä tarvitse lukea kuin yhtä taulun osaa koko taulun sijaan. Esimerkiksi miljoonan rivin taulu voidaan jakaa kymmeneksi sadan tuhannen rivin osioksi ja jos kysely osuu vain tiettyihin osioihin, ei muita osioita avata.

Osiointi voidaan tehdä joko horisontaalisesti tai vertikaalisesti. Horisontaalisella osioinnilla taulun rivit ryhmitellään osioiksi. Vastaavasti vertikaalisella osioinnilla ryhmitellaan taulun sarakkeet osioiksi. Vertikaalisella osioinnilla voidaan esimerkiksi erottaa usein käytetyt sarakkeet omaan osioon ja harvoin käytetyt omaansa.

Tällä hetkellä kehitysasteella oleva MySQL 5.1 versio tukee taulun horisontaalista osiointia. Osioinnilla voidaan saada tehollisia ja hallinnallisia hyötyjä tietovarastoissa. Osioinnissa taulun eri osiot voidaan myös tallentaa eri levyasemille, jolloin kyselyssä jokaista eri levyä voidaan lukea rinnakkain. Tämä mahdollistaa taulun lukemisen huomattavasti nopeammin silloin, kun kysely kohdistuu useisiin osioihin. (Schumacher 2006)

Jos osioitua taulua verrataan MERGE-tauluun, erona on se, että osioitu taulu tietää, mitä dataa on missäkin osiossa ja sopivan osion etsiminen jää silloin tiedonhallintajärjestelmän vastuulle. MERGE-moottori on tietämätön datan jakautumasta ja logiikka joudutaan tekemään ohjelmallisesti. Osioitu taulu näkyy käyttäjälle yhtenä tauluna, mutta todellisuudessa jokaisella taulun osiolla on omat tiedostonsa.

MySQL:ssa osioinnin voi tehdä neljällä eri tavalla: range, list, hash tai key. Osioiden sisällä voi olla myös aliosioita. Aliosioiden sisällä ei voi kuitenkaan olla uusia aliosioita. Taulussa voi olla enintään 1024 osiota aliosiot mukaan luettuna. Range-osioinnissa taulu jaetaan osiin tiettyjen rajojen mukaan. Tieto voidaan osioida esimerkiksi vuosittain osioihin { < 2000, <= 2005, > 2005 }, jossa ensimmäinen sisältää kaiken ennen vuotta 2000 tulleen datan ja seuraava sisältää datan vuodesta 2000 vuoteen 2005 asti ja viimeinen sisältää kaiken datan, jota tulee vuodesta 2005 eteenpäin. List-osioinnissa jokaisella osiolla on lista niistä arvoista, joita kentällä voi

olla kyseisessä osiossa. Hash-osioinnilla taulun data saadaan jaettua automaattisesti yhtä suuriin osiin. Voidaan esimerkiksi osioida taulu tiettyjen kenttien perusteella 10 osaan eikä käyttäjän itse tarvitse määritellä osiin kuuluvia kenttien arvoja. Key-osiointi on kuten hash, mutta sen muodostuksessa on käytettävä ainakin osittain taulun pääavainta. Tämä takaa datan tasaisemman jakautumisen osioihin. (Schumacher 2006)

Hallinnallisia hyötyjä osioinnilla saadaan mm. siten, että esimerkiksi vanhentuneen osion voi pudottaa taulusta vaikuttamatta muuhun taulun sisältöön. Jos osiointi on tehty epätehokkaasti, voidaan siihen tehdä muutoksia jälkikäteen ALTER TABLE -kyselyllä. Osioita voidaan mm. liittää yhteen, jakaa osiin ja lisätä. Osioituun tauluun ei voi laittaa sellaista riviä, jolle ei ole olemassa osiota. (Schumacher 2006) Tätä ominaisuutta voidaan hyödyntää esimerkiksi silloin, kun tietoa tuodaan lähdetietokannoista tietovarastoon. Taulu voidaan osioida mittausarvon perusteella niin, että järjellisten mittausarvojen ulkopuolelle jäävät mittaukset (mittausvirheet) eivät tallennu tauluun. Toisaalta jos halutaan tietää myös mittausvirheet, voidaan niille tehdä omat osiot, joista ne ovat erikseen haettavissa.

MySQL-osioinnissa on vielä paljon puutteita, joita varmaan tulevissa versioissa tullaan korjaamaan. Esimerkiksi osioidussa MyISAM-taulussa lukitaan kaikki osiot, joka periaatteessa tarkoittaa samaa asiaa kuin koko taulun lukitseminen. Tulevaisuudessa tämä tulee muuttumaan siten, että ainoastaan ne osiot lukitaan joita tarvitaan. Näin useammat käyttäjät voivat käyttää rinnakkain samaa taulua, kunhan heidän hakunsa kohdistuvat eri osioihin. (Ronstrom 2006)

3.6.5 Scale-up vs. Scale-out

Vaikka MySQL onkin pääasiassa OLTP-tiedonhallintajärjestelmä, on sitä käytetty myös monissa tietovarastoprojekteissa. MySQL-tietovarastoissa käytetään yleensä rinnakkaista arkkitehtuuria, jossa tietovarasto koostuu useista tavallisista palvelintietokoneista. Kapasiteettia voidaan lisätä tasaisesti tarpeen mukaan, koska yksi palvelin ei tuo suuressa järjestelmässä suurta kapasiteetin lisäystä, mutta ei myöskään suurta kulujen lisäystä. Tämä tekee järjestelmästä myös virheensietokykyisemmän, koska yhden palvelimen kaatuminen aiheuttaa vain pientä järjestelmän hidastumista. Huonona puolena on se, että järjestelmästä tulee monimutkainen. Tällaista tietovaraston skaalaustapaa kutsutaan scale-out-

skaalaukseksi. MySQL-tietovarastossa se kannattaa siksi, koska MySQL-lisenssin hinta yhtä palvelinta kohti on erittäin pieni. (Schiff 2005; MySQL 2005)

Scale-up-arkkitehtuurissa on vähän palvelimia, mutta ne ovat erittäin tehokkaita ja kalliita. Esimerkiksi 12 prosessorin SMP-palvelin (Symmetric Multiprocessing) voisi maksaa 600 000$ ja siihen Oracle Enterprice Edition tiedonhallintajärjestelmät yhteensä 585 600$ (48 800$ / prosessori). Toisaalta 20 yhden prosessorin peruspalvelinta voisi maksaa esimerkiksi 75 000$ ja näihin MySQL Network Gold lisenssi yhteensä 60 000$. Tässä esimerkissä säästö olisi noin miljoonan dollarin luokkaa. On arvioitu, että esimerkiksi Cox Communications, joka varastoi dataa noin 1,2 miljoonasta asiakkaidensa kaapelimodeemista, säästää noin 2,2 miljoonaa dollaria pitäessään tietovarastonsa MySQL-arkkitehtuurilla. Nämä arviot ovat MySQL AB:n itse tekemiä. (MySQL 2005)

MySQL-tietovaraston rinnakkainen tiedon tallennus tehdään yleensä käyttäen replikointia. Replikoinnissa tieto kopioidaan useille identtisille palvelimille. Käytännössä replikointi toimii siten, että kokoonpanossa on yksi master-palvelin, joka tallentaa kaikki tapahtumat binäärilogiin. Slave-palvelimet (replikaatit) on asetettu lukemaan ja suorittamaan master-palvelimen binäärilogiin tallennettuja tapahtumia. Replikointi toimii hyvin silloin kun järjestelmässä luetaan paljon, mutta kirjoitetaan vähän. Tämä johtuu siitä, että kirjoitus tehdään aina pelkästään master-palvelimen kautta eikä kirjoitustehokkuutta voi skaalata. Kun tämä raja tulee vastaan, tarvitaan useampia master-palvelimia. Silloin tieto joudutaan jakamaan osiin esimerkiksi sovellusalueittain. Tietovarastossa tämä osakokonaisuus voisi hyvin olla esimerkiksi yksi paikallisvarasto. Mitä riippumattomampia eri replikaatio-kokoonpanojen tiedot ovat toisistaan, sen paremmin järjestelmä toimii. Myös puskuripalvelin voidaan jakaa osiin replikoimalla, jos lähdejärjestelmiä on paljon tai tietoa tulee niin paljon, ettei yksi puskuripalvelin ehdi sitä käsittelemään. (MySQL 2005; Sennhauser 2006)

4 MITTAUSTIETOKANNAN RAKENTEEN SUUNNITTELU

Diplomityön tavoitteena on kehittää tietovarasto tehdasmittausten varastointiin. Rakenteellisessa suunnittelussa suunnitellaan tietokantajärjestelmän arkkitehtuuri, jossa tietokanta sekä sen ohjelmistot jaetaan loogisiin osiin. Tämän jälkeen tietovaraston kohdeympäristöä kuvataan käsitteellisesti. Loogisessa suunnitteluvaiheessa suunnitellaan moniulotteinen malli ja mallin kuvauksessa käytettävä metatieto. Fyysisessä suunnittelussa valitaan käsitteellisessä ja loogisessa vaiheessa suunnitelluille osille fyysiset MySQL-tietokannan rakenteet.

4.1 Tallennustilan tarve moniulotteisella tietomallilla

Moniulotteisessa tietomallissa mittauksia ei laiteta rinnakkain, jos ne eivät liity vahvasti toisiinsa. Rinnakkain voisi olla esimerkiksi hakepalan pituus ja leveys, koska nämä yhdessä kuvaavat hakepalan kokoa. Nämä tapaukset ovat kuitenkin harvinaisempia ja yleisintä on, että mittaukset tallennetaan yhteen sarakkeeseen.

Moniulotteisella tietomallilla toteutetun tietokannan koon voidaan olettaa kasvavan suoraan verrannollisesti tallennettavien näytteiden määrän suhteen, koska jokainen näyte vaatii lähes täsmälleen saman verran tallennustilaa (dimensioviittaukset + näyte + indeksi). Dimensiotaulujen ja esimerkiksi metatiedon kapasiteettivaatimuksia ei tarvitse laskea, koska ne muodostavat häviävän pienen osan tietokannan koosta. Tässä kappaleessa on arvioitu, kuinka paljon tallennuskapasiteettia yhden näytteen tallennus vie vähintään MyISAM-tallennusmoottorilla ja laskettu kuinka paljon kaikkien näytteiden tallentaminen vaatii vuosittain tallennuskapasiteettia.

Jos yhteen tauluun tallennetaan yli 16 miljoonaa mittausta niin, että kaikki näytteet laitetaan samaan sarakkeeseen, täytyy pääavaimen olla vähintään 4 tavua. Jos mittaus on esimerkiksi liukuluku, joka tarvitsee myös 4 tavua, saadaan yhden rivin tallennustilaksi ilman indeksiä 8 tavua. MyISAM-taulun B-puu indeksin yhden rivin vaatiman tilatarpeen yläraja saadaan laskettua kaavan 4 avulla, jossa l_{key} on avaimen pituus (MySQL 2007, s. 777). Lähteessä ei ole tarkemmin selvitetty, mistä tämä kaava on johdettu. Kaavan tuntemattomat jäsenet ovat kuitenkin pääteltävissä. Indeksin jokaisessa tietueessa tarvitaan avainarvo ja riviviittaus. MyISAM-taulu voi olla 2^{32}

rivin korkuinen eli kuvattavissa neljällä tavulla, mistä osoittajan vakio 4 on todennäköisesti johdettu. B-puuindeksin sivut eivät ole aivan täysiä, koska indeksin arvojen lisäys ja poisto jakaa ja yhdistää indeksisivuja. Koska kaava antaa indeksin vaatiman tilan pahimman tilanteen mukaan, niin oletettavasti vakio 0,67 on johdettu pahimman tilanteen täyttöasteesta.

$$V_{index} = \frac{(l_{key} + 4)}{0.67} \tag{4}$$

Indeksin vaatimaksi tilaksi saadaan näin 12 tavua. Kun datan ja indeksin tarvitsemat tilat lasketaan yhteen, saadaan yhden näytteen tai rivin vaatimaksi tilaksi noin 20 tavua. Arvio on varmasti todellista määrää paljon pienempi, koska faktataulun pääavain koostuu erillisistä sijaisavaimista, mikä on yhteensä pidempi kuin yhden sarakkeen pääavain. Faktataulussa on myös dimensioita, jotka eivät osallistu rivin yksilöintiin. Tämä arvio antaakin meille vain sen alarajan, kuinka paljon tilaa näyte vie vähintään.

Järjestelmän täytyy kestää vähintään 6 miljardin näytteen vuosittainen kasvu. Tilatarve saadaan laskettua kaavan 3 avulla kertomalla näytteiden määrä yhden näytteen tilatarpeella. Tästä saadaan noin 117 GB vuosittainen tallennuskapasiteettitarve. Nykyisillä tallennusmedioilla tällainen määrä on vielä aika pieni. Näytteen tallennukseen saakin mennä vielä moninkertaisesti tätä arviota enemmän tilaa eikä se silti aiheuta ongelmia. Tämän vuoksi tilankäytön puolesta moniulotteista mallia voidaan käyttää ja tiedot voidaan pääasiassa säilyttää pysyvästi. Myös tallennusmedioiden kapasiteetit nousevat ja hinnat laskevat jatkuvasti. Tämän vuoksi tulevaisuudessa voidaan varastoida yhä suurempi osa mittauksista.

Maksimaalista kasvua arvioidessa täytyy ottaa huomioon muutakin kuin tallennuskapasiteetti. Esimerkiksi kuinka paljon tietoa verkko ehtii siirtämään tietovarastoon yön ja mahdollisesti viikonlopun aikana ja kuinka paljon tietoa ehditään puhdistamaan, muokkaamaan ja tallentamaan käytettävissä olevana aikana. Näillä mittausmäärillä viikossa dataa täytyisi siirtää ja käsitellä vain muutamia gigatavuja. Se on suhteellisen vähän, kun otetaan huomioon, että muunnosoperaatioita ei tarvita paljon, tietolähteiden yksinkertaisen tallennustavan vuoksi. Oletettavasti tiedonsiirto ja muunnosoperaatiot eivät näillä näytemäärillä aiheuta mitään ongelmia.

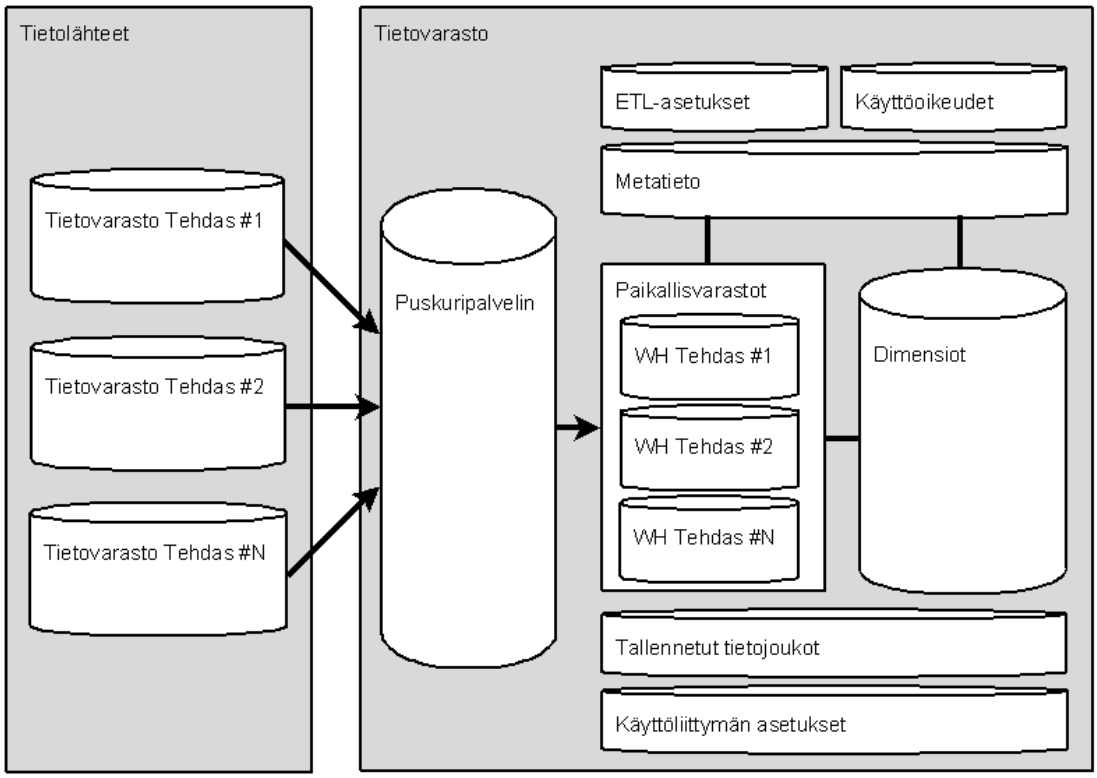
4.2 Arkkitehtuurisuunnittelu

Arkkitehtuurisuunnittelussa tieto jaetaan loogisina osakokonaisuuksina eri tietokantoihin, jotta sen hallinta olisi helpompaa. Myös tietovaraston toiminnalliset osat jaetaan loogisiin osiin. Tietokantaa varten on käytössä yksi Linux-palvelin, jossa käytetään MySQL-tiedonhallintajärjestelmää tiedon tallentamiseen.

4.2.1 Tietokannat

Tietovaraston väyläarkkitehtuurin ideana on, että tieto on jaettu erillisiin paikallisvarastoihin loogisina osakokonaisuuksina, usein liiketoimintaprosesseittain. Sellunvalmistusprosessia voidaan pitää yhdenlaisena liiketoimintaprosessina eikä paikallisvarastojakoa tältä kannalta ajateltuna tarvitsisi tehdä. Sellunvalmistusprosessit tehtaittain eroavat kuitenkin toisistaan yllättävän paljon. Myös mittauksia on niin paljon, että hallittavuudenkin kannalta ne kannattaa luokitella eri tietokantoihin. Näistä syistä johtuen tiedonhallintajärjestelmään tehtiin jokaista tehdasta varten oma paikallisvarasto, kuten kuvassa 14. Jokaiseen paikallisvarastoon tulee sekä atomista että summattua tietoa. Summattujen ja atomisten taulujen nimeäminen tehdään kuitenkin erilailla, jotta ne voidaan erottaa toisistaan. Karkeustaso ilmenee myös metatiedon avulla.

Jokainen paikallisvarasto hyödyntää yhteisiä dimensiotauluja, jotka sijaitsevat omassa tietokannassaan. Vaikka ulottuvuuksia onkin alussa vain muutamia, tulee niiden määrä tulevaisuudessa kasvamaan. Lisää ulottuvuuksia tulee esimerkiksi silloin, kun summatauluja varten tehdään osajoukkodimensioita tai kun dimensiolle tehdään rooleja näkymien avulla.



Kuva 14. Tiedon hajautus eri tietokantoihin

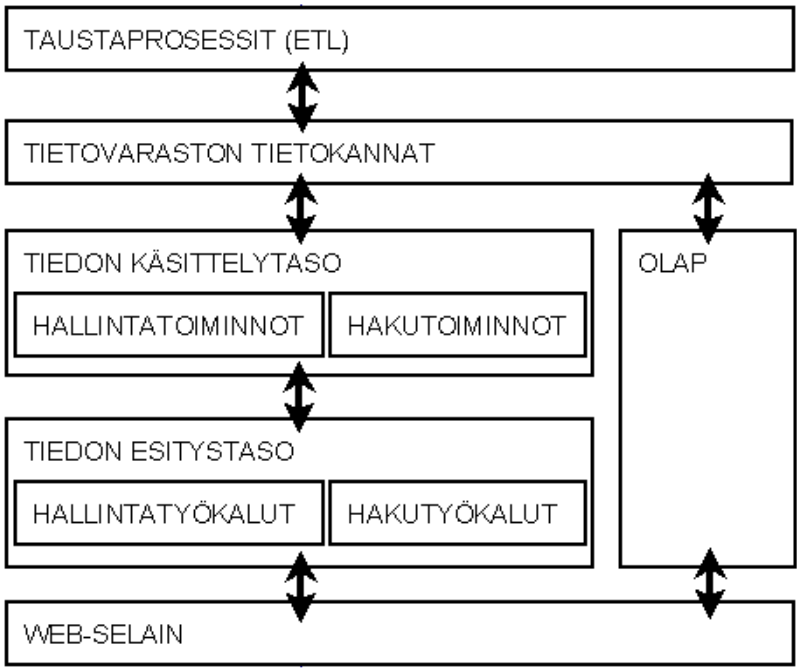
Relaatiotietokannalla toteutetussa tietovarastossa tarvitaan metatietoa moniulotteisen mallin kuvaamiseen. Metatietoa on paljon ja se muuttuu kehitysvaiheessa jatkuvasti ja sen päätyttyäkin ajoittain. Metatiedolla kuvataan mm. dimensiotaulut, paikallisvarastojen faktataulut, lähdetietokantojen lähdetaulut ja faktataulujen linkitykset lähdetietoihin ja dimensioihin. Myös taustaprosessien metatieto tallennetaan muun metatiedon kanssa samaan tietokantaan. Taustaprosessien asetukset ja niiden tarvitsemat lookup-taulut tallennetaan omaan tietokantaansa (kuvassa ETL-asetukset). Tiedon tuonti on monimutkainen tehtävä ja oletettavasti tulevaisuudessa sitä varten taulujen määrä tulee myös kasvamaan.

Tietovarastossa eri käyttäjille voidaan määritellä erilaisia käyttöoikeuksia. Joku käyttäjä saa päästä esimerkiksi vain yhden tehtaan tietoihin käsiksi. Käyttöoikeustaulut pidetään erillään muista tiedoista muodostamalla niitä varten oma tietokantansa. Tietovarastoon tullaan kehittämään erilaisia käyttöliittymiä ja sen vuoksi myös käyttöliittymien asetuksille tehtiin oma tietokantansa. Käyttäjät voivat tallentaa tietokannasta hakemiaan tietojoukkoja myöhempää käyttöä varten. Jokaista

tietojoukkoa varten luodaan uusi taulu. Koska tauluja tulee paljon, tallennetaan kaikki haetut tietojoukot selkeyden vuoksi yhteen tietokantaan.

4.2.2 Toiminnalliset osat

Tietokannan toiminnallisuus voidaan käyttäjien näkökulmasta jakaa tiedon hallintaan ja tiedon hakuun. Arkkitehtuuri jaetaan tiedon esitys- ja käsittelytasoihin siten, että tiedonkäsittelytaso tarjoaa toimintoja esitystason käyttöliittymille. Tiedonkäsittelytasolla on tietokantaliittymä, jonka kautta tiedonkäsittelytaso keskustelee tietokannan kanssa, kuten kuvassa 15. Käyttöliittymä erotetaan tiedonkäsittelystä, koska tiedon hauissa halutaan erilaisia käyttöliittymiä erilaisille käyttäjille. Jokainen käyttöliittymä voi hyödyntää esimerkiksi samoja kyselynmuodostustyökaluja. Tiedonkäsittelytasolla tarvitaan hallinnalliset toiminnot ainakin taustaprosessien hallintaan, mittausten hallintaan ja käyttöoikeuksien hallintaan. Mittausten hallinta kattaa faktojen ja ulottuvuuksien luomisen ja muokkaamisen sekä faktojen linkittämisen ulottuvuuksiin ja tietolähteisiin. Tietokantojen takana pyörivät taustaprosessit, jotka pumppaavat tietoa lähdetietokannoista tietovarastoon.



Kuva 15. Tietovaraston toiminnalliset osat

Tietokantaan voidaan tulevaisuudessa myös lisätä OLAP-toiminnallisuus. OLAP-moduuli toimii itsenäisesti ja tarvitsee ainoastaan liittymän tietokantaan. Pentaho

Mondrian on tällä hetkellä ainoa varteenotettava vapaan lähdekoodin OLAP-työkalu, joka soveltuu MySQL-tietokannan kanssa käytettäväksi.

Kaikkia toimintoja käytetään web-selaimen avulla, joka mahdollistaa käyttäjille paremman liikkuvuuden ja tiedon saannin eri paikoissa. Web-selain itsessään hoitaa tiedon esittämisen, joka myös helpottaa käyttöliittymän suunnittelua. Tiedon esitystason tarvitsee vain huolehtia sopivien sivujen tuottamisesta selaimen esitettäväksi.

4.2.3 Nimeämiskäytännöt

Koska tietokantaan tulee paljon tauluja, täytyy niiden nimeämisessä käyttää yhtenäistä linjaa. Esimerkiksi taulujen nimet aloitetaan aina pienellä kirjaimella, mutta etu- ja jälkiliitteet kirjoitetaan aina isolla. Jälkiliitteitä käytetään tiedon karkeuden kuvaamiseen. Etu- ja jälkiliitteet erotetaan taulun nimestä ja toisistaan ’_’-merkillä.

Erityyppisissä tauluissa käytetään etuliitteitä tiedon hallinnan helpottamiseksi. Esimerkiksi kaikki dimensiotaulut on nimetty DIM etuliitteellä ja faktataulut FACT etuliitteellä. Puskuripalvelimella olevat lähde- ja kohdetaulut nimetään erilaisilla etuliitteillä. Lisäksi aggregoiduissa tauluissa täytyy ilmaista mille tasolle tieto on aggregoitu. Sen lisäksi, että nimeämiskäytännöt helpottavat taulujen löytämistä, saadaan niillä myös varmistettua, että käsitellään oikeaa taulua. Jos vahingossa suoritetaan julkaisutietokannassa puskuritietokannan komentoja, ei tapahdu mitään katastrofaalista, koska julkaisutietokannassa ei ole samannimisiä tauluja kuin puskuritietokannassa.

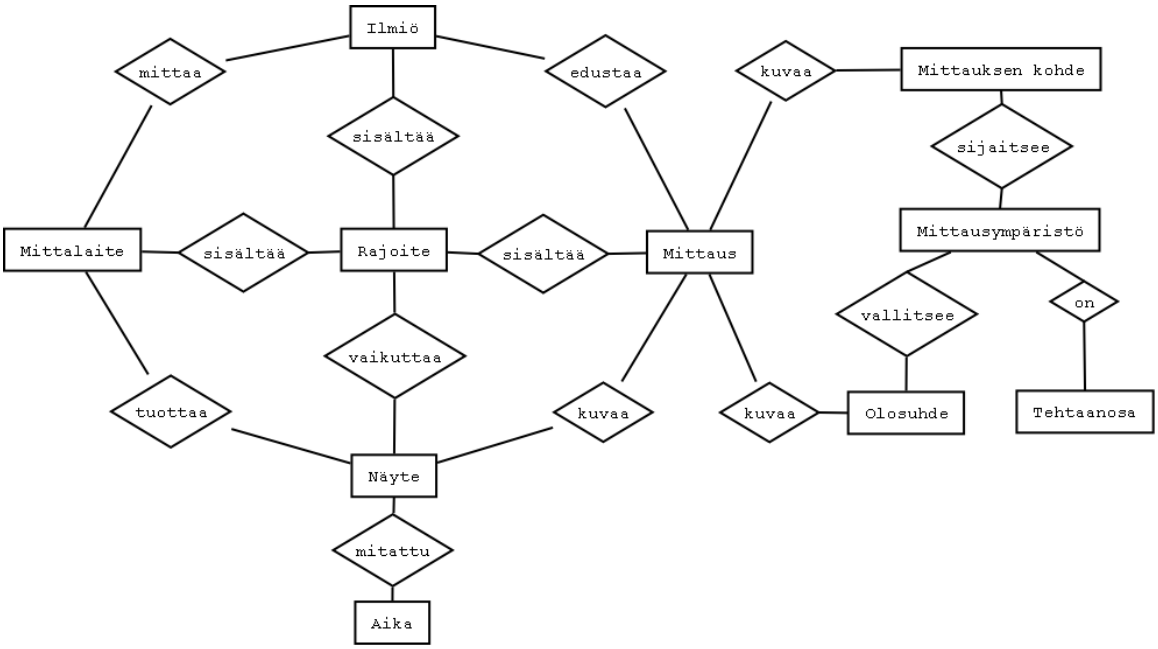
4.3 Mittausympäristön käsitteellinen analysointi

Tavallisesti relaatiotietokantoja suunnitellaan käyttäen käsitekaavioita. Käsitekaavioiden avulla suunniteltu ja normalisoitu rakenne takaavat sen, etteivät tiedot toistu tietokannassa. Tietojen toistumista pyritään välttämään mm. siksi, että tiedon hallinta olisi helpompaa. Jos sama tieto on monessa paikassa, täytyy aina varmistua, että tiedot pysyvät yhtenäisinä. Tällä säästetään myös tallennustilaa. Tietovarastoa ei kannata suunnitella pyrkien mahdollisimman pieneen redundanssiin. Pieni redundanssisella suunnittelulla tiedot menevät atomisiksi ja taulujen määrä on suuri. Tällaisesta rakenteesta tiedon hakeminen on vaikeaa ja hidasta, johtuen useista

kyselyissä tarvittavista liitos-operaatioista, jotka joudutaan tekemään suurelle tietomassalle. (Kimball et al. 1998, s. 140-144)

Käsitekaaviot ovat helppolukuisia ja niitä käytettiin tietovaraston suunnittelun alkuvaiheessa jäsentämään ja kartoittamaan siihen tosimaailman osaan liittyvät käsitteet ja käsitteiden väliset suhteet, jota tietokanta kuvaa. Jos käsitekaavioita käytetään tietovaraston suunnittelussa, pitää kaaviot denormalisoida dimensioihin moniulotteisen mallin suunnitteluvaiheessa. Kohdeympäristön käsitteet on määriteltävä tarkasti, ettei tietokannasta jää uupumaan oleellisia asioita ja jotta tietokannassa puhutaan asioista niiden oikeilla termeillä.

Mittaustietokannan keskeisin käsite on mittaus. Kuvan 16 kaaviossa on kuvattu yleisellä tasolla mittaamiseen liittyvät tärkeimmät käsitteet ja relaatiot. Kaavioon ei ole piirretty relaatioiden kardinaliteetteja eikä objektien attribuutteja. Relaatioiden lukusuunnat on pääteltävissä intuitiivisesti. Tämä kuva voidaan kääntää myös luonnolliselle kielelle.



Kuva 16. Mittaamiseen liittyvät käsitteet

Mittausympäristössä pyritään kuvaamaan tiettyjä luonnonilmiöitä näytteiden avulla. Nämä luonnonilmiöt yhdessä kuvaavat ympäristössä vallitsevia olosuhteita. Luonnonilmiöllä voi olla teoreettisia rajoitteita, kuten lämpötilalla negatiivinen raja on absoluuttinen nollapiste. Mittausympäristö ja mittauksen kohde voivat myös asettaa mitattavalle ilmiölle omat järjelliset rajoitteensa, jotka ovat teoreettisia rajoja

tiukemmat. Esimerkiksi veden massa voi tietyssä kattilassa olla tietyllä rajatulla välillä, koska kattilaan mahtuu vain tietty määrä vettä. Jos se on yli tai alle rajojen, kyseessä on varmasti mittausvirhe. Tietystä ympäristöstä mitattavaa, tietyn kohteen ilmiötä sanotaan mittaukseksi. Mittaus voi olla esimerkiksi sellun (kohde) lämpötila (ilmiö) sellutehtaan X keittimessä Y (ympäristö). Näyte kuvaa mittausta tietyllä ajanhetkellä. Mittauksella voi olla prosessiin liittyviä hälytysrajoja, joiden sisällä mittausarvon täytyy olla, jotta prosessi voi toimia normaalisti. Kaikki rajoitteet vaikuttavat lopulliseen näytteen arvoon.

Mittausympäristö on lähes aina jokin tehtaan osa. Tehtaan rakenne voidaan kuvata hierarkiana, jossa jokainen tehtaan osa sisältyy johonkin toiseen tehtaan osaan. Esimerkiksi tehtaaseen kuuluu tuotantolinjoja, joihin kuuluu osastoja joihin taas kuuluu joitain toisia pienempiä osakokonaisuuksia ja niin edelleen, kunnes lopulta päädytään jakamattomaan tehtaan komponenttiin. Tehdasta ei voi kuitenkaan jakaa yleispäteviin abstraktiotasoihin, koska hierarkia ei ole kaikkialla yhtä syvä. Joskus mittausympäristö voi myös olla tehtaan ulkopuolella sijaitseva laboratorio, jossa mitataan jotain tehdasympäristöstä kerättyä vaihetuotetta. Tällöin mittausympäristö ja tuotantoympäristö eivät ole samoja.

Mittaaminen suoritetaan tietyllä mittalaitteella, joka oletetaan passiiviseksi eikä sen mittausympäristöön aiheuttamia häiriöitä siksi huomioida. Tehdasympäristön automaattisilla mittalaitteilla on aina tietty mittaustarkkuus ja näytteenottotaajuus. Mittalaitteella on myös mittaustekniikan asettamat tekniset rajat, joiden sisällä sen antamat arvot voivat olla. Mittalaitteen raaka mittaus on skaalattu tiettyyn mittayksikköön, joka kuvaa mitattavaa suuretta. Mittalaitteet on kytketty automaatiojärjestelmään tiettyyn porttiin. Usein mittareita etsitään positionimellä. Joistain mittareista tiedetään myös valmistaja, malli ja tyyppi.

Mittauksen kohde on yleensä tuotantoprosessin vaihetuote, prosessia ohjaava toimilaite tai yleisesti osa ympäristöä. Tuotannossa käytettävät raaka-aineet voivat vaihdella, jolloin on tärkeää tietää, mitä raaka-aineita käytettiin vaihetuotteen muodostukseen. Ilman tätä tietoa esimerkiksi eri raaka-aineiden tuotantoa tai hyötysuhdetta olisi vaikea vertailla. Sellua voidaan esimerkiksi tehdä eri puulajeilla, joilla on erilaisia ominaisuuksia. Kohde voi yleisesti ottaen olla esimerkiksi puu tai tarkemmin havupuu ja vielä tarkemmin esimerkiksi kuusi. Toimilaite voi olla

esimerkiksi venttiili jonka aukioloastetta mitataan. Ympäristömittaus voi olla huoneilman lämpötila, kosteus tai valoisuus. Jotkut kohteet voivat myös muuttua ja niihin voi liittyä olomuoto, kuten kiinteä, neste tai höyry. Myös olomuoto voi muuttua ympäristön olosuhteiden muuttuessa.

Aika on mittaustietokannassa yksi tärkeimmistä käsitteistä. Ilman aikaa, näytteet muuttuvat arvottomiksi, koska niiden tuloksista ei voida tehdä päätelmiä eikä niitä myöskään voida vertailla keskenään. Jokaiseen mittaukseen täytyykin liittyä tieto siitä, milloin se on kerätty. Mittausympäristöä valvovat ja ohjaavat tehtaan työmiehet, jotka tekevät vuorotyötä. Yleisimmin käytössä on kolmivuoroinen järjestelmä jonka aamuvuoro alkaa klo 6:00 aamulla ja päättyy klo 14:00. Iltavuoro alkaa klo 14:00 ja päättyy klo 22:00, josta taas yövuoro jatkuu klo 6:00 asti. Vuorot on jaettu viidelle eri ryhmälle, joilla voi olla minä päivänä tahansa aamuvuoro, iltavuoro, yövuoro tai vapaapäivä. On myös ns. tehdasseisokkeja, jolloin sellua ei tuoteta. (Paperiliitto 2007)

4.4 Looginen suunnittelu

Moniulotteisella tietomallilla suunnitellun tietovaraston kulmakiviä ovat yleiset dimensiot, joita voidaan käyttää mahdollisimman monissa faktatauluissa. Ensin suunnitellaan käytettävissä olevat dimensiot, jonka jälkeen suunnitellaan faktataulut ja niiden liitokset dimensioihin. Käsitteellisessä analyysissä on jo selvitetty tärkeimmät käsitteet, joita moniulotteiseen malliin tullaan laittamaan. Tässä tapauksessa tärkeintä on, että dimensiot vastaavat kysymyksiin mitä, millä, missä ja milloin mitattiin. Koska suurin osa mittauksista kerätään automaattisesti, niin vastausta kysymykseen, kuka mittasi ei tarvita. On kuitenkin hyödyllistä tietää, mikä vuororyhmä oli näytteenkeräyshetkellä töissä.

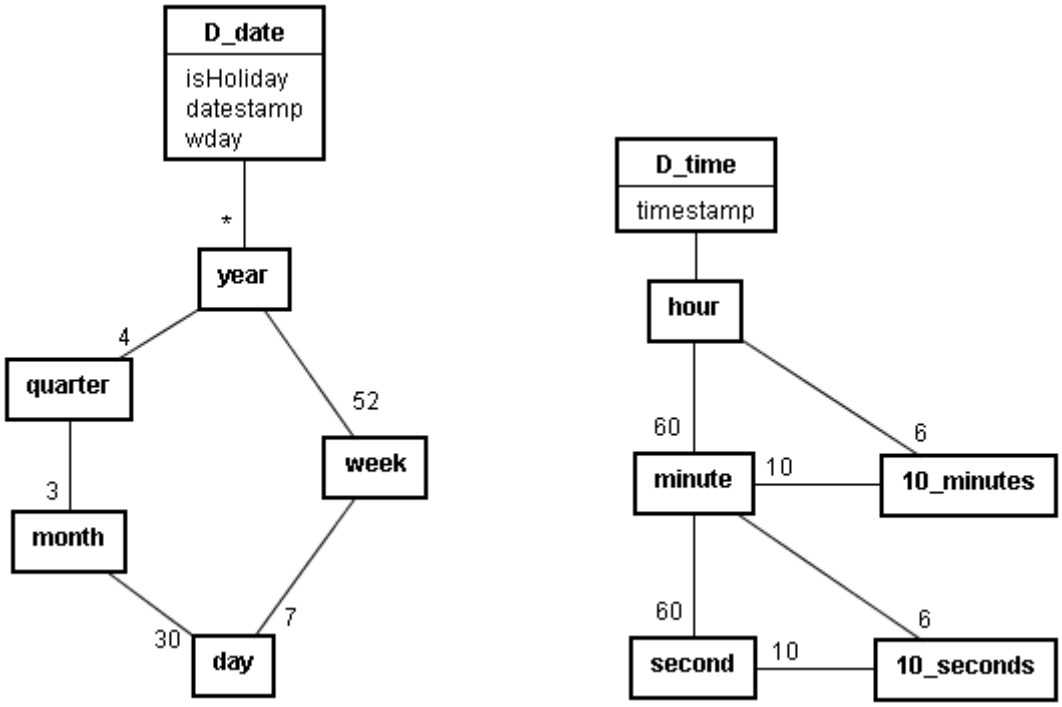
4.4.1 Aikadimensiot

Aika on jatkuva käsite, mutta mittarit keräävät näytteitä mittarille ominaisella näytteenottotaajuudella. Jos näytteenottotaajuus yhdellä mittarilla on 1 sekunti, kehittää se näytteitä 365*24*60*60 = 31,5 miljoonaa vuodessa. Jos aikadimensio muodostettaisiin yhdellä taululla, olisi siinä viiden vuoden jänteellä oltava noin 158 miljoonaa riviä. Dimensiossa tällainen rivimäärä ei ole hyväksyttävä. Rivien määrä saadaan pienemmäksi tekemällä erikseen dimensiot päivämäärää ja kellonaikaa varten. Päivämäärädimensioon tulee 365 riviä vuodessa ja sekuntitarkkuuden

kellonaikadimensioon tulee yhteensä 86400 riviä. Näin aikadimensiotaulut eivät kasva liian suuriksi ja mittaustaulussa voidaan silti osoittaa mihin tahansa ajan hetkeen. Aikadimensiosta on hyötyä tilatarpeen ja tehon kannalta, koska datetime-kenttä on yleensä kuvattu kahdeksalla tavulla ja dimensioviittauksilla tämä saadaan huomattavasti pienemmäksi ja siten säästetään faktataulun leveydessä.

Kellonaikadimensiossa tarvitaan tunti-, minuutti- ja sekuntisarakkeet ja muutama näiden väliin jäävää tasoa. Ylimääräisiä aggregointitasoja tarvitaan, jotta saadaan valittua parhaimmin dataa kuvaava taso. Tarkkoja mittauksia aggregoidaan, jotta nähdään kohinan alta todellinen trendi. Tieto halutaan kuitenkin säilyttää tietokannassa mahdollisimman hienojakoisella tasolla, jolloin menetetään vähemmän tietoa ja näin myös tiedon louhinta ja mittausarvojen välisten korrelaatioiden tutkiminen on mahdollista.

Päivämäärädimension hierarkiasta on olemassa kirjallisuudessa paljon esimerkkejä, joista useat muistuttavat hyvin läheisesti myös kuvassa 17 nähtävää hierarkiaa. Kuvan juurialkioon on liitetty kaikki ne attribuutit, jotka eivät osallistu hierarkiaan. Näistä nähdään esimerkiksi, mikä viikonpäivä on kyseessä ja onko vapaapäivä. Vaikka aikadimensiot ovatkin kahdessa eri taulussa, ovat ne periaatteessa samaa dimensiota. Siksi myös hierarkia alkaa päivämäärän hierarkiasta ja jatkuu kellonaikahierarkiaan.



Kuva 17. Päivämäärän ja kellonajan käsitehierarkiat

Vuoden viimeinen viikko voi osittain sisältyä molempiin vuosiin, mikä aiheuttaa hieman ongelmia hierarkiassa. Viimeinen viikko numeroidaan siihen vuoteen, jonka aikana se on alkanut ja jatkuu seuraavan vuoden puolella numerolla 0. Hakutyökalussa täytyy päättää, kuinka se esitetään käyttäjälle. Näytetäänkö haussa vain pyydettyyn vuoteen liittyvät päivät vai liitetäänkö siihen myös seuraavan vuoden nollaviikko, jolloin saadaan koko viikko. Myös nollaviikkoa pyydettäessä voidaan joko esittää tai olla esittämättä myös edellisen vuoden viimeinen viikko.

4.4.2 Työvuorodimensio

Mittauksiin haluttiin liittää myös työvuorotietoja. Vuorotietoja ei kannata liittää suoraan aikaulottuvuuteen, koska yövuoron kuuluminen kahteen eri päivään ja useat eri työaikamuodot aiheuttaisivat aikaulottuvuuden koon moninkertaistumisen. Vuoroja varten päätettiinkin tehdä kokonaan oma ulottuvuus. Tällöin mistä tahansa näytteestä nähdään suoraan, mikä viidestä ryhmästä on ollut töissä näytteen keräämisen aikana ja onko tuolloin ollut tehdasseisokki tai jokin muu erikoislaatuinen päivä. Faktatauluihin tarvitaan yksi sijaisavain lisää, mutta vuorotiedon liittäminen aika ulottuvuuksiin olisi yhtälailla kasvattanut faktataulun leveyttä, koska myös aika ulottuvuudet olisivat kasvaneet. Erillinen vuoroulottuvuus antaa enemmän joustovaraa. Vuorotaulusta tuli niin matala, että sijaisavain saadaan kuvattua yhdellä tavulla.

4.4.3 Mittalaitedimensio

Tehdastyöläinen yleensä tietää mittalaitteen positionimen ja haluaa hakea mittauksia suoraan sen perusteella. Mittalaitedimensioon täytyy tämän vuoksi sisällyttää positionimi. Useimmat käyttäjät eivät välttämättä tiedä haluamansa mittarin positionimeä ja ilman lisäinformaatiota he joutuisivat selvittämään asian ensin esimerkiksi prosessikaaviosta. Tämän vuoksi mittalaitedimensioon täytyy yhdistää myös tietoja, jotka helpottavat halutun laitteen löytämistä. Mittalaitteen löytämistä helpottavat mm. se, mitä ilmiötä mittari mittaa ja mikä on sen mittayksikkö.

Laitteen toimintaan vaikuttavat sen asetukset. Varsinkin, kun käytetään kameraa mittalaitteena, on asetuksilla suuri merkitys mittauksiin. Asetukset vaihtelevat kuitenkin laitteittain niin suuresti, ettei niistä voida muodostaa yhtenäistä dimensiota eikä myöskään liittää mittalaitedimensioon. Jos jostain syystä tietyntyyppisistä

mittauksista halutaan nähdä laiteasetukset, täytyy kyseisiä laitteita varten tehdä sellainen dimensio, joka tukee juuri kyseisten laitteiden asetuksia. Tällaisia tapauksia voisivat olla esimerkiksi kaikki kameramittaukset, joiden yleisimmät asetukset olisi mahdollista koota dimensioon.

4.4.4 Tehtaanosadimensio

Näytteen kannalta merkityksellistä on se, minkälaisessa ympäristössä näyte on otettu ja missä tämä ympäristö sijaitsee. Sijainti ja ympäristö käsitteinä menevät hieman päällekkäin, mutta yleisesti ottaen ne ovat aina joitain tehtaan osia. Useimmin mittauksia etsitään nimenomaan niiden sijainnin perusteella. Tässä dimensiossa selvästi hyödyttäisiin rivien välisen hierarkian kuvaavasta siltataulusta jolla selitettäisiin tehtaan monimutkainen rakenne. Dimensio pyritään kuitenkin alkuvaiheessa kuvaamaan suurpiirteisemmin pelkän käsitehierarkian avulla, johon laitetaan tehtaan yleiset osakokonaisuudet. Näin malli saadaan helpommin ja nopeammin toimimaan. Hierarkian kuvaava siltataulu voidaan tehdä ja ottaa käyttöön myöhemmin eikä sen käyttöönotto vaikuta dimensio- tai faktataulujen rakenteisiin tai sisältöön.

4.4.5 Mittauksenkohdedimensio

Mittauksen kohdetta varten tehtiin myös oma dimensio. Kohdedimensioon ei tullut kovinkaan montaa kenttää ja siihen liittyy erittäin yksinkertainen hierarkia. Kohdedimensiossa on sarakkeet kohteen tyyppiä (toimilaite, vaihetuote, ympäristö), yleistä kategoriaa (venttiili, puu, sellu jne.), lajia (lehtipuu, havupuu, ruskea massa jne.) ja yksilöivää nimeä esimerkiksi koivu varten. Nämä muodostavat dimensiossa hierarkian. Kaikki kohteet eivät kuitenkaan täysin asetu tällaiseen rakenteeseen. Näissä tapauksissa yli jääneisiin kenttiin tulee teksti ”tuntematon”. Jos kentät eivät jostain syystä riitä, jää kuvaus yleisemmälle tasolle. Näiden lisäksi dimensiossa kerrotaan myös kohteen olomuoto. Kaikilla kohteilla kuten toimilaitteella ei varmastikaan ole olomuotoa tai sen ilmaisemisesta ei saada mitään hyötyä. Näille voidaankin laittaa olomuoto sarakkeeseen esimerkiksi teksti ”ei ole”.

4.4.6 Tiedostodimensio

Jotkut mittaukset on jalostettu sellaisesta datasta, joka on saatavilla tiedostossa. Esimerkiksi kuva-analyyseissä analysoitu kuva on saatavilla tiedostossa. Näytteistä

halutaan päästä takaisin käsiksi niihin tiedostoihin, joihin ne perustuvat. Tiedostoja varten kehitettiin erillinen dimensio. Tiedostoviittaukset koostuvat tiedostotyypistä, suhteellisesta polusta ja tiedoston nimestä. Suhteellinen polku mahdollistaa koko hakemistorakenteen siirtämisen toiseen paikkaan ilman dimension muokkaamista. Suhteellisen polun juuripolkua voidaan säilyttää esimerkiksi metatietona tai suoraan ohjelmistoissa globaalina vakiona.

4.4.7 Mittausdimensio

Mittausdimensioon liitetään kaikki suoraan mittaukseen liittyvät dimensiot yhdeksi dimensioksi eli denormalisoidaan rakennetta vielä entisestään. Mittauksen kohde, mittauksen sijainti ja mittauksessa käytetty mittalaite voidaan yhdistää yhteen dimensioon, joka helpottaa tietokannan ylläpitoa huomattavasti. Tämä on mahdollista, koska aika, mittalaite ja mittaus riittävät yksilöimään näytteen. Tämä ilmenee epäsuorasti myös kuvan 16 käsitekaaviosta. Yhdistäminen kaventaa myös faktatauluja ja pienentää faktataulujen indeksien kokoa. Dimensioon tulee suunnilleen yhtä paljon rivejä, kuin mittalaitedimensioon olisi tullut.

Tämän dimension päätarkoitus on helpottaa ja nopeuttaa tietovaraston käyttöä sen alkuvaiheessa. Myöhemmin, kun tietovaraston kaikki eri osa-alueet saadaan toimimaan, se on mahdollista hajottaa pienemmiksi dimensioiksi. Tähän ei välttämättä kuitenkaan ole tarvetta. Varastoitavia mittareita ei ole luokiteltu ja alkuvaiheessa olisi epärealistinen tavoite yrittää hallita montaa dimensiota, joilla on keskinäisiä riippuvuuksia. Kun ajan lisäksi faktataulussa tarvitaan vain yksi viittaus, jolla näyte saadaan yksilöityä, on tiedon luokittelu helpompaa. Varsinkin tiedon tuominen tietokantaan helpottuu, koska selvitettäviä sijaisavaimia on vähemmän. Koska näytteitä on paljon, on yksinkertaisuudesta myös etua esimerkiksi tilatarpeen ja hakunopeuden suhteen.

Näiden dimensioiden yhdistämisestä voi koitua myös hieman haittaa. Yhdistetty dimensio voi alkaa kasvamaan nopeasti, jos tehdasympäristöön kohdistuu paljon muutoksia. Muutokset eivät kuitenkin ole kovinkaan yleisiä. Suurempia ongelmia syntyy, jos jossain faktataulussa ei tarvitakaan mittausdimensiota, mutta sijaintidimensiota tarvitaan. Silloin sijaintia varten tarvitaan oma dimensio. Tämmöinen tilanne voi tulla esimerkiksi viivefaktassa. Viivehän ei oikeastaan kohdistu kahden mittalaitteen välille vaan paremminkin kahden eri prosessin vaiheen

välille (ympäristöt). Jos tietyille näytteille halutaan hakea viiveajat sijainnin perusteella, ei viivefaktataulussa olevaa sijaintidimension ja mittausfaktataulussa olevaa mittausdimension avainta voida vertailla suoraan keskenään. Silloin joudutaan vertaamaan dimensioiden ominaisuuksien arvoja. Ongelma voidaan kuitenkin kiertää laittamalla mittausdimensioon myös sijaintidimension viiteavain. Periaatteessa siis tehdään lumihiutale. Myös lumihiutaleesta löytyvät tieto on suoraan mittausdimensiossa, jolloin ei menetetä hakutehokkuutta ja yksinkertaisuutta. Avaimien ja tietojen synkronointi voi kuitenkin olla työlästä.

4.4.8 Mittausfaktataulun suunnittelu

Tietokuutioissa käytettävät ulottuvuudet on tunnistettu ja määritelty. Tässä kappaleessa suunnitellaan, kuinka näytteet liitetään dimensioihin faktataulujen avulla.

Faktojen valinnassa täytyy miettiä, mitä organisaatiossa mitataan tai halutaan mitata. Esimerkiksi vähittäistavarakaupassa voidaan mitata asiakkaiden ostokäyttäytymistä, rahallista tuottoa ja markkinoinnin vaikutusta. Nämä ovat varsin yleisiä käsitteitä. Tehdasympäristössä mittaukset ovat yksityiskohtaisempia ja niitä on tuhansia. Niistä on vaikea tunnistaa sellaisia yleisiä luokkia, joissa mittaukset olisivat keskenään vertailukelpoisia. Tällaisia yleispäteviä mittauksia voi olla esimerkiksi tuotantomäärät tai tuotantokulut. Mittauksia voidaan myös luokitella esimerkiksi tuotannollisiin ja laadullisiin mittauksiin. Luokkia voidaan jakaa yhä pienempiin osiin, jolloin lopulta päädytään yksittäisiin mittareihin. Tämän jaon voi tehdä monella tavalla ja tärkeintä on, että yhteen faktatauluun saataisiin yhdenmukaisia näytteitä.

Yksinkertaisinta olisi tallentaa kaikki näytteet yhteen faktatauluun allekkain, jolloin taulujen määrä pysyisi pienenä ja tietokannan rakenne selkeänä. Lisäksi näin tietoa voitaisiin aina hakea samasta faktataulusta, joka olisi helppoa. Faktan tallennukseen jouduttaisiin kuitenkin käyttämään liukuluku- tai pahimmillaan tekstikenttää. Taulu kasvaisi niin korkeaksi, ettei sitä pystyisi hallitsemaan enää edes horisontaalisella osioinnilla vaan se jouduttaisiin lopulta jakamaan erillisiksi, mutta samanlaisiksi tauluiksi. Ei voida myöskään olettaa, että jokainen mittaus voitaisiin rajata käyttämään samoja dimensioita ja yhtä mittausarvoa. Jotkin mittaukset halutaan kuvata käyttäen useampaa kuin yhtä mittausarvoa tai jossain tapauksessa mittauksen arvoa ei tarvita ollenkaan, koska se ilmenee dimensioista. Esimerkiksi ”on”/”off” mittauksissa riittää, että esitetään faktataulussa kaikki sellaiset tilanteet, jolloin jonkin

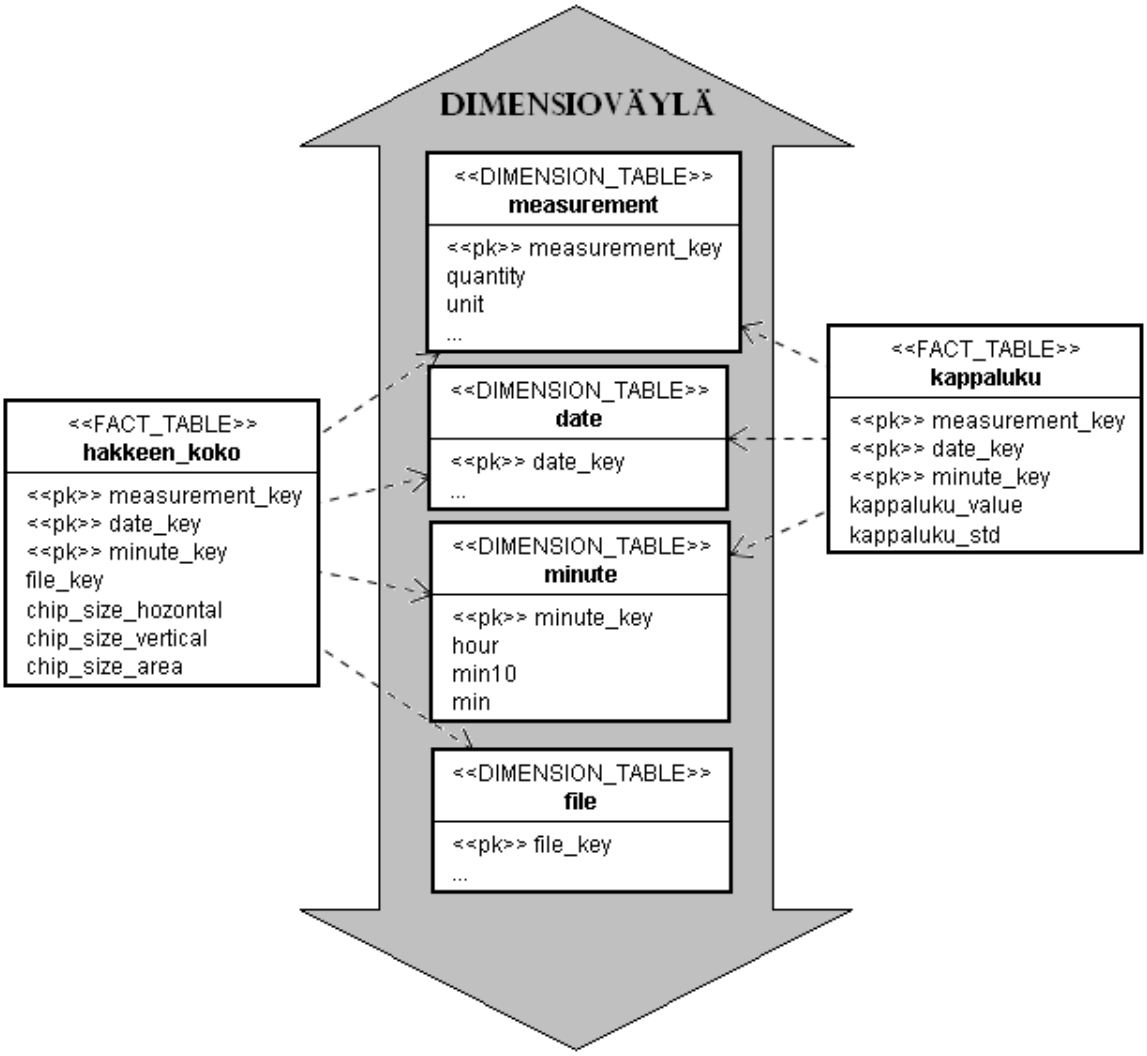
laitteen tila on ollut ”on”. Tällöin fakta olisi jokaisella rivillä ”on” ja siksi sitä varten on turha tehdä saraketta. Tämän takia mittauksia ei kannata eikä voi tallentaa yhteen tauluun.

Erilaisilla tiedoilla pitää olla omanlaisensa faktataulu ja periaatteessa jopa jokaisella mittarilla voisi olla oma taulunsa. Näin yhdessä taulussa olisi varmasti aina samanlaista tietoa. Taulujen määrä kasvaisi kuitenkin erittäin suureksi, koska tehdasympäristössä on tuhansia mittareita. Tämän takia myös taulujen hallinta muuttuisi hankalammaksi ja myös metatietoa tarvittaisiin enemmän.

Samantyyppiset näytteet voidaan tallentaa samantyyppisellä rakenteella. Yksinkertaisin luokittelu saataisiin suureen perusteella, koska sellaiset mittaukset, joilla on sama suure, tarvitsevat tallennuksessa samanlaista tietoa. Vertailevat kyselyt yksinkertaistuvat, koska samantyyppiset näytteet löytyvät samasta taulusta. Tällaisella jaolla tietokantaan ei tule suurta määrää tauluja ja rakenne on looginen. Lisäksi näin mittauksiin voidaan aina liittää kaikki juuri niille ominaiset tiedot, jolloin tallennustapa on aina tehokas. Myös hakutyökalut voivat nopeasti päätellä, mistä tieto löytyy, jolloin metatietoakaan ei tarvita niin paljon. Erilaisia tehdasmittauksia tallentavia faktatauluja voisivat olla esimerkiksi paine, tuotantomäärä tai kappaluku. Tällöin dimensioiden avulla esimerkiksi painetaulusta voidaan pyytää tuotantolinjan X säiliön Y paine aikavälillä Z.

Kuvassa 18 on esitetty esimerkkinä kaksi erilaista mittausfaktataulua. Molemmat faktataulut hyödyntävät yhteisiä dimensioita, joiden kautta tiedot on myös mahdollista yhdistää toisiinsa. Tässä kellonajan kuvaamiseen on riittänyt minuuttitaso ja sitä varten on käytössä minuuttitason osajoukkodimensio. Hakkeen koko on määritelty kuva-analyysin avulla. Siksi hakkeen koko -faktataulussa on viittaus myös tiedostodimensioon. Hakepalan koko on kuvattu vertikaalisella ja horisontaalisella komponentilla sekä hakepalan pinta-alalla. Kappaluku-taulu on summataulu ja siinä kappaluku on keskiarvotettu tarkemmalta tasolta ja samalla on laskettu erikseen jokaisen minuutin aikana tapahtunut arvojen hajonta. Tauluissa on eroja niin dimensioiden kuin faktojenkin osalta ja selvästi erityyppisillä mittauksilla on erilaisia tietotarpeita. Siksi taulujako mittaustyypeittäin on perusteltua. Jos vastaavat asiat kuvattaisiin yhdenlaisella taululla, jouduttaisiin jokaista eri faktaa varten tekemään mittausdimensioviittaus. Ei ole mielekästä tehdä esimerkiksi kappaluvun_hajontaa

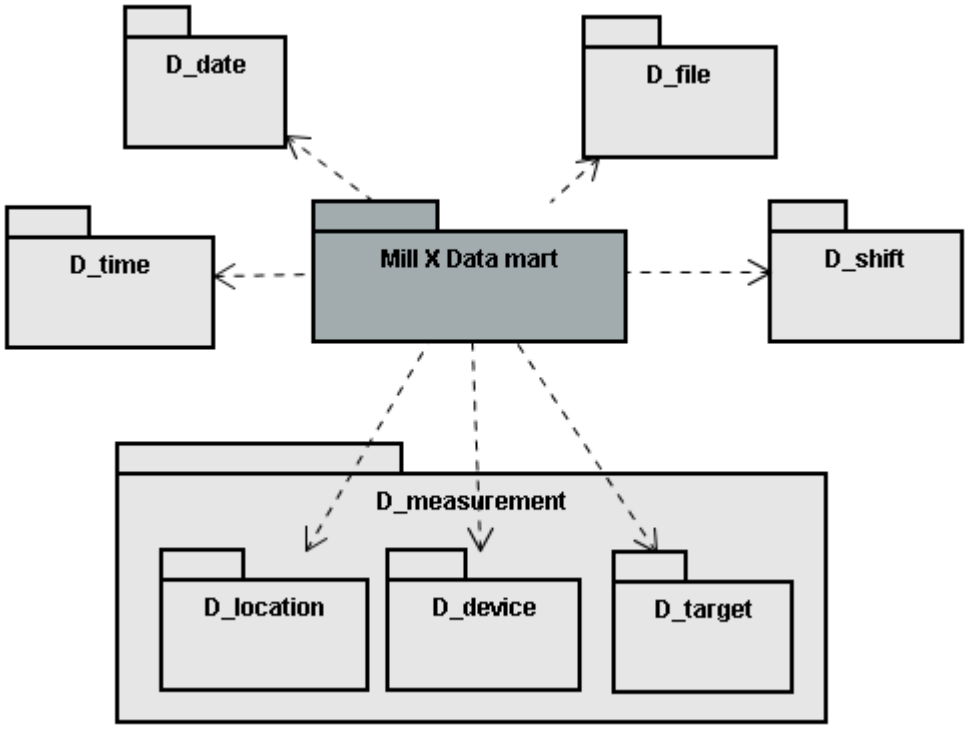
varten omaa mittausta. Faktataulussa täytyisi olla aina viittaukset kaikkiin dimensioihin. Tämä taas kasvattaisi faktataulujen kokoa.



Kuva 18. Mittausfaktojen erilaiset tietotarpeet

Väyläarkkitehtuuriajattelussa tavallisesti eri paikallisvarastoissa on erilaisia faktatauluja joihin voi liittyä mitä tahansa dimensioita. Nyt kaikissa paikallisvarastoissa on samantyyppistä tietoa ja kaikissa paikallisvarastoissa tarvitaankin kaikkia dimensioita. Kaikissa mittausfaktoissa on aina oltava ainakin aikadimensiot ja mittausdimensio, jotka yksilöivät näytteen. Tämä tietenkin tekee hauista ja vertailuista helppoja, koska kaikista mittauksista löydetään näiden yhteisten dimensioiden avulla yhteisiä tekijöitä, joiden avulla mittauksia voidaan ryhmitellä. Kuvassa 19 on esitetty mittausfaktojen liittyminen yleisiin dimensioihin. Dimensiot on kuvattu paketeilla, koska joissain dimensioissa voi olla useita tauluja. Keskellä oleva paketti kuvaa jonkin tuntemattoman tehtaan paikallisvarastoa, joka sisältää

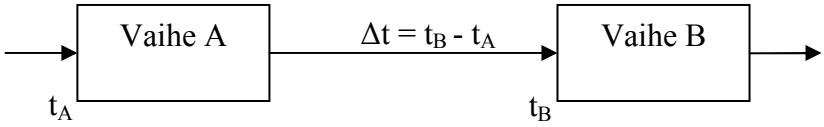
kaikki kyseisen tehtaan faktataulut. Alimpana sijainti-, mittalaite- ja mittauksen kohdedimensiot, jotka on yhdistetty yhdeksi mittausdimensioksi.



Kuva 19. Dimensioiden liitokset paikallisvarastoihin

4.4.9 Viivefaktataulu

Sellunvalmistusta voidaan pitää kokonaisuudessaan melko kaoottisena prosessina. Sen jokainen vaihe on riippuvainen sitä edeltävästä vaiheesta ja eri vaiheissa on useita eri muuttujia jotka vaikuttavat yksittäisen vaiheen tuloksiin. Jotta eri vaiheissa havaittavien ilmiöiden keskinäisiä riippuvuuksia voitaisiin vertailla, täytyy tietää, kuinka paljon vaihetuotteella kuluu aikaa vaiheesta A vaiheeseen B, kuten kuvassa 20.



Kuva 20. Viiveaika tuotantoprosessin vaiheiden välillä

Viivefakta mittaa tuotteen etenemiseen kuluvaa aikaa kahden eri tehtaanosan välillä. Faktataulussa tarvitaan dimensiot sekä alku- että lopputilanteille. Mittausarvoa ei välttämättä tarvita, koska se voidaan laskea yksinkertaisesti B:n ja A:n aikojen erotuksena. Voi kuitenkin olla käyttäjän kannalta mielekästä liittää viiveaika faktaksi. Jos käyttäjä haluaa esimerkiksi suodattaa haustaan kaikki negatiiviset viiveajat

(virheelliset viiveet) pois, on se helpompaa silloin, kun viiveaika on jo valmiiksi laskettu.

Kun kaikki viiveajat ovat yhdessä taulussa, on eri viiveiden hakeminen helppoa, koska kaikki rajaukset voidaan tehdä dimensioiden avulla. Viiveitä voidaan hakea vapaasti alku- ja lopputilanteiden perusteella. Myös viivefaktassa käytetään samoja dimensioita kuin mittaustauluissa. Siksi mittaustietoihin voidaan yhdistää viivetietoja helposti. Esimerkiksi kuvan 18 hakepalan koon vaikutusta voitaisiin vertailla kappalukuun. Tällöin tiedot voidaan yhdistää toisiinsa käyttämällä suoraan viivefaktan dimensioviittauksia.

4.4.10 Tiedon summautuvuus ja summataulut

Kehitetyn mittaustietovaraston käytössä olisi mahdollista hyödyntää kaikkia kappaleessa 3.4.2 mainittuja E. F. Coddin ja kumppanien määrittelemiä OLAP-työkalujen ominaisuuksia. Tässä tapauksessa suurin hyöty saadaan mahdollisuudesta tiedon monenlaiseen ryhmittelyyn, koska tietoa on mielettömän paljon ja ryhmittelyillä sen löytäminen on helpompaa. Myös tietoon porautuminen ja sen summaaminen on mahdollista. Porautumiseen ja summaamiseen ei kuitenkaan voida käyttää mitään yleistä aggregaattoria jolla tiedot yhdistettäisiin aina samalla tavalla. Tämä johtuu siitä, että tieto on osittain summautuvaa koska, tietojen summaustapa on aina riippuvainen siitä, kuinka tietoa käytetään tutkimuksessa. Ainoastaan aika on sellainen dimensio, jonka suhteen voidaan valita yleinen aggregaattori erityyppisille mittauksille. Useimmin se on keskiarvo, mutta joissain tapauksissa se voi myös olla summa tai lukumäärä.

Päivämäärä ja kellonaika eli aikadimensiot ovat varmasti potentiaalisimmat dimensiot tiedon summaamista varten, koska kaikki mittaukset joko summautuvat tai keskiarvottuvat niiden suhteen. Aikadimensioiden suhteen faktataulut ovat myös erittäin tiheitä ja näin ollen summaamisella saadaan suurin mahdollinen hyöty. Tämän osoittaa myös kuvan 17 aikadimensioiden hierarkia, jossa eri tasojen välillä on suuria kertoimia. Tämän takia summataulut ovat merkittävästi perustauluja pienempiä ja kyselyissä ei tarvita niin paljon laskentaa ja myös kyseltäviä rivejä on vähemmän. Aikadimensioiden lisäksi summatauluja ei kannata tehdä muiden dimensioiden suhteen, koska yleisesti ottaen ei voida tietää, miten käyttäjä haluaisi yhdistellä tietoja niiden suhteen. Summataulut vaativat myös jonkin verran ylläpitoa. Kun tietoa

lisätään perustauluun, täytyy muistaa päivittää myös summataulu näiden näytteiden osalta. Tämä täytyy ottaa huomioon silloin, kun tietoa tuodaan lähdetietokannasta tietovarastoon. Jos summaukset tehdään aina pelkästään aikadimensioiden suhteen, on summataulujen ylläpito helpompaa. Ajan suhteen voidaan myös löytää kiinteät summatasot, mikä vielä entisestään helpottaa ylläpitoa.

Kaikki mittaukset säilytetään tietovarastossa ainakin tuntitasolla. Sen lisäksi tehdään päivätason summataulut, joihin summataan tuntitaulujen tiedot päivätasolle. Päivätason summatauluissa tietoa on 24 kertaa vähemmän kuin tuntitason summatauluissa ja siinä päästään myös eroon kellonaikadimension viittauksesta. Päivätason summataulu on niin karkealla tasolla, että hakunopeuden parantamiseksi sitä karkeammalle tasolle ei varmasti tarvitse summatauluja tehdä. Tuntitasoa tarkemmalle tasolle tietoa tallennetaan ainoastaan tietyistä mittareista. Koska suurin osa näytteistä on tuntitasolla, tehtiin sitä varten myös oma dimensio aikadimension osajoukkona. Tämä on 60*60=3600 kertaa pienempi (yhteensä 24 riviä/tuntia) kuin täydellinen kellonaikadimensio ja siksi myös erittäin nopea. Näin ollen faktataulu ja mahdollisesti myös sen indeksi kapenevat, koska kellonajan esittämiseen riittää yksi tavu kolmen tavun sijaan. Tuntidimensioon ei jää paljonkaan tietoa ja siksi voisi olla perusteltua degeneroida se eli ottaa faktataulusta viittaus pois ja laittaa tunti suoraan numeerisena arvona faktatauluun. Toisaalta tulevaisuudessa tuntidimensioon voi tulla lisää tietoa ja se voi siksi osoittautua hyödylliseksi. Tuntidimensio kannattaakin tehdä niin, että sen avain kuvaa myös tuntia itseään. Näin faktataulusta haettaessa ei välttämättä tarvitse käyttää dimensiotaulua tunnin rajaamiseen, mutta sekin on mahdollista.

4.4.11 Metatieto

Metatiedon tallentamisessa päädyttiin käyttämään tietovaraston kanssa samaa tiedonhallintajärjestelmää, jonne luotiin oma tietokanta metatietoa varten. Metatiedon sijaitsemiseen samassa tiedonhallintajärjestelmässä todellisten taulujen kanssa on hyötyä siksi, että myös tiedonhallintajärjestelmään tallennetut rutiinit (proseduurit ja funktiot) pääsevät siihen helposti käsiksi. Rutiinit ovat MySQL-komentotiedostoja joita on mahdollista tallentaa ja suorittaa MySQL versiosta 5.1 lähtien.

Kaikkiin faktatauluihin, joissa on tehdasmittauksia sisältyy viittaukset mittaus- ja aikadimensioihin. Siksi voidaan ajatella, että kaikki tehdasmittaukset sijaitsevat

näiden rajaamassa tehdasmittauskuutiossa. Toisaalta erityyppiset mittaukset sijaitsevat erilaisissa faktatauluissa ja siksi niitä voitaisiin pitää erillisinä kuutioina tai em. kuution osina. Yhtenäinen kuutio on tässä tapauksessa vaikea määritellä ja siksi ei ole mielekästä puhua kuutioista vaan enemmänkin dimensioista, jotka yhdistävät kaikki tiedot toisiinsa. Oikeat faktataulut on mahdollista löytää tietovarastosta, vaikka kuutiomäärittelyjä ei olisikaan. Riittää, että tiedetään, mitä tietoa on missäkin faktataulussa. Tämän vuoksi metatiedolla ei määritelty tietokuutioita. Metatieto pyrittiin muutenkin pitämään minimaalisena, jotta sen hallinta olisi helppoa, koska resursseja monimutkaisen metatiedon ylläpitämiseen ei ole. Lähteessä Kimball et al. 1998 on listattu paljon asioita, joita kannattaa tallentaa metatietona. Kyseisiin listoihin verrattuna metatietoa tuli hyvin vähän. Liitteessä 2 on lueteltu kaikki tärkeimmät tietovarastossa huomioidut julkiseen osaan liittyvät metatiedot ja liitteessä 3 taustaprosesseihin liittyvät metatiedot.

4.5 Fyysinen suunnittelu

Tietokannan tehokkuuteen voidaan vaikuttaa merkittävästi vielä loogisen suunnittelun jälkeenkin tallentamalla tieto tehokkaasti ja indeksoimalla se siten, että tieto on löydettävissä nopeasti. Tässä kappaleessa on kerrottu, kuinka eri-ikäiset tiedot tallennetaan ja indeksoidaan.

4.5.1 Näytteiden tallennusmoottorit

Tietovaraston data voidaan tallentaa eri tavoilla riippuen siitä, kuinka vanhaa tieto on ja kuinka aktiivisesti sitä käytetään. Vähemmällä käytöllä olevat tiedot voidaan tallentaa tehokkaammalla tavalla ja täysin tarpeettomat tiedot voidaan poistaa. Kaikkiin taulujen sarakkeisiin valitaan aina pienimmät tilanteeseen soveltuvat tietotyypit, koska varsinkin faktataulut ovat niin korkeita, että pieni rivikoon muutos vaikuttaa paljon nopeuteen ja tilatarpeeseen.

Tietovarastossa viite-eheydet tarkistetaan siinä vaiheessa, kun tietoa tuodaan lähdetietokannoista tietovarastoon. Tämän jälkeen tietoa ei enää muuteta, eikä viite-eheyden tarkastuksia siksi tarvita tallennetuissa mittauksissa. Tiedon saantinopeus on tärkein kriteeri. Faktataulujen tallennuksessa käytetään MyISAM-tallennusmoottoria, koska se on nopea eikä se silti vie yhtä paljon tilaa kuin InnoDB-taulu. MyISAM-taulun voi myös pakata myisampack-komennolla, jolloin se vie vielä huomattavasti

normaalia vähemmän levytilaa. Pakattua MyISAM-taulua voidaan ainoastaan lukea, mutta siinä on myös indeksit käytössä, toisin kuin ARCHIVE-tallennusmoottorissa.

Yksi merkittävimmistä hitauden aiheuttajista on taulun lukeminen levyltä ja teoriassa pakattu tieto voisi olla jopa pakkaamatonta nopeammin haettavissa, riippuen pakkausalgoritmista ja levyjärjestelmästä. Tässä tapauksessa MyISAM-faktataulun pakkaus pienensi taulun vaatimaa tilaa noin 33-40%, mutta samalla hidasti kyselyjä noin 18%. Pakkauksen purkaminen vie siis niin paljon prosessoritehoa, että pelkästään nopeuden kannalta ajateltuna taulun pakkaaminen ei ollut perusteltua. Pakkauksessa ei voi säätää pakkaustehokkuuden tasoa nopeamman toiminnan toivossa. Pakattuun tauluun ei voi kirjoittaa eikä tiedon pakkaus pienentänyt kovinkaan paljon hakutehokkuutta. Pakattuja tauluja kannattaakin käyttää sellaisten faktataulujen tallentamiseen, jotka ovat melko aktiivisessa käytössä, mutta joihin ei enää lisätä uutta tietoa. Esimerkiksi vuoden vaihteessa voidaan pakata tietyt edellisen vuoden tiedot ja alkaa tallentaa tietoa taas uuteen faktatauluun.

Taulujen arkistoinnissa kannattaa käyttää ARCHIVE-tallennusmoottoria, jossa tieto on pakattu tehokkaasti. ARCHIVE-moottorilla tallennettu faktataulu vei jopa 83% vähemmän levytilaa verrattuna perinteiseen MyISAM-tauluun, jossa on tarvittavat indeksit. Tästä muodosta taulu on tarpeen vaatiessa palautettavissa takaisin saataville muutamalla SQL-komennolla. Noin 30 miljoonan näytteen taulu tarvitsi 150 megatavua tallennustilaa. Käytännössä suuri osa näytteistä voidaan tallentaa pysyvästi, kun ne aktiivisen käytön jälkeen pakataan ARCHIVE-tauluihin.

Faktatauluja kannattaa osioida, jotta tietoa olisi helpompi hallita ja nopeampi hakea. Faktataulut ovat yleensä erittäin suuria ja vaikka ne ovatkin indeksoituja, saadaan osioinnilla nopeuden kannalta hyötyä. Tämä johtuu siitä, että myös indeksit osioidaan jolloin niistä tulee matalampia ja nopeampia. Varsinkin ARCHIVE-tauluihin tallennetut mittaustiedot kannattaa osioida, koska niitä ei voida indeksoida. Faktatauluissa osiointi kannattaa yleensä tehdä dimensioiden perusteella (sijaisavaimet), koska niiden perusteella data saadaan jakautumaan tasaisesti ja ennalta arvattavasti. Esimerkiksi aikadimension perusteella tieto jakautuu varmasti tasaisiin osiin, koska tieto lisääntyy yleensä tietyn vakiomäärän aikayksikköä kohti. Jos kyseessä on tuntidata, niin käytössä voisi olla esimerkiksi kuukausittaiset osiot. Faktataulujen osioinnissa täytyy kuitenkin myös miettiä, kuinka tauluja käytetään. Jos

faktataulusta haetaan usein tietoa mittausarvojen perusteella, voi osiointi myös mittausarvon mukaan olla perusteltua. Mittausarvoa ei yleensä ole indeksoitu ja siksi sen perusteella osiointi voi myös nopeuttaa sellaista hakua merkittävästi, jossa tietoa rajataan mittausarvon perusteella. Useimmissa tapauksissa tietoa kuitenkin haetaan dimensioiden perusteella ja niiden mukaan osiointi on aina varmin vaihtoehto.

Dimensiotaulut vievät tallennustilaa vain joitain kymmeniä megatavuja. Dimensiotauluja voidaan tämän takia tallentaa MEMORY-moottorilla, jolloin ne ovat jatkuvasti keskusmuistissa saatavilla. Tämä on hyödyllistä, koska dimensiotauluja luetaan paljon ja ne vaikuttavat paljon myös hakutehokkuuteen. Taulut täytyy kuitenkin säilyttää myös kiintolevyllä. Kaikki muutokset tehdään kiintolevyllä sijaitsevaan tauluun jonka jälkeen muutokset päivitetään keskusmuistiin. Muutosten päivitys voidaan yksinkertaisimmin tehdä siten, että muistissa oleva taulu tyhjennetään ja kiintolevyllä olevan taulun tiedot luetaan sinne kokonaan uudelleen. Dimensioihin tulee muutoksia harvoin eikä taulujen pitäminen muistissa aiheuta kovinkaan paljon ylläpidollisia toimenpiteitä. MEMORY-taulut tukevat B-puuindeksien lisäksi myös HASH-indeksejä, joilla voidaan hakea nopeasti yksittäisiä arvoja.

Metatiedon tallentamiseen valittiin InnoDB-tallennusmoottori, koska sen avulla voidaan määritellä taulujen välille viiteavaimia. Metatauluilla on paljon riippuvuussuhteita ja siksi täytyy varmistaa myös niiden viite-eheydet. Metatiedon ei tavallisesti pitäisi muuttua paljon, mutta muutostilanteissa on tärkeää, että tieto pysyy ehjänä. InnoDB-taulujen käyttöä tukee sekin, että metatietoa saattaa yhtäaikaisesti käyttää monet eri prosessit. Rivikohtaiset lukitukset mahdollistavat sujuvan käytön näissäkin tilanteissa.

4.5.2 Indeksoinnit

Dimensioiden indeksoinnissa ei tarvitse huolehtia tallennuskapasiteetista, koska dimensiot ovat yleensä melko matalia eivätkä ne kasva kovinkaan paljon. Niissä kannattaakin indeksoida mahdollisimman monia kenttiä. Lisäksi kannattaa suosia usean sarakkeen yhdistettyjä indeksejä, koska niillä saadaan indeksin kardinaliteetti suuremmaksi, joka on hyödyllistä käytettäessä B-puuindeksointia. Usean sarakkeen indekseissä kannattaa kuitenkin muistaa, että indeksiä ei käytetä ollenkaan, jos indeksin ensimmäinen termi ei esiinny hakuehdoissa. Esimerkiksi indeksiä, joka

koostuu vuodesta, kuukaudesta ja päivästä (tässä järjestyksessä), ei käytetä, jos haetaan pelkästään kuukauden ja/tai päivän perusteella. Koska tallennus tehdään MEMORY-tallennusmoottorilla, sarakkeita voidaan indeksoida myös HASH-indeksillä. Sitä kannattaa käyttää silloin, kun tiedetään, että sarakkeesta haetaan yksittäisiä arvoja joukkojen sijaan. B-puu toimii paremmin esimerkiksi numeerisissa kentissä, joista haetaan tiettyjen rajojen sisälle jääviä arvoja.

Faktataulujen indeksoinnissa joudutaan tehokkuuden lisäksi huomioimaan myös indeksin tarvitsema tila. Pääavainta varten ei kannata tehdä omaa kenttäänsä, koska se voidaan muodostaa sijaisavaimista. Mittausfaktojen tapauksessa käytetään aina päivämäärä-, aika- ja mittausviittauksia faktataulujen pääavaimessa. B-puuindeksi vie vähiten tilaa ja toimii nopeimmin silloin, kun selektiivisimmät eli suurimman kardinaliteetin omaavat kentät ovat indeksissä ensimmäisinä. Tältä kannalta ajateltuna MyISAM-taulun indeksi saattaisi olla joissain tauluissa muotoa *<kellonaika, päivämäärä, mittaus>*. Kellonaikaa tarvitaan kyselyissä harvemmin, jonka takia tämä indeksi olisi monissa tilanteissa tarpeeton ja johtaisi taulun läpikäyntiin riviriviltä. Pääavain kannattaakin tehdä juuri päinvastoin, koska kyselyissä mittauskenttää tarvitaan kaikkein varmimmin ja kellonaikakenttää harvimmin. Näin indeksi on useimmissa hakuissa hyödyllinen.

4.5.3 Tallennuspolut

Jokaiselle MyISAM-taululle, taulun indeksitiedostolle ja taulun osiolle on mahdollista määritellä tiedostojärjestelmän tallennuspolku, jonne tiedostot tallennetaan. Näin voidaan esimerkiksi useimmin tarvittava data pitää nopeimmilla medioilla ja harvemmin tarvittava data hitaammilla. Tiedon hajauttamisella eri levyasemille voidaan optimoida tietokanta toimimaan entistä nopeammin, koska yhtäaikaisesti voidaan lukea/kirjoittaa useita eri levyjä. Tällöin täytyy kuitenkin tietää, mitä tietoja tarvitaan useimmin yhtäaikaisesti.

Jos halutaan, että tietovarasto on helposti hallittavissa, voidaan kaikki tiedot tallentaa yhdelle loogiselle asemalle. Hakutehokkuutta saadaan käyttämällä RAID-lomitusta (Redundant Array of Independent Drives). Sillä tieto hajautetaan eri levyille, joilta tiedon luku tapahtuu rinnakkaisesti. Tällaisella kokoonpanolla saadaan hyvä hakunopeus ja jonkinlainen varmistuskin. On vaikea päätellä ennalta, mitä tiedostoja tarvitaan yhtäaikaisesti ja sen perusteella jakaa tiedot eri levyille. Se voidaan

kuitenkin todeta, että julkaisupalvelimen tiedostot kannattaa erottaa puskuripalvelimen tiedostoista eri levyohjaimelle, koska silloin tietoa voidaan tuoda lähdetietokannoista aiheuttamatta kuormaa julkaisupalvelimen kiintolevyille. Lisäksi tiedon siirtäminen julkaisupalvelimelle tapahtuu nopeammin, kun lukeminen tehdään eri levyillä kuin kirjoittaminen. (Kimball et al. 1998, s. 599-602)

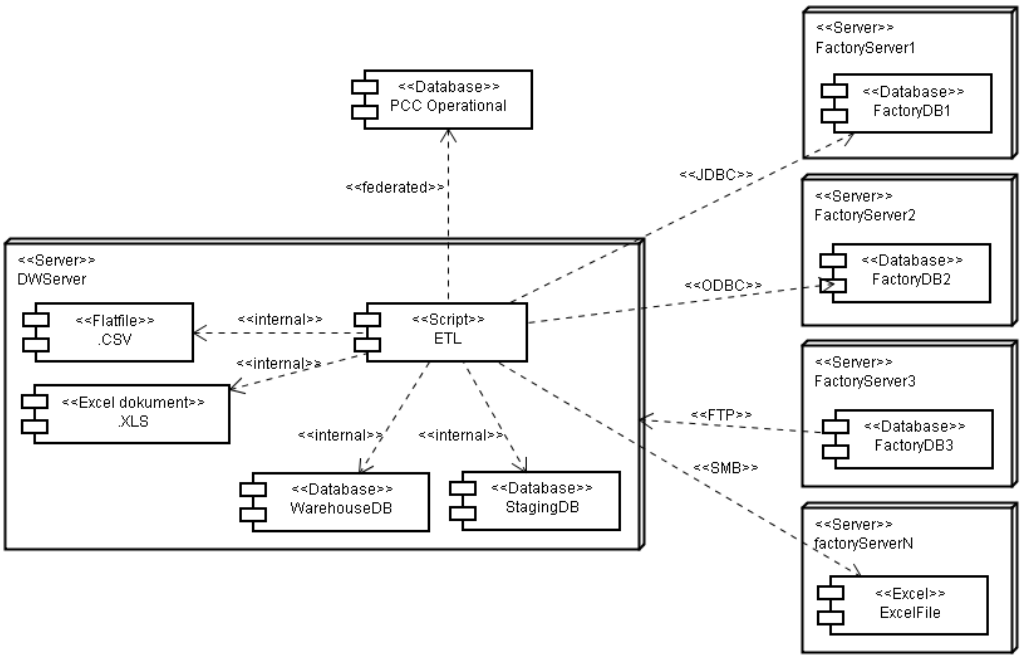
Tietovaraston palvelimella on kaksikanavainen levyohjain. Toiseen kanavaan tehtiin suuri kuuden levyn lomitettu kokoonpano ja toiseen kanavaan kahden levyn pieni peilattu kokoonpano. Suuremmalle osiolle tullaan tallentamaan kaikki tietovaraston data, koska sieltä tiedon saantinopeus on lomituksen takia parempi ja tallennuskapasiteetti suurempi. Pienemmällä kokoonpanolla pidetään ainoastaan puskuripalvelimen tietoja, millä pyritään minimoimaan ETL-prosessien tietokantaan aiheuttamaa kuormaa.

5 MITTAUSTIETOKANNAN TOIMINNAN SUUNNITTELU

Tietokannan rakenteellisen suunnittelun rinnalla suunniteltiin ja toteutettiin tärkeimpiä tietokannan käyttöön liittyviä toimintoja. Taustatoiminnot ovat tietovarastoissa vaativimpia prosesseja. Koska rakenteellinen suunnittelu oli työn tärkein kohde, tehtiin taustatoiminnot ainoastaan siinä mittakaavassa, että tietokannan rakennetta pystyttiin testaamaan oikealla datalla. Hakutyökalujen täytyy toimia nopeasti ja selkeästi, jotta tietovarasto olisi käyttäjien kannalta hyödyllinen. Tässä kappaleessa suunnitellaan myös kyselyt siten, että tieto saadaan tietokannasta mahdollisimman nopeasti ulos.

5.1 Taustatoiminnot (ETL)

Kohdeympäristöön liittyy useita tehtaita, joiden vastuulla on oman tietokantansa ylläpitäminen. Eri tehtailla on erilaisia tietokantoja ja erilaisia tietokantaliittymiä. Tämä tilanne on yleinen tietovarastoinnissa ja pelkästään tiedon poiminta, muunto ja lataus toiminnoista onkin kirjoitettu kokonaisia kirjoja. Kuvassa 21 on kuvattu osittain ETL-toimintoihin liittyvät komponentit.

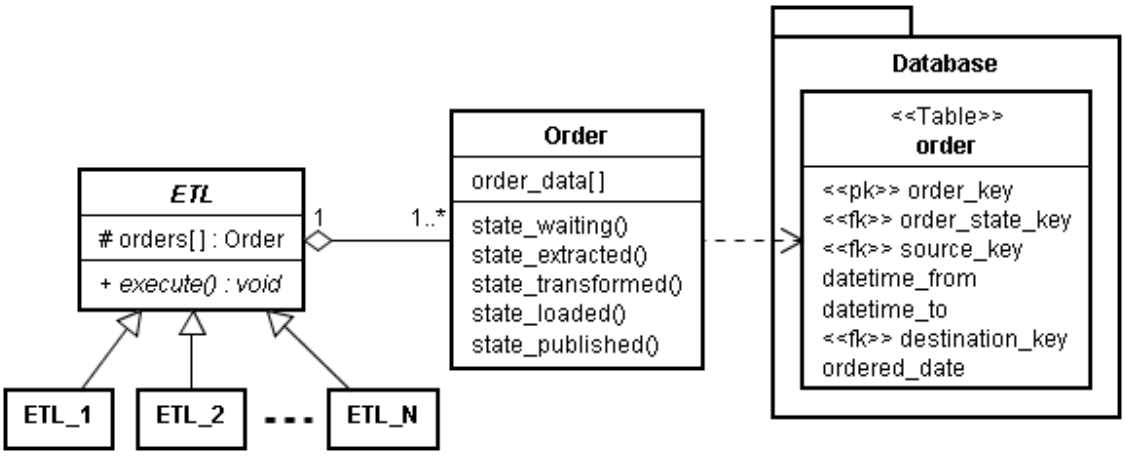


Kuva 21. ETL esimerkki

Joistain lähteistä tietoa voidaan hakea ja joistain läheistä tieto lähetetään tietovarastoon. Myös vapaan lähdekoodin ETL-työkaluja on saatavilla kuten Pentahon Kettle. Usein valmiiden ETL-työkalujen käyttöönotto on kuitenkin lähes yhtä työlästä, kuin kohdeympäristöön soveltuvan ETL-työkalun tekeminen itse (Kimball et al. 1998, s. 356.). ETL-toiminnot toteutettiin PHP-ohjelmointikielellä, koska organisaatiossa on kokemusta sen käytöstä ja siinä on integroituna riittävät toiminnot MySQL-tietokannan käyttöön.

ETL-toimintoja varten tehtiin abstrakti ETL-luokka, josta jokainen konkreettinen ETL-luokka periytyy. Tämän luokan tarkoitus on tarjota kehys kaikille tarvittaville ETL-luokille ja se mahdollistaa myös sen, että tiedonhakua pyytävän koodin ei tarvitse tietää tarkalleen, minkälainen ETL-objekti on kyseessä suorittaessaan sitä. Kuvassa 22 on esitetty yksinkertaistettu kaavio ETL-luokista ja niiden välisistä yhteyksistä.

Tietokannasta voidaan selvittää, millä objektilla tieto on haettava. Tämä tieto on kerrottu lähdetaulujen ja kohdetaulujen linkityksiä kuvaavassa taulussa. Periaatteessa ETL-noudattaa aina kolmea vaihetta, mutta usein nämä vaiheet menevät hieman päällekkäin ja joskus on esimerkiksi tehokkaampaa tehdä muunnosoperaatioita jo latauksen aikana (Kimball et al. 1998, s. 351). Tämän takia tehtiin yksi abstrakti metodi nimeltään *execute()*, jolla ETL-prosessi käynnistetään sen sijaan, että eri vaiheita varten olisi tehty omat abstraktit metodinsa. Näin kehittäjälle jää vapaus suorittaa ETL-vaiheet tehokkaimmalla mahdollisella tavalla.



Kuva 22. ETL luokat

Tilausten tietoja varten ETL-luokkaan määriteltiin taulukko, joka sisältää tilausobjektilistan. Jokainen tilausobjekti sisältää kaikki tarvittavat tiedot, joita ETL-olio tarvitsee tilauksen tietojen poimintaan, muokkaamiseen ja julkaisemiseen. Tilausluokassa on funktiot, joita ETL voi käyttää tilauksen tilan muuttamiseen prosessin edetessä. Tila kirjoitetaan tietokantaan, josta sitä on mahdollista seurata.

Koska ETL hakee lähdetietokannoista tai tiedostoista aina tiettyjä tehdasmittauksia tietyltä aikaväliltä, on tietokannan order-taulu melko yksinkertainen. Siinä on viittaukset lähde- ja kohdesarakkeiden kuvauksiin. Tämän lisäksi on datetime kentät, jotka kertovat, miltä aikaväliltä tilatut mittaukset haetaan ja milloin tilaus on tehty.

INSERT-komentojen lisätessä tauluun tietoa, muuttuu indeksien jakauma. On tärkeää, että latauksen jälkeen kohdetauluille suoritetaan ainakin ANALYSE TABLE -komento. Taulun analysointi tutkii MyISAM-taulun indeksien jakauman ja tallentaa sen. Kyselyn optimoija hyödyntää tätä tietoa suorittaessaan liitos-operaatioita. Jos jakauman analysointia ei tehdä, voi tiedonhaku toimia kummallisesti ja kestää kauan, koska indeksejä käytetään epätehokkaasti. OPTIMIZE TABLE -komento täytyy suorittaa jos ETL-prosessi jostain syystä päivittää rivejä tai jos rivejä poistetaan. Poistetut rivit jättävät tauluun tyhjiä kohtia (jotka eivät näy käyttäjälle), joista taulua täytetään seuraavilla INSERT-kyselyillä. OPTIMIZE TABLE -komento poistaa tyhjät taulun rivit. (MySQL 2007, s. 729-730, 732-733)

5.2 Hakutoiminnot

Tietokannan käyttöä varten kehitettiin yksi tiedonhakusovellus, jolla tietoa voidaan rajata ja hakea. Sovellus osaa yhdistää monen faktataulun tiedot yhteen kyselyyn. Tätä sovellusta varten täytyi suunnitella kuinka kyselyt toteutetaan. MySQL on OLTP-käyttöön optimoitu relaatiotietokanta. Se sisältää SELECT-kyselyn optimoijan. Koska MySQL ei tue suoraan OLAP-kyselyjä, täytyy tutkia kuinka hyvin se suoriutuu SQL-kyselyistä, joilla tietoa haetaan OLAP-tyyppisesti ja miten kyselyt pitää suorittaa, että MySQL osaa optimoida ne. MySQL-kyselyn optimoijaan ei ole kehitetty optimointia tähtiliitoksia varten (Schumacher 2007). Jos kyselyt saadaan yksinkertaisiksi, on tietokannan ylläpitäjän helpompi tarjota käyttäjille uusia raportteja ja hakutyökalun helpompi muodostaa kyselyjä.

MySQL:n EXPLAIN-komennolla voidaan selvittää, kuinka kyselyn optimoija suorittaa kyselyn ja esimerkiksi missä järjestyksessä JOIN-operaatiossa taulut käydään läpi. MySQL versiosta 5.0.36 lähtien käytössä on ollut myös kyselyn profiloija. Profilointi tallentaa kyselyjen suoritusvaiheista yksityiskohtaista tietoa erilliseen tauluun. Näin tietokannan kehittäjä pystyy selvittämään hyvin tarkasti, mikä vaihe kyselyn suorituksessa aiheuttaa ongelmia ja voi sitä kautta optimoida kyselyä, tietokannan rakennetta ja palvelimen asetuksia.

Kyselyiden suunnittelussa käytettiin apuna 17 miljoonan rivin kokoista faktataulua, johon liittyi viisi dimensiota. Taulussa oli noin 1,5 vuoden minuuttikeskiarvot 22:sta eri mittarista. Apuna käytettiin EXPLAIN-komentoa, joka kertoo, kuinka taulut käydään läpi ja liitetään toisiinsa kyselyn optimoijan käsiteltyä kyselyn. Kaikissa kyselyissä haettiin samaa tietoa, mutta hieman eri tavoilla.

5.2.1 Yksinkertainen SQL-tähtiliitos

Ensimmäinen kysely tehtiin mahdollisimman yksinkertaisesti ilman JOIN operaatioita.

```
SELECT [HALUTUT_SARAKKEET]
FROM [DIMENSIO_TAULUT, FAKTA_TAULU]
WHERE [HAKU_RAJOITTEET, YHDISTÄMIS_SÄÄNNÖT]
GROUP BY [RYHMITTELY_SÄÄNNÖT]
```

Kysely osoittautui käyttökelvottomaksi ja sen suoritus täytyi lopulta keskeyttää. EXPLAIN-komento paljastaa hitauden syyksi sen, että MySQL käy järjen vastaisesti ensimmäisenä suuren faktataulun kokonaan rivi riviltä läpi, jonka jälkeen se rajoittaa haettua tietoa dimensioilla.

Koska kyselyn suoritusjärjestys aiheutti hitauden, pyrittiin tätä korjaamaan pakottamalla järjestys optimaalisemmaksi. MySQL:ssa voidaan käyttää STRAIGHT_JOIN sanaa JOIN sanan sijasta, jolloin vasen operandi käydään liitoksessa varmasti läpi ennen oikeaa. Tehtiin kysely, jossa kaikki dimensiot käydään ensin läpi ja vasta sitten liitetään faktatauluun.

```
SELECT [HALUTUT_SARAKKEET]
FROM ( [DIMENSIO_TAULUT] )
STRAIGHT_JOIN [FAKTA_TAULU]
ON [YHDISTÄMIS_SÄÄNNÖT]
WHERE [HAKU_RAJOITTEET]
GROUP BY [RYHMITTELY_SÄÄNNÖT]
```

Tämä kysely palautti tuloksia järjellisessä ajassa, mutta on kuitenkin vielä aika hidas. Todennäköisesti MySQL rajaa faktataulun indeksiä erikseen jokaisella dimensiolla ja lopuksi yhdistää rajatut indeksit keskenään. Riittäisi kuitenkin, että faktataulun indeksiä rajataan ensin yhdellä dimensiolla ja tämän tuloksena saatua joukkoa edelleen seuraavalla dimensiolla jne., jolloin rajaus tehdään jokaisessa liitoksessa aina pienempään ja pienempään joukkoon. Tämä pyrittiin saavuttamaan alla näkyvällä sisäkkäisellä rakenteella.

```
SELECT [HALUTUT_SARAKKEET]
FROM
[DIMENSIO_TAULU1]
JOIN (
      [DIMENSIO_TAULU2]
      JOIN (
          ...
          [FAKTA_TAULU]
          ...
      )
      ON [YHDISTÄMIS_SÄÄNTÖ2]
)
ON [YHDISTÄMIS_SÄÄNTÖ1]
WHERE [HAKU_RAJOITTEET]
```

Tämä toimii edellisiin kyselyihin verrattuna huomattavasti nopeammin. MySQL-kyselyn optimoija osaa optimoida tällaisen kyselyn eikä liitosjärjestyksellä ole kyselyssä väliä. Kyselyn optimoija huomasi, että sen ei kannata käydä suurta faktataulua läpi aivan viimeisenä. Se rajoitti ensin faktataulun indeksijoukkoa neljällä pienimmällä dimensiolla ja vasta tämän jälkeen yhdisti indeksit suurimman dimensiotaulun kanssa. EXPLAIN-komento osoittaa myös, että kyselyssä käydään erittäin vähän rivejä yhteensä läpi ja rivien liitoksissa käytetään tehokkaasti indeksejä.

Kolme erilaista kyselyä, joilla haettiin täysin samaa asiaa, tuottivat hakutehokkuudessa kolme aivan erilaista tulosta. Hakutehokkuuden puolesta moniulotteista mallia voidaankin käyttää MySQL-tietovarastossa, kunhan kyselyn rakenne on suunniteltu siten, että MySQL osaa suorittaa sen tehokkaasti.

Tähtiliitos voidaan laajentaa toimimaan myös useamman faktataulun läpi. Hakeminen monesta faktataulusta käyttäen jotain kaikille tauluille yhteisiä ja yksilöllisiä hakuehtoja voidaan käytännössä tehdä MySQL:lla seuraavalla tavalla. Ensin haetaan ensimmäisestä faktataulusta kaikilla siihen liittyvillä rajoitteilla (kuten edellä) ja

tämän jälkeen kaikista jäljelle jääneistä faktatauluista niiden omilla rajoitteillaan yhdistäen niihin ensimmäisessä vaiheessa rajautuneet yhteiset hakuehdot. Näin kaikille yhteiset ehdot rajautuvat ensimmäisessä tähtiliitoksessa, jolloin niitä ei turhaan haeta jokaiselle faktataululle erikseen.

5.2.2 Pivot-kysely

Usein tietoja halutaan vertailla keskenään ja helpoimmin se onnistuu, kun tiedot ovat rinnakkain. Haku tähtiliitoksella tuottaa paljon rivejä, jotka on ryhmitelty allekkain. Ryhmien hakemiseen rinnakkain eli ns. pivot-kyselyyn ei MySQL tarjoa mitään yksinkertaista menetelmää ja tämä voidaan tehdä monella eri tavalla. Voidaan käyttää alikyselyjä, joissa erikseen kysellään kaikki ryhmät eri sarakkeisiin. Kaikissa alikyselyissä hyödyttäisiin joistain samoista indekseistä ja näin ne haettaisiin turhaan useaan kertaan. Kyselyn voi tehdä myös siten, että jokaista saraketta varten tehdään taululle alias, joiden avulla tiedot haetaan eri sarakkeisiin käyttäen lähes samaan kyselyä kuin edellisessä kappaleessa. Tällainen haku on tehokas, mutta jos tauluista pyydetään monia eri ryhmiä, tarvitaan monia eri aliaksia ja monia JOIN-operaatioita, joiden määrä on kyselyssä rajallinen.

Kolmas ja varmasti paras tapa on suorittaa kysely ensin normaalisti ja tämän jälkeen suorittaa ryhmien uudelleenjärjestely erillisellä pivot-kyselyllä. Sen ideana on että sarakkeissa on ehtoja joilla taulusta luettu tieto asetetaan aina sille kuuluvaan sarakkeeseen ja muut sarakkeet jätetään tyhjiksi kuten taulukossa 4. Näin ryhmittelemättömiin tuloksiin jää aukkoja. Kun tulokset lopuksi ryhmitellään käyttäen MAX-operaatiota, jää ryhmään aina voimaan siihen liittyvä oikea arvo ja tulokset saadaan rinnakkain kuten taulukossa 5. MAX toimii kaikkien tietotyyppien kanssa, jonka takia se soveltuu tähän käyttöön. (Eaton 2006)

Taulukko 3. Tulokset päällekkäin

Mittari	*Neljännes*	*Arvo*
A	1	1,4
A	2	3,7
A	3	4,6
A	4	2,5
B	1	1,1
B	2	1,5
B	3	2,9
B	4	3.1

Taulukko 4. Tulokset rinnakkain

Neljännes	*A*	*B*
1	1,4	-
1	-	1,1
2	3,7	-
2	-	1,5
3	4,6	-
3	-	2,9
4	2,5	-
4	-	3.1

Taulukko 5. Tulokset ryhmiteltynä rinnakkain

Neljännes	*A*	*B*
1	1,4	1,1
2	3,7	1,5
3	4,6	2,9
4	2,5	3,1

Taulukossa 3 tulokset on ryhmitelty mittarin ja vuosineljänneksen perusteella normaalisti. Ne voidaan uudelleen järjestellä taulukon 5 mukaisesti käyttäen alla olevaa pivot-kyselyä. Taulukon 3 eri mittareiden arvoja on vaikea vertailla keskenään vuosineljänneksittäin, kun taas taulukossa 5 vertailu on helppoa. Pivot-kysely on tässä tapauksessa tärkeä, koska lähes aina, kun haetaan useamman mittarin tietoja, halutaan ne rinnakkain ryhmiteltynä ajan suhteen.

```
SELECT Neljännes,
       MAX(CASE WHEN Mittari = 'A' THEN Arvo END) AS A,
       MAX(CASE WHEN Mittari = 'B' THEN Arvo END) AS B
FROM Taulu
GROUP BY Neljännes
```

5.2.3 Summataulujen hyödyntäminen kyselyissä

Oikeastaan ei voida puhua erikseen summatauluista ja faktatauluista, koska kaikki faktataulut ovat lopulta jonkin karkeustason summatauluja. Hakutyökalun on osattava päätellä, minkä karkeustason faktataulusta tietoa haetaan. Koska summaukset tehdään ainoastaan aikadimension suhteen, on taulujen käyttö yksinkertaista. Aina kannattaa valita karkein tilanteeseen soveltuva taulu, jotta tietokantaan tulee pienin mahdollinen kuorma. Karkeimmat tasot ovat tunti- ja päivätasot. Näytteet on summattu näille tasoille kaikista mittausdimension tuntemista mittareista, jolloin haettaessa sopivalta aikaväliltä saadaan varmasti tuloksia. Koska tuntitasoa tarkemmilla tasoilla ei ole kaikkien mittareiden näytteitä, voi käyttäjä helposti osua tyhjään tai saada vain osan haluamistaan tiedoista. Tämä on hyväksyttävää, koska tarkemman tason tietojen tarkastelu on harvinaisempaa ja ne käyttäjät, jotka niitä tarvitsevat tietävät myös niiden olemassaolon. Jos summauksia olisi tehty muidenkin dimensioiden suhteen,

olisi summataulun valinta heti huomattavasti vaikeampaa. Tällöin jouduttaisiin aina päättelemään, minkä dimension suhteen summattu taulu on tilanteessa tehokkain. Lisäksi on hyvä, että ajan suhteen on kaksi kiinteää summatasoa, joilta löytyy kaikki tieto. Sillä saadaan ohjelmien logiikkaa yksinkertaisemmaksi ja myös hakutyökalun virheiden mahdollisuus pienemmäksi.

6 TOTEUTUS, TESTAUS JA KÄYTTÖÖNOTTO

Tietokantaan suunniteltuja ratkaisuja testattiin samalla, kun tietokantaa suunniteltiin. Tietokantaan tehtiin suunnitelmien mukaiset taulut, jotka kuvattiin metatiedolla. Tietoa tietokantaan ladattiin vanhasta tietovarastosta, joka toimii uuden kehitetyn tietovaraston tietolähteenä niin kauan, kunnes jokaiselle oikealle tietolähteelle on saatu kehitettyä toimivat ETL-työkalut. Tietokantaan tehtiin myös yksinkertainen hakutyökalu, jolla tietoa voi hakea web-selaimen avulla.

Tietokannan toteutus jaettiin kahteen vaiheeseen. Koska mittausten lajittelu on suuri ja toistaiseksi tekemätön työ, mittauksia ei jaeta aluksi faktatauluihin mittaustyypeittäin. Kaikkien mittausdimension tuntemien mittalaitteiden näytteet ovat tuntitasolla ja taulujako tehdäänkin siten, että jokaisen tehtaan paikallisvarastoon tehdään jokaista vuotta kohti taulu, johon tallennetaan kaikki kyseisen tehtaan vuoden tuntidata. Tämä pudottaa taulujen määrää dramaattisesti aiemmasta järjestelmästä, jossa taulujako oli tehty sijainnin perusteella nykyistä pienempinä osina. Tämän hetkisellä näytteiden määrällä eräänkin tehtaan kaikkien tarvittavien mittareiden yhteenlaskettu vuosittaisten tuntikeskiarvojen määrä oli hieman yli 10 miljoonaa. Tämä määrä voidaan helposti tallentaa yhteen tauluun dimensiomallilla. Myös päiväkeskiarvot tallennetaan vuosittain. Näin myös hakusovellusten on helppo päätellä, mistä taulusta tieto löytyy. Faktatauluissa käytetään yleisiä dimensioita ja kaikki näytteet tallennetaan liukulukuina.

Toisessa käyttöönoton vaiheessa tunnetaan kaikkien mittareiden kategoriat ja taulujako tehdäänkin mittauskategorian perusteella. Siten jokaiseen faktatauluun voidaan valita näytteiden tietotyyppi tehokkaasti ja taulujen rakenne voi hieman vaihdella tarpeen mukaan. Taulujen määrä kasvaa, mutta tiedot jakautuvat aiempaa yhtenäisempiin osakokonaisuuksiin ja yhdessä taulussa on aina yhden tyyppistä tietoa. Ensimmäisessä vaiheessa tauluja on vähemmän, koska jokaisessa taulussa on tuntemattoman tyyppisiä mittauksia ja taulujako halutaan pitää yksinkertaisena, jotta mittaukset löytyisivät helpommin. Kun mittaukset ovat tauluissa tyypeittäin, on hakutyökalun helppo löytää oikea taulu mittauskategorian, tehtaan ja vuoden perusteella ja tarkemmat haun rajoitteet tehdäänkin dimensioilla.

6.1 Palvelimen asetusten säätö

Palvelimen asetukset pitää säätää tukemaan MyISAM-tauluihin tehtäviä massiivisia hakuja. Tietokannan metatieto on InnoDB-tauluissa ja metatiedon lukemisen pitäisi myös toimia nopeasti, jotta käyttöliittymät voivat sujuvasti esittää käyttäjälle, mitä tietoa tietovarastosta löytyy. Ensimmäinen tavoite on näistä kahdesta tärkeämpi, koska OLAP-haut kohdistuvat suureen tietomäärään ja voi helpommin hidastaa koko järjestelmää.

Indeksivälimuistit ovat erittäin tärkeitä tietovarastoissa. MySQL-palvelimelle on mahdollista määritellä useita indeksivälimuisteja ja tauluille voidaan myös kertoa, mitä välimuistia niiden tulisi käyttää. Useiden välimuistien käyttö on kannattavaa, koska silloin taulut, jotka käyttävät välimuistia A, eivät estä välimuistin B yhtäaikaista käyttöä. Faktataulut ovat suuria kuten myös niiden indeksitiedostotkin. Siksi ne dominoisivat yhtä välimuistia. Faktatauluille onkin parempi tehdä kokonaan oma välimuistinsa. Palvelimelle tehtiin kaksi indeksivälimuistia, joista toinen on faktatauluja varten ja toinen on yleinen indeksivälimuisti, jota käytetään kaikille muille tauluille.

Jos kyselyssä on ryhmittelysääntöjä, järjestetään tulokset automaattisesti niiden mukaan, jollei erikseen anneta myös järjestely (ORDER BY) sääntöjä. Tähtiliitoksella haettaessa käytetään lähes aina ryhmittelysääntöjä. MySQL tallentaa järjestetyt tulokset puskuriin, jonka koko on määritelty sort_buffer_size-muuttujassa. Jos puskuri tulee täyteen, kirjoitetaan se kiintolevylle. Tämän muuttujan pitäisi olla riittävän suuri, että kiintolevyn käyttö pysyisi pienenä. Se ei kuitenkaan saa olla liian suuri, koska jokaiselle säikeelle määritellään oma järjestelypuskurinsa. Tässä tapauksessa siitä tehtiin muutaman kymmenen megatavun kokoinen.

6.2 Testaus

Tietovarastoon tehtävät kyselyt ovat hakutavaltaan hyvin samanlaisia, mutta hakurajoitteet vaihtelevat paljon. Suoritusnopeutta testattiin erilaisilla tiedonhakukyselyillä. Ennen testien suoritusta indeksivälimuistit ja tauluvälimuisti asetettiin nollaksi, jotta kaikki kyselyt olisivat keskenään vertailukelpoisia. Testauksessa selvitettiin myös, kuinka paljon tilaa mittausten tallentaminen vaatii ja kuinka se vastaa aiemmin tehtyjä arvioita. ETL-prosessien nopeuksia ei mitattu, koska

tässä vaiheessa tietokannassa tarvittiin ainoastaan toiminnallisuus tiedon tuontiin vanhasta tietovarastosta uuteen tietovarastoon.

6.2.1 Nopeus

Ensin testattiin kyselyn nopeutta haettaessa mittauksia yhdestä taulusta. Testit tehtiin käyttäen kolmea eri taulua, joissa on samanlaista tietoa. Kaikissa kyselyissä kyseltiin vuoden 2006 dataa, koska se sisältyi jokaiseen tauluun. Pienimmässä taulussa oli dataa ainoastaan vuodelta 2006 ja rivejä siinä oli noin 11,5 miljoonaa. Suuremmat taulut sisältävät reilun 30 miljoonaa riviä ja niissä on data vuodesta 2005 vuoteen 2007 asti siltä osin, kun sitä oli saatavilla. Kahden suurimman taulun erona on se, että toinen niistä on tavallinen MyISAM-taulu ja toinen on muodostettu MERGE-tallennusmoottorilla yhdistäen kolme pienempää taulua, joissa jokaisessa on yhdeltä vuodelta dataa. Jos MERGE-moottorille voisi jotenkin kertoa, että data on jaettu osiin vuosittain, voisi se kohdistaa haun nopeasti kyselyn perusteella oikeaan tauluun. Käytännössä MERGE-taulu ei kuitenkaan ymmärrä, kuinka tieto on jakautunut tauluihin ja siksi se hakee kyselystä riippumatta aina kaikista taulusta.

Liitteessä 4 on esitetty suoritettujen kyselyiden kestot. Kyselyissä muutettiin haettavien mittareiden määrää, joka kasvattaa samassa suhteessa faktataulusta haettavien rivien määrää, koska muut dimensiorajoitteet pysyvät muuttumattomina. Selvästi suoritusajat kasvavat lineaarisesti sen mukaan, kuinka monen mittarin tietoja haetaan. Kahden suuremman taulun välillä MERGE-taulusta hakeminen oli hieman hitaampaa. Taulujen pilkkominen osiin kuitenkin kannattaa, koska pienimmästä taulusta hakeminen oli huomattavasti nopeampaa kuin kahdesta suuremmasta. Päivä- ja tuntikeskiarvohakujen erot ovat pieniä, koska molemmat rajaavat jo niin suuren osan taulusta pois, että vaikka näiden keskinäinen ero onkin suuri, ei se enää näy kyselyajassa. Kun näitä verrataan kuukausikeskiarvojen hakuun, ero on jo huomattavasti suurempi, koska siinä ajan suhteen haetaan suurin osa taulusta.

Taulun koko vaikuttaa edellisten testien perusteella kyselyn suoritusnopeuteen. Tämän takia tehtiin tarkempi tutkimus, jolla selvitettiin taulun koon vaikutusta kyselyn kestoon. Tulokset on nähtävissä liitteessä 5. Testissä käytettiin edellisessäkin testissä käytettyjä 192 mittariin kohdistuvia päivä- ja tuntikeskiarvokyselyjä. Kyselyissä muutettiin ainoastaan taulua, johon kysely kohdistui. Testissä käytettiin viittä erikokoista, mutta rakenteeltaan samanlaista taulua, joista jokaisen pienemmän

taulun data löytyy aina myös sitä suuremmista tauluista. Taulun koko näyttää vaikuttavan lineaarisesti kyselyn suorituksen kestoon ja tässä tapauksessa taulun rivikoon kasvaminen miljoonalla kasvatti kyselyn suorituksen kestoa noin 0,4-0,5 sekuntia.

Koska taulun koko vaikuttaa merkittävästi suoritusnopeuteen, kannattaa data pitää pienissä tauluissa. Pieniä tauluja tarvitaan kuitenkin paljon ja se aiheuttaa lisää työtä ohjelmakoodiin. Tämän takia tauluja kannattaa ehdottomasti osioida. Osiointia ei testattu, koska käytössä oleva vakaa MySQL versio ei vielä tukenut sitä. Tulevaisuudessa osiointi on kuitenkin yksi tärkeimmistä tietovaraston fyysisistä optimointikeinoista. Taulun koko vaikutti paljon kyselynopeuteen ja vaikutusten pitäisi olla osioidussa taulussa vastaavanlaiset.

Tiedot voidaan hakea rinnakkain siten, että todellinen kysely suoritetaan pivot-kyselyn alikyselynä FROM-osassa. Näin oikea kysely suoritetaan aina ensin ja sen tuloksien esitys muutetaan pivot-kyselyllä, jolloin kokonaissuoritusaika koostuu kahdesta osasta. Mitä enemmän todellinen kysely palauttaa tuloksia, sitä enemmän pivot-kysely joutuu ryhmittelemään tuloksia. Liitteessä 5 on esitetty pivot-kyselyn suoritusajat. Vaikuttaisi siltä, että pivot-kyselyn kesto kasvaa polynomisesti todellisen kyselyn palauttaman rivimäärän kasvaessa. Yleensä kuitenkin haetaan niin pieniä tietomääriä, että pivot-operaation osuus on suorituksen kokonaisajasta häviävän pieni. Kun tietoa haetaan erittäin paljon, esimerkiksi tiedon louhintaa varten, on mahdollista, että pivot-kyselyn osuus kasvaa kyselyssä hallitsevaksi.

6.2.2 Tilatarve

Kappaleessa 4.1 arvioitiin, että MyISAM-tallennusmoottorilla tallennetussa faktataulussa tilatarve olisi vähintään 20 tavua yhdelle näytteelle. Käyttöönotetussa tietokannassa yhden sekuntitason näytteen tallennukseen tarvittiin 30 tavua. Tämä on yllättävän lähellä arvioitua teoreettista minimiä ja faktataulut ovatkin erittäin tiheitä. Tuntikeskiarvot veivät 22 tavua yhtä riviä kohti, joka on jo todella lähellä arvioitua minimiä. Tuntitason mittaukset tarvitsivat sekuntitason mittauksia vähemmän tilaa, koska aikadimension tuntitason osajoukon avain kuvataan yhdellä tavulla ja sekuntitaso kolmella. Osajoukkodimensio kaventaa faktataulun rivikokoa kuten myös indeksin kokoa ja siksi eri karkeustason näytteiden tallentaminen vaatii eri määrän tallennustilaa riviä kohti.

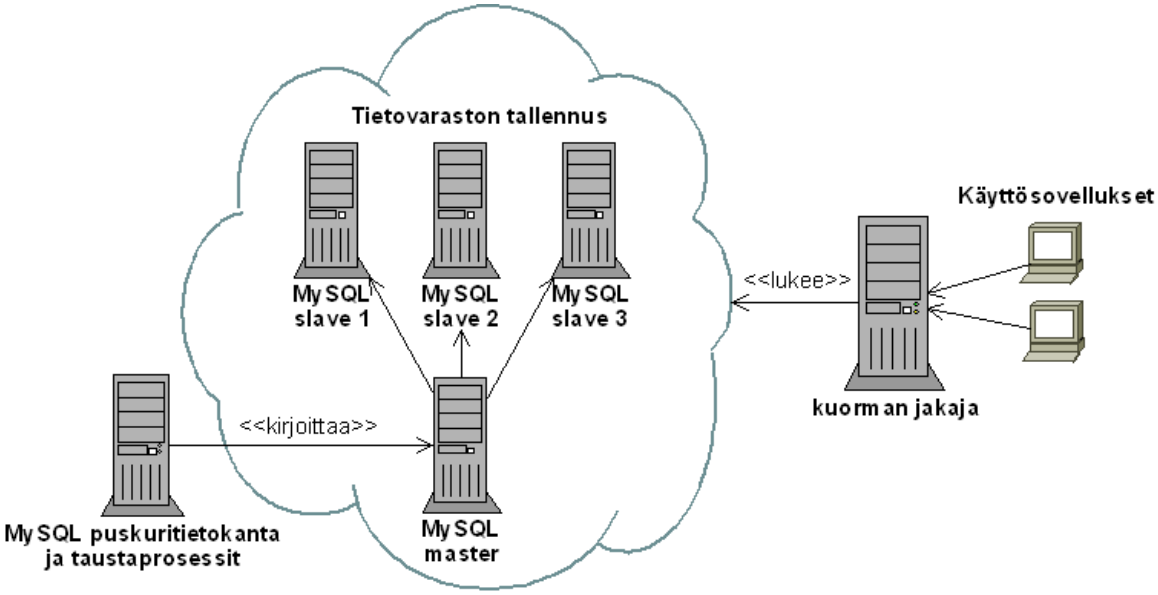
6.3 Tietovaraston skaalaus

Suoritusaika kasvaa lineaarisesti riippuen taulun koosta ja siitä, kuinka suureen tietomäärään kysely kohdistuu. Tämän perusteella tietokannan koon kasvun aiheuttamaa kapasiteetin ja tehon tarvetta voidaan kompensoida kasvattamalla palvelinten määrää, tehoa ja kapasiteettia. Jossain vaiheessa tullaan siihen pisteeseen, että yhden palvelimen tallennustila ei yksinkertaisesti enää riitä kaikkien taulujen tallentamiseen eikä myöskään teho riitä hakuihin. ”Ajatus siitä, että tietovarasto voisi toimia yhden palvelimen varassa, on aivan yhtä realistista kuin 1950-luvun ajatus siitä, että yrityksissä tarvitaan vain yksi tietokone.” (Kimball et al. 1998, s. 28)

Organisaatiossa tarvitaan keskitettyä tehdasmittausten tallentamista. Nyt kehitetty tietovarasto on pieni vain yhden palvelimen tietokanta, joka on kehitetty halvimmalla mahdollisella tavalla. Jos tietovaraston kehittämiseen olisi käytettävissä riittävät varat, voisi se palvella koko organisaatiota yhdellä yhtenäisellä tavalla ja samalla poistaisi kuormaa tehtaiden omista tietovarastoista. Kun kaikki tiedot on saatavilla yhdellä tavalla, helpottuu työntekijöiden työskentely. Jos esimerkiksi työntekijän toimipiste vaihtuu, ei hänen tarvitse opetella uutta tiedonhaku menetelmää mittausten löytämiseksi.

Jos tietovarastoa halutaan tulevaisuudessa laajentaa, kannattaa se tehdä käyttäen MySQL-tiedonhallintajärjestelmää ja replikointia. Muiden tiedonhallintajärjestelmien prosessorikohtaiset lisenssit ovat huomattavasti kalliimpia ja tietovarastossa tarvitaan joka tapauksessa useita palvelimia. Kuvassa 23 tietovaraston julkisen osan tallennus on jaettu neljälle palvelimelle. Lähdetietokannoista haettu ja muokattu tieto kirjoitetaan master-palvelimelle taustaprosessien toimesta. Tämä tieto kopioituu jokaiselle slave-palvelimelle. Jos master-palvelin ei ehdi käsittelemään kaikkia kirjoitusoperaatioita, täytyy tietovarastoa jakaa edelleen osiin sovellusalueittain. Silloin esimerkiksi jokaisen tehtaan tiedoista voidaan tehdä oma replikaatiokokoonpanonsa, eli kuvan 23 pilven sisältöä vastaava rakenne. Näin saadaan käyttöön useita master-palvelimia joissa on eri tietoa ja siten myös kirjoitettavan tiedon määrä on pienempi palvelinta kohti. Sovelluspalvelimet pyytävät tietoa kuormanjakajalta, joka jakaa kyselyn slave-palvelimille. Kuvan tapauksessa kyselyn suoritusaika voi pienentyä jopa kolmannekseen, jos oletetaan, että master-palvelinta ei käytetä tiedon lukemiseen. Lukunopeutta voidaan skaalata lisäämällä

slave-palvelimia. Tavallisesti kuormanjako perustuu virtuaalisiin IP-osoitteisiin, joilla eri käyttäjien kuorma jaetaan tasaisesti eri palvelimille. Tietovarastossa käyttäjiä on vähän, mutta niiden suorittamat kyselyt ovat raskaita. Siksi kuormanjako kannattaa tehdä mieluummin jakamalla kyselyt osiin ja jakamalla niiden eri osat eri palvelimille. Tietovarastossa kyselyn toiminta ja palautuvat rivit ovat ennalta arvattavissa ja kysely on helppo jakaa osiin esimerkiksi aikadimension tai mittausdimension perusteella.



Kuva 23. Scale-out tietovarastoskenaario

Kuvan 23 skenaario on kaukana nykyisestä tilanteesta ja antaa vain suurpiirteisen kuvan siitä, millainen järjestelmä voisi olla jos huomattavasti suurempi osa tiedoista haluttaisiin varastoida. Mitä enemmän tietoa halutaan tallentaa ja mitä enemmän tällä tiedolla on käyttäjiä, niin sitä enemmän tarvitaan varoja tietovaraston laitehankintoihin, kehittämiseen ja ylläpitoon.

7 JOHTOPÄÄTÖKSET

Työssä suunniteltiin ja toteutettiin tietovarasto tehdasmittausten varastointiin. Lähtötilanteena oli tietokanta, jonka rakenne oli suunniteltu pelkästään tiedon tallentamista varten. Rakenteessa ei ollut huomioitu tiedon luokittelua ja ryhmittelyä, jonka takia tiedon löytäminen tietokannasta oli vaikeaa. Tietokannan rakenne ei myöskään kestänyt kohdeympäristössä tapahtuvia muutoksia.

Tietovarastoja on tutkittu paljon ja nykyään niistä onkin saatavilla paljon tietoa. Yleisesti ottaen tehdasmittausten varastoinnista ei kuitenkaan löytynyt julkaisuja tai tutkimuksia. Tämän takia olikin mielenkiintoista selvittää, kuinka liiketoimintaympäristöistä tuttu ja hyväksi havaittu tähtimalli soveltuu tehdasmittausten varastointiin. Työn toteutuksessa käytetystä MySQL-tiedonhallintajärjestelmästä on saatavilla paljon tietoa, mutta vain vähän siitä, kuinka se soveltuu tiedon varastointiin. Tässä työssä käsiteltiin myös tätä aihetta ja siksi hyötyä saavat myös ne tahot, jotka haluavat kehittää kustannustehokkaan tietovaraston MySQL-tiedonhallintajärjestelmän avulla.

7.1 Käytettyjen tekniikoiden soveltuvuus

Tietokanta suunniteltiin moniulotteisen mallin mukaisesti tähtimallilla. Moniulotteinen malli ratkaisi kaikki lähtötilanteessa havaitut ongelmat. Tiedon ryhmittely on mahdollista useiden eri kriteerien perusteella ja siksi myös tiedon löytäminen ja analysoiminen on helpompaa. Kohdeympäristössä tapahtuvat muutokset voidaan täydellisesti jäljittää dimensioiden avulla ja tieto pysyy ehjänä kaikkina aikoina. Hakutyökalun tekeminen on myös helppoa, koska tietokantaan tehtävät haut ovat aina samankaltaisia. Vaatimuksena oli, että tietomallilla voidaan tallentaa sellunvalmistusprosessin mittauksia. Kehitetty tietokanta on kuitenkin laajennettavissa minkä tahansa tehdasmittauksen tallentamiseen yksinkertaisesti lisäämällä dimensioihin uutta tietoa.

Käytännössä merkittävin asia, jonka moniulotteinen tietomalli tarjosi, ovat erilaiset tiedon ryhmittelytavat. Yhteen faktatauluun liitetyissä dimensiotauluissa on kymmenittäin tekstikenttiä. Faktataulujen tietoja voidaan rajata minkä tahansa sarakkeen perusteella. Teoriassa SQL-kysely voisi pyytää tietokannasta esimerkiksi:

”hae tehtaan T linjan L säiliön S paine P aikavälillä A”. Tässä huomataan, että välttämättä ei tarvitse tuntea mittalaitteen positionimeä (vaikeita koodeja), kunhan tiedetään esimerkiksi mitä, mistä ja milloin halutaan. Toisaalta, jos halutaan vertailla eri sijaintien tuotantomääriä, voitaisiin pyytää: ”hae tehtaittain vaihetuotteen T tuotantomäärä M prosessin P vaiheesta V aikaväliltä A”.

Suhtauduin alussa hieman varauksella moniulotteisen mallin toimivuuteen MySQL-ympäristössä, koska sitä ei ole suunniteltu OLAP-tyyppiseen käyttöön eikä se tarjoa esimerkiksi bittikarttaindeksejä. Eniten epäilytti, että se ei joko toimi riittävän nopeasti tai sen käyttö vaatii monimutkaisia kyselyjä. Monipuolinen tallennusmoottorivalikoima antoi kuitenkin riittävät työkalut tietovaraston rakentamiseen ja kyselyistä tulikin melko yksinkertaisia. Dimensiot saatiin tallennettua kätevästi suoraan keskusmuistiin, josta niiden läpikäynti tapahtuu niin nopeasti, että rajoittavaksi tekijäksi jäi vain faktataulun indeksin läpikäyntitehokkuus. Faktataulun fyysisiä rakenteita varten ei ollut mitään huipputehokasta tallennusmoottoria käytössä. MyISAM-tallennusmoottorilla ja monisarakkeisella pääavaimella faktataulut saatiin kuitenkin tallennettua niin, että lopputulos oli yllättävän hyvä. Tulevaisuudessa faktataulujen hakuaikoja saadaan osioinnin avulla vielä entistä matalammiksi. MySQL on varmasti tulevaisuuttakin ajatellen hyvä valinta, koska se on halpa (tai ilmainen) ja erittäin skaalautuva. Jos tietovarasto tukehtuu, voidaan sitä melko helposti elvyttää rautahankinnoin ja replikoimalla tekemättä tietokantaan juuri mitään muutoksia.

7.2 Muut havainnot

Ei ollut yllätys, että jokaisen tehtaan paikallinen tietovarasto on erilainen, koska tehtaat on rakennettu eri aikoina ja markkinoilla on useita eri toimittajia. Oli kuitenkin hieman yllättävää ettei organisaatiossa ollut tehtaiden omien tietovarastojen lisäksi kollektiivista tietovarastoa, jonne kerättäisiin kaikki tiedot tehtaiden järjestelmistä yhtenäisellä tavalla pitkältä aikaväliltä. Lisäksi tehdastietokannoista saatavan tiedon epäyhtenäisyys ja epätäydellisyys oli yllättävää. Näihin asioihin täytyy todella panostaa, koska muutoin tulevaisuudessakin joudutaan tekemään aina yhtä paljon työtä, jotta päästään tutkimuksessa alkua pitemmälle. Tuntuisi loogisemmalta, että tehtailla säilytettäisiin tietoa vain lyhyeltä ajalta, mutta erittäin tarkalla tasolla, koska

siellä tietoa käytetään prosessin ohjaamiseen ja säätöjen virittämiseen. Tämä tieto siirtyisi tasaisesti näistä prosessin hallinnassa käytettävistä ns. reaaliaikaisista tietovarastoista kollektiiviseen historiatietovarastoon, jossa kaikki tieto olisi yhtenäistä, mutta mahdollisesti hieman karkeampaa. Tällainen visio on kallis toteuttaa, mutta se laajentaisi tiedon käyttötavat aivan uusiin ulottuvuuksiin ajatellen nyt muitakin tahoja kuin tutkijoita.

Yleisin tietovarastoinnin esimerkkitapaus on vähittäistavarakauppa, jossa varastoidaan tietoja myyntitapahtumista. Tällä saralla tietovarastoilla onkin saavutettu suuria hyötyjä. Varmasti yhtä suuria tai suurempia hyötyjä voidaan saavuttaa prosessimittausten varastoinnilla. Voidaan vain kuvitella, mitä tuhannet sellunvalmistusprosessin mittarit voivat meille kertoa, kunhan niiden tuottamaan dataan päästäisiin helposti ja yhtenäisellä tavalla käsiksi. Näitä piileviä ominaisuuksia ei aina tarvitse selvittää itse, koska tiedon louhinnalla voidaan havaita piileviä ominaisuuksia lähes automaattisesti. Kärjistettynä voitaisiin sanoa, että tutkijan vastuulle jää enää vain havaittujen ominaisuuksien ja vaikutusten nimeäminen. Tiedon louhinta edellyttää toimivaa tietovarastoa, josta tutkijat voivat rajata louhinnalla tutkittavan tietojoukon ja massiivista laskentatehoa, jolla tätä tietoa käsitellään.

7.3 Jatkotoimenpiteet

Tämä työ sisälsi pääasiassa tietorakenteiden suunnittelun. Tulevaisuudessa tiedon käyttöä täytyy tehostaa kehittämällä tietovaraston toiminnallisia osia ja lisäämällä tietovarastoon tietoa. Tärkeimpiä kehityskohteita ovat mittausdimension tietojen eheyttäminen, taustatoimintojen kehittäminen ja käyttöliittymän kehittäminen. Mittausdimension tietojen eheyttäminen on tärkeää, koska tiedon hakeminen ja luokittelu perustuu dimensioiden tietoihin eivätkä ne voi toimia täydellisesti, jos käytössä ei ole täydellisiä tietoja mittauksista. Olemassa olevan tiedon avulla voidaan automaattisesti selvittää suuri osa puuttuvista tiedoista, joka helpottaa selvitystyötä. Tämä työ ei kattanut taustatoimintojen kehitystä. Se on vaativa urakka, koska lähdejärjestelmiä on paljon. Taustatoimintojen kehittäminen on jatkon kannalta tärkeintä, koska ilman taustatoimintoja tietovarastoon ei saada uutta tietoa. Jotta

käyttäjät pääsevät lopulta tietoihin käsiksi, tarvitaan myös käyttöliittymä. Tietomalli antaa nyt paremmat edellytykset myös käyttöliittymän kehittämiseen.

LÄHTEET

Burst, A. J. & Forte, S. Programming Microsoft SQL Server 2005. Microsoft Press, Washington 2006. ISBN 978-0-7356-1923-4

Codd, E. F. A Relational Model of Data for Large Shared Data Banks. Communications of the ACM, Volume 13, Number 6, June 1970, s. 377-387.

Codd, E. F., Codd, S. B. & Salley, C. T. Providing OLAP to User-Analysts: An IT Mandate. Codd & Date, Inc. San Jose 1993.

Computerworld. MySQL gets another database engine - NitroEDB is being optimized for big databases. 15.11.2006. Viitattu: 5.11.2007. http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=9005113

Darmawikarta, D. Dimensional Data Warehousing with MySQL: A Tutorial 1st Edition. BrainySoftware Corp, 2007. ISBN 0-9752128-2-6

DeSouza, E., Walter, L., Hyde, J. OLAP for MySQL using Pentaho's Mondrian. 24.10.2006. Viitattu 15.11.2007. http://www.mysql.com/news-and-events/on-demand-webinars/pentaho-20061024.php

Eaton, C. 11.6.2006. Viitattu 3.12.2007. http://blogs.ittoolbox.com/database/technology/archives/pivot-query-12757

Elmazri, R. & Navathe, S. B. Fundamentals of Database Systems 4th Edition. Pearson Education, Inc., 2004. ISBN 0-321-20448-4

MySQL. Guide to Cost-effective Database Scale-Out using MySQL - For Web and Data Warehouse Applications, July 2005, A MySQL® Business White Paper.

Han, J. & Kamber, M. Data Mining – Concepts and Techniques 2nd Edition. Elsevier Inc., USA 2006. ISBN 978-1-55860-901-3

Hernandez, M. J. Tietokannat – Suunnittelu käytännössä. IT Press., Suomi 2000. ISBN 951-826-137-7

Hovi, A. Data Warehousing – Tietovarastotekniikka. Suomen Atk-kustannus., Espoo 1997. ISBN 951-762-509-X

Infobright kotisivu. Ei päiväystä. Viitattu 12.11.2007. http://www.infobright.com/products.php

Kimball, R., Reeves, L., Ross, M. & Thornthwaite, W. The data warehouse lifecycle toolkit 1st Edition. John Wiley & Sons, Inc., USA 1998. ISBN 0-471-25547-5

KnowPulp - sellutekniikan ja automaation oppimisympäristö. Viitattu 15.11.2007. http://www.knowpulp.com/suomi/

Lenz, A. MySQL Storage Engine Architecture, Part 1, 2 & 3. 2004. Viitattu 16.10.2007. http://dev.mysql.com/tech-resources/articles/storage-engine/part_1.html, http://dev.mysql.com/tech-resources/articles/storage-engine/part_2.html, http://dev.mysql.com/tech-resources/articles/storage-engine/part_3.html

Mattila, J. K. Sumean logiikan oppikirja – johdatus sumean matematiikkaan. Art house, Vantaa 2002. ISBN 951-884-300-7

MySQL 5.0 reference manual revision: 9063. 3.12.2007. Viitattu 3.12.2007. http://downloads.mysql.com/docs/refman-5.0-en.a4.pdf

Narasimhaiah, G. Features to consider in a data warehousing system. Communications of the ACM November 2003 / Vol. 46, No. 11, s. 111-115.

Pachev, S. Understanding MySQL Internals 1st Edition. O'Reilly Media, Inc., USA 2007. ISBN 0-596-00957-7

Paperiliiton kotisivu, työvuorokalenterit. Ei päiväystä. Viitattu 25.10.2007. http://www.paperiliitto.fi/paperiliitto/suomeksi/tyovuorokalenteri/index.php

Ronstrom, M. Partitioning and Locking. 27.3.2006. Viitattu 12.11.2007. http://mikaelronstrom.blogspot.com/2006/03/partitioning-and-locking.html

Schiff, M. MySQL Data Warehouse Web Presentation. 20.10.2005. Viitattu 15.11.2007. http://www.mysql.com/news-and-events/on-demand-webinars/data-warehouse.php

Schumacher, R. More on MySQL 5.1 partitioning. MySQL developer articles 23.5.2006. Viitattu 22.10.2007. http://dev.mysql.com/tech-resources/articles/mysql_5.1_partitioning.html

Schumacher, R. MySQL 5.0's Pluggable Storage Engine Architecture, Part 1: An Overview. MySQL developer articles 19.10.2005. Viitattu 22.10.2007. http://dev.mysql.com/tech-resources/articles/mysql_5.0_psea1.html

Sennhauser, O. MySQL Scale-Out by application partitioning. MySQL developer articles 8.11.2006. Viitattu 20.10.2007. http://dev.mysql.com/tech-resources/articles/application_partitioning_wp.pdf

Stockinger, K., Rotem, D., Shoshani, A. & Wu K. Analyzing Enron Data: Bitmap Indexing Outperforms MySQL Queries by Several Orders of Magnitude. Lawrence Berkeley National Laboratory 23.1.2006.

Stora Enson kotisivut, Mills. 7.11.2007. Viitattu 15.11.2007. http://www.storaenso.com/CDAvgn/main/0,,1_EN-1926-14331-,00.html

Stora Enson kotisivut, Pulp. 17.4.2007. Viitattu 8.10.2007. http://www.storaenso.com/CDAvgn/main/0,,1_EN-3047-13986-,00.html

Teorey, T., Lightstone, S. & Nadeau, T. Database Modeling & Design 4th Edition. Elsevier Inc., USA 2006. ISBN 0-12-685352-5

Wikipedia, Tuple. 30.10.2007. Viitattu 15.11.2007. http://en.wikipedia.org/wiki/Tuple

LIITE 1. MySQL-tallennusmoottoreiden ominaisuudet

Ominaisuus	InnoDB	MyISAM	MERGE	ARCHIVE	MEMORY	CSV	NDB CLUSTER	FEDERATED
Lyhyesti	OLTP-moottori. Transaktiot, viiteavaimet jne.	Nopea tiedon tallennus ja luku	Yhdistää useita samanlaisia MyISAM-tauluja	Zlib-menetelmällä pakattu data	Taulu muistissa	Ns. ”flatfile”-tallennusmoottori	Hajauttava tallennusmoottori	Linkki ulkoiseen MySQL-tauluun
Indeksointi	Ryvästetty pääavain, B+-tree, (hash lookup)	B-tree, R-tree, Fulltext	Indeksit kohdetauluissa	Ei indeksointia	B-tree, Hash	Ei indeksointia	B-tree, Hash	Kohdetaulun vastuulla
Paikallinen levytilan tarve	Iso	Pieni	Kuvaus	Erittäin pieni	Erittäin pieni (vain kuvaus)	Iso	Pieni	Erittäin pieni (vain kuvaus)
Pakkaus	Ei	Mahdollista	Kohdetauluissa mahdollinen	Aina pakattu	Ei	Ei	Ei	Kohdetaulun vastuulla
Muistin käyttö	Korkea	Matala	Matala	* (Pakkaaminen / purkaminen vaatinee muistia)	Datan lisäksi matala muistin käyttö	* Matala	Korkea	Kohdetaulun vastuulla
SELECT nopeus	Hidas	Nopea	Nopea	Hidas yleisesti, mutta nopeampi taulun skannaus, kuin MyISAM moottorissa.	Erittäin nopea	*Hidas	Erittäin nopea	Riippuu kohdetaulusta
INSERT nopeus	Hidas	Nopea. Hidas jos useita käyttäjiä.	Kuten MyISAM	Nopea (nopeampi, kuin MyISAM)	Erittäin nopea	*Hidas	Erittäin nopea	Riippuu kohdetaulusta
UPDATE nopeus	Nopea	Hidas	Hidas	Ei tuettu	Erittäin nopea	*Hidas	Erittäin nopea	Riippuu kohdetaulusta
Tiedon tallennus	Oma tiedosto tai tauluavaruus	Oma tiedosto	MyISAM-tiedostot	Oma tiedosto	Muistiin	Oma tiedosto	Taulut viipaloitu eri palvelimille	Ulkopuolisella palvelimella
Viiteavaimet	Kyllä	Ei	Ei	Ei	Ei	Ei	Ei	Ei
Lukitus	Rivi	Taulu	Taulu	Rivi	Taulu	*	Rivi	Kohdetaulun vastuulla
Potentiaalinen käyttötapa tietovarastossa	Metatieto ja muu hallinnassa tarvittava data	Aktiivisest faktataulut	Periodisen datan yhdistäminen	Vanhentuneet faktataulut	Väliaikainen tallennus, optimointi, dimensiot	Tiedon tuonti lähdetietokannasta	Kriittisten toimintojen nopeuden skaalaus	Tiedon tuonti lähdetietokannasta

* Varmaa tietoa ei löytynyt

LIITE 2. Frontroom-metatieto

Faktataulu

- Taulun nimi
- Tietokannan nimi
- Tallennusmoottori
- Käyttöönotto päivämäärä
- Sisällön kuvaus

Fakta (faktataulun sarake)

- Sarakkeen nimi
- Faktan kuvaus
- Viittaus faktatauluun johon sarake kuuluu
- Suure ja yksikkö (jos kiinteät)
- Tietotyyppi

Dimensiotaulu

- Nimi
- Taulun nimi

Attribuutti (Dimensiotaulun sarake)

- Viittaus dimensiotauluun johon sarake kuuluu
- Sarakkeen nimi
- Tietotyyppi

Käsitehierarkia

- Viittaukset karkeistavaan ja tarkentavaan dimensiosarakkeeseen
- Kardinaliteetti (kuvaa tiedon määrän moninkertaistumista)

Dimension ja faktataulun liitos

- Avaimet faktataulun kuvaukseen ja dimensiotaulun kuvaukseen
- Sijaisavainsarakkeen nimi faktataulussa
- Sijaisavainsarakkeen nimi dimensiotaulussa

LIITE 3. Backroom-metatieto

Lähdejärjestelmä

- Järjestelmän nimi
- osoite
- portti
- protokolla
- käyttäjätunnus ja salasana

Lähdesarake

- Viittaus lähdejärjestelmään
- Tietokannan nimi
- Taulun nimi
- Sarakkeen nimi

Lähdesarakkeen ja faktasarakkeen liitos

- Viittaukset lähdesarakkeeseen ja faktasarakkeeseen
- Viittaus mittausdimensioon (kertoo, mitä mittausta ko. sarakkeessa)
- ETL luokan nimi

Tilaus

- Viittaus lähde- ja faktasarakkeen liitokseen
- Tilauksen tila (poimittu, muunnettu, ladattu)
- Aikaväli, jolta mittaukset haetaan

LIITE 4. Kyselyt 1

Tarvittavan faktatiedon määrän vaikutus kyselyn kokonaissuoritusaikaan

Tauluihin tehtävät kyselyt

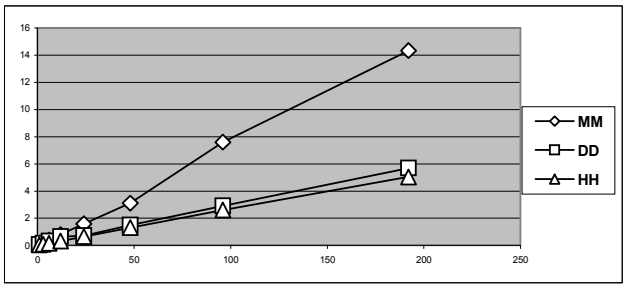
MM	Kuukausikeskiarvot yhdeltä vuodelta
DD	Päiväkeskiarvot yhdeltä kuukaudelta
HH	Tuntikeskiarvot seitsemältä päivältä (keskiarvotusta ei kyselyssä tarvita, koska data valmiiksi tuntikeskiarvoina)

Taulut

MyISAM (11 457 888) taulussa on yhden tehtaan yhden vuoden kaikki keräyksessä olevat tuntimittaukset
MyISAM (30 375 743) taulussa on yhden tehtaan noin kahden ja puolen vuoden kaikki keräyksessä olevat tuntimittaukset
MERGE (30 375 743) samat mittaukset kuin edellisessä, mutta jaettuna kolmeen eri tauluun vuosittain ja yhdistettynä MERGE:llä

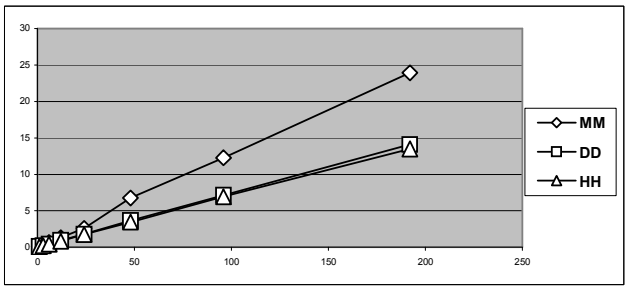
MYISAM (11 457 888)

Mittareita	MM	DD	HH
1	0,06	0,06	0,03
3	0,2	0,16	0,08
6	0,41	0,34	0,16
12	0,81	0,65	0,32
24	1,59	0,72	0,66
48	3,11	1,49	1,31
96	7,59	2,9	2,6
192	14,34	5,69	5,05



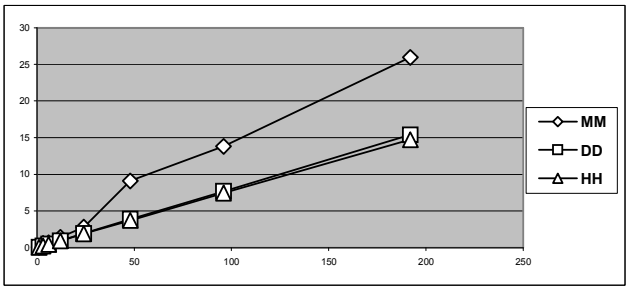
MYISAM (30 375 743)

Mittareita	MM	DD	HH
1	0,1	0,14	0,07
3	0,32	0,23	0,21
6	0,67	0,45	0,43
12	1,33	0,93	0,86
24	2,59	1,82	1,75
48	6,75	3,66	3,46
96	12,26	7,15	6,94
192	23,94	14,1	13,49



MERGE (30 375 743)

Mittareita	MM	DD	HH
1	0,1	0,08	0,08
3	0,59	0,24	0,24
6	0,7	0,49	0,46
12	1,43	0,98	0,94
24	2,8	1,96	1,93
48	9,12	3,9	3,72
96	13,81	7,69	7,47
192	25,96	15,39	14,74

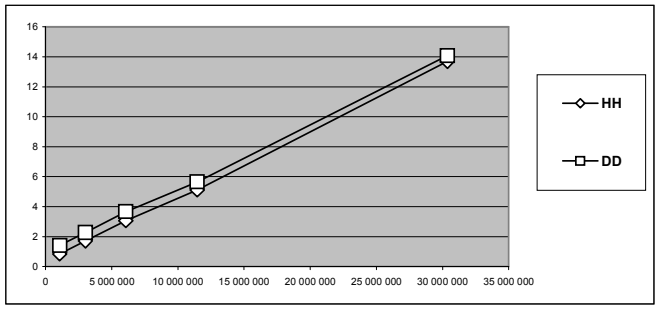


LIITE 5. Kyselyt 2

Taulun koon vaikutus kyselyn suoritusaikaan

Tauluihin tehdyt kyselyt	
DD	Päiväkeskiarvot yhdeltä kuukaudelta
HH	Tuntikeskiarvot seitsemältä päivältä (keskiarvotusta ei kyselyssä tarvita, koska data valmiiksi tuntikeskiarvoina)

Taulun koko		**Suoritusaika (sek)**	
Ajallisesti	*Riviä*	*HH*	*DD*
Kuukausi	1 063 971	0,85	1,41
Vuosineljännes	3 002 682	1,72	2,26
Puoli vuotta	6 061 371	3,07	3,66
Vuosi	11 457 888	5,12	5,65
Noin 2,5 vuotta	30 375 743	13,69	14,07



Kyselyn palauttaman tiedon määrän vaikutus pivot operaation suoritusnopeuteen

Mittauksia	**Palautettuja rivejä**	**Kesto(sek)**
3	21249	0,07
6	42498	0,18
12	85338	0,55
24	168931	1,83