TEKNILLINEN KORKEAKOULU
Sähkö- ja tietoliikennetekniikan osasto

Jouni Karppinen

Tietovarasto automaattisten tarkastusjärjestelmien keräämälle datalle

Diplomityö, joka on jätetty opinnäytteenä tarkastettavaksi diplomi-insinöörin tutkintoa varten Espoossa 16.1.2007

Työn valvoja: professori Lauri Malmi
Työn ohjaaja: TkT Lauri Malmi

TEKNILLINEN KORKEAKOULU Diplomityön tiivistelmä

Tekijä: Jouni Karppinen

Työn nimi: Tietovarasto automaattisten tarkastusjärjestelmien keräämälle datalle

Päivämäärä: 16.1.2007 Sivumäärä: 80

Osasto: Sähkö- ja tietoliikennetekniikan osasto

Professuuri: T-106 Ohjelmistotekniikka

Työn valvoja: prof. Lauri Malmi

Työn ohjaaja: TkT Lauri Malmi

Tiivistelmäteksti:
Tässä diplomityössä luodaan tietovarasto Teknillisen korkeakoulun tietotekniikan perusopetuksessa käytettävien automaattisten tarkastus-järjestelmien datalle. Työn tärkeimpiä erityispiirteitä ovat sovellusalue, pyrkimys käyttää avoimen lähdekoodin työkaluja ja keskittyminen tietovaraston luomisvaiheeseen sekä teknologiavalintoihin. Tyypillisesti tietovarastointia käytetään suurissa yrityksissä yhdistämään hajanaisia tietojärjestelmiä ja kehittämään raportointia ja datan analysointia liiketoiminnan kehittämiseksi. Tässä työssä menetelmät ovat samoja, mutta tavoitteena on auttaa tutkimuksessa ja opetusmenetelmien kehittämisessä.

Tietovarastoon kerätään dataa kolmesta eri järjestelmästä. Teknologisesti vertaillaan eri tietokantatekniikoita. Avoimen lähdekoodin tietokanta-sovelluksista ja tietovarastointityökaluista valitaan soveltuvimmat, ja myös analysointivälineitä käydään läpi.

Viimeisten kymmenen vuoden aikana tietovarastoinnissa vakiintuneet relaatiotietokannat ovat tekniikkana edelleen selvästi vahvimmat erityisesti kehittyneisyyden ja yhteensopivuuden ansiosta. Avoimen lähdekoodin tietovarastointiratkaisut kehittyvät kovalla vauhdilla, ja tietovaraston pystyy jo rakentamaan niiden varaan.

Tässä työssä toteutetun tietovaraston arkkitehtuuri perustuu yksittäiseen keskustietovarastoon, joka noudattaa skeemaltaan denormalisoitua eli toisteista dataa sisältävää tähtimallia. Jatkossa tietovarastoa voidaan kehittää parantamalla datan analysointimahdollisuuksia ja laajentamalla sitä uusiin järjestelmiin.

Avainsanat:
tietovarasto, data warehouse, automaattinen tarkastaminen, dimensiomalli, ETL

HELSINKI UNIVERSITY OF TECHNOLOGY Abstract of the Master's Thesis

Author: Jouni Karppinen

Title: A Data Warehouse for the Data of Automatic Assessment Systems

Date: 16.1.2007 Number of pages: 80

Department: Department of Electrical and Communications Engineering

Professorship: T-106 Software Technology

Supervisor: Professor Lauri Malmi

Instructor: Lauri Malmi, Dr.Sc.(Tech)

Abstract:
Several automatic assessment systems are used at Helsinki University of Technology in basic computer science education. In this thesis a data warehouse is created for the data collected by those systems. Typically data warehousing is used in large companies to integrate scattered database systems and to enhance reporting and data analysis to improve competitiveness. This work uses the same methods but aims to help in research and teaching. In addition to the unusual field, other properties of this work include using open source software and concentrating on the creation of the data warehouse and its technological choices.

The data warehouse collects data from three separate systems. Several different database technologies are compared to find the most suitable one. The best database and data warehousing tools are selected and some analysis tools are explored as well.

During the last decade data warehouses have been most commonly built upon relational databases. They are still the most prominent technology for the job, which derives from the maturity of the technology and good connectivity to other software and systems. Open source data warehousing solutions are developing at a quick pace right now and a data warehouse can already be built with open source tools.

In this work the architecture of the data warehouse is based on a single central data warehouse. It uses the star schema, which is a denormalised model containing redundant data. In the future, the data warehouse can be improved by developing the analysis capabilities and by expanding it to new data sources.

Keywords:
data warehouse, automatic assessment, dimensional model, ETL

Alkulause

Tämä diplomityö on tehty Teknillisen korkeakoulun Ohjelmistotekniikan laboratoriossa.

Kiitokset työni ohjaajalle ja valvojalle professori Lauri Malmille. Kiitokset myös kaikille työtovereilleni heiltä saamastani avusta työn eri vaiheissa.

Espoossa 16.1.2007

Jouni Karppinen

Sisällysluettelo

Sanasto

ACID
- neljä ominaisuutta joita edellytetään kunnolliselta tietokannalta
- Atomicity = atomisuus, operaatio tehdään kokonaan tai ei ollenkaan
- Consistency = konsistenssi, data on aina eheää eli vastaa tietokannan määrityksiä
- Isolation = isolaatio, samanaikaiset operaatiot eivät häiritse toisiaan
- Durability = pysyvyys, tehdyt muutokset eivät katoa (lokit, varmuuskopiot)

automaattinen tarkastaminen
- opiskelijoiden harjoitustehtävät tarkastetaan ja pisteytetään tietokoneella automaattisesti eli ilman ihmistyötä

dimensiomalli, tähtimalli (dimensional model, star schema)
- relaatiotietokantapohjaisten tietovarastojen yleisin, muodoltaan denormalisoitu eli toisteista tietoa sisältävä skeema, joka simuloi moniulotteisuutta kaksiulotteisessa relaatiotietokannassa

dimensiotaulu, ulottuvuustaulu (dimensional table)
- tähtimallin sakaroina olevat tietokantataulut, jotka kategorisoivat haettavia faktoja

ETL
- tietovaraston luomisprosessi
- Extract = poiminta, kerätään tiedot lähdejärjestelmistä
- Transform = siirto, siistitään ja yhtenäistetään tieto
- Load = lataus, syötetään tieto tietovarastoon

faktataulu (fact table)
- tähtimallin keskipisteenä oleva tietokantataulu, joka sisältää numeerisia mittaustuloksia eli faktoja

FASMI (Fast Analysis of Shared Multidimensional Information)
- määrittää OLAP:n vaatimukset viiden termin avulla

heterogeeniset tietokannat
- keskenään erilaiset tietokannat, jotka voivat poiketa toisistaan mm. skeeman ja datan muodon suhteen

hienovaraiset muutokset (graceful modifications)
- tavat joilla tietovaraston rakenne voi muuttua ilman muutoksia vanhoihin ETL-määrityksiin tai tietovarastoa hyödyntäviin sovelluksiin

hitaasti muuttuva dimensio (slowly changing dimension)
- käsittää erilaisia keinoja käsitellä tietovaraston vanhaa dataa päivitysten yhteydessä

JDBC (Java Database Connectivity)
- Java-pohjainen rajapinta sovelluksista tietokantoihin

lumihiutalemalli (snowflake schema)
- tähtimallista kehitetty tietokantamalli, jossa ulottuvuustaulut normalisoidaan eli vältetään tiedon toisteisuus

ODBC (Open Database Connectivity)
- järjestelmästä ja ohjelmointikielestä riippumaton rajapinta sovelluksista tietokantoihin

OLAP (Online Analytical Processing)
- moniulotteisen tiedon käsittely ja analysointi

OQL (Object Query Language)
- oliotietokantojen SQL:ään pohjautuva kyselykieli

paikallisvarasto (data mart)
- tietovarastosta luotu pienempi tietokanta tiettyyn analysointitarpeeseen

SQL (Structured Query Language)
- relaatiotietokantojen kyselykieli

TAPAS (Text and program analysis)
- TKK:n ja Joensuun yliopiston yhteistyöprojekti, jossa kehitetään ohjelmointiin liittyvien tekstien automaattista arviointia

XQuery
- XML-tietokantojen kyselykieli

1 Johdanto

Nykyään yritykset ja muut organisaatiot keräävät valtavasti tietoa asiakkaistaan ja toiminnastaan. Tiedon suuresta määrästä ja hajanaisuudesta johtuen syntyy tarve tiivistää ja koostaa tietoa, jotta sitä voidaan käyttää hyväksi esim. toiminnan kehittämiseen. Monesti tätä ei kuitenkaan ymmärretä tai osata tehdä, ja data jää lojumaan hyödyttömänä. Lisäksi tiedon keräämistä ei ole usein suunniteltu riittävän hyvin. Tiedon hyödyntämisellä voitaisiin kuitenkin saavuttaa kilpailuetua tai kehittää toimintaa.

1990-luvulla on kehittynyt tietovarastoinnin (data warehousing) käsite, jossa pyritään juuri datan koostamiseen yhteen paikkaan analysointia ja raportointia varten. Keskenään erilaisista operatiivisista järjestelmistä kerätään hyödyllistä dataa yhteen paikkaan, josta voidaan esim. automaattisesti koota raportteja johdolle tai johon voidaan tehdä toimintaa analysoivia kyselyjä.

Eräitä sovellusalueita datan keräämiselle ovat tieteellinen tutkimus ja tilastollinen analyysi. Yliopistoissa ja muissa tieteellisissä laitoksissa kertyy paljon tutkimustietoa, jonka tehokkaaseen hyödyntämiseen tarvitaan edistyneitä menetelmiä. Tässä työssä pyritään luomaan edellytykset erään tieteellisen tutkimus- ja opetusyhteisön datan hyödyntämiselle.

1.1 Tutkimuksen tarkoitus

Teknillisessä korkeakoulussa käytetään automaattisia tarkastusjärjestelmiä tietotekniikan opetuksessa. Ohjelmoinnin sekä tietorakenteiden ja algoritmien peruskursseilla käy vuosittain satoja opiskelijoita, ja erilaisia järjestelmiä on käytetty jo yli kymmenen vuoden ajan helpottamaan henkilökunnan työtaakkaa ja parantamaan opetuksen laatua.

Tarkastusjärjestelmiin liittyy tietokantoja, joihin kerätään sekä opiskelijoiden palauttamia harjoitustehtävävastauksia että arvioinnin tuloksia. Kussakin järjestelmässä tallennetaan hieman eri tietoja ja erilaisiin tietokantoihin. Tätä dataa on aikaisemmin analysoitu järjestelmäkohtaisesti opetusmenetelmien

kehittämiseksi (esim. Torvinen 2003; Malmi & Korhonen 2004). Nyt on havaittu tarve verrata kursseja keskenään. Tällöin voidaan esim. selvittää, vaihteleeko opiskelijoiden palautuskäyttäytyminen eri kursseilla. Nykyiset järjestelmät ovat kuitenkin niin erilaisia keskenään, että vertailu suoraan olisi hyvin työlästä.

Tutkimuksen tarkoituksena on luoda yhteinen tietovarasto, johon voidaan kerätä eri järjestelmien keräämä data yhtenäisessä muodossa. Tämän pohjalta dataa voidaan analysoida, jolloin opetus- ja arviointimenetelmiä pystytään kehittämään edelleen. Alkuvaiheessa tietovarastoon kerätään muutamien TKK:n Ohjelmistotekniikan laboratorion pitämien kurssien vanhaa ja nykyistä dataa. Myöhemmin siihen liitetään yhteistyöyliopistojen vastaavia järjestelmiä sekä tulevan tutkimustyön tuloksena syntyviä tarkastimia. Samalla luodaan katsaus tietovarastointiin yleisesti, sillä laboratoriossa ei ole ennestään juurikaan osaamista aiheesta.

Työ on osa TAPAS-projektia (Text and program analysis), joka on yhteistyöprojekti Teknillisen korkeakoulun ohjelmistotekniikan laboratorion COMPSER-tutkimusryhmän ja Joensuun yliopiston EdTech-tutkimusryhmän välillä. Projektin tarkoituksena on kehittää uusia automaattisia tai puoliautomaattisia tarkastusjärjestelmiä ohjelmoinnin opetukseen. Tällaisia voivat olla mm. ohjelman dokumentoinnin tai pseudokoodin tarkastaminen. Näissä yhdistyvät TKK:n kokemus ohjelmointitehtävien automaattisesta tarkastamisesta ja Joensuun yliopistossa kehitetyt vapaan tekstin tarkastusmenetelmät.

1.2 Tutkimusongelma ja rajaukset

Jokainen tietovarastoprojekti on yksilöllinen, sillä aihealue, käyttötarkoitus, käytetyt sovellukset ja monet muut tekijät vaihtelevat usein paljonkin. Tietovaraston pystyttäminen on yleensä myös monimutkainen prosessi kokonaisuudessaan. Siksi järjestelmän luonti itsessään on jo oikea tutkimusongelma.

Yleensä tietovarastointia käytetään suurissa kaupallisissa ja julkisissa organisaatioissa, ja sillä halutaan saada aikaan automaattisia raportteja sekä taloudelliseen hyötyyn tähtääviä analyyseja. TAPAS-projekti keskittyy tieteelliseen tutkimukseen, jonka perimmäinen tarkoitus on opetus- ja arviointimenetelmien kehittäminen. Tarkoituksen lisäksi myös aihealue ja projektin koko ovat poikkeuksellisia. Pieni koko näkyy datan suhteellisen vähäisessä määrässä, tietovarastoa hyödyntävän organisaation koossa sekä käyttäjien lukumäärässä ja taustoissa. Voidaan siis todeta, että tässä työssä on lukuisia erityispiirteitä, jotka saattavat vaikuttaa työn aikana tehtäviin ratkaisuihin, kuten analysointityökalun valintaan tai tietokannan suunnitteluun.

Tutkimusongelma voidaan kiteyttää seuraavalla tavalla:

Miten voidaan rakentaa tietovarasto automaattisten tarkastusjärjestelmien keräämien tietojen yhdistämiseen?

Tämä voidaan pilkkoa pienempiin kysymyksiin:

- Millainen prosessi tiedon kerääminen lähdejärjestelmistä on?
- Millaisia valmiita tietovarastointisovelluksia on olemassa ja voitaisiinko niitä hyödyntää?
- Mikä tietokanta- ja muu teknologia soveltuu tehtävään parhaiten?

Kysymysten asettelusta nähdään, että työn pääpaino on tietovaraston luomisprosessissa. Tärkeää on siis datan kerääminen ja ns. ETL-prosessi, joka selitetään tarkasti työn teoriaosassa. Vähemmälle huomiolle jää, millainen lopullinen tietovarasto on ja millä keinoin sitä hyödynnetään. Tämä on siinä mielessä poikkeuksellinen lähestymistapa, että tietovaraston luomisen varsinainen syy ovat lähtökohtaisesti loppukäyttäjien tarpeet. Syyt painotukselle perustuvat pitkälti työn erityispiirteisiin.

Seuraavassa on mainittu rakennettavan tietovaraston tärkeimpiä ominaisuuksia. Ne tarkentuvat myöhemmin varsinkin vaatimusmäärittelyn yhteydessä.

Ensinnäkin käytettävät sovellukset pyritään rajaamaan avoimen lähdekoodin ohjelmiin. Toiseksi lähdedatasta pitää pystyä erottamaan oleellinen osa ja muuntamaan se yhteiseen muotoon muiden lähteiden kanssa. Tämä datan siirtäminen lähteistä tietovarastoon pitää olla helposti määriteltävissä, toistettavissa ja automatisoitu mahdollisimman pitkälle. Jälkikäteen dataa tulee voida hakea tietovarastosta nopeasti ja monipuolisesti eri lähteitä yhdistäen. Tulevaisuuden tarpeita ajatellen tietovaraston täytyy olla lisäksi helposti laajennettavissa ja muokattavissa.

Tietovarastointiprojektit ovat yleensä työläitä, joten tätä työtä on rajattu seuraavin tavoin. Ensinnäkin analysointipuoli jätetään vähemmälle huomiolle. Dataa on tärkeää pystyä hakemaan ja hyödyntämään tietovarastosta, mutta tähän ei pyritä tarjoamaan kovin hienostuneita välineitä tämän työn puitteissa. Toiseksi tietovarastoon kerätään vain muutaman kurssin vanha ja nykyinen data. Laajennettavuus ja tiedossa olevien muiden datalähteiden tarpeet otetaan silti suunnittelussa huomioon.

1.3 Työn rakenne

Työssä kuvataan ensin tarvittava teoria ja sen jälkeen toteutuksen eri vaiheita. Luku 2 esittelee tietovarastoinnin teoriapuolen. Itse tietovarastoinnin lisäksi käydään läpi tietokantatekniikoita sekä aikaisempia tutkimustuloksia. Teoria keskittyy tässä työssä tarvittaviin osiin, eikä aivan kaikkia tietovarastoinnin osa-alueita siten käydä läpi.

Luvut 3–6 käsittelevät toteutuksen osia. Luku 3 esittelee alkutilanteen ja lähtökohdat sekä vaatimukset toteutettavalle järjestelmälle. Luvussa 4 käydään läpi tietovaraston rakentamisprosessia yleisesti, ja muuten luku keskittyy tämän työn teknologiavalintoihin. Luku 5 kuvaa järjestelmän arkkitehtuurin ja toteutuksen. Tätä analysoidaan luvussa 6 peilaamalla järjestelmää asetettuihin vaatimuksiin ja tutkimuskysymyksiin.

Työn tärkeimmät asiat vedetään yhteen luvussa 7. Lopuksi luvussa 8 esitetään jatkokehitysehdotuksia järjestelmälle.

Tietovarastoinnista ei ole olemassa kovin paljon suomenkielistä kirjallisuutta. Tähän työhön sanastoa on otettu erityisesti Hovin (1997) teoksesta, mutta joitakin termejä on suomennettu itse. Tämän vuoksi on mahdollista törmätä samoihin sanoihin eri nimillä muissa yhteyksissä. Käsitteiden esittelyn yhteydessä on kuitenkin annettu englanninkielinen vastine väärinkäsitysten välttämiseksi.

2 Tietovarastoinnin teoria

Suurilla yrityksillä ja organisaatioilla on usein lukuisia tietokantoja, jotka sisältävät tietoa toiminnan eri osa-alueilta. Näitä voivat olla mm. kirjanpito, asiakastietorekisteri ja varastonhallintajärjestelmä. Kukin tällainen tiedonhallintajärjestelmä toimii tyypillisesti itsenäisesti, ja ne on suunniteltu vain kyseiseen tarkoitukseen erillään muista. Kun tietoa on tarpeen siirtää toiseen järjestelmään, tämä tehdään joko käsin tai tarkoitukseen luodaan jokin yksinkertainen automaattinen työkalu. Tiedot ovat kuitenkin niin hajallaan, että kokonaiskuvaa kaikesta on lähes mahdotonta saada matalan tason datasta. Johtajat joutuvat turvautumaan kausittain saataviin raportteihin eri järjestelmistä, mutta nekin ovat keskenään täysin erillisiä, ja yhtenäistä kokonaiskäsitystä voi olla vaikea muodostaa.

Kyse on siis ennen kaikkea yritysjohdon tarpeesta kaupallisessa maailmassa. Kilpailun kiristyessä tarve ajantasaisen ja tarkan tiedon saamiseen on kasvanut. Myöhemmin samankaltaisia tarpeita on havaittu muissakin organisaatioissa ja yhteyksissä, mistä tämä diplomityö on hyvänä osoituksena.

Ongelmaa ratkaisemaan kehittyivät ja yleistyivät 1990-luvulla tietovarastot (data warehouse), joiden teoriaa käydään läpi tässä luvussa. Liikkeelle lähdetään kuitenkin erilaisten tietokantatekniikoiden esittelystä, sillä tietokannat toimivat tietovarastojen pohjana, ja sopivan tekniikan valinta on myös tämän työn ensimmäisiä ratkaistavia kysymyksiä. Tämän jälkeen tietovarastointia käsitellään ensin yleisesti ja sen jälkeen pureudutaan tarkemmin tärkeimpiin alueisiin. Luvun lopussa käydään lyhyesti läpi aiempaa tutkimusta aiheesta. Esitettävä teoria pysyttelee perusasioissa eikä pureudu tutkimuksen erityiskysymyksiin, sillä ne eivät ole juurikaan oleellisia tämän työn kannalta. Niinpä useissa aliluvuissa pääasiallisena lähteenä ovat oppikirjat. Valittu näkökulma painottaa teknistä puolta ja pyrkii siinä käytännönläheisyyteen, ja mm. käyttäjien ja ”liiketoiminnan” tarpeet jäävät vähemmälle huomiolle.

2.1 Tietokantatekniikat

Erilaisia tapoja toteuttaa tietokanta on lukuisia. Relaatiotietokannat ovat muodostuneet selvästi yleisimmäksi ratkaisuksi, mutta aikojen kuluessa monia muitakin tapoja on kehitetty ja kehitetään edelleen vaihtelevalla menestyksellä. Tässä käydään läpi tämän hetken yleisimpiä tekniikoita, jotka ovat yksinkertainen hakemistopuu, relaatiotietokanta, oliotietokanta sekä XML-tietokanta. Lukijan odotetaan tuntevan perusasiat tietokannoista ennestään. Aiheesta on lukuisia Internet-sivustoja sekä oppikirjoja (esim. Silberschatz *ym.* 2006; Ullman & Widom 2001). Tässä pyritään keskittymään vain tekniikoiden tärkeimpiin ominaisuuksiin ja keskinäisiin eroihin. Mainittujen tekniikoiden lisäksi on olemassa erityisiä moniulotteisia tietokantoja, mutta ne joko ovat kaupallisia tai niitä ei ole saatavilla halutuille käyttöjärjestelmille. Lisäksi ne rajoittavat datan hyväksikäyttömahdollisuuksia enemmän erikoistuneisuutensa vuoksi. Moniulotteisista tietokannoista kerrotaan lyhyesti kohdassa 2.8.

Hakemistopuu

Hakemistopuun mainitseminen tietokantaratkaisuna saattaa kuulostaa yllättävältä. Käytännössä kuitenkin monet pienet tiedon säilyttämistarpeet voidaan toteuttaa ilman monimutkaisia, oikeita tietokantasovelluksia. Hakemistopuulla tarkoitetaan tässä käyttöjärjestelmän tiedostojärjestelmää, johon tieto tallennetaan erillisiin tiedostoihin ja hakemistoihin. Erikoistapauksena tieto voi sijaita kokonaan yhdessä tiedostossa, jonka sisäinen rakenne voi olla mielivaltainen. Esimerkkinä tällaisesta on XML-tiedosto, jossa data jäsennetään tiedoston sisällä puun muotoon.

Lisäykset, päivitykset ja haut on käytännössä toteutettava itse ohjelmoimalla tai valmiilla sovelluksella, sillä valmiina tarjolla ovat vain käyttöjärjestelmän omat, melko matalan tason operaatiot. Toteutus on yleensä kuitenkin melko suoraviivaista, sillä tarpeet ovat usein suhteellisen vähäiset. Toisaalta erikoisemmassa projektissa ohjelmoimalla voidaan toteuttaa harvinaisempia ominaisuuksia, joita kehittyneetkään tietokantasovellukset eivät tarjoa.

Oikeisiin tietokantoihin verrattuna puutteita on melko paljon, minkä vuoksi näin yksinkertaista ratkaisua voi käyttää vain hyvin rajatuissa tilanteissa. Tiedosta saattaa olla olemassa useita kopioita eri paikoissa, jolloin kaikki versiot eivät ole välttämättä ajan tasalla, ja päivittäminen on monimutkaisempaa. Myös datan yhtäaikainen käyttö saattaa aiheuttaa odottamattomia ongelmia. Koska tiedon sisältö ja rakenne eivät mitenkään määräydy järjestelmän mukaan, voivat eri tiedostot esittää tiedon keskenään yhteensopimattomissa muodoissa. Nämä kohdat liittyvät melko vakavaan puutteeseen, ns. ACID-ominaisuuksien puuttumiseen. Niitä pidetään luotettavan tietokannan perusedellytyksinä, ja ne on kuvattu taulukossa 1. Nämä kaikki ovat vain muutamia esimerkkejä mahdollisista ongelmista.

Taulukko 1. Tietokantojen ACID-ominaisuudet

Ominaisuus	Kuvaus
Atomisuus (Atomicity)	Kukin tietokantaan suoritettava operaatio tehdään kokonaan tai ei ollenkaan. Operaation epäonnistuessa palataan tilaan ennen sen suorittamista tai yritetään uudestaan.
Konsistenssi, eheys (Consistency)	Data on aina eheää eli vastaa tietokannan määrityksiä ja rajoitteita. Muutokset vievät tietokannan eheästä tilasta eheään.
Isolaatio, eristyneisyys (Isolation)	Eri käyttäjien samanaikaiset operaatiot eivät häiritse toisiaan.
Pysyvyys (Durability)	Tietokantaan tehdyt muutokset eivät katoa ongelmatilanteissa, vaan ne voidaan palauttaa esim. lokien tai varmuuskopioiden avulla.

Relaatiotietokanta

Relaatiotietokannat kehitettiin käytännössä 1970-luvulla, mutta ne yleistyivät vasta 1980-luvun aikana, kun teknologia kehittyi tarpeeksi nopeaksi (Silberschatz *ym.* 2006, 28–29). Ne ovat olleet kaikkein yleisin tietokantatekniikka jo pitkään. Tietokannan looginen rakenne noudattaa relaatiomallia, ja haut ja päivitykset tehdään tehokkaalla SQL-kyselykielellä.

Relaatiotietokannoissa taulut normalisoidaan yleensä vähintään kolmanteen normaalimuotoon, jotta päällekkäistä dataa ei tallentuisi. Tämä helpottaa päivitysten tekemistä ja tiedon säilyttämistä eheänä. Tietovarastoinnissa normaalimuotovaatimuksista usein poiketaan, joten taulukossa 2 kerrataan

kolmen ensimmäisen normaalimuodon määritelmät. Näiden lisäksi on olemassa neljä tiukempaa normaalimuotoa, joiden merkitys on vähäisempi.

Taulukko 2. Relaatiomallin kolme ensimmäistä normaalimuotoa

Normaalimuoto	Selitys
1. (1NF)	Taulun riveillä ei saa olla toistuvia tai moniarvoisia attribuutteja.
2. (2NF)	Ensimmäisen normaalimuodon vaatimus, ja lisäksi taulun attribuuttien on oltava riippuvaisia kaikista taulun avaimista.
3. (3NF)	Toisen normaalimuodon vaatimukset, ja lisäksi taulun attribuutit eivät saa riippua muista attribuuteista kuin avaimista.

Tekniikalla on melko vähän heikkouksia. Relaatiotietokannoissa on rajoittuneet (ja oliokielten kanssa erilaiset) tietotyypit, joten niiden käsittely ohjelmointikielistä käsin voi aiheuttaa ylimääräistä vaivaa. Ne kuitenkin mahdollistavat tehokkaan optimoinnin ja suojaavat paremmin ohjelmointivirheiltä. Malli on skeeman suhteen melko joustamaton, eli datalähteestä riippumatta jokaisella taulun rivillä on kaikki samalla tavoin määritetyt attribuutit. Niinpä voi syntyä paljon vähän käytettyjä kenttiä, joita käyttää hyväksi vain murto-osa lähdejärjestelmistä. Monissa tietokantatuotteissa tämä ei ole varsinainen ongelma, sillä vaihtelevanpituisista kentistä tyhjiä kohtia varten ei tarvitse varata tilaa.

Vahvuuksia sen sijaan on useita. Tekniikka on kehittynyt hyvin pitkälle, ja erilaisia työkaluja ja muuta tukea on valtavasti. JDBC- ja ODBC-liitännät mahdollistavat tietokantojen käsittelyn muista sovelluksista helposti. Pitkällisen kehityksen tuloksena erikoisetkin haut on optimoitu suhteellisen nopeiksi. Suurin osa tietovarastoista käyttääkin nykyään relaatiotietokantaa pohjanaan.

Oliotietokanta

Oliotietokannat syntyivät 1980-luvulla (Bancilhon 1992). Kuten edellä todettiin, oli joissakin sovelluksissa relaatiotietokantojen kanssa ongelmina rajoittuneet tietotyypit ja tietokannan käsittely ohjelmointikielestä käsin. Oliotietokannan rakenne perustuu oliomalliin kuten olio-ohjelmointikielissä, ja tekniikkaa käytetäänkin nimenomaan tällaisten kielten kanssa. Tieto tallennetaan olioina, joita voi manipuloida vastaavan luokan metodeilla. Kyselyt tehdään SQL:ään

pohjautuvalla OQL-kielellä. On myös olemassa olio- ja relaatiotietokantojen välimuoto, oliorelaatiotietokanta (object-relational database), joka lisää relaatiokantaan olio-ominaisuuksia kuten itse määriteltäviä tietotyyppejä (Silberschatz 2006, 361).

Olioista on suoria viittauksia toisiin olioihin, ja näitä käytetään myös tietokannan hauissa, jolloin haut ovat nopeampia ja relaatiokannoissa käytettäviä taululiitoksia ei tarvita. Vastaavasti ennalta määrittelemättömät haut, joihin olioissa ei ole valmiita suhteita, ovat selvästi hitaampia. Tästä seuraa, että oliotietokannat ovat tehokkaampia kuin relaatiotietokannat tietynlaisissa sovelluksissa. Ne eivät ole kuitenkaan yleistyneet yhtä laajalti.

Varsinaiset oliotietokannat vaativat olio-ohjelmointikieleltä persistenttiä tiedonkäsittelykykyä, eli ohjelmointikielen on pystyttävä suoraan manipuloimaan tietokannan pysyviä tietorakenteita (Silberschatz *ym.* 2006, 379–380). Tavallisestihan ohjelmointikielet säilyttävät dataa vain väliaikaisessa muistissa ohjelman suorituksen ajan ja unohtavat sen jälkeen kaiken.

XML-tietokanta

XML:ää ei alun perin tarkoitettu tietokantatekniikaksi. Se on kuitenkin yleistynyt niin paljon tiedon tallennus- ja siirtotekniikkana, että viime vuosina on jouduttu pohtimaan, kuinka XML-dataa voitaisiin parhaiten käsitellä tietokannoissa. (Silberschatz 2006, 395, 408–409.)

Tekniikan voi jakaa natiiveihin XML-tietokantoihin ja relaatiotietokantojen päälle rakennettuihin XML-ominaisuuksiin (XML enabled database). Natiivin tietokannan määritelmä vaihtelee jonkin verran, mutta se perustuu aina tiedon tallennus- tai käsittelytapaan (Westermann & Klas 2003). Natiiveissa tietokannoissa kyselyt suoritetaan XQuery-kielellä (W3C 2006), jonka hakulauseiden FLWOR-rakenne (for-let-where-order by-return) vastaa karkeasti SQL:n select-from-where -rakennetta. Kieli käyttää hyväkseen Xpath-määritystä (W3C 1999), jolla voidaan helposti viitata XML-dokumentin eri osiin. Fyysisesti

data voidaan tallentaa tavallisiin XML-tiedostoihin, erityiseen XML-tietokantaan tai relaatiotietokantaan. Lisäksi XML Schemalla (W3C 2004) voidaan määrittää relaatiomallin mukaisia rakenteita, mikä mahdollistaa rajoitetusti relaatiotietokannan skeeman muuntamisen XML-muotoon.

Relaatiotietokantaan sulautettuna tieto voidaan esim. tallentaa kokonaisina XML-tiedostoina, ja varsinaiset haut tehdään näiden sisältä. SQL:lle on olemassa laajennus SQL/XML, jossa kyselyjä voidaan ulottaa XML-dokumenttien sisälle. Uusissa relaatiojärjestelmissä voi olla myös oma tietotyyppi XML-dokumenteille. (Silberschatz 2006, 422–428.)

XML:n käyttö tietokannoissa on edelleen melko tuore ilmiö, ja siksi tekniikka ei ole kehittynyt yhtä pitkälle kuin vaikkapa relaatiotietokannat. Niinpä sovellukset eivät ole vielä täysin kypsiä, ja niihin tulee edelleen merkittäviäkin muutoksia. Edes W3C:n XQuery 1.0 ei ole edennyt vielä lopulliseksi suositukseksi, vaikka tämä tapahtuneekin lähitulevaisuudessa (Eisenberg & Melton 2005).

2.2 Mikä on tietovarasto

Tietovaraston käsitteen esitteli ensimmäisen kerran Bill Inmon 1990-luvun alussa. Hän määrittelee sen seuraavasti:

> A data warehouse is a subject-oriented, integrated, nonvolatile, and time-variant collection of data in support of management's decisions. (Inmon 2002, 31.)

Toisena tietovarastoinnin isähahmona pidetään Ralph Kimballia, jonka määritelmä on seuraava:

> A data warehouse is a system that extracts, cleans, conforms, and delivers source data into a dimensional data store and then supports and implements querying and analysis for the purpose of decision making. (Kimball & Caserta 2004, 23.)

Lyhyesti sanottuna kyseessä on siis kokoelma tietoa, jonka on tarkoitus tukea päätöksentekoa. Näissä kahdessa näkemyksessä on kuitenkin jonkin verran painotuseroa. Inmon luettelee tietovaraston tärkeimpiä ominaisuuksia, joita

hänen mielestään ovat siis keskittyminen tiettyyn aiheeseen eikä yrityksen toimintoihin yleisesti, datan kokoaminen yhtenäiseksi kokonaisuudeksi erilaisista lähteistä, datan muuttumattomuus tietovarastossa sekä aikaulottuvuuden ottaminen mukaan dataan. Kimballin lähestymistapa on teknisempi ja keskittyy enemmän tietovaraston luomisprosessiin, sillä hän mainitsee sekä varaston muodostamisen vaiheet että lopullisen tietokannan suositellun muodon, dimensiomallin. Molemmat silti painottavat valmiin tietovaraston perimmäistä tarkoitusta johdon päätöksenteon apuvälineenä. Voidaan jopa ajatella, että jos lähtökohtana eivät koko ajan ole loppukäyttäjien tarpeet, on tietovarastoprojekti turha (Törmänen 1999, 28).

Tietovarastolla halutaan koota erilaisten operatiivisten järjestelmien keräämä data samaan paikkaan ja yhtenäiseen muotoon, jotta sitä voitaisiin tehokkaasti hyödyntää. Pitkään on nimittäin ollut vallalla taipumus kerätä kaikenlaista tietoa asiakkaista, valmistusprosesseista, markkinoinnista, kilpailijoista ja monista muista asioista. Tätä tietoa ei ole kuitenkaan osattu hyödyntää kunnolla, ja se on jäänyt monissa yrityksissä ja organisaatioissa lojumaan unohduksiin magneettinauhojen ja muiden tallentimien syövereihin. Nämä operatiiviset tietokannat on yleensä luotu toisistaan riippumattomasti ja hyvin vaihtelevin ratkaisuin. Tällöin puhutaan heterogeenisista tietokannoista, joiden ominaisuuksia tarkastellaan lähemmin luvussa 2.4. Kaiken tämän heterogeenisen datan tuominen yhteen paikkaan yhtenäisessä muodossa on tietovaraston luomisen selvästi työläin vaihe, ETL-prosessi, johon tutustutaan luvussa 2.5. Valmiista tietovarastosta saatavia hyötyjä ovat mm. parempi täsmämarkkinointi, toimintojen tehostaminen, asiakaspalvelu ja nopeampi tiedon saanti (Hovi 1997, 4). Yksi niiden vaikutus on myös datan hyödyntämisen muuttaminen itsepalvelun suuntaan, eli sen sijaan että IT-ammattilaiset tuottaisivat raportteja käyttäjille, käyttäjät suorittavat itse hakuja suoraan tietovarastoon (Moody & Kortink 2000).

Koska tietovarasto on tietokokoelma, on luontevaa rakentaa se tietokannan päälle, ja näin tehdäänkin käytännössä aina. Tällöin voidaan kysyä, miten se

eroaa tavallisesta tietokannasta. Koska data kerätään useista lähdejärjestelmistä yhteen paikkaan, on tietovaraston koko keskimäärin selvästi suurempi kuin operatiivisen tietokannan, yleensä gigatavujen mutta joskus jopa teratavujen luokkaa (Hovi *ym.* 2001, 48). Data kerätään varastoon usein suurina eräajoina, joissa operatiivisten järjestelmien muutokset edellisen tietovarastopäivityksen jälkeen tulevat kaikki kerralla mukaan. Päivitykset ovat siis harvinaisia mutta suuria. Haut tietovarastoon eroavat tavallisesta puolestaan siinä, että ne voivat olla hyvin vaihtelevia, mielivaltaisia ja monimutkaisia. Tästä seuraa, että hakutehokkuus on tärkeää, mutta päivitystehokkuus ei. Taulukko 3 kokoaa yhteen tietovaraston ja operatiivisen tietokannan eroja. Osa tekijöistä tulee tarkemmin esiin myöhemmin tässä luvussa.

Taulukko 3. Operatiivisen tietokannan ja tietovaraston tärkeimpiä eroja

Tekijä	Operatiivinen tietokanta	Tietovarasto
kokoluokka	megatavuja/gigatavuja	yleensä gigatavuja
käyttöaste	suuri	matala, purskeinen
päivitykset	paljon yksittäisiä päivityksiä hajallaan	päivitykset harvinaisia ja ne suoritetaan suurina eräajoina
käyttäjien operaatiot	sekä luku- että kirjoitusoperaatioita	yleensä vain lukuoperaatioita
ajantasaisuus	aina ajan tasalla	päivitetään harvoin, tasaisin jaksoin, esim. kerran päivässä
tiedon toisteisuus	normalisoitu tieto eli ei toisteisuutta	denormalisoitu tieto eli osa datasta toistuu; lisäksi valmiiksi laskettua summatietoa
tiedon ikä	lähinnä tuoretta tietoa	sekä tuoretta että historiatietoa

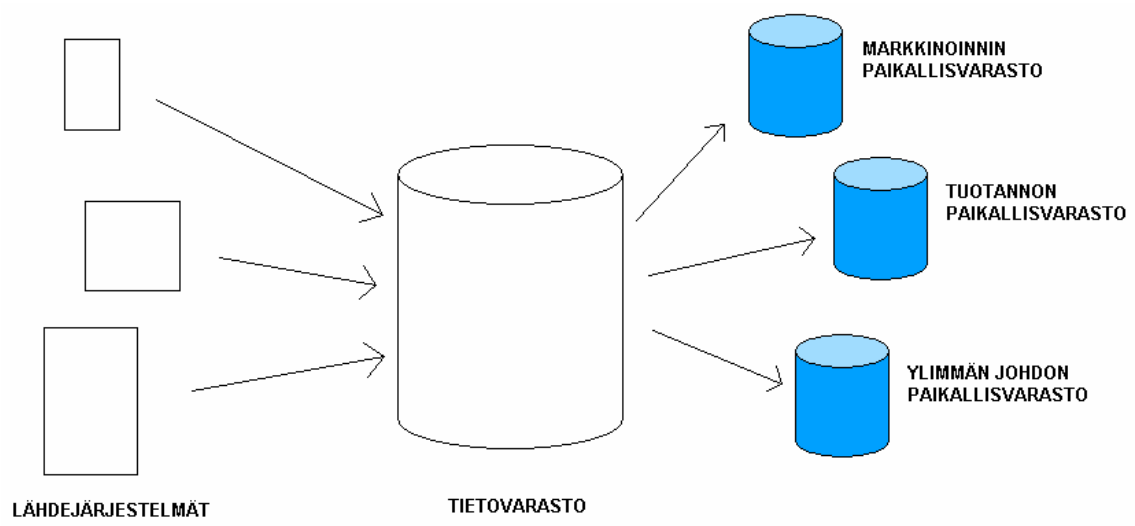
Lähde: Hovi ym. *2001, 46–48*

Tietovaraston voi jakaa piilotettuun ja julkiseen osaan. Piilossa olevaan osaan kuuluu erityisesti ETL-prosessi eli tiedon tuottaminen tietovarastoon, ja loppukäyttäjän ei pitäisi tietää tästä osasta mitään. Se on siis kuten muiden ATK-järjestelmien konehuone. Julkisella osalla tarkoitetaan käyttäjän näkemää puolta, joka muodostuu erityisesti valmiista tietovarastosta sekä analysointi- ja muista hyödyntämistyökaluista. Näistä kerrotaan luvussa 2.8. Erikoistapaus tietovarastosta (tai oikeastaan sen julkisesta osasta) on paikallisvarasto, joka määritellään seuraavaksi.

2.3 Paikallisvarasto

Paikallisvarasto eli data mart on erikoistunut versio tietovarastosta. Valmiista tietovarastosta (ns. keskusvarastosta) voidaan muodostaa pienempiä, esim. osastotason kantoja, joihin voidaan rajata tiedon määrää vain tiettyyn erityistarpeeseen ja laskea helpommin pidemmälle vietyjä tunnuslukuja valmiiksi (Hovi 1997, 35–36). Hyötynä ovat nopeammat haut, kun paikallisvarastot toteutetaan eri paikkaan ja ovat kooltaan selvästi pienempiä. Ne on myös mahdollista toteuttaa vain näkyminä alkuperäisestä tietovarastosta (Moody & Kortink 2000).

Kuvassa 1 on esimerkki siitä, miten tieto kulkee paikallisvarastoihin, jos sellaisia päätetään toteuttaa. Paikallisvarastoja ei siis yleensä luoda suoraan, vaan vasta tietovarastoon kerätyn datan kautta.



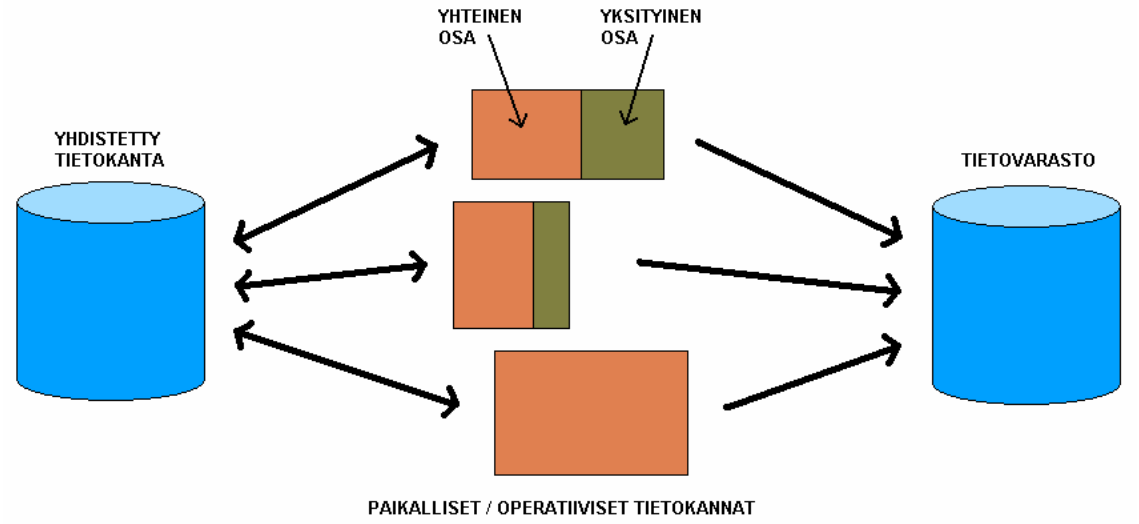
Kuva 1. Data kulkee paikallisvarastoihin keskusvaraston kautta.

Kimball ja Caserta (2004, 20–21) huomauttavat myös, että paikallisvarastojen jaottelun tulee perustua datalähteisiin eikä siihen, miten data yrityksessä nähdään. Lisäksi paikallisvarastojen pitää sisältää myös atomisen tason data, vaikka siitä laskettaisiinkin pitkälle vietyjä summa-arvoja. Muussa tapauksessa niiden suunnittelija tekisi oletuksia käytetyistä kyselyistä, vaikka kyselyiden tulisi voida olla täysin mielivaltaisia.

2.4 Heterogeeniset tietokannat

Heterogeenisiksi tietokannoiksi kutsutaan järjestelmiä, joiden sisältämä data (tai järjestelmä itse) on keskenään erilaista. Tyypillisesti ainakin tietomallit poikkeavat toisistaan (Thomas *ym.* 1990). Koska tiedonhallintajärjestelmiä luodaan eri aikoina ja eri tarkoituksiin, voi saman organisaation sisällekin kertyä vuosien saatossa useita heterogeenisia järjestelmiä, joita ei ole alun perin ymmärretty suunnitella yhtenäisiksi. Myöhemmin esim. tietovarastoprojektin yhteydessä tämä monimuotoisuus kostautuu, kun dataa halutaan integroida.

Integroinnin yhteydessä tietokannan datan voi kuvitteellisesti jakaa kahteen osaan. Toisaalta on yhteinen osa, jonka data halutaan yhdistää muiden lähteiden kanssa, ja toisaalta yksityinen osa, josta integroinnissa ei olla kiinnostuneita. Osat voivat luonnollisesti olla suhteessa toisiinsa kuinka suuria hyvänsä riippuen siitä, kuinka suuri osa tietokannasta on yhdistämisen kannalta hyödyllistä, ja ääritapauksessa yksityistä osaa ei ole ollenkaan. Yksityisen osan datan ongelmista ei tarvitse välittää ollenkaan, paitsi jos tarkoituksena on yhdistetyssä järjestelmässä tehdä päivityksiä alkuperäisiin järjestelmiin. Tällaisessa yhdistetyssä tietokannassa (federated database) pyritään tarjoamaan käyttäjälle yksi yhteinen rajapinta kaikkiin lähdejärjestelmiin. Tietovarastoinnissa sen sijaan pitäydytään datan lukemisessa lähdetietokannoista. Kuvassa 2 nähdään tämä ero ja lähteiden jakautuminen kahteen osaan.



Kuva 2. Dataliikenteen ero yhdistetyssä tietokannassa ja tietovarastossa

Taulukossa 4 on listattu joitakin mahdollisia eroja heterogeenisten tietokantojen välillä. Eri tekijöistä on mainittu myös esimerkit. Taulukosta huomataan, että erot lähtevät laitteistotasolta ja päätyvät siihen, miten yksittäinen data-alkio on tietokantaan tallennettu. Väliin mahtuu sovellustason ja suunnittelun aiheuttamia tekijöitä. Lisäksi yksittäisissä järjestelmissä on tyypillisesti likaista dataa, joka aiheuttaa omat ongelmansa järjestelmien integrointiin. Tällaista dataa muodostuu mm. testiajoista ja datan muuttamisesta käsin.

Nykyään kaikki heterogeenisuustekijät eivät ole varsinaisia ongelmia. Tiedon siirto eri käyttöjärjestelmien välillä on tullut helpoksi, ja monet sovellukset tietokannat mukaan lukien tarjoavat standardirajapintoja tiedon käsittelyyn. Niinpä suurimmaksi hankaluudeksi jäävät suunnittelun hajanaisuudesta johtuvat tekijät, jotka taulukossa 4 ovat alemmassa puoliskossa.

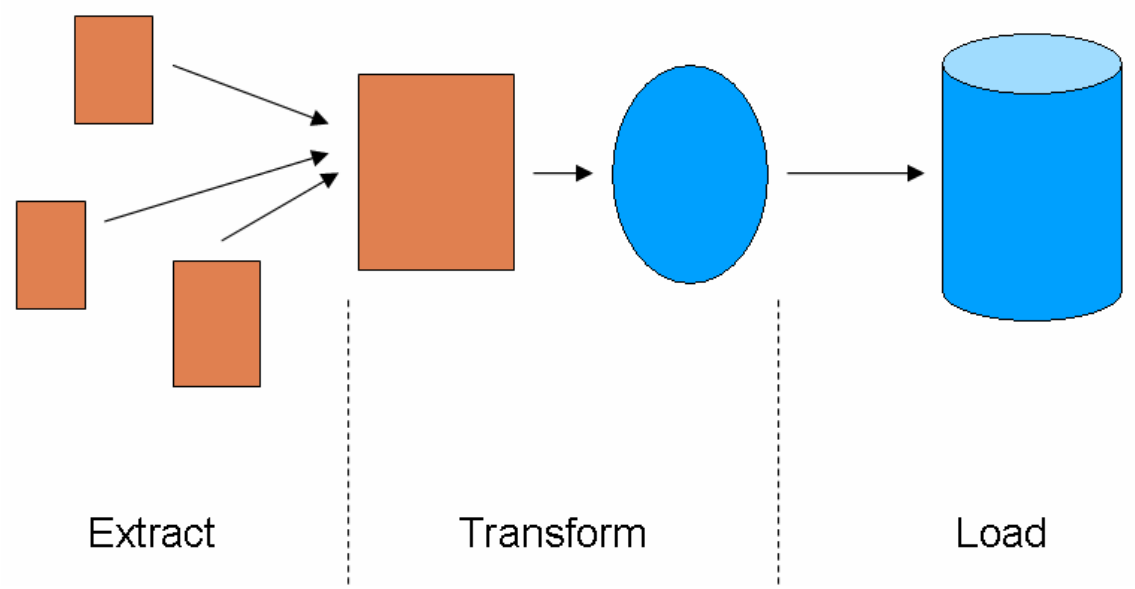
Taulukko 4. Heterogeenisten datalähteiden keskinäisiä eroja

Tekijä	Esimerkki
tietokonearkkitehtuuri ja muu laitteisto	Macintosh – PC
käyttöjärjestelmä	Windows – Linux
tiedonhallintajärjestelmän tyyppi	relaatiotietokanta – oliotietokanta
tietokantasovellus (sisältäen tallennusmuodot, pakkaamisen, protokollat jne.)	MySQL – Oracle
data hajautettu useampaan paikkaan järjestelmän sisällä	tietokanta + tiedostot + lokit
skeema (tietokannan looginen rakenne, taulujen suhteet)	kahden eri suunnittelijan tietokannat ovat käytännössä aina erilaiset skeemaltaan
tietosisältö (osittain)	kirjanpidon ja myynnin tietokannat, yhteisenä tekijänä vain myynnin raha- ja tavaraliikenne
päällekkäinen data (sama data eri tietokannoissa)	henkilötiedot sekä työntekijöiden että asiakkaiden tietokannassa
datan karkeustaso	myynti ilmoitettu päivätasolla – myynti ilmoitettu yksittäisinä myyntitapahtumina
data vain yhdessä tietokannassa	tuotteesta on tietoja myynnin tietokannassa mutta ei varastotietokannassa vaikka pitäisi
sama nimi eri datalla	pid: henkilön (person) tunniste – tuotteen (product) tunniste
eri nimet samalla datalla	henkilötunnus – id
eri avaimet samalla datalla	henkilön tiedot löytyvät yhdestä tietokannasta henkilötunnuksen avulla ja toisesta keinotekoisen työntekijänumeron mukaan
eri tietotyypit samalla datalla (tai muuten eri muoto)	liukuluku – kokonaisluku; amerikkalainen päivämäärä – suomalainen päivämäärä

2.5 ETL

Tietovaraston luomisen ydin on ETL-prosessi. Lyhenne tulee sanoista extract, transform ja load, eli poiminta, siirto ja lataaminen. Lyhyesti sanottuna poiminta tarkoittaa datan lukemista lähdejärjestelmistä, siirto sen siistimistä ja muokkaamista yhtenäiseen muotoon ja lataus sen kirjoittamista tietovarastoon (kuva 3). Toisesta vaiheesta käytetään toisinaan muitakin suomenkielisiä nimityksiä, kuten muokkaus. Näitä vaiheita käsitellään tarkemmin tässä luvussa. ETL:n ohella toinen tietovaraston suuri kokonaisuus on käyttäjälle näkyvä eli julkinen lopputuote ja siihen kerätyn datan hyödyntäminen, ja sitä puolta esitellään erityisesti kohdissa 2.6 ja 2.8. Kuten jo tietovaraston perusperiaatteita esiteltäessä tuli ilmi, ei loppukäyttäjillä tule olla minkäänlaista pääsyä ETL-prosessiin ja sen käsittelyltään keskeneräiseen dataan, ei suoraan eikä analysointityökalujen kautta. Tämä voisi rikkoa datan eheyden, sillä

tietokantaan saattaisi tulla epätoivottuja muutoksia. Lisäksi käyttäjät saattaisivat tietämättään käyttää virheellistä dataa, jota ei ole vielä puhdistettu.



Kuva 3. ETL-prosessin kolme päävaihetta

Tavallisesti tietovarastoa päivitetään tasaisin väliajoin ajamalla ETL-prosessi läpi. Tämä voi tapahtua vaikkapa kerran vuorokaudessa. Koska lähdetietokantojen lukeminen vie suuren osan näiden operatiivisten tiedonhallintajärjestelmien suorituskyvystä prosessin aikana, se pyritään tekemään hiljaisina aikoina, kuten yöllä, tai sitten järjestelmät suljetaan käyttäjiltä siksi aikaa (vrt. huoltokatko). Tästä aiheutuu ETL:lle kaksi tärkeää suorituskykytavoitetta. Ensiksikin datan siirtäminen lähteistä tietovarastoon tulee tapahtua mahdollisimman nopeasti. Toiseksi, koska erilaisia virhetilanteita voi tulla eteen prosessin aikana, on mahdollisimman nopea ja tehokas elpyminen virhetilanteista tärkeää.

ETL-prosessia suunniteltaessa joudutaan alussa tekemään muutamia valintoja, jotka määrittävät koko suunnitteluprosessia pitkälti. On olemassa paljon työkaluja, jotka voivat helpottaa työtä selvästi. Toisaalta koko prosessin toteuttaminen ja ohjelmoiminen itse voi olla jossain tapauksessa järkevää. Taulukossa 5 vertaillaan jotakin valikoituja kummankin lähestymistavan hyötyjä käyttäen lähteenä pääasiassa Kimballia ja Casertaa (2004, 10–12).

Ominaisuuksia on ryhmitelty, ja jokaiselle ryhmälle on annettu arvosana yhdestä kolmeen plusmerkein.

Taulukko 5. ETL:n eri toteutustapojen vertailua

Ominaisuus	Valmis ETL-työkalu	Itse tehty ETL
Käytettävyys	**++**	**+**
nopeus ja yksinkertaisuus	hyvä	heikohko
riippumattomuus ohjelmointitaidosta	useimmilla työkaluilla hyvä	heikko
osaaminen työntekijöiden keskuudessa	heikko	keskinkertainen
Kustannukset	**++(+)**	**++**
hankinta ja alkuvaiheen toteutus	heikohko, paitsi ilmaisilla ohjelmilla erittäin hyvä	hyvä
suuret projektit ja pitkä tähtäin	hyvä	heikohko
Skaalautuvuus ja laajennettavuus	**++**	**++**
suorituskyky suurilla datamäärillä	hyvä	heikohko
toiminnallisuuden laajentaminen omalla ohjelmakoodilla	keskinkertainen	erittäin hyvä
Ominaisuudet	**+++**	**+**
valmiit liitännät moniin tietokantasovelluksiin	erittäin hyvä	keskinkertainen
metadatan tuottaminen automaattisesti	erittäin hyvä	heikko
salaus- ja pakkausominaisuudet	hyvä	heikohko
muutoksenhallintaominaisuudet	hyvä	heikko

Valmiin ETL-työkalun hyödyt riippuvat viime kädessä valitusta sovelluksesta. Ne ovat kuitenkin helppokäyttöisiä, nopeita opetella, suorituskyvyltään hyviä, sisältävät useimpia tarvittavia ominaisuuksia valmiiksi sekä tulevat suurissa ja pitkissä projekteissa helposti halvemmiksi. Sen sijaan toteutettaessa prosessi kokonaan itse työntekijöillä on usein valmiiksi kokemusta ohjelmoinnista ja tietokannoista, ja ETL:n erityisominaisuudet eivät tuo kokeneelle ohjelmistotiimille merkittävästi uutta teknisesti. Aloituskustannukset ovat todennäköisesti pienemmät, ja kokonaiskustannuksetkin jäävät helposti matalammiksi pienissä projekteissa. Vaikka kaiken joutuu periaatteessa tekemään itse, JDBC- ja ODBC-liitännät tietokantoihin sekä muut valmiit kirjastot helpottavat toteutusta jonkin verran.

Yhteenvetona voidaan todeta, että valmiin työkalun hyödyt näyttävät suuremmilta, joten sellaista kannattaa lähtökohtaisesti etsiä ensin. ETL:n

toteuttaminen kokonaan itse sopii lähinnä tilanteisiin, joissa projekti on pieni ja yksinkertainen tai niin poikkeuksellinen, että valmis työkalu ei tarjoa haluttuja ominaisuuksia.

Kun edellä oleva päätös on tehty, päästään käsiksi muihin valintoihin. Yksi päätös on, tehdäänkö tietovaraston lataaminen eräajoina, vai pyritäänkö sitä päivittämään jatkuvasti. Ensin mainittu on selvästi yleisempää, sillä se on teknisesti helpompi toteuttaa ja rasittaa lähdejärjestelmiä vähemmän. Toiseksi on päätettävä, pyritäänkö ETL:n osaprosessit toteuttamaan horisontaalisella vai vertikaalisella tehtäväriippuvuudella (Kimball & Caserta 2004, 14). Horisontaalisuus tarkoittaa, että tehtävät ovat keskenään riippumattomia ja voidaan siten ajaa täysin erikseen. Vertikaalisessa tehtäväriippuvuudessa lähteet synkronoidaan siten, että kaikki lataukset voidaan suorittaa samanaikaisesti. Riippumaton toteutus ei aina ole mahdollinen, sillä osa yksittäisistä tietokannan datariveistä muodostuu helposti useasta lähteestä. Kolmanneksi, koska tietovaraston lataaminen tehdään yleensä säännöllisin väliajoin, tulee helposti kysymykseen sen ajastaminen automaattisesti. Tällöin on selvitettävä, tarvitaanko päivityksiä joka vuorokausi vai harvemmilla väleillä, kuten kuukausittain. Näiden koko ETL-prosessia koskevien tekijöiden jälkeen siirrytään yksittäisiin vaiheisiin aloittaen poiminnasta.

Poiminta

Poiminta on ETL:n ensimmäinen vaihe, ja siinä data kerätään yhteen käsittelyä varten. Usein poiminta tehdään väliaikaisiin tietokantatauluihin, joissa dataa voidaan helposti käsitellä samanaikaisesti. Näin lähdejärjestelmät rasittuvat mahdollisimman vähän, kun niitä joudutaan lukemaan vain kerran. Samoin tekstitiedostot voi olla hyödyllistä lukea tietokantatauluihin, jos myöhemmät vaiheet käsittelevät tietoa helpoiten siinä muodossa.

Ennen kuin varsinaista poimintaa voidaan aloittaa, on kuitenkin päätettävä, mitkä lähdejärjestelmät tietovarastoon otetaan mukaan. Näiden järjestelmien datan laatua on syytä analysoida etukäteen. Näin virheet ja poikkeukset

datassa voidaan dokumentoida ja ottaa huomioon myös ETL:n myöhemmissä vaiheissa. Tähän datan profilointiin on olemassa omat työkalunsa. Samalla voidaan selvittää lähdejärjestelmien muita ominaisuuksia, kuten ER-kaaviot, tiedonhallintajärjestelmät ja ylläpitäjät, sekä luoda datan looginen kuvaus lähteistä tietovarastoon. Nämä ovat tärkeää metadataa, jota kuvataan tarkemmin kohdassa 2.7.

Itse fyysinen poiminta voidaan suorittaa joko täydellisenä tai inkrementaalisesti. Täydellinen poiminta tarkoittaa, että kaikki data luetaan joka kerta ETL-prosessi ajettaessa lähteistä uudestaan. Tämä on luonnollisesti helppo toteuttaa. Toisaalta ylimääräisen datan kerääminen myöhempiin vaiheisiin hidastaa ja monimutkaistaa niitä. Inkrementaalisessa poiminnassa tätä ongelmaa ei ole, sillä lähdejärjestelmistä haetaan vain edellisen kerran jälkeen muuttunut data. Muuttuneen datan tunnistaminen vaatii kuitenkin ylimääräistä vaivannäköä. Keinoja tämän toteuttamiseen ovat mm. aikaleimojen käyttäminen lähdetietokannoissa, muutosten tarkistaminen transaktiolokista sekä vertailu edellisen poiminnan tulokseen (Kimball & Caserta 2004, 106–109). Riippumatta siitä, toteutetaanko poiminta täydellisenä vai inkrementaalisena, on lähteistä ensimmäisellä kerralla joka tapauksessa luettava kaikki data.

Siirto

Datan siirto on ETL-prosessin varsinaisesti lisäarvoa tuottava vaihe. Lyhyesti sanottuna siinä ensin tarkastetaan jokaisen datalähteen oma laatu, ja tämän jälkeen kaikki tieto yhdistetään. Kimball ja Caserta (2004, 17–19) jakavatkin vaiheen kahteen osaan: clean ja conform eli puhdistus ja yhtenäistäminen, ja he puhuvat mieluummin ECCD-prosessista (Extract, Clean, Conform, Deliver), vaikka ETL on käsitteenä muuten vakiintunut.

Puhdistusosassa datan oikeellisuutta tutkitaan usealla tasolla. Yksinkertaisimmalla ja matalimmalla tasolla valvotaan, että yksittäisten sarakkeiden ja kenttien arvot ovat kunnollisia. Tähän sisältyy mm. oikeinkirjoitus, kenttien pituuksien järkevyys, kenttien arvojen mahtuminen

sallittuihin rajoihin ja arvojen olemassaolo kentissä, joissa ei saa olla null-arvoa. Hieman korkeammalla tasolla on taulujen ominaisuuksien eli tietokannan rakenteen valvonta. Tällä tarkoitetaan, että taulun avainten ja viittausten muihin tauluihin täytyy olla kunnossa. Viimeisenä asiana tiedon merkitykseen liittyvien loogisten sääntöjen on pädettävä. Esim. jos opiskelijalla ei ole suoritusmerkintää kaikista kurssin osista, hänellä ei voi yleensä olla kokonaisarvosanaakaan. Puhdistuksen aikana ilmenevässä ongelmatilanteessa voidaan toimenpiteinä joko merkitä virhekenttä ottaen rivi silti mukaan sellaisenaan, hylätä rivi tai keskeyttää koko ETL-prosessi.

Yhtenäistämisosassa kaikki heterogeenisyystekijät pyritään poistamaan saattamalla data yhteen muottiin. Näitä tekijöitä käytiin läpi kohdassa 2.4. Toteutus riippuu täysin lähdedatasta sekä käytetyistä työkaluista, mutta joitakin periaatteita on aina hyödyllistä noudattaa. Jos useista lähteistä saadaan samaksi tunnistettua dataa, valitaan tiedoista luotettavin, esim. tuorein. Jos tietovarastossa on useita faktatauluja (ks. seuraava aliluku), niiden käyttämien samoja attribuutteja sisältävien dimensiotaulujen tulee olla vähintään rakenteeltaan samoja, ja yleensä kyseessä on yksi ja sama taulu. Näin eri faktataulujen tietoja pystytään yhdistämään toisiinsa.

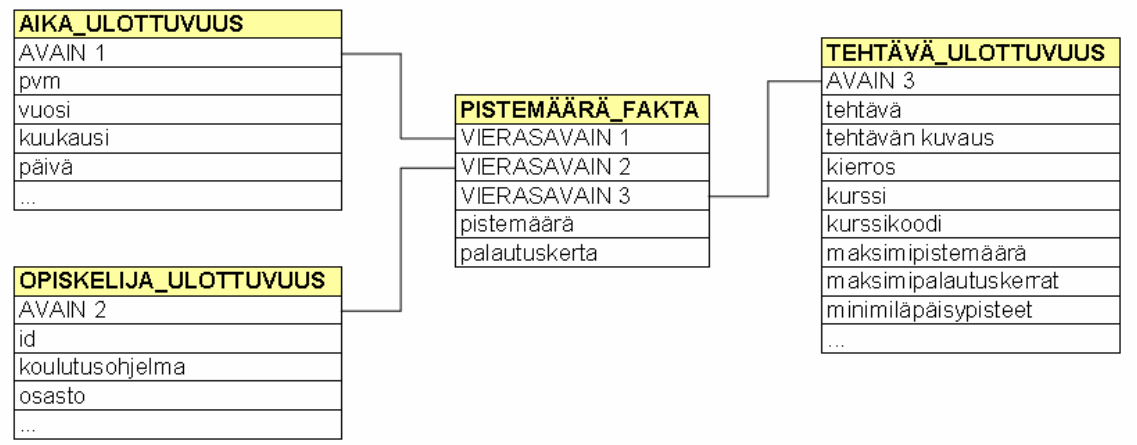
Lataus

Latausvaiheessa on tärkeää huomioida lähinnä joitakin tehokkuustekijöitä sekä taulujen latausjärjestys. Taulujen indeksit kannattaa tehokkuussyistä poistaa ennen latausta ja latauksen jälkeen luoda ne uudelleen. Kannattaa myös käyttää tietokannan bulk load -toimintoa, koska se on paljon nopeampaa kuin yksittäiset päivitykset ja lisäykset. Latausjärjestyksessä ensimmäisenä tulevat mahdolliset dimensioihin kytkeytyvät alidimensiot, sillä suhteiden riippuvuusketju lähtee niistä liikkeelle. Seuraavaksi ladataan varsinaiset dimensiotaulut, joiden vierasavaimet riippuvat alidimensioiden avaimista. Faktataulut ladataan käytännössä viimeisenä, koska ne riippuvat kaikista muista tauluista. Taulut, jotka eivät riipu muista tauluista, voidaan ladata aikaisessa vaiheessa rinnakkain muiden kanssa.

2.6 Dimensiomalli

Tietovarastoissa tiedon looginen rakenne suunnitellaan yleensä eri periaatteilla kuin operationaalisissa tietokannoissa. Tässä esitellään relaatiomalliin perustuva Kimballin kehittämä dimensio- eli tähtimalli (dimensional model, star schema). Se ei perustu mihinkään teoriaan, mutta on käytännössä todettu hyväksi, ja niinpä dimensiomallista on muodostunut vallalla oleva tietovarastojen rakenne (Moody & Kortink 2000).

Tähtimallissa keskellä on faktataulu (fact table), ja sakaroissa ovat dimensio- eli ulottuvuustaulut (dimension table) (kuva 4). Faktataulun ytimenä on yksi tai useampi, yleensä mitattava, numeerinen arvo. Esimerkkejä tästä ovat päivän myynti euroissa ja kappaleissa mitattuna, tai kuvassa opiskelijan harjoitustehtävästä saama pistemäärä. Muut kentät ovat vierasavainviittauksia (foreign key) dimensiotauluihin. Taulun avain (primary key) koostuu osasta näitä vierasavaimia, eli vierasavaimet yksilöivät kunkin faktan.



Kuva 4. Tähtimalli

Ulottuvuus on jokin tekijä, josta käsin valittua faktaa voidaan tarkastella tai rajata. Niiden määrä faktataulua kohti vaihtelee kolmesta ylöspäin, ja aika on lähes aina yksi ulottuvuus. Kuvan esimerkissä ajan lisäksi muita ovat opiskelijan taustatiedot sekä tehtävien kuvaukset. Ulottuvuuksia voi myös tarpeen mukaan lisätä malliin jälkikäteen. Ne sisältävät tarkastelunäkökulmaa kuvaavia attribuutteja, jotka ovat yleensä diskreettejä lukuarvoja tai tekstimuotoisia.

Tietovaraston tietokannassa voi olla useita tähtimallin mukaisia tähtiä, ja osa niiden sakaroista saattaa olla yhteisiä. Jos ulottuvuuteen tulisi vain yksi attribuutti, ei omaa dimensiotaulua kannata tehdä, vaan attribuutti on tapana sijoittaa faktatauluun, jolloin sitä kutsutaan degeneroituneeksi ulottuvuudeksi (degenerate dimension).

Kullekin ulottuvuustaululle annetaan ETL-prosessissa avaimeksi merkityksetön mutta yksilöivä kokonaisluku, jota kutsutaan sijaisavaimeksi tai surrogaatiksi (surrogate key). Tiedon alun perin lähdejärjestelmissä yksilöivistä avainattribuuteista tulee tavallisia datakenttiä, ja niitä kutsutaan mallissa luonnollisiksi avaimiksi (natural key). Luonnollinen avain yksilöi myös edelleen kunkin datarivin taulussa, eikä se periaatteessa voi koskaan muuttua. Sijaisavain sen sijaan saattaa muuttua joissakin tilanteissa, varsinkin käytettäessä hitaasti muuttuvia dimensioita, joista kerrotaan myöhemmin tässä luvussa. Hitaasti muuttuvat dimensiot ovat ylipäänsä tärkeä syy sijaisavainten käyttämiseen. Toisena perusteena niiden käytölle haut ja taululiitokset nopeutuvat, kun käytössä on vain yksi avainattribuutti ja se on kokonaislukumuotoinen. (Kimball & Caserta 2004, 162–163, 170, 211.)

Tähtimalli on pitkälle denormalisoitu eli sisältää toisteista tietoa. Tämä vähentää taulujen määrää ja lisää hakunopeutta, kun taululiitoksia tarvitaan vähemmän. Dimensiomallin mukainen tietokanta ei siis ole enää kolmannessa normaalimuodossa. Malli on myös käyttäjille helpompi ymmärtää kuin perinteinen normalisoitu tietokanta (Hovi *ym.* 2001, 94).

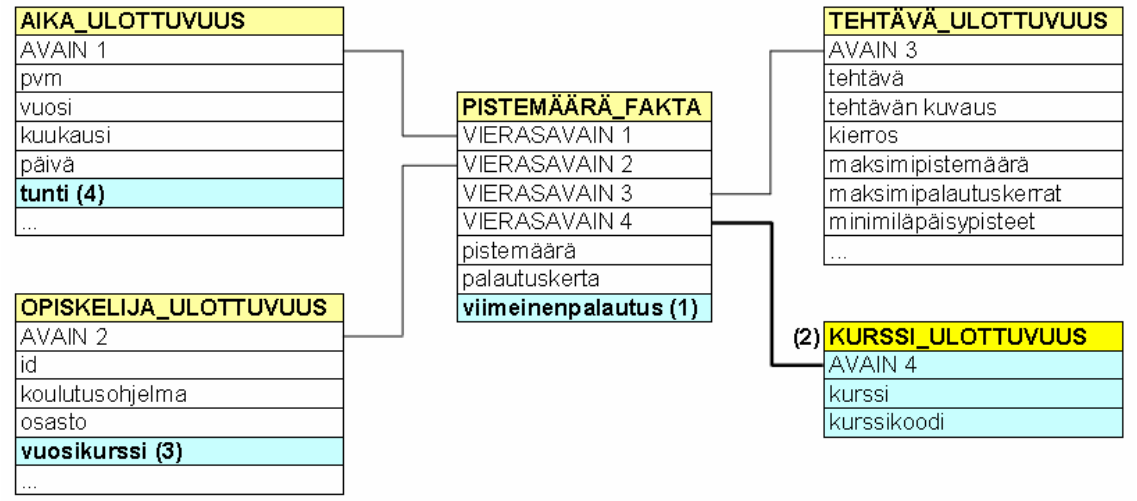
Faktoja voi olla kolmea eri perustyyppiä, ja suunnittelun selkeyttämiseksi sekä virheiden välttämiseksi niissä on hyvä pitäytyä. Transaktiotaulu on yleisin kolmesta tyypistä, ja se tallentaa tavallisia mitattavia suureita valitulta hetkeltä. Taulun rivien määrä on vaikeasti ennustettavissa ja sitä kautta myös hakutulosten määrä. Jaksollinen tilannekuva puolestaan tallentaa hetkellisen tiedon tasaisin väliajoin, esim. kuukauden välein. Tällainen tieto on vaikkapa kuukausittaisessa tiliotteessa näkyvä saldo. Koska jokaiselta ajanjaksolta tulee

yksi uusi tieto, on taulun rivien määrä ennustettavissa, ja kaikki valitun tarkkuustason aika-avaimet ovat käytössä. Viimeinen perustyyppi, kertyvä tilannekuva, sopii datalle tai prosessille, johon liittyy selkeä alku ja loppu, kuten tuotteen tilaaminen ja toimittaminen. Joukko päivämääräkenttiä määrittää standardiskenaarion taululle, esim. tilauspäivä, toimituspäivä ja maksupäivä, ja nämä kentät otetaan käyttöön sitä mukaa, kun ne tulevat ajankohtaisiksi. Toistaiseksi käyttämättömät vierasavaimet ovat tyyppiä N/A, eli ne merkitään käyttämättömiksi. Varsinaiset faktat korvataan uusilla päivitysten yhteydessä. (Kimball & Caserta 2004, 217–224.)

Yksi dimensiomallin vahvuuksista on mahdollisuus tehdä neljänlaisia muutoksia tietokannan skeemaan ilman muutoksia tietovarastoa hyödyntäviin ohjelmiin tai kyselyihin. Näitä kutsutaan hienovaraisiksi muutoksiksi (graceful modifications), ja ne ovat seuraavat (Kimball & Caserta 2004, 235–236):

1. uuden faktan lisääminen vanhaan faktatauluun
2. uuden ulottuvuuden lisääminen faktatauluun
3. attribuutin lisääminen dimensiotauluun
4. fakta- tai dimensiotaulujen muuttaminen suuremmalle tarkkuustasolle

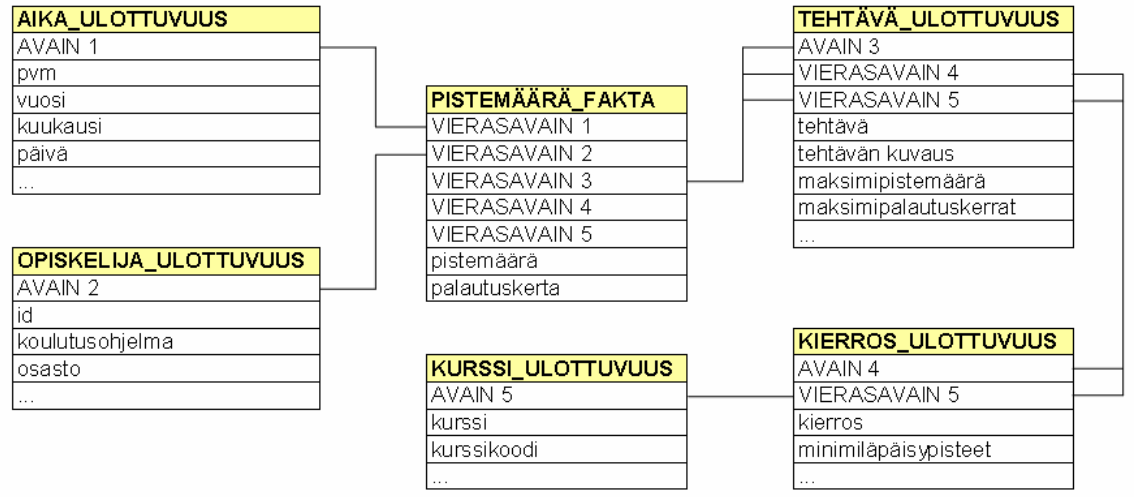
Kuva 5 havainnollistaa näitä muutoksia.



Kuva 5. Kukin hienovaraisten muutosten tyyppi on lisätty numeroituna kuvaan 4 verrattuna.

Mallia vastaan on esitetty myös kritiikkiä. Moody ja Kortink (2000) huomauttavat erityisesti seuraavaa. Ensinnäkin malli perustuu sellaisten faktojen tuntemiseen, joista käyttäjät ovat kiinnostuneita, mutta käyttäjien tarpeet ovat vaikeasti ennakoitavissa ja voivat muuttua ajan myötä. Mallin suunnittelijan täytyy muutenkin ymmärtää datan keskinäiset suhteet hyvin, tai malliin voi tulla virheitä. Jos dataa summataan valmiiksi, menetetään informaation tarkkuutta, mikä voi rajoittaa analysointimahdollisuuksia. Lopuksi lähestymistapa on perusteltavissa lähinnä menestyksekkäiden esimerkkien eikä niinkään järjestelmällisen suunnitteluprosessin avulla.

Tähtimallista eteenpäin kehitetty malli on lumihiutalemalli (snowflake schema), jossa dimensiot on normalisoitu 3. normaalimuotoon (Hovi 1997, 74). Tämä tarkoittaa kuvan 6 mukaisesti, että faktataulut pysyvät ennallaan, mutta ulottuvuustaulut haarautuvat eteenpäin ja muuttuvat kooltaan pienemmiksi (vertaa kuvaan 4). Näin mallin kaavio alkaa muistuttaa hieman lumihiutaletta. Lumihiutalemallilla voidaan saavuttaa tehokkaampia hakuja tähtimalliin verrattuna silloin, kun dimensiotaulut kasvavat hyvin suuriksi (Martyn 2004). Tämä johtuu juuri taulujen pienemmästä koosta, mikä sallii nopeammat liitokset muiden taulujen kanssa.



Kuva 6. Lumihiutalemalli

Dimensiomallin vahva puolestapuhuja Kimball esittää Casertan kanssa kuitenkin kaksi kritiikkiä lumihiutalemallille (2004, 168). Ensinnäkin jos taulujen suhteet muuttuvat, tästä aiheutuu muutoksia myös dataa hyödyntävään ohjelmistoon. Dimensiomallissa samaa ongelmaa ei yhtä helposti tule hienovaraisten muutosten ansiosta. Toiseksi malli on vaikeaselkoisempi ihmiselle lukea. Tällainen vaikeaselkoisuus joudutaan erikseen piilottamaan loppukäyttäjiltä, mikä lisää työtä.

Tähtimallin ja lumihiutalemallin lisäksi puhutaan toisinaan mm. galaksimallista (galaxy schema) (Moody & Kortink 2000) ja tähtihiutalemallista (starflake schema) (You *ym.* 2001). Nämä ovat kuitenkin vain yksinkertaisia muunnelmia edellisistä. Galaksimalli sisältää useita tähtimallin tähtiä, ja tähtihiutalemallissa puolestaan osa ulottuvuuksista on lumihiutalemalliin verrattuna jätetty normalisoimatta. Näiden erottaminen omiksi käsitteikseen vaikuttaisi lähinnä turhaan kasvattavan käsiteviidakkoa.

Hitaasti muuttuvat dimensiot

Vaikka data tietovarastossa periaatteessa pysyy aina samana, kun se on kerran sinne ladattu, niin käytännössä pieniä muutoksia yksittäisiin kenttiin kuitenkin tulee. Syynä voi olla vaikkapa aikaisemman virheellisen tiedon korjaaminen tai yksinkertaisesti lähdetiedon muuttuminen. Tätä ilmiötä kutsutaan hitaasti muuttuviksi dimensioiksi (slowly changing dimensions, SCD). Muutoksia on kolmea eri tyyppiä, ja Kimball ja Caserta kertovat niistä seuraavaa (2004, 183–195).

Ensimmäinen tyyppi on korvaaminen, mikä tarkoittaa, että yksi tai useampia jonkin rivin kenttiä korvataan uusilla arvoilla. Tähän käytetään SQL:n UPDATE-komentoa, tai INSERT-komentoa uuden rivin ollessa kyseessä, jolloin pitää luoda myös uudet avaimet. Tätä ensimmäistä tyyppiä käytetään, kun historiaa ei haluta säilyttää tai kun virheellistä dataa korjataan oikeaksi.

Toinen mahdollisuus on historian säilyttäminen. Tällöin datan muuttuessa uudelle tiedolle luodaan uusi rivi uudella sijaisavaimella. Vanha tieto jää nyt elämään tietovarastossa, ja jälkeenpäin attribuutin muutoshistoriaa voidaan tarkastella vertaamalla sellaisia rivejä keskenään, joissa on sama (muuttumaton) luonnollinen avain. Poistetut rivit muodostavat nyt kuitenkin erikoistapauksen. Jos operationaalisesta tietokannasta poistettu tieto halutaan säilyttää tietovarastossa, se on jotenkin merkittävä tuhotuksi, esim. ylimääräisellä attribuutilla. Lisäksi poistettujen rivien löytäminen vaatii ylimääräisen toimenpiteen, kun täytyy käydä läpi, mitä rivejä tietovarastossa on mutta lähdedatassa enää ei. Näihin toisen tyypin muutoksiin voi olla hyödyllistä tallentaa myös ylimääräistä tietoa muutoksesta, kuten muutoksen aika ja syy, seuraavan muutoksen aika, ja onko kyseinen tieto uusin.

Kolmas hitaasti muuttuvien dimensioiden tyyppi on vaihtoehtoiset arvot. Siinä vanha data säilytetään toisena vaihtoehtona uuden rinnalla. Toteutus eroaa tyypistä 2 siinä, että uuden rivin sijaan tauluun lisätään (tarvittaessa) uusi sarake vanhalle attribuutin arvolle, ja vanha arvo siirretään aina sinne uuden tieltä. Nyt avaimet tai mikään muu kuin muuttuneet arvot eivät vaihdu. Tämä kolmas tyyppi on luonnollisesti laajennettavissa useammankin kuin kahden vaihtoehtoarvon tallentamiseen.

Kaikkia kolmea hitaasti muuttuvien dimensioiden tyyppiä on havainnollistettu kuvassa 7. Siinä (1) henkilön virheellinen nimitieto on korjattu, (2) uusi palkkatieto on lisätty, sekä (3) henkilö on nimitetty uuteen tehtävään. Näin vanhasta virheellisestä nimitiedosta ei tarvitse enää välittää, palkkakehitystä voidaan seurata, ja edellinen työtehtävä jää talteen. Kannattaa myös huomioida, että yksittäisessä ulottuvuudessa eri kentillä voi olla eri tyypit. Niinpä eri tyypit voivat aiheuttaa ylimääräisiä päivityksiä. Esim. tyypin 1 muutos pitää tehdä kaikkiin asiaankuuluviin tyypin 2 historiatietoihin, jos kyse on virheellisen tiedon korjaamisesta. Kuvassa tämä tarkoittaa, että henkilön etunimi on korjattu molemmille riveille lopputilanteessa.

avain	henkilötunnus (luonnollinen avain)	nimi	tehtävä	palkka
...				
837453	010150-001A	Martti Möttönen	tutkija	3 000 €
...				

(1) (3) (2)

avain	henkilötunnus (luonnollinen avain)	nimi	tehtävä	vanha tehtävä	palkka
...					
837453	010150-001A	**Matti** Möttönen	professori	**tutkija**	3 000 €
839724	010150-001A	**Matti** Möttönen	professori	**tutkija**	**4 000 €**
...					

Kuva 7. Hitaasti muuttuvien dimensioiden tyypit

Summaaminen

Summaamisella (aggregation) tarkoitetaan usein kyseltävien perusdatasta johdettujen tietojen laskemista valmiiksi. Tällaisia tietoja voivat olla vaikkapa myyntitulot päivä- tai kuukausitasolla yksittäisen myyntitapahtuman sijaan, tai niistä laskettuja tilastollisia tunnuslukuja. Summaaminen on tehokkain yksittäinen tapa lisätä hakujen nopeutta tietovarastossa, mutta mielekästä lähinnä dimensiomallin kanssa eikä niinkään normalisoiduissa tietokannoissa. Nopeusetu saavutetaan sekä säästyneillä laskutoimituksilla että kyselyiden ohjaamisella paljon pienempiin tauluihin. Summaukset suositellaankin tehtävän omiin faktatauluihinsa, ja jokaiselle summauksen tarkkuustasolle tehdään oma taulu. Näin niitä voidaan käsitellä riippumattomasti muista tauluista. Myös summatauluihin liitettävät dimensiotaulut on syytä pitää kutistettuina versioina alkuperäisistä, ja yksi ratkaisu on jättää osa niistä pois kokonaan. Summatauluja hyödyntämään on hyvä olla tähän tarkoitettu väliohjelmisto. Tällöin loppukäyttäjän ei tarvitse tietää summatauluista mitään, ja hänen tekemänsä tavalliset kyselyt muutetaan väliohjelmistossa mahdollisuuksien mukaan koskemaan summatauluja alkuperäisen datan sijaan. (Kimball & Caserta 2004, 241–247.)

Summataulut voidaan toteuttaa joko erillisinä tauluina tai materialisoituina näkyminä (materialised view). Ne ovat kuten tavalliset näkymät (view) tietokannassa, mutta niistä luodaan todelliset taulut hakujen nopeuttamiseksi. Avoimen lähdekoodin tietokannat eivät vielä suoraan tue materialisoituja näkymiä, mutta ne on ohjelmoitavissa itse.

Selvästä tehokkuushyödystä huolimatta Hovi (1997, 83, 87) muistuttaa joistakin summaukseen liittyvistä ongelmistakin. Näistä ehkä merkittävimpänä voidaan pitää edellytystä tuntea, mitä summauksia ja minkä dimensioiden suhteen loppukäyttäjät tarvitsevat. Summataulujen käyttäminen lisää tietokannan koon lisäksi myös sen monimutkaisuutta. Lisäksi summataulujen luominen ja ylläpitäminen vaatii jonkin verran ylimääräistä työtä.

2.7 Metadata

Metadatalla tarkoitetaan yksinkertaisesti tietoa, joka kuvaa jotakin dataa. Esim. kirjaston kirjatietokanta kuvaa kaikista kirjaston kirjoista nimen, tekijän, julkaisijan ja monia muita asioita. Tietokonemaailmasta esimerkkinä voidaan ottaa ohjelmakoodin kommentointi ja dokumentointi. Metadatan tarkoituksena on siis helpottaa halutun tiedon löytämistä tai ymmärtämistä, jotta sitä voidaan käyttää hyväksi ja muokata.

Metadatan ja varsinaisen järjestelmän tulee noudattaa organisaation standardeja, kuten mahdollista nimeämiskäytäntöä sekä arkkitehtuuri- ja laitteistovaatimuksia. Itse asiassa nimeämiskäytännöstä tulee päättää, ennen kuin mitään aletaan toteuttaa. Jos erityistä käytäntöä ei vanhastaan ole olemassa, voi olla hyvä ottaa mallia lähdejärjestelmistä.

Edellä on kuvattu ETL-prosessia, lopullisen tietovaraston mallia sekä datalähteiden heterogeenisuutta. Kaikki ovat monimutkaisia asioita, joista järjestelmää tuntemattoman olisi vaikea päästä perille vain tutkimalla suoraan tietokantatauluja. On siis helppoa perustella melko mittavaakin tarvetta muodostaa metadataa tietovarastosta. Toisaalta metadatan luomisen ja

ylläpidon työläys ovat yksi peruste pitää itse järjestelmäkin mahdollisimman yksinkertaisena. Tässä käydään läpi tärkeimpiä kohtia, joista metadataa voi olla hyödyllistä muodostaa. Ensin keskitytään tekniseen ja tämän jälkeen käyttäjille suunnattuun metadataan. Lähteenä on hieman mukaillen Kimball ja Caserta (2004, 58–66, 351–381).

Teknisen metadatan voi edelleen jakaa järjestelmän teknisiin määrityksiin ja erilaiseen lokitietoon. Tärkein jälkimmäiseen kategoriaan kuuluvista asioista on ETL-prosessin suorituksesta syntynyt loki. Siitä pitäisi nähdä ainakin, milloin prosessi on suoritettu, kuinka kauan se on kestänyt, kuinka paljon tietoa (rivejä) tietovarastoon on ladattu, sekä missä ja minkälaisia virheitä on tapahtunut. Näiden avulla voidaan paitsi tarkistaa prosessin suorituksen virheettömyys ja paikantaa virhekohdat niin myös mm. tarkkailla datamäärien ja suoritusaikojen muutoksia järjestelmän kehittämistä silmällä pitäen.

Yksi teknisiin määrityksiin kuuluvista, koko ETL-prosessia kuvaavista punaisista langoista on datan looginen kuvaus lähteistä tietovarastoon eli aivan alusta aivan loppuun. Se on taulukko, joka kertoo tarvittavat tekniset yksityiskohdat sekä lähdetietokantojen että tietovaraston sarakkeista ja kuvaa lyhyesti muunnokset, joita tässä välillä tarvitaan. Samaan taulukkoon on mahdollista koota myös summadatan laskukaavat. Alla on pieni esimerkki siitä, miltä tällainen kuvaustaulukko voi näyttää. Jos siihen lisää vielä kunkin attribuutin selväkielisen merkityksen kertovan sarakkeen, taulukosta tulisi hyödyllisempi myös loppukäyttäjille.

target table	target column	datatype	source DB or file	source table or sheet	source column	datatype	transformation / note
course_dim	course_key	int	-				surrogate key
	id	int	trakla2 2004 trakla2 2005-	course	id	int	SELECT id, code FROM course WHERE code like 'T-%';
	code	varchar(20)	trakla2 2004 trakla2 2005-	course	code	varchar(20)	same as above
	name	varchar(255)	-				specified manually

Kuva 8. Datan looginen kuvaustaulukko lähteistä tietovarastoon

Päivitettäessä ETL-prosessia nousevat tärkeiksi edellisenkaltaisten taulukoiden lisäksi sekä lähdetietokantojen että tietovaraston skeemat. Tähän on syytä sisällyttää myös tietovaraston indeksit, näkymät ja muu vastaava tekninen tieto. Samoin lähteinä käytettävistä tiedostoista ja lokeista tarvitaan tarkat kuvaukset. Jos käyttäjien annetaan tehdä omia SQL-kyselyitään tietovarastoon, myös he tarvitsevat skeeman siitä.

Lopuksi teknisellä puolella tarvitaan tieto lähdedatan omistajista, ylläpitäjistä, keinoista päästä käsiksi dataan tarvittavine ohjelmineen, protokollineen, tunnuksineen ja salasanoineen, tietovaraston järjestelmäkohtainen päivitysaikataulu sekä summataulujen määritykset. Summataulut tosin saattavat tulla kuvatuiksi riittävän tarkasti aikaisemmissakin kohdissa.

Kuten edellä oikeastaan kävi ilmi, loppukäyttäjät hyötyvät pitkälti samoista dokumenteista, jos niiden tekemisessä on otettu myös heidän tarpeensa huomioon. Erityisesti on syytä korostaa asioiden selväkielisyyttä ja helposti ymmärrettävää esitystapaa, sillä käyttäjät eivät todennäköisesti tunnista teknisiä termejä tai lue monimutkaisia kaavioita kovin hyvin. Jos näitä asioita ei saada helposti liitettyä muuhun metadataan, voidaan harkita erityisen sanakirjan luomista, jossa kerrottaisiin selväkielisesti asioiden ja termien nimet ja merkitykset. Tämän lisäksi käyttäjille täytyy luoda ja luovuttaa tietovaraston käyttäjätunnukset.

2.8 OLAP ja datan analysointi

Datan siirtäminen tietovarastoon ei itsessään ole kovin hyödyllistä, vaan sitä täytyy pystyä analysoimaan ja hyödyntämään eri keinoin. Tähän on olemassa ns. OLAP-sovelluksia (Online Analytical Processing), jotka perustuvat tietovaraston tietokantaan ja tarjoavat loppukäyttäjälle hyödyllisen näkymän tietoon. Tavallisista operatiivisista tietokannoista käytetään tällöin nimitystä OLTP (Online Transaction Processing). Tällä jaolla halutaan korostaa operatiivisen tietokannan ja tietovaraston eroa. Operatiivisen kannan tyypillisiä ominaisuuksia ovat mm. suuri käyttöaste, päivitysten suuri määrä, matalan

tason data, yksinkertaiset kyselyt sekä tietokannasta saatavat vakiomuotoiset raportit. Tietovarastossa puolestaan on yleensä matala käyttöaste, päivitykset hyvin harvinaisia, dataa summattu korkeammalle tasolle sekä vaihtelevat ja monimutkaiset kyselyt.

Koska datan hyödyntäminen ei ole tämän työn painopisteenä, OLAP pyritään määrittämään tiiviisti lähinnä FASMI-käsitteen (Fast Analysis of Shared Multidimensional Information) kautta. Kyseessä ovat viisi tekijää, jotka määräävät OLAP:n yleisiä ominaisuuksia. Tällaisen rinnakkaisen käsitteen luominen on ollut tarpeellista, sillä OLTP-termistä johdettu OLAP ei itse auki kirjoitettuna kerro paljon mitään (Pendse 2005). FASMI-ominaisuudet on esitelty taulukossa 6. Näistä shared-kohta vaikuttaa jonkin verran kyseenalaiselta ja lienee oleellinen vain erikoistapauksissa, sillä kuten myös Hovi *ym.* (2001, 57) toteavat, niin tietovarastot ovat vain lukua varten. Monen käyttäjän yhtäaikaiset lukuoperaatiot nimittäin eivät ole mikään ongelma.

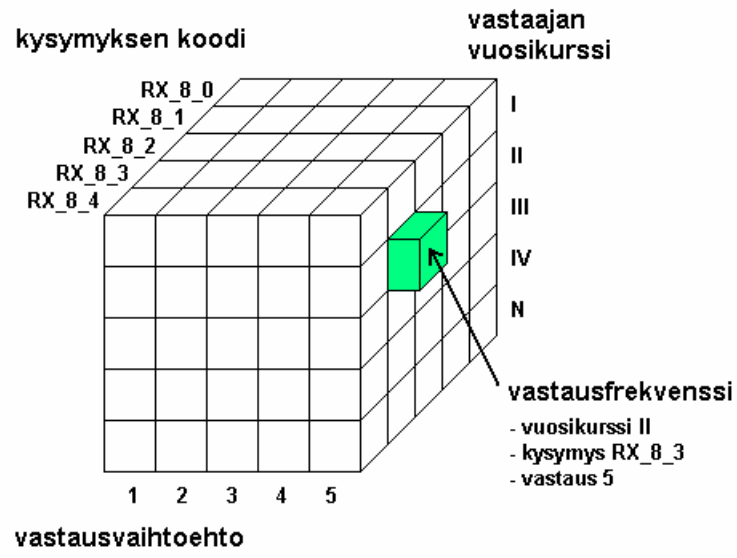
Taulukko 6. OLAP:n määrittävät FASMI-ominaisuudet

Tekijä	Kuvaus
Fast (nopeus)	Vastaus suurimpaan osaan kyselyistä saadaan alle 5 sekunnissa, nopeimpiin alle sekunnissa ja äärimmäisen harvoihin yli 20 sekunnissa. Muuten käyttäjät olettavat helposti järjestelmän jumiutuneen tai kadottavat keskittymisensä. Yksi tärkeä apuväline tähän on tiedon summaaminen.
Analysis (analysointikyky)	Analysointityökalu tarjoaa riittävän helposti käyttäjälle mahdollisuuden tehdä kaikkia aihealueeseen liittyviä oleellisia analyyseja sekä omia ad hoc -kyselyitä.
Shared (jakaminen)	Monta käyttäjää voi käyttää järjestelmää samanaikaisesti (vrt. ACID-ominaisuuksien isolaatio). Erikoistapauksissa moni käyttäjä voi nimittäin yrittää kirjoittaa tietoa samanaikaisesti.
Multidimensional (moniulotteisuus)	Järjestelmän on pystyttävä esittämään data moniulotteisessa, käsitteellisessä ja hierarkkisessa muodossa. Faktataululla on monta ulottuvuustaulua eli tarkastelunäkökulmaa, kuten aika, sijainti jne. Hierarkkisuus tarkoittaa, että vaikkapa aikaulottuvuudesta voi valita päivän, viikon, kuukauden jne. tiedon tarkkuustasoksi.
Information (tietomäärä)	Järjestelmän on pystyttävä käsittelemään riittävästi dataa sovellusalueen tarpeisiin.

Lähde: Pendse 2005

Analysoinnissa on havaittu hyväksi käsitellä tietoa moniulotteisesti, kuten yksi edellisen taulukon ominaisuuksistakin kertoo. Tästä syystä dimensiomalli on

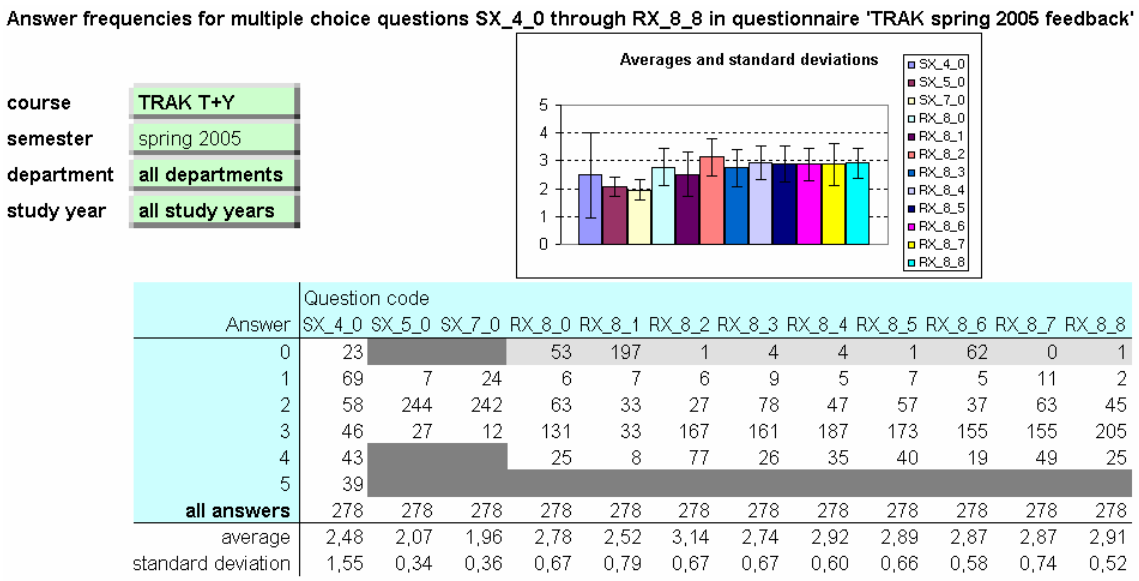
hyödyllinen relaatiotietokantaan perustuvassa tietovarastossa, sillä sen avulla relaatiomallin taulujen tarjoamaa kahta ulottuvuutta voidaan laajentaa. Tiedon moniulotteinen esittäminen onnistuu siis relaatiotietokantoja pohjana käyttäen, mutta tarkoitukseen on myös olemassa erityisiä moniulotteisia tietokantoja. Moniulotteisten tietokantojen vahvuus on niiden erikoistuminen pelkästään OLAP-kyselyihin, mutta heikkoutena toteutukset vaihtelevat valmistajan mukaan eivätkä usein ole avoimia, minkä lisäksi ne eivät pysty käsittelemään yhtä suuria datamääriä kerrallaan kuin relaatiotietokannat (Hovi *ym.* 2001, 59). Moniulotteista tiedon esittämistapaa kutsutaan myös kuutioksi (cube), vaikka nimestä huolimatta ulottuvuuksia on yleensä enemmän kuin kolme. Moniulotteisuutta havainnollistetaan kuvassa 9, jossa tarkastellaan kyselystä saatuja opiskelijoiden vastausfrekvenssejä. Koska kuvassa on kolme eri ulottuvuutta, joista jokaisessa on viisi eri vaihtoehtoa, on frekvenssejä eli faktoja yhteensä 125, joista yksi on poimittu esiin. Tämä fakta kertoo, kuinka moni toisen vuosikurssin opiskelija on valinnut kysymykseen RX_8_3 vastausvaihtoehdon 5.



Kuva 9. OLAP-kuutio

Kuva 10 esittää, miltä OLAP-sovellus saattaa käyttäjälle näyttää. Alhaalla on taulukko, jossa sarakkeissa on kyselyn kysymysten koodit ja riveillä eri vastausvaihtoehdot 0-5 ja näiden alapuolella vastausten kokonaismäärä sekä

pari käyttäjän omaa taulukon avulla laskemaa tunnuslukua. Rivit ja sarakkeet edustavat kukin yhtä ulottuvuutta datassa. Loput ulottuvuudet ovat vasemmassa ylänurkassa (kurssi, vuosi, koulutusohjelmat sekä opiskeluvuodet). Oikealla ylhäällä on taulukon datasta piirretty kuvaaja. Ulottuvuuksien valintoja voidaan nyt vaihtaa siten, että taulukon ja kuvaajan tiedot päivittyvät välittömästi vastaamaan uusia valintoja. Esim. kaikkien koulutusohjelmien sijaan voitaisiin päättää katsoa vain tietotekniikan opiskelijoiden vastauksia. Tästä käy ilmi myös tiedon hierarkkinen jäsentäminen kuutiossa. Yksittäiset koulutusohjelmat muodostavat ryhmän ”kaikki koulutusohjelmat” ja vastaavasti yksittäiset vastausvaihtoehdot taulukossa muodostavat niiden alla olevat ”kaikki vastaukset” -ryhmän. Siirtymistä alemmalle hierarkiatasolle kutsutaan porautumiseksi (drill down) ja nousemista korkeammalle tasolle karkeistamiseksi (roll up).



Kuva 10. Esimerkki OLAP-sovelluksesta (Palo Excel-plugin)

2.9 Aikaisempi tutkimustieto

Tietovarastointia on ehditty tutkia kohtalaisen paljon, ja kirjallisuus näyttäisi jonkin verran keskittyvän 1990-luvun loppupuolelle. Varsinkaan perusteoria ei näytä juurikaan muuttuneen tämän vuosituhannen puolella. Useimmat tieteelliset artikkelit keskittyvätkin erityiskysymyksiin, jotka eivät suuresti kosketa tätä työtä. Suositut aiheet ovat liittyneet mm. kyselyjen optimointiin, indeksointiin

ja näkymien hallintaan. Lähimmäksi tulevat tietovaraston mallintamista koskevat paperit.

Tryfona *ym.* (1999) luovat tavan mallintaa tietovarastoa käsitteellisesti yhdistämällä ER-mallin monipuolisuuden vakiintuneeseen tähtimalliin. Boehnlein ja Ulbrich-vom Ende (1999) käyttävät puolestaan laajennettua ER-mallia (SERM, Structured Entity Relationship Model), Jones ja Song (2005) yhdistävät suunnittelumallit (design patterns) tähtimallin suunnitteluun, ja Moody esittää Kortinkin kanssa (2000) keinon muuntaa ER-malli tähtimalliksi. Vassiliadis *ym.* (2002) ehdottavat ETL-prosessin mallintamista erityisiä kaavioita käyttäen. Osassa ehdotetuista malleista on takana käytännön kokeiluja. Mikään tutkijoiden esittämistä keinoista ei näytä kuitenkaan yleistyneen, sillä ne eivät ole päätyneet esim. Internetin tietovarastointia esitteleville sivustoille tai oppikirjoihin. Niinpä niiden hyödyt jäävät hieman kyseenalaisiksi.

Varsinaisista ETL-työkaluista suurin osa on kaupallisia tuotteita tai avoimen lähdekoodin sovelluksia, joilla ei ole tekemistä tieteellisten julkaisujen kanssa. Pienenä poikkeuksena Haas *ym.* (2005) kuvaavat IBM:n kehittämän mielenkiintoisen Clio-työkalun, jolla voidaan muuntaa relaatiomallin skeemoja toisiksi, mitä voisi hyödyntää ETL-prosessissa. Cliossa määritetään graafisesti skeemamuunnos kahden skeeman välillä, ja sovellus luo sitä vastaavat SQL-lauseet. Valitettavasti itse sovellus ei ole julkisesti tarjolla.

Tieteelliset artikkelit eivät siis sisällä kovin paljon hyödyllistä tutkimustietoa tälle työlle. Jonkinlaisen vertailukohdan löytää sen sijaan muista tietovarastoinnin opinnäytetöistä. Nekin ovat silti aiheiltaan vaihtelevia ja painottavat hieman eri asioita, kuten tietovaraston tyypillisiä hakujen nopeuttamiskeinoja (Kostiainen 2003), tiedon laadun kehittämistä valmiissa tietovarastossa (Ollila 2005), tai sitten ne eroavat sovellusalueen ja työkalujen osalta (Mäkinen 1999; Karhumäki 2001). Tässä työssä pyritään sen sijaan antamaan melko yksityiskohtainen perusasioihin keskittyvä tekninen kuvaus aiheesta.

Horstmann (2005) ei käsittele suoranaisesti tietovarastointia vaan siirtymistä kaupallisesta tietokannasta avoimen lähdekoodin tietokantaan. Vaihtaminen uuteen tietokantaan muistuttaa kuitenkin tietovaraston luomista ETL-prosessin osalta, sillä siinäkin luodaan uusi tietokanta eri tietokantasovellukseen kuin lähdejärjestelmät. Työ sisältää erityisesti hyödyllistä analyysia avoimen lähdekoodin tietokannoista. Tietovarastoinnista Horstmann toteaa, että sitä ei ole paljon tehty avoimen lähdekoodin tietokannoilla, mutta työkaluja on juuri nyt kehitteillä useitakin (2005, 31–34).

3 Vaatimukset ja lähtökohdat

Tässä luvussa tuodaan esiin ensin työn erityispiirteet sekä käyttäjäryhmä. Kuten jo johdannossa todettiin, tämä työ poikkeaa monin tavoin tavallisesta tietovarastointiprojektista. Nämä piirteet ovat yksi tärkeä lähtökohta vaatimuksille, jotka tarkentavat alussa esitettyjä tutkimuskysymyksiä. Tämän jälkeen käydään läpi mukaan otettavia tarkastusjärjestelmiä ja niiden dataa. Muut järjestelmät, joihin tietovarasto mahdollisesti myöhemmin laajennetaan, esitellään vain pintapuolisesti. Näillä lähdejärjestelmilläkin on silti vaikutusta vaatimuksiin. Lopuksi pohditaan lyhyesti projektin tietosuojakysymyksiä.

3.1 Työn erityispiirteet

Tämä työ eroaa tyypillisestä tietovarastosta sovellusalueen, käyttäjäryhmän, datan määrän ja monen muun pienemmän tekijän suhteen. Näitä tekijöitä on syytä käydä lyhyesti läpi, sillä ne vaikuttavat tietovaraston suunnitteluun ja toteutukseen ensimmäisistä valinnoista alkaen. Käyttäjistä kerrotaan kuitenkin vasta kohdassa 3.2.

Tietovarastoja on tähän mennessä ehditty toki rakentaa kaikenlaisiin aihepiireihin liittyvästä datasta ja hyvin monenlaisissa organisaatioissa. Suurin osa kirjallisuudesta näyttäisi kuitenkin keskittyvän suurten yritysten kaupallisiin tarpeisiin. Niinpä ei ole varmuutta, vaikuttaako tämän työn sovellusalue jollain odottamattomalla tavalla tietovaraston rakentamiseen. Soveltuuko esim. dimensiomalli opiskelijoiden ja heidän suoritustensa kuvaamiseen ja hakemiseen yhtä hyvin kuin tuotteiden ja niiden myyntimäärien käsittelemiseen? Datassa saatetaan olla suhteellisesti enemmän kiinnostuneita keskiarvoista ja muista tilastollisista tunnusluvuista kuin yksinkertaisista summista, joita OLAP-sovellukset olettavat käyttäjien haluavan.

Datan määrä on tietovarastoksi vähäinen. Liikkeelle lähdetään varovaisesti muutaman järjestelmän voimin, ja suurimmassakin yksittäisessä taulussa on vain reilut satatuhatta riviä, kun suuressa yrityksessä voidaan helposti puhua useista miljoonien rivien tauluista. Tämä seikka itse asiassa mahdollistaa

projektin läpi viemisen yhden henkilön voimin. Kimball ja Caserta (2004, 383–384) luettelevat joukon tarvittavia eri henkilöiden työtehtäviä tietovarastoprojektin toteuttamiseen olettaessaan projektin lähtökohtaisesti olevan hyvin suuri. Toisaalta tämä helpottava tekijä kannattaa hyödyntää erityisesti pohtimalla, mitä asioita voisi tehdä yksinkertaisemmin, jos siihen on mahdollisuus. Esim. metadataa ei tarvita niin paljon, koska pienempää kokonaisuutta on helpompi hallita vähemmilläkin määrityksillä, ja samalla helpotetaan ylläpitäjän työtä.

Kerättävä data koostuu lähinnä opiskelijoiden tekemistä harjoitustehtävätuloksista ja muista suorituksista valituilla kursseilla. Kurssien pituudet ovat melko tarkasti rajattuja ja kestävät tyypillisesti puolisen vuotta. Vanha data säilötään sellaisenaan, eikä se muutu enää kurssin päätyttyä. Niinpä dataa ei ole tarkoitus lisätä yhdestä lähteestä lataamisen jälkeen, jos eri kurssit säilötään eri tietokantoihin. Tämän seurauksena voidaan harkita hitaasti muuttuvien dimensioiden jättämistä määrittämättä, jos kaikki data näyttäisi olevan tyyppiä 1 (ks. luku 2.6). Jos korjauksia kuitenkin tulee, olisi luultavasti helpompaa suorittaa koko lataus uudestaan, ja tehdä latausprosessista mahdollisimman automaattinen. Tästä kaikesta joka tapauksessa nähdään, että työn erityispiirteillä voi olla huomattavia vaikutuksia tietovarastoprojektiin.

3.2 Käyttäjäryhmä

Kuten edellä kävi ilmi, ovat loppukäyttäjätkin tavallisesta tietovarastointiprojektista poikkeava ryhmä. He ovat kaikki tietotekniikan opiskelijoita tai jo valmistuneita tutkijoita. Niinpä heidän osaamistasonsa on selvästi vaikkapa tyypillistä yritysjohtoa korkeammalla tasolla. He kykenevät mm. luomaan itse SQL-kyselyitä. Tällä voidaan pitkälti perustella mahdollisuutta jättää analysointi- ja raportointityökalujen mukaan ottaminen vähemmälle huomiolle. Käyttäjien korkea osaamistaso tulee muutenkin tarpeeseen, sillä heidän on myös osallistuttava tietovaraston ylläpitoon ja laajentamiseen tämän diplomityön valmistuttua. Käyttäjäryhmän työtehtäviin kuuluu varsinaisen

tutkimustyön lisäksi lähinnä opetusta, opiskelijoiden ohjausta sekä hallinnollisia tehtäviä.

Ensimmäisessä vaiheessa tietovarastoa käytetään vain Teknillisen korkeakoulun Ohjelmistotekniikan laboratoriossa. Käyttäjät tuntevat ainakin jollain tasolla myös osan lähdejärjestelmistä, ja he ovat mahdollisesti olleet kehittämässäkin niitä. Koska osa näistä järjestelmistä on käytössä myös muissa Suomen yliopistoissa, ja TAPAS on myös yliopistojen välinen yhteistyöprojekti, laajenee käyttäjäkunta jossakin vaiheessa myös toisten koulujen tutkijoihin.

Tietovaraston loppukäyttäjät ovat ilmoittaneet seuraavanlaisia esimerkinomaisia tarpeita järjestelmän käytölle. Nämä tulevat yksityiskohtaisemmin esiin vielä luvun 3.3 vaatimusmäärittelyssä. Ensinnäkin ollaan kiinnostuneita tekemään pitkittäistutkimusta yksittäisille kursseille. Tämä on mahdollista ilman tietovarastoakin, mutta uuden järjestelmän myötä datan hyödyntäminen pitäisi tulla huomattavasti helpommaksi. Kaiken tutkimusdatan löytäminen yhdestä paikasta on saattanut aikaisemmin olla ongelma yksittäisenkin kurssin kohdalla, mutta nyt useiden kurssien tiedot saadaan kaivettua esiin samasta paikasta. Tämä mahdollistaa tietenkin useiden kurssien vertaamisen keskenään valittujen tekijöiden ja niiden keskinäisten korrelaatioiden osalta. Tarkastelun kohteeksi voidaan ottaa myös yksittäisiä opiskelijoita, jotka ovat suorittaneet useita tietovarastoon liitetyistä kursseista, ja tarkastella heidän menestyksensä tai palautuskäyttäytymisensä kehittymistä.

3.3 Vaatimukset

Tässä kuvataan työn vaatimukset tavallisen ohjelmistoprojektin tapaa mukaillen. Kovin yksityiskohtaiselle tasolle ei kuitenkaan mennä, vaan tarkoitus on tuoda esiin tärkeimpiä projektin onnistumiseen vaikuttavia tekijöitä. Vaatimukset on jaoteltu kolmeen eri tärkeysluokkaan taulukon 7 mukaisesti. Vaatimusten toteutumista lopullisessa tietovarastossa tarkastellaan luvussa 6.1 tulosten analysoinnin yhteydessä.

Taulukko 7. Vaatimusmäärittelyn prioriteetit

#	Tärkeys	Kuvaus
1	Välttämätön	Järjestelmän toiminnan ja käytön kannalta olennainen asia, jonka puuttuminen selvästi haittaa tai estää tietovaraston käyttöä.
2	Tärkeä	Tärkeä ominaisuus, joka tuo järjestelmälle selvää lisäarvoa ja jonka pois jättäminen saattaa heikentää esim. käyttömukavuutta.
3	Ylimääräinen	Lisäominaisuus, josta on jonkinlaista hyötyä järjestelmälle, mutta jonka puuttuminen ei varsinaisesti vähennä sen laatua. Nämä toteutetaan, jos on käytössä ylimääräisiä resursseja.

Vaatimukset jaetaan karkeasti toiminnallisiin ja ei-toiminnallisiin vaatimuksiin. Toiminnalliset vaatimukset (taulukko 8) kuvaavat, mitä sekä loppukäyttäjien että ylläpitäjien on pystyttävä tietovaraston kanssa tekemään. Ei-toiminnallisia vaatimuksia (taulukko 9) ovat kaikki muut sisältäen lähinnä teknisiä ominaisuuksia, suorituskykymittareita ja käytettävyysvaatimuksia. Taulukoissa annetaan kullekin vaatimukselle koodi, itse vaatimuksen lyhyt kuvaus, perustelu tai tarkennus sekä prioriteetti.

Taulukko 8. Toiminnalliset vaatimukset

#	Vaatimus	Perustelu tai lisätieto	Priori-teetti
T1	ETL-prosessi määrityksineen kullekin lähdejärjestelmälle on käytettävissä uudestaan ja muokattavissa jälkeenpäin.	ETL on niin suuritöinen prosessi, että sen tekeminen alusta joka kerta olisi järjetöntä.	1
T2	Tietovarastoon voi liittää uusia lähdejärjestelmiä.	Laajentamisen on oltava mahdollista. Usein tietovarastoihin lisätään järjestelmiä vähitellen sen perustamisen jälkeen.	1
T3	Tietovarastosta voidaan hakea tietoa vakiintuneella kyselykielellä tai vastaavalla tavalla vapaasti.	Tietoa on pystyttävä hyödyntämään jotenkin. FASMI:n Analysis-ominaisuus (osittain). Esim. SQL.	1
T4	Tietovarastosta voidaan hakea tietoa graafisella käyttöliittymällä.	Tiedon helppokäyttöinen hakeminen lisää käytettävyyttä huomattavasti, mutta se ei ole tämän työn painopisteitä. Esim. OLAP-sovellus.	3
T5	Tietovarastolla voi olla useita samanaikaisia käyttäjiä.	Tavallisesti käyttäjäkunta voi olla suurikin, mutta tässä työssä näin ei ole. FASMI:n Shared-ominaisuus.	3

Taulukko 9. Ei-toiminnalliset vaatimukset

#	Vaatimus	Perustelu tai lisätieto	Prioriteetti
E1	Järjestelmän tietokantapalvelimen voi asentaa sekä Linux- että Windows-ympäristöön.	Kehitystyö tehdään Windowsilla, mutta tutkimusryhmässä Linux on tavallisin palvelinkäyttöjärjestelmä.	1
E2	Tietovarasto on loppukäyttäjälle käytettävissä Linux-, Macintosh- ja Windows-ympäristöistä käsin.	Nämä kolme käyttöjärjestelmää ovat loppukäyttäjien käytössä.	1
E3	Vastaus suurimpaan osaan kyselyistä saadaan alle 5 sekunnissa, nopeimpiin alle sekunnissa ja äärimmäisen harvoihin yli 20 sekunnissa.	FASMI:n Fast-ominaisuus. Jos haut eivät ole tarpeeksi nopeita, käyttäjät kokevat järjestelmän helposti käyttökelvottomana.	1
E4	Tietovaraston ETL-prosessista luodaan loki, josta nähdään ongelmat suorituksen aikana.	ETL-prosessin loki on tärkein osa lokimuotoista metadataa. Ilman sitä virheiden jäljittäminen on työlästä.	1
E5	Järjestelmä pystyy käsittelemään riittävän suuren määrän dataa sovellusalueen tarpeisiin.	FASMI:n Information-ominaisuus.	1
E6	Järjestelmässä käytetään ainoastaan avoimen lähdekoodin tai muita ilmaisia sovelluksia.	Tämä vaatimus tulee projektin johdolta.	2
E7	Järjestelmä on modulaarinen siten, että yksittäinen osa tai sovellus on helppo vaihtaa.	Tämä vaatimus tulee projektin johdolta.	2
E8	Tietovaraston lataaminen valmiista ETL-määrityksestä kestää alle puoli tuntia.	Järjestelmien käytön rajoittuminen latauksen aikana ei ole ongelma, mutta ETL:n ajamisen on silti tapahduttava kohtuullisessa ajassa.	2
E9	Tietovarastosta tehdään riittävästi metadataa helpottamaan järjestelmän käytön, toiminnan ja muokkaamisen ymmärtämistä.	Vaadittua metadataa ovat: ▪ datan looginen kuvaus lähteistä tietovarastoon ▪ tietovaraston skeema ja arkkitehtuuri ▪ tiedot lähdejärjestelmistä ▪ loppukäyttäjälle tietovaraston sisällön selväkielinen kuvaus	2
E10	Järjestelmän hyödyntäminen datan analysointiin on nopeasti opittavissa ja vähintään yhtä helppokäyttöistä kuin yksittäisissä lähdejärjestelmissä.	FASMI:n Analysis-ominaisuus (osittain).	2
E11	Järjestelmä on englanninkielinen.	Tämä vaatimus tulee projektin johdolta.	2
E12	Uusien järjestelmien liittäminen tietovarastoon ja ETL-prosessien muokkaaminen on opittavissa ja tehtävissä kohtuullisella vaivalla.	Tietokantoja tuntevan henkilön pitäisi kahden viikon työllä saada lisättyä uusi lähde tietovarastoon.	2
E13	Järjestelmä pystyy esittämään datan moniulotteisessa ja hierarkkisessa muodossa.	FASMI:n Multidimensional-ominaisuus.	3

3.4 Lähdedata

Alkuvaiheessa tietovarastoon kerätään kolmen datankeruujärjestelmän sisältämää dataa. Tässä pyritään antamaan noista järjestelmistä yleiskuva, jotta niiden tärkeimmät ominaisuudet ja erot tulisivat selville. Nämä on koottu taulukkoon 10. Kaikki järjestelmät ovat nykyisessä muodossaan korkeintaan muutaman vuoden vanhoja, joten niiden teknologia on kohtalaisen tuoretta, mikä helpottaa datan poimimista. Taulukossa on myös mainittu kirjallisuusviite, jonka kautta järjestelmiin voi tutustua tarkemmin. Osasuoritusrekisteriä (OSR) ei ole tehty opinnäytetyönä tai osana tieteellistä tutkimusta, joten siitä ei ole julkaistu kirjallista materiaalia.

Taulukko 10. Alkuvaiheessa tietovarastoon sisällytettävät datankeruujärjestelmät

datankeruu-järjestelmä	TRAKLA2	Goblin	OSR (osa-suoritusrekisteri)
kurssit	tietorakenteet ja algoritmit (TRAK); erilliset kurssit tietotekniikan ja muiden koulutusohjelmien opiskelijoille	eräät ohjelmoinnin peruskurssit	kaikkien tietotekniikan osaston kurssien osasuoritukset ja tulokset
tietokanta	PostgreSQL 7.4.1	MySQL 4.0.16 (LAMP-alusta = Linux, Apache, MySQL ja PHP)	Solid
arvio datan loogisesta mallista	tietokanta suunniteltu datan keräämistä varten	suuri määrä tauluja sisältäen paljon tietovaraston kannalta turhaa dataa, kuten käyttöoikeuksien hallintaa	kohtalaisen yksinkertainen ja selkeä tietovarastoa ajatellen
datan hajautus kurssien kesken	kaikkien kurssien data samassa tietokannassa ml. muiden yliopistojen kurssit	kaikkien kurssien data samalla palvelimella mutta erillisissä tietokannoissa	kaikki data samassa tietokannassa
kirjallisuus-viite	Silvasti 2003	Hiisilä 2005	-

TRAKLA2 liittyy tietorakenteiden ja algoritmien perusteita opettaviin kursseihin. Kyseinen järjestelmä antaa opiskelijoiden tehdä harjoitustehtäviä itsenäisesti verkossa olevalla Java-sovelmalla ja arvostelee tehtävät automaattisesti.

Järjestelmän datankeruu on suunniteltu opetuksen kehittämistä ajatellen, joten tietokanta sisältää paljon sellaisenaan hyvää dataa tietovarastoa varten. Tietokanta on kokenut muutoksia vuoden 2005 alussa, ja sitä vanhempi data on hieman niukempaa kuin uusi. Uuden osan saaminen tietovarastoon on tärkeämpää, mutta myös vanhempi data pyritään liittämään mukaan. Lisäksi joka vuosi on kerätty dataa palautekyselyissä, ja vuoden 2005 keväällä kursseilla järjestettiin erityinen koeasetelma opetuksen kehittämistä varten. Tähän koeasetelmaan liittyi opiskelijoiden täyttämiä kyselyitä kaiken tavallisen TRAKLA2-datan lisäksi. Kaikkien näiden kyselyiden tulokset otetaan tietovarastoon mukaan. Itse asiassa juuri kevään 2005 tulosten analysoinnista ollaan kiinnostuneita tällä hetkellä, jolloin kyseinen data muodostaa hyvän ja käyttäjiä kiinnostavan lähtökohdan tietovaraston prototyypille ja käyttöönotolle. On huomattava, että TRAKLA2-tietokannassa säilötään myös muiden yliopistojen kurssien tietoja. Ne on kuitenkin alkuvaiheessa rajattu työn ulkopuolelle, sillä kyseiset yliopistot eivät ole alussa mukana projektissa.

Goblin on automaatio- ja systeemitekniikan osastolla kehitetty järjestelmä yksinkertaisten ohjelmointitehtävien automaattiseen tarkastamiseen. Se on käytössä useilla ohjelmoinnin peruskursseilla. Datan kerääminen Goblinista poikkeaa toisista järjestelmistä siinä, että eri kurssit on hajautettu eri tietokantoihin ja tietokannan looginen malli on melko monimutkainen. Toisaalta kurssien hajauttaminen antaa mahdollisuuden hallita ja rajata tietovarastoon otettavia kursseja paremmin. Sekä Goblinin että TRAKLA2:n tietokantojen malleihin suunnitellaan tällä hetkellä pieniä parannuksia, mikä tarkoittaa tietovaraston kannalta, että datan poimintaprosessia joudutaan tulevaisuudessa korjaamaan näiden järjestelmien osalta.

OSR ei ole tarkastusjärjestelmä kuten muut, vaan siihen kerätään kaikkien tietotekniikan osaston kurssien osasuoritukset, kuten harjoitustyö- ja tenttiarvosanat, sekä tietysti kokonaisarvosanat. Kyseessä on osaston hallinnon työkalu. OSR on otettu mukaan tietovarastoon sen tarjoaman laajemman ja korkeamman tason datan vuoksi tukemaan muiden järjestelmien antamaa

tietoa. Näin voidaan vaikkapa yrittää selittää opiskelijoiden menestystä ohjelmointiharjoituksissa heidän käymiensä aikaisempien kurssien ja niiden arvosanojen perusteella. Aluksi OSR-järjestelmästä poimitaan vain Ohjelmistotekniikan laboratorion kurssien suoritustietoja, mutta myöhemmin yhteistyö muidenkin laboratorioiden kanssa voi tulla kysymykseen. Teknisesti OSR poikkeaa siinä muista kahdesta järjestelmästä, että se käyttää kaupallista Solid-tietokantaa. Sen vaihtamista avoimen lähdekoodin tietokantaan kuitenkin harkitaan.

Vaikka tietovarastoon liitetään alussa vain kolmen järjestelmän data, on muita vastaavia järjestelmiä jo tiedossa. Myös niiden asettamat vaatimukset pyritään huomioimaan suunnittelussa tulevaisuutta varten. Eniten muista järjestelmistä poikkeaa Joensuun yliopiston kehittämä AEA-järjestelmä (Automatic Essay Assessor) (Kakkonen 2003), joka arvioi automaattisesti esseevastauksia. Järjestelmä eroaa tietovaraston kannalta erityisesti siinä, että se käsittelee pitkiä tekstipätkiä sekä muodostaa arvioinnissa tarvittavaa erityistä apudataa.

Toisena ulkopuolisena datalähteenä on muiden yliopistojen TRAKLA2-data. Kyseistä järjestelmää on toistaiseksi käytetty Teknillisen korkeakoulun lisäksi Turun yliopistossa, Tampereen teknillisessä yliopistossa, Lappeenrannan teknillisessä yliopistossa sekä Helsingin ammattikorkeakoulu Stadiassa. Kaikki nämä oppilaitokset käyttävät TRAKLA2:ta TKK:lla sijaitsevalla yhteisellä palvelimella, joten data on tarpeen tullen helposti saatavilla, ja TKK:n kursseja varten tehty ETL-prosessi vaatii vain pientä hiomista uuden datan mukaan liittämiseksi.

TKK:lla on aikaisemmin käytetty itse muokattua Ceilidh-tarkastusjärjestelmää ohjelmointikursseilla. Nykyään sitä ei juurikaan käytetä, sillä Goblin on kehittyneempänä korvannut järjestelmän. Vanha data on kuitenkin enimmäkseen tallessa, ja sen liittäminen tietovarastoon voisi tulla jatkossa kysymykseen. Koska tieto ei ole enää tuoretta, sen hyödyllisyys on kuitenkin kyseenalaista. Suurin poikkeavuus muihin järjestelmiin verrattuna

tietovarastoinnin näkökulmasta on, että kaikki data on säilötty tiedostojärjestelmään ja lähinnä tekstimuotoisena.

Viimeisenä voidaan mainita TAPAS-projektin tulevat hedelmät. Tarkoituksena projektissa on luoda ohjelmointiin liittyville teksteille automaattisia tarkastimia, esim. dokumentaation tai pseudokoodin arviointiin. Tässä vaiheessa niiden yksityiskohdista ei ole kuitenkaan mitään tietoa, joten suunnittelussa on vaikea varautua niiden tarpeisiin. Voidaan kuitenkin olettaa, että ne pohjautuvat jotenkin Joensuun yliopiston AEA-järjestelmään.

3.5 Tietosuojakysymykset

Tietovaraston käyttämä lähdedata sisältää opiskelijoiden henkilötietoja sekä tietoa heidän opiskelumenestyksestään. Henkilötunnuksia mikään järjestelmä ei kuitenkaan sisällä, vaan pelkästään nimiä ja opiskelijanumeroita. Lait ja korkeakoulun omat säännöt säätelevät silti niiden käyttömahdollisuuksia. Henkilötietolaissa todetaan henkilötietojen käsittelyä koskevista yleisistä periaatteista seuraavaa:

> – – Myöhempää henkilötietojen käsittelyä historiallista tutkimusta taikka tieteellistä tai tilastotarkoitusta varten ei pidetä yhteensopimattomana alkuperäisten käsittelyn tarkoitusten kanssa. (Henkilötietolaki 1999, 7 §.)

Tietovaraston tarkoitus on auttaa kehittämään opetusmenetelmiä tilastollisen analyysin ja tieteellisen tutkimuksen kautta. Niinpä tietojen käyttötarkoituksen voidaan katsoa sopivan edelliseen lain määritelmään, vaikka alun perin tietoja kerättäessä tämän projektin tarpeet eivät ole olleet tiedossa, jolloin lupaa tietojen käyttämiseen ei ole voitu erikseen kysyä. Alkuperäinen tarkoitushan on useimmissa tapauksissa opiskelijoiden suoritustietojen käsittely ja säilyttäminen tutkinnon suorittamista varten. Toisaalta joidenkin kurssien yhteydessä erillinen lupa tietojen käyttämiseen tutkimustarkoituksiin on saatettu kysyä.

Varsinaisilla yksilöivillä tunnistetiedoilla (nimi ja opiskelijanumero) ei ole mitään käyttöä tilastollisen tutkimuksen kannalta. Joissakin yhteyksissä on kerätty muitakin tietoja, kuten ikä tai sukupuoli, mutta näiden avulla opiskelijoita ei

pystytä yksilöimään. Jonkinlainen yksilöintikeino tietovarastoon kuitenkin tarvitaan, jotta voitaisiin verrata samojen henkilöiden suorituksia eri kursseilla tai eri vuosina samalla kurssilla. Tätä varten opiskelijanumerot siirretään tietovarastoon, mikä on mahdollista, sillä tietovarastoa alkuvaiheessa käyttävillä henkilöillä on muutenkin oikeus käsitellä opiskelijoiden suoritustietoja. Myöhemmin tarpeen vaatiessa opiskelijanumeroista voidaan laskea salaisella yksisuuntaisella hajautusfunktiolla suuria kokonaislukumuotoisia tunnisteita.

Koska ETL-prosessissa ja tietovarastossa käsitellään todellisia henkilötietoja, on pääsy niihin tietoturvasyistä rajattava tarkasti. Tämä on tärkeä syy, miksi loppukäyttäjät saavat käsitellä vain lopullista tietovarastoa, ja sinnekin heillä riittää lukuoikeus. Lisäksi on pohdittava teknisiä turvallisuustekijöitä. Verkon yli siirrettävä data poimintavaiheessa voi olla syytä salata siirron ajaksi. Tiedostojärjestelmässä, tietokannassa sekä palomuurissa on muistettava asettaa käyttöoikeudet riittävän tiukoiksi. Erikoistapauksen muodostaa valtavasti opiskelijoiden kurssisuoritustietoja sisältävä OSR-lähdejärjestelmä. Sen tietoturvallisuus on määritetty niin tiukaksi, ettei järjestelmään saa tietovarastoa varten ottaa suoraa yhteyttä ollenkaan, vaan asia on hoidettava ottamalla ajoittain tilannekuvia koko tietokannasta OSR:n ylläpitäjän toimesta.

4 Menetelmät ja työkalut

Kun työn lähtökohdat ja vaatimukset on esitelty, on aika pohtia käytettäviä menetelmiä ja työkaluja. Ensimmäiseksi käydään läpi prosessia, jolla tietovarasto luodaan. Tämän jälkeen siirrytään teknologiavalintoihin tiedonhallintajärjestelmän ja muiden sovellusten osalta.

Työssä käytetään lähes yksinomaan avoimen lähdekoodin ohjelmistoja, ja tähän on useita perusteluja. Ensimmäinen on tietysti lisenssien ilmaisuus, ja toinen on riippumattomuus yhdestä valmistajasta. Samalla ohjelmistojen kehitykseen on mahdollista vaikuttaa itse, ja sitä on helppo seurata. Ohjelmistot pyrkivät käyttämään enemmän avoimia standardeja (esim. XML), mitä kaupallisissa ohjelmistoissa tosin paikataan tarjoamalla erikseen tehtyjä liitäntöjä muihin sovelluksiin. Tässä tapauksessa myös laboratorion työntekijöillä on eniten kokemusta avoimen lähdekoodin sovelluksista varsinkin tietokantojen osalta, sillä lähdejärjestelmätkin pohjautuvat enimmäkseen avoimen lähdekoodin tietokantoihin. Merkittävimpänä haittana on, että tällaiset ohjelmistot eivät ole usein yhtä pitkälle kehitettyjä, niille ei tarjota yleensä tukipalveluita, eikä tiettyihin erikoistarkoituksiin ole kunnollisia työkaluja tarjolla ollenkaan. Varsinkin tietovarastointiin niitä pidetään vielä melko kehittymättöminä (Horstmann 2005, 33–34).

Poikkeuksena avoimen lähdekoodin käytöstä ovat Windows-käyttöjärjestelmä sekä Excel-taulukkolaskentaohjelma. Työ kehitetään Windows-alustalla, ja tulevaisuudessa mahdollisesti tapahtuvia alustanvaihtoja varten pidetään silmällä myös siirtymistä Linuxiin. Lisäksi datan analysoinnissa voi käyttää eri järjestelmää kuin tietovarastossa, sillä standardoidut tietokantaliitännät mahdollistavat siirtämisen käyttöjärjestelmien välillä helposti. Niinpä työkalut olisi oltava saatavilla molemmille käyttöjärjestelmille, ja analysointipuolella mahdollisesti myös Macintoshille. Exceliä on aiemmin käytetty kerätyn datan analysointiin, joten sen hyödyntämisestä on valmiiksi kokemusta ja osaamista. Jos siis varsinkin analysointipuolella työkaluissa on hyvät liitännät Exceliin, sitä voidaan pitää myönteisenä tekijänä. Myös Excelin käyttö itsessään voi tulla

kysymykseen sen tarjoamien tietokantayhteyksien ansiosta. Tällöin jouduttaisiin valitettavasti osittain luopumaan Linux-tuesta tämän osa-alueen kohdalla.

4.1 Tietovaraston rakentamisprosessi

Kirjallisuudessa on jonkin verran esitetty malleja koko tietovaraston rakentamisprosessille (esim. Hovi *ym.* 2001, 135–181), mutta ne ovat aivan liian raskaita näin pieneen projektiin. Niinpä niistä on pyritty vain hakemaan suuntaa ja poimimaan sopivimmat osat. Lisäksi tässä keskitytään enemmän tekniseen puoleen kuin esim. organisointiin ja sidosryhmiin, joita on hyvin vähän. Kohtien järjestys ei monilta kohdin ole tiukka, vaan niitä voidaan tehdä limittäin tai hieman eri järjestyksessä.

Tässä työssä liikkeelle lähdetään luvun 3 asioista, eli työn ominaisuuksista, erityispiirteistä ja taustoista. Erityisesti määritetään luvussa 3.3 kuvatut vaatimukset, jotka saattavat työn edetessä vielä tarkentua, ja valitaan tietovarastoon mukaan otettavat datalähteet (ks. luku 3.4). Lähteet asetetaan toteuttamisjärjestykseen, ja lisäksi varaudutaan tuleviin laajennuksiin käymällä läpi muita mahdollisia lähdejärjestelmiä. Vaiheesta syntyy vaatimusmäärittelyn ohella metadataa kunkin lähdejärjestelmän tärkeimmistä tiedoista sekä toteuttamisjärjestyksestä tietovarastossa.

Toisena suurena kokonaisuutena ovat teknologiavalinnat, jotka ovat yksi työn painopistealue. Tavallisessa projektissa käytetyt sovellukset luultavasti määräytyisivät vanhojen järjestelmien eli ostettujen lisenssien perusteella, mutta tällä kertaa on mahdollisuus lähteä liikkeelle puhtaalta pöydältä. Vaihe sisältää kaiken tietokantatekniikasta aina OLAP-sovellukseen ja aputyökaluihin. Tämän luvun loppupuoli käsittelee teknologiavalinnan eri osia.

Kun lähdejärjestelmäehdokkaat tunnetaan, on syytä analysoida niiden laatua datanprofilointityökalulla (data-profiling tool) (Kimball & Caserta 2004, 5-6, 67). Tällä tavoin saadaan selville, mikä osa datasta vaatii puhdistamista ja millä tavoin. Jos jonkin järjestelmän laatu osoittautuu liian heikoksi, se on syytä jättää

kokonaan tietovarastosta pois. Tämä voi pahimmillaan kaataa koko projektin, joten tämä vaihe kannattaa hoitaa osittain rinnakkain ensimmäisen kanssa. Laadun tarkistamisen voi myös jättää ETL-prosessin yhteyteen. Vaihe tuottaa raportin löydetyistä laatupoikkeamista sekä mahdollisesti lähdejärjestelmien kuvaukset esim. ER-kaavioiden muodossa.

Edelliset kolme kohtaa muodostavat oikeastaan pohjan tietovarastoprojektille. Tämän jälkeen suunnitellaan varsinainen tietovaraston ydin sekä ETL-prosessin looginen kuvaus. Taulukkomuotoinen looginen kuvaus kertoo tietovaraston taulujen ja kenttien määritykset, niitä vastaavat lähdejärjestelmien kentät sekä tarvittavan muunnoksen. Samalla luodaan tietovarastosta graafinen kaavio (esim. ER-kaavio), josta rakenne näkyy selkeämmin. Nämä muodostavat pohjan ETL-prosessin käytännön toteutukselle.

Toisena suunnittelun osana järjestelmälle luodaan arkkitehtuuri. Siitä nähdään korkealla tasolla järjestelmän toteutustapa ja osien keskinäiset suhteet, mitkä riippuvat pitkälti teknologiavalinnoista. Arkkitehtuuri esitetään yhtenä tai tarvittaessa useampana kaaviona.

Tietovaraston toteutus tapahtuu inkrementaalisesti, eli lähdejärjestelmät lisätään yksi kerrallaan. Toteutuksen yksityiskohdat riippuvat kuitenkin täysin käytetyistä työkaluista. Ensimmäisen järjestelmän lisäämisen jälkeen voidaan puhua prototyypistä, jota voi esitellä käyttäjille. Niinpä tietovarasto saadaan käyttöön vähitellen jo toteutuksen aikana. Käyttäjiltä saatua palautetta voidaan hyödyntää välittömästi, ja jo valmiiksi saatuun osaankin on kohtalaisen helppo tehdä muutoksia. Tästä seuraa, että myös tietovaraston testaus tapahtuu toteutuksen ja käyttöönoton rinnalla, ja loppukäyttäjätkin osallistuvat siihen jossain määrin. Testaus käsittää tässä projektissa lähinnä erilaisten kyselyiden tekemistä tietovarastoon ja datan virheettömyyden selvittämistä.

Projekti päättyy varsinaisesti luovutukseen, kun halutut lähdejärjestelmät on saatu liitettyä tietovarastoon. Tästä alkaa ylläpitovaihe, jota varten tarvittaessa

annetaan koulutusta. Tietovaraston luovuttamiseen sisältyy itse järjestelmän lisäksi dokumentaatiota ja metadataa riittävässä määrin.

Kaikkien vaiheiden lomassa käydään ajoittain läpi projektin etenemistä käyttäjien ja muiden työntekijöiden kanssa. Näin esim. teknologiavalintoihin ja suunnitteluratkaisuihin saadaan laajempaa näkemystä ja mahdolliset virheet karsiutuvat helpommin.

4.2 Tietokantavalinta

Tässä aliluvussa lähdetään käymään läpi työn teknologiavalintaa. Se keskittyy tietovaraston ytimenä toimivan tietokantatekniikan päättämiseen. Sopivan tekniikan jälkeen on päätettävä myös käytettävä sovellus, joka kyseistä tekniikkaa käyttää.

Tietokantatekniikoiden vertailu

Taulukko 11 on tehty aliluvun 2.1 asioiden perusteella sekä tutustumalla muutamien, lähinnä avoimen lähdekoodin tuotteiden ominaisuuksiin (pl. hakemistopuu tietenkin). Kohdista on mahdollista olla eri mieltäkin, varsinkin olio- tai XML-tietokantojen kohdalla, mutta erot ovat lopulta melko selviä. Vertailua varten on mietitty 12 tietovarastoinnin kannalta tärkeää ominaisuutta. Valitut ominaisuudet on ryhmitelty neljään joukkoon ja pisteytetty asteikolla yhdestä kolmeen plusmerkein.

Yksinkertaisessa hakemistopuussa on joitakin vahvuuksia, mutta kaiken ohjelmoiminen itse on turhaa ja virhealtista, kun tarjolla on paljon valmiita sovelluksia. Oliotietokannat näyttäisivät soveltuvan joihinkin erityistarkoituksiin, joissa varsinkin tunnetaan tarvittavia hakuja hyvin ennalta. Tämä ei kuitenkaan toteudu tietovarastoissa tyypillisesti. Ne myös edellyttäisivät käytännössä Java-ohjelmointia, jotta oliomallista saataisiin kaikki hyöty irti. XML-teknologia on tietokannoissa kaiken kaikkiaan vielä liian kehittymätöntä tietovarastointiin.

Taulukko 11. Tietokantatekniikoiden vertailu

Tekniikka	Hakemistopuu	Relaatio-tietokanta	Olio-tietokanta	XML-tietokanta
Yleinen kehitys	++	+++	++	+
ACID-ominaisuudet	ei	kyllä	kyllä	vaihtelee
kehittyneisyys ja vakiintuneisuus	ei kehittynyt, mutta vakiintunut muissa sovelluksissa	hyvä	keskin-kertainen	heikohko
kirjallinen materiaali (tieteelliset artikkelit, oppikirjat jne.)	paljon, mutta ei tietovaras-tointiin liittyen	paljon	vähän	vähän
Käytettävyys	++	++	+	++
osaaminen työntekijöiden keskuudessa	hyvä	hyvä	heikko	heikko
oppimisen helppous	hyvä	keskin-kertainen	keskin-kertainen	keskin-kertainen
loogisen mallin selkeys ja ymmärrettävyys käyttäjälle	hyvä hakemis-topuulla, mutta ei yleensä tiedostojen sisällä	heikohko	keskin-kertainen	hyvä
Yhteensopivuus	+	+++	+	+
yhteys lähdejärjestelmiin	ei	kyllä	ei	ei
tuki ETL-työkaluissa	hyvä	erittäin hyvä	heikko	heikko
tuki analysointi-työkaluissa	ei	erittäin hyvä	heikohko	heikohko
Tietomallin ominaisuudet	+	++	++	++
loogisen tietomallin monipuolisuus ja joustavuus	heikko	keskin-kertainen	hyvä	hyvä
tietotyyppien monipuolisuus	heikko	keskin-kertainen	erittäin hyvä	hyvä
hakujen tehokkuus	heikko, mutta riippuu toteu-tuksesta	hyvä	vain ennalta tunnetut haut nopeita	keskin-kertainen

Kuten yllä olevasta taulukosta nähdään, relaatiotietokannat ovat tasaisen vahvoja lähes kaikilla osa-alueilla. Varsinkin työkalujen suhteen ne vievät selvän voiton kaikista muista tekniikoista. Käytännössä suurin osa tietovarastoista rakennetaan relaatiotietokantojen päälle, ja kirjallisuudessakin

tätä pidetään usein itsestään selvänä lähtökohtana. Lisäksi suuri osa lähdedatasta on relaatiotietokannoissa, mikä helpottaa siirtämistä, kun muotoa ei tarvitse paljon muuttaa. Niinpä hyväksi havaitusta ratkaisusta ei tässäkään tapauksessa nähdä syytä poiketa.

Relaatiotietokantatuotteiden vertailu

Relaatiotietokannoissa on lukuisia vaihtoehtoja avoimen lähdekoodin sovellustenkin joukossa. Horstmann (2005, 23–28) on tehnyt äskettäin vertailun seuraavan viiden ratkaisun kesken: MySQL (2006), PostgreSQL (2006), Firebird (2006), Ingres (2006) ja MaxDB (2006). Tuota vertailua käytetään lähtökohtana myös tässä työssä, mutta mukana on luonnollisesti erilaisia tekijöitä. Näitä ovat erityisesti lähdejärjestelmien käytössä olevat tietokannat sekä tietovarastointityökalujen tuki.

Firebird-, Ingres- ja MaxDB-tietokantoja ei ole tiettävästi paljon käytetty tietovarastointiprojekteissa maailmalla (Horstmann 2005, 31). Vaikka niilläkin on useita vuosia ikää, ne eivät ole yleistyneet yhtä paljon kuin MySQL ja PostgreSQL. Niiden käytöstä ei työntekijöillä ole myöskään valmiiksi kokemusta. Ominaisuuksiltaan ne olisivat kuitenkin keskitasoa.

MySQL:stä ja PostgreSQL:stä kokemusta sen sijaan on, sillä molemmat ovat käytössä yhdessä lähdejärjestelmässä, MySQL Goblinissa ja PostgreSQL TRAKLA2:ssa. Molemmista on tosin tullut suuri versiopäivitys käyttöönottojen jälkeen. Perinteisesti MySQL:ää on pidetty nopeana mutta hyvin vähän ominaisuuksia sisältävänä ratkaisuna. PostgreSQL taas noudattaa standardeja hyvin pitkälti ja sisältää paljon kehittyneitä tietokantaominaisuuksia. Ero näyttäisi kuitenkin kaventuneen, kun mm. MySQL 5.0 toi vuoden 2005 lopulla mukanaan paljon uutta toiminnallisuutta. Toisaalta edes PostgreSQL ei sisällä kaikkea, mistä tietovarastoinnissa olisi hyötyä, kuten materialisoituja näkymiä. Tämä korjaantuisi käyttämällä sovelluksesta eteenpäin jalostettua Bizgres-versiota, joka on kehitetty juuri tietovarastoja silmällä pitäen. Ongelmaksi muodostuu vain sen saatavuus eri käyttöjärjestelmille, sillä tässä työssä

tietovarasto kehitetään Windows-alustalla, ja sitä Bizgres ei tue. Lisensseistä PostgreSQL käyttää hyvin sallivaa BSD-lisenssiä ja MySQL pitkälti (sisäisenkin) levittämisen kieltävää GPL-lisenssiä. Koska tietovarasto ei ole sen kaltainen kokonaisuus, jota olisi syytä jakaa kenellekään, ei tämä ole ongelma. Kun sekä MySQL:lle että PostgreSQL:lle on vielä melko hyvä tuki aputyökaluohjelmissa, muodostuu valinta näiden kahden välillä vaikeaksi. Edellä esitetyt vertailukohdat on koottu taulukkoon 12.

Taulukko 12. MySQL:n ja PostgreSQL:n vertailu

Tietokanta	MySQL	PostgreSQL
Saatavuus käyttöjärjestelmille	kaikki tavallisimmat käyttöjärjestelmät	kaikki tavallisimmat käyttöjärjestelmät
Arvioitu versio	5.0	8.1.4
Lisenssi	GPL (myös kaupallinen lisenssi tarjolla)	BSD
Työntekijöiden kokemus	kohtalainen (käytössä Goblinissa)	melko suuri (käytössä TRAKLA2:ssa)
Tuki muissa työkaluissa	hyvä	hyvä
Ominaisuudet	kohtalaiset ominaisuudet ja hyvä suorituskyky	melko laajat ominaisuudet ja melko hyvä suorituskyky; Bizgres-versiossa hyvin laajat ominaisuudet

Kun molempia tietokantaratkaisuja kokeiltiin lyhyesti käytännössä, havaittiin merkittävimmäksi eroksi PostgreSQL:n runsaammat ominaisuudet. Osa näistä (mm. monipuolisemmat taululiitokset) on sellaisia, joilla näyttäisi olevan vaikutusta tietovarastoinnin toteuttamiseen. Tällä perusteella tietokannaksi valittiin PostgreSQL.

4.3 Tietovarastointityökalut

Myös itse tietovarastointiin on tarjolla useita avoimen lähdekoodin työkaluja. Näissä voidaan tehdä jako ETL:ään ja datan analysointiin, ja tässä vertaillaan noihin tarkoituksiin tehtyjä sovelluksia sekä valitaan perustellusti sopivimmat tähän työhön. Yhteisesti kaikista työkaluista voidaan sanoa, että ne ovat melko nuoria ja aktiivisen kehityksen kohteina. Kaikista on mm. julkaistu tuore versio muutamien kuukausien sisällä. Tämä osoittaa, kuinka tuore ilmiö tietovarastointi on avoimen lähdekoodin yhteisöissä.

Kolmas osa-alue olisi datan puhdistus ja profilointi, joka tulisi järjestyksessä ensimmäisenä. Siihen ei kuitenkaan löytynyt kunnollisia ilmaisia työkaluja. Niinpä lähdejärjestelmien datan laatua joudutaan arvioimaan kevyemmin mm. silmäilemällä data-arvoja ja tekemällä joitakin yksinkertaisia datan ominaisuuksia luotaavia SQL-kyselyitä. Osa virheistä tulee esiin varmasti myös ETL-prosessin aikana, jolloin ne on vielä nopeaa korjata.

ETL

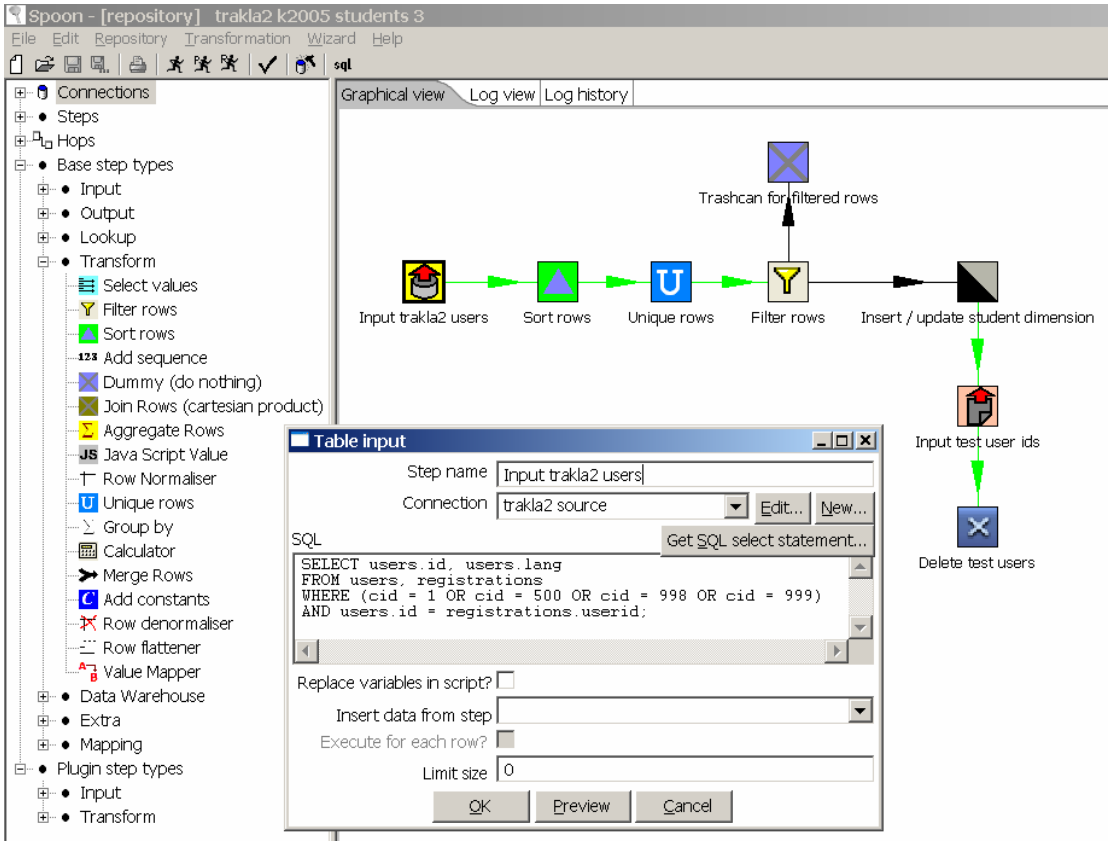
Kuten aiemmin on jo todettu, ETL-prosessi on tietovarastoinnin yleensä selvästi työläin vaihe. Niinpä sopivan työkalun löytäminen vaikuttaa kriittisesti sekä projektin onnistumiseen että kestoon. Taulukko 13 vetää yhteen kolmen varteenotettavimman työkalun ominaisuuksia. Tärkeimmiksi nousevat saatavuus halutuille käyttöjärjestelmille sekä käyttöliittymän helppokäyttöisyys ja opittavuus. Jälkimmäinen on olennaista siksi, että tulevien ylläpitäjien on kyettävä oppimaan ja käyttämään työkalua ilman kohtuutonta työmäärää. Tästä syystä myös dokumentaatiolle asetetaan painoarvoa.

Taulukko 13. ETL-työkalujen ominaisuuksia

Työkalu	KETTLE	Clover.ETL	Enhydra Octopus
Saatavuus käyttöjärjestelmille	Windows+Linux (Java-pohjainen)	Windows+Linux (Java-pohjainen)	Windows+Linux (Java-pohjainen)
Arvioitu versio	2.3.1	1.8.3 + 1.3 (Eclipse-plugin)	3.6
Lisenssi	LGPL	LGPL	LGPL
Käyttöliittymä	graafinen sovellus	Java-kirjasto eli vaatii Java-ohjel-mointia; erikseen tarjolla graafinen käyttöliittymä Eclipse-pluginina	määritykset XML:ää ja tehtävä enimmäkseen käsin; jotkut asiat määritettävissä graafisesti
Dokumentaatio	PDF-muodossa, mutta ominaisuu-det esitellään melko suppeasti	Javadoc-dokumen-taatio; graafiseen käyttöliittymään opetusvideoita	PDF- ja HTML-muodoissa
Muuta	paljon ominai-suuksia, joissa kuitenkin vielä kehittämisvaraa ja pieniä bugeja	graafinen käyttö-liittymä hitaampi ja monimutkaisempi kuin KETTLE:ssä ja tarjoaa vähemmän ominaisuuksia	hankalin käyttää

Jokainen työkalu on saatavissa ainakin Windowsille ja Linuxille, ja Java-pohjaisuuden ansiosta käytännössä muillekin käyttöjärjestelmille tarvittaessa. Kaikissa on myös melko salliva LGPL-lisensointi, joten tämän asian suhteen ei tule ongelmia. Eniten eroja on käyttöliittymässä, ja se on KETTLE:ssä (2006) kehittynein. Ruudulle vedetään melko kattavasta listasta palikoita, jotka käsittävät datan lukemista eri lähteistä, muotoilua, yhdistämistä, karsintaa ja paljon muuta. Näille palikoille määritetään datan mukaiset ominaisuudet ja niitä yhdistellään sopivaan järjestykseen suunnatuksi graafiksi (kuva 11). Clover.ETL:n (2006) Eclipse-pluginina saatava graafinen käyttöliittymä on samankaltainen, mutta ominaisuuksia on tarjolla selvästi vähemmän. Lisäksi asetuksia joudutaan syöttämään käsin huomattavasti enemmän, mikä tekee käytöstä monimutkaisempaa. KETTLE sen sijaan on automatisoitu paljon pidemmälle, ja mm. käyttöliittymäkomponentit osaavat automaattisesti lukea järjestyksessä edellisestä palikasta tulevat datakentät ja näyttää ne käyttäjälle. Clover.ETL:ää ei kokeiltu ohjelmoinnin kautta, jolloin käyttömahdollisuudet olisivat saattaneet olla monipuolisemmat. Enhydra Octopusissa (2006) graafinen käyttöliittymä on tarjolla lähinnä tietokantayhteyksien määrittämiseen, ja varsinainen datan käsittely on tehtävä kirjoittamalla käsin XML-määrityksiä. Työkalun käyttäminen esim. ohjelmoimalla Java-kirjastona on myös mahdollista.

Sekä KETTLE:en että Octopusiin on tarjolla tavallinen käyttöohje. Clover.ETL:ään sen sijaan on Javadoc-dokumentaatio sekä graafiseen käyttöliittymään opetusvideoita, jotka käyvät läpi työkalun perusominaisuuksia. Jälkimmäinen on hyödyllinen varsinkin Eclipseä ennestään tuntemattomalle.



Kuva 11. KETTLE:n käyttöliittymä

Octopus vaikuttaa selvästi hankalimmalta käyttää, sillä graafista käyttöliittymää ei varsinaisesti ole. Lyhyen kokeilun aikana Clover.ETL tuntui monimutkaisuuden lisäksi KETTLE:ä hitaammalta käyttöliittymän osalta. Vaikka KETTLE tarjoaa paljon ominaisuuksia, siinäkin on vielä kehitettävää, ja kokeilukäytössä tuli vastaan joitakin pieniä vikoja. Näistä heikkouksista huolimatta se vaikutti selvästi helppokäyttöisimmältä, joten se päätettiin valita tähän projektiin. Versiosta 2.3.0 alkaen KETTLE:n nimi muuttui muotoon Pentaho Data Integration seurauksena sulautumisesta avoimen lähdekoodin liiketoimintasovelluksia kehittävään Pentahoon.

Neljäntenä olisi ollut olemassa BEE-niminen työkalu (BEE 2006), mutta sitä ei ole tarjolla Windowsille ollenkaan. Tämän vuoksi se jätettiin vertailun ulkopuolelle. Myös ETL:n toteuttaminen kokonaan itse olisi ollut mahdollista. Tämä vaihtoehto olisi muodostunut kuitenkin turhan työlääksi, joten lähtökohdaksi otettiin valmiin työkalun käyttäminen.

Datan analysointi

Datan analysointityökalut määrittävät, miten käyttäjät voivat lopulta hyödyntää tietovarastoon kerättyä tietoa. Tällä alueella avoimen lähdekoodin sovelluksissa on hyvin vähän valinnan varaa, joten vaihtoehtona on syytä tarkastella myös muita keinoja päästä dataa käsiksi. Varsinaiset OLAP-sovellukset, jotka tässä tulevat kysymykseen, ovat Mondrian (2006) ja Palo (2006). Näiden rinnalla pohditaan Excel-taulukkolaskentaohjelman, sitä vastaavan ilmaisen OpenOffice Calcin sekä aikaisemmin Ohjelmistotekniikan laboratoriossa diplomityönä tehdyn visuaalisen SEEQ-tietokantaselaimen (Rontu 2004a; Rontu 2004b; Rontu *ym*. 2006) käyttöä tiedon hakemiseen. Näiden viiden ominaisuuksia on kerätty taulukkoon alla.

Taulukko 14. Analysointisovellusten ominaisuuksia

Työkalu	Mondrian	Palo	Excel	OpenOffice Calc	SEEQ
Saatavuus käyttö-järjestelmille	Windows+ Linux	Windows+ Linux (Excel-plugin vain Windowsille)	Windows+ Macintosh	kaikki	kaikki (Java-pohjainen)
Arvioitu versio	2.1	1.0c	Excel 2003	2.0.4	release 1
Lisenssi	CPL	GPL	kaupalli-nen	LGPL	ei varsinaista lisenssiä; vapaassa levityksessä
Tyyppi	ROLAP (relaatio-malliin pohjautu-va OLAP)	MOLAP (moniulottei-seen malliin pohjautuva OLAP)	taulukko-laskenta	taulukko-laskenta	visuaalinen tietokanta-selain
Vaatimukset	Tomcat-palvelin	plugin vaatii Windowsin ja Excelin	ODBC-ajuri	JDBC- tai ODBC-ajuri	JDBC-ajuri
Käyttö-liittymä	web-pohjainen	itse ohjel-moitavissa ohjelmointi-rajapintojen avulla; Excel-plugin sulautuu osaksi Exceliä	tietokanta-hakuihin graafinen MS Query	tietokanta-hakuihin graafinen sovellus	itsenäinen graafinen sovellus

Mondrian perustuu relaatiotietokantaan ja tarjoaa moniulotteisen näkymän dataan webin kautta. Se pyörii Tomcat-palvelimen päällä websovelluksena. Datan esitysmuoto määritetään XML:llä, ja hauissa käytetään Microsoftin kehittämää MDX-kieltä, joka muistuttaa rakenteeltaan hieman SQL:ää. Mondrian on saatavilla Windowsille ja Linuxille. Sovellus vaikuttaa hieman raskaalta käyttää, sillä se vaatii erikseen Tomcatin asentamista, ja XML-määritykset on kirjoitettava käsin. Myönteisenä seikkana web-sovelluksena se olisi käytettävissä mistä tahansa.

Palo käyttää puolestaan moniulotteista tietokantaa, ja se toimii myös Windowsilla ja Linuxilla. Vahvimpana puolena siinä on Windowsille tarjolla Excel-plugin (joka ei tosin ole avoimen lähdekoodin sovellus mutta freewarea kuitenkin). Käytännössä plugin lisää Excelin kaksiulotteisiin taulukoihin lisää ulottuvuuksia. Plugin ei käytä makroja eikä vaadi ohjelmointia, joten se on varsin helppokäyttöinen. Valitettavasti samaa pluginia ei ole tehty Macintoshille, ja Linuxilla joudutaan turvautumaan pelkästään ohjelmointirajapintoihin esim. Javalle tai PHP:lle.

Suurin etu käyttäjälle sekä Mondrianissa että Palon Excel-pluginissa olisi siinä, että SQL:ää ei tarvitsisi osata yhtään, toisin kuin kaikissa muissa vaihtoehdoissa. Tämä on myös periaate, jolla OLAP-puoli tietovarastoissa yleensä toteutetaan. Tällöin tehtävät kyselyt pitäisi kuitenkin tuntea melko hyvin etukäteen, mikä ei tässä projektissa toteudu. Niinpä mahdollisuudet näiden työkalujen käyttöön ainakin alkuvaiheessa ovat hyvin rajalliset.

Excel on kaupallinen, mutta se käytännössä yleensä sisältyy Windowsiin ja on saatavana myös Macintoshille. Sen suurimmat hyödyt ovat aikaisempi käyttökokemus analysoinnissa sekä helpotettu datan hakeminen tietovarastosta, kun SQL-kyselyitä ei tarvitse kirjoittaa kokonaan käsin. Exceliä vastaava ilmainen OpenOffice Calc olisi tarjolla myös Linuxille, mutta sen tietokantayhteys vaikutti kokeiltaessa niin hitaalta ja kehittymättömältä, ettei siitä

luultavasti olisi riittävää hyötyä käyttäjälle. Sen käyttöliittymä on kuitenkin hyvin samankaltainen kuin Excelin, joten opettelu olisi melko vaivatonta.

SEEQ:llä tietokantaan voidaan tehdä kyselyitä täysin graafisesti, eikä se vaadi juurikaan etukäteisvalmisteluja kuten OLAP-sovellukset. Ohjelmassa on kuitenkin rajoituksia, ja esim. suurien datamäärien hakeminen ja analysointi eivät kunnolla onnistu. Lisäksi käyttöliittymässä on vielä hiottavaa, ja SEEQ:n käyttäminen vaatii erikseen koulutusta. Vahvuutena taas SQL-kieltä ei periaatteessa tarvitse osata, ja ohjelmaa voisikin parhaiten hyödyntää selvittämään käyttäjälle tietokannan rakennetta ja datasisältöä tiedonhaun alkuvaiheessa. Vaikka kyselyitä ei tarvitsekaan kirjoittaa käsin, tuntuu monimutkaisten kyselyiden tekeminen käytännössä silti vaativan SQL:n periaatteiden ymmärtämistä.

Osa tarkastelluista sovelluksista vaatii tietyn käyttöjärjestelmän toimiakseen, ja toisaalta ne on tehty osittain erilaisiin tarkoituksiin. Tämän vuoksi ei ole järkevää valita niistä mitään yksittäistä tietovaraston analysointityökaluksi. Sen sijaan kukin käyttäjä voi valita sopivan työkalun omien tarpeidensa ja mieltymystensä mukaan. Ratkaisu on perusteltu myös työn tavoitteiden kannalta, sillä tiedon analysointiin ei ollut tarkoitus kiinnittää yhtä paljon huomiota kuin muihin osiin.

4.4 Muut työkalut

Relaatiokaavioiden suunnitteluun ja visualisointiin käytettiin DBDesigner 4:ää. Se on helppokäyttöinen graafinen sovellus, joka myös tekee suunnitelluille tauluille valmiiksi SQL-luontilauseet. DBDesigner 4:ää ei ole kehitetty enää muutamaan vuoteen, mikä näkyy lähinnä ongelmina luettaessa uusien tietokantaversioiden rakennekuvauksia.

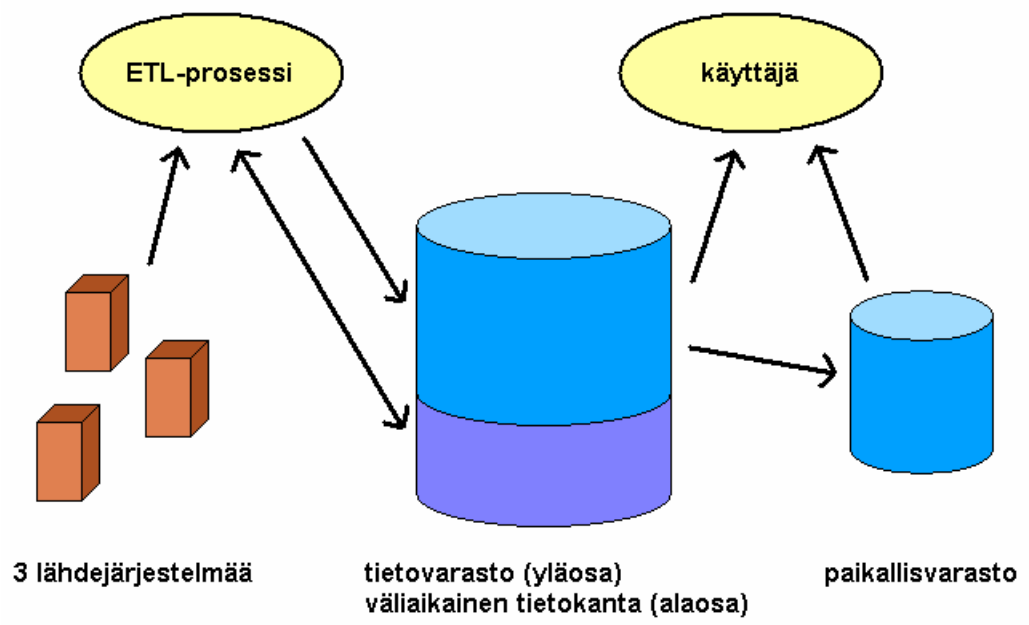
5 Toteutus

Tässä luvussa esitellään koko järjestelmän fyysinen ja tietovaraston sisäinen looginen arkkitehtuuri. Lisäksi käydään läpi testausta, käyttöönottoa ja ylläpitoasioita.

5.1 Arkkitehtuuri

Hovi *ym*. (2001, 66–70) esittelevät kolme eri perusarkkitehtuuria tietovarastoille. Ns. inmonilaisessa (Bill Inmonin mukaan) tietovarastossa on yksi yhteinen keskusvarasto, josta tarvittaessa johdetaan paikallisvarastoja. Toisena vaihtoehtona ns. kimballilainen (Ralph Kimballin mukaan) tietovarasto sisältää joukon yhdenmukaisia paikallisvarastoja, joissa yhtenäisyys tulee erityisesti paikallisvarastoille yhteisten ulottuvuustaulujen kautta. Kolmas ja riskialttein vaihtoehto ovat täysin erilliset paikallisvarastot. Tällöin tietojen keskinäistä eheyttä on vaikea enää valvoa. Koska tässä työssä ei alustavasti käytetä juurikaan paikallisvarastoja, noudattaa toteutettu tietovarasto lähinnä inmonilaista mallia.

Korkealla tasolla tietovaraston arkkitehtuuri näyttää kuvan 12 mukaiselta. Käytössä on erillisiä datalähteitä, jotka ovat enimmäkseen tietokantoja, mutta osittain myös tavallisia tiedostoja. Näitä luetaan ja työstetään ETL-prosessissa KETTLE-sovelluksesta käsin. KETTLE itse on hyvä sijoittaa samalle tietokoneelle ainakin tiedostomuotoisten lähteiden kanssa. Sovellus käyttää väliaikaisena tallennuspaikkana samaa tietokantaa kuin varsinainen tietovarasto. Taulut ovat toki erillisiä, ja vasta valmis tieto kirjoitetaan tietovaraston tauluihin. Tietovarasto on loppukäyttäjien ulottuvilla mm. SEEQ:n tai Excelin tietokantayhteyksien kautta. Osa tiedosta voidaan valita edelleen Palon muodostamaan paikallisvarastoon. Paikallisvaraston tehtävä tässä vaiheessa on lähinnä tuoda esiin erilainen tapa käsitellä ja hyödyntää tietoa. Palon moniulotteisesta tietokannasta käyttäjät saavat vastaavasti haettua tietoa Excel-pluginin avulla taulukkolaskentaohjelmaan.



Kuva 12. Tietovaraston arkkitehtuuri

Fyysisesti tietovaraston voi sijoittaa erilliselle koneelle tai yhteiselle koneelle jonkin muun järjestelmän kanssa. Pienen koon ja kuorman takia järjestelmien liittäminen samalle koneelle on mahdollista. Tällöin on vain pidettävä huoli, ettei tietovaraston lataaminen häiritse muiden järjestelmien käyttöä. Palon paikallisvarasto voidaan sijoittaa mille tahansa Windows-koneelle, eikä se vaadi erityisiä resursseja tai jatkuvaa päällä olemista. Niinpä esim. käyttäjät voivat asentaa Palon omille tietokoneilleen niin halutessaan. Joka tapauksessa tietovaraston osat sijoitetaan laboratorion tietokoneille, ja käyttäjät pääsevät tietoihin käsiksi vain sisäverkon kautta.

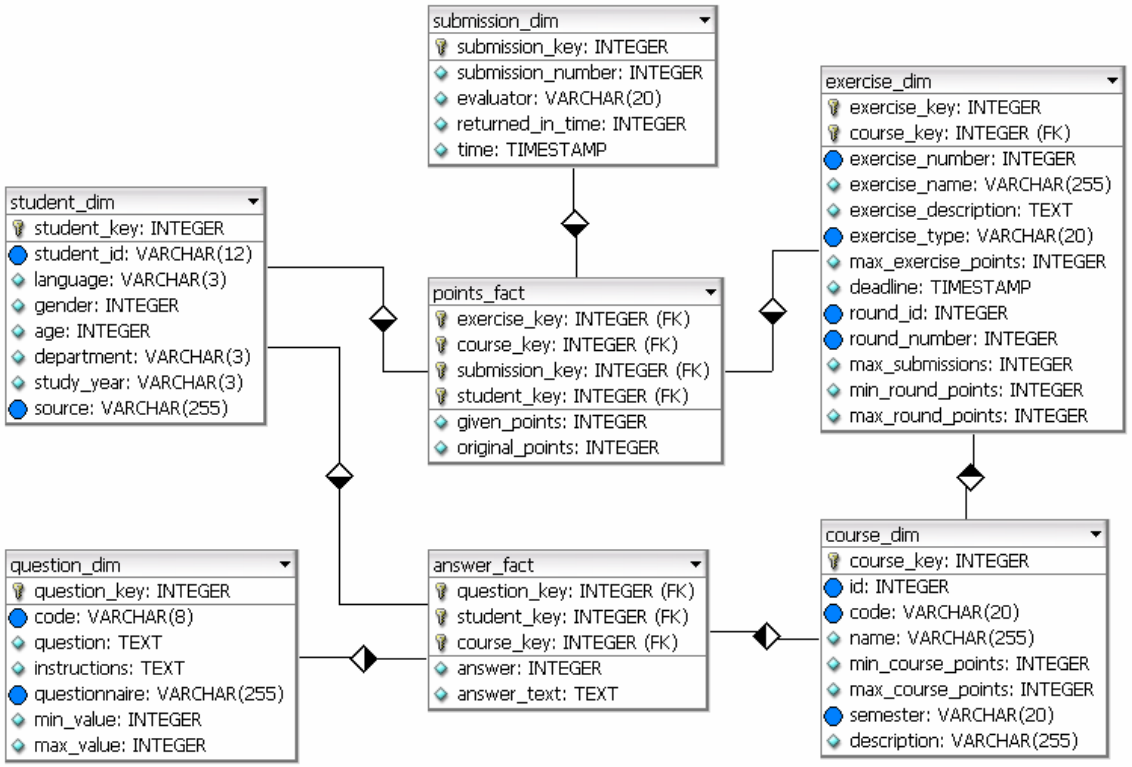
Tietovaraston latausta ei suoriteta automaattisesti tietyin välein, vaan se käynnistetään aina käsin tarvittaessa, sillä uusia dataeriä valmistuu vain muutaman kerran vuodessa. Ladatessa valitut lähteet käydään aina kokonaan läpi, ja muutokset kirjoitetaan tietovarastoon. ETL:n eri lähdejärjestelmien välillä vallitsee horisontaalinen riippuvuus, koska ne ovat täysin erillisiä, eli ne voidaan ladata valinnaisessa järjestyksessä tai yksitellen. Samoin eri data järjestelmien sisällä on yleensä riippumatonta lukukausien suhteen. Virhetilanne pysäyttää ETL-prosessin. Kun ongelma on ratkaistu, on koko prosessi käynnistettävä uudestaan kyseisen lähdejärjestelmän tai sen osan osalta. Aloittaminen alusta

on hyväksyttävää, sillä kokonaisen lähdejärjestelmänkin latausaika on korkeintaan reilut kymmenen minuuttia, ja usein voidaan aloittaa alusta vain jokin pienempi osaprosessi. Mukaan otettavalle datalle valittiin melko hieno karkeisuus, eli kaikki hyödyllinen matalan tason data otetaan mukaan, mikä on helppoa datan pienen kokonaismäärän ansiosta.

Tietovarastossa käytetään yleisesti suosittua dimensiomallia kyseenalaistamatta sitä erityisemmin. Yksi vaihtoehto olisi ollut lumihiutalemalli, mutta mm. datan pieni määrä puoltaa pientä taulujen määrää ja yksinkertaista suunnittelua. Toisena mahdollisuutena suunnittelu olisi voitu tehdä tavallisen tietokannan tavoin, mutta alan kirjallisuus kannustaa yleisesti käyttämään tietovarastoille luotuja malleja. Tietovaraston skeeman suunnittelu ei noudattanut mitään tarkkaa menetelmää. Tähän ei ollut tarvetta, koska aihealue oli melko suppea, esim. matalan tason faktat olivat hyvin ilmeisiä, ja inkrementaalisen toteutuksen ansiosta muutosten ja korjausten tekeminen oli alussa helppoa. Tietovaraston attribuuteista muutamia on indeksoitu sillä perusteella, että niitä hyvin todennäköisesti tarvitaan useimmissa kyselyissä.

Lopullinen skeema on esitetty kuvassa 13. Faktatauluja on kaksi (points_fact harjoitustehtävien pisteille tai arvosanoille ja answer_fact kyselyiden vastauksille), joista kummallakin on 3-4 dimensiota. Kurssiulottuvuus erotettiin tehtäväulottuvuuden alidimensioksi, koska kurssi on itsessään myös kyselyvastausten ulottuvuus. Ulottuvuustauluissa suuremmilla ympyröillä merkityt kentät muodostavat luonnollisen avaimen, jonka avulla kukin rivi voidaan yksilöidä. Poikkeuksena on tehtäväpalautuksen ominaisuuksia kuvaava submission_dim, jonka rivien ei tarvitse olla yksilöitävissä. Useimmiten tämän taulun riveillä on 1:1 suhde pistefaktojen kanssa. Hitaasti muuttuvien dimensioiden suhteen kaikki taulut ovat tyyppiä 1 (vanhat tiedot korvataan uusilla) paitsi opiskelijaulottuvuus. Opiskelijatiedot muodostavat hieman mukaillun tyypin 2 (uudelle tiedolle muodostetaan uusi rivi) ulottuvuuden. Niinpä samalla opiskelijalla voi olla useita rivejä taulussa. Syynä tähän on se, että mm. vuosikurssi ja koulutusohjelma voivat muuttua kurssista ja vuodesta toiseen.

Kunkin kurssin datassa on kuitenkin mielenkiintoista se, mikä attribuutin arvo on ollut kyseistä kurssia suoritettaessa.



Kuva 13. Tietovaraston tietokantaskeema

Nyt opiskelijadimensiossa otetaan huomioon vain yksittäiset opiskelijat, vaikka osa tehtävistä tehdään ryhmätöinä. Tämä saattaa tulevaisuudessa muuttaa kyseistä ulottuvuutta tai synnyttää uuden ulottuvuuden sen yhteyteen. Tehtävä- ja kurssidimensioissa puolestaan voidaan pohtia tarvetta erottaa kierrokset tehtävätaulusta tai kurssi-instanssit kursseista, jolloin siirryttäisiin lähemmäksi lumihiutalemallia. Kolmantena muutostekijänä TAPAS-projektin tulevat tekstinarviointijärjestelmät vaativat todennäköisesti lisää kenttiä joihinkin tauluihin tai kokonaan uusia ulottuvuuksia, mikä ei ole kuitenkaan ongelma hienovaraisten muutosten ansiosta.

Koska samanaikaisia käyttäjiä ei ole, ei siitä aiheudu kyselyjä hidastavaa tekijää, mikä vähentää summataulujen tarvetta. Tällä hetkellä ainoat valmiiksi lasketut arvot ovat tehtäväkierrosten ja kurssien maksimipisteet.

Tulevaisuudessa summadataksi voitaisiin luoda toinen pisteitä vastaava faktataulu, johon on laskettu yksittäisten pistemäärien sijaan esim. tehtäväkohtaisia keskiarvoja tai kierroskohtaisia pistejakaumia.

Alkuperäisistä suunnitelmista metadatan tuottamisessa poikettiin vähän. Käyttäjiä varten luotiin oma käyttöohje, joka sisältää ohjeita päästä tietovarastoon käsiksi ja selittää tietovaraston taulujen sisältöä. Tähän sisältyy kaikkein tärkeimpänä tietokantaskeema. Suurin osa metadatasta on kuitenkin järjestelmän ylläpitäjää ja kehittäjää varten. Siihen sisältyy luettelo alkuperäisistä datalähteistä (ja miten niihin pääsee käsiksi), KETTLE:n työtiedostot (jotka korvaavat erillisen kuvauksen siitä, mitä datalle on tehty), lokit KETTLE:n eli ETL-prosessin ajamisesta, hierarkkiset taulukot KETTLE:llä tehdyistä ETL-määrityksistä, ylläpitokäyttöohje sekä tietysti myös tietokannan skeema. Perusteena valita juuri nämä asiat metadataksi on se, että ne ovat työn edetessä vaikuttaneet tärkeimmiltä dokumentointikohteilta.

5.2 Testaus

Tietovaraston data oikeellisuuden tarkastaminen on tietovarastoinnin tärkeimpiä testauskohteita. Hovi *ym.* (2001, 114–115) esittävät muutamia hyödyllisiä tapoja tähän. Yksinkertaisinta on latauksen jälkeen tarkistaa, että tietovaraston tauluihin on tullut järkevän näköinen määrä rivejä. Siivoamisen ja yhdistelyn vuoksi tarkkaa vertailua operatiivisiin järjestelmiin voi olla kuitenkin vaikea tehdä. Toiseksi tauluista voi katsoa pieniä otoksia ja tarkistaa, että tiedot näyttävät järkeviltä ja mitään ei puutu. Tässä voi käyttää apuna ryhmittelyä (SQL:n group by), jolloin virheelliset arvot tulevat usein joko alkuun tai loppuun. Lisäksi datasta voidaan laskea joitakin summa-arvoja tai tilastollisia tunnuslukuja, ja varmistua niiden järkevyydestä. Testauksessa voidaan käyttää apuvälineinä mm. SQL-kyselyitä suoraan tietokantaan, datan hakemista Exceliin tai jopa OLAP-työkaluja.

Käytännössä tärkein osa testauksesta suoritettiin ETL-prosessin luomisen yhteydessä. Suurin osa lähdedatasta silmäiltiin läpi varsinkin pienempien

taulujen ja tiedostojen osalta. Näin selvästi virheellinen data pystyttiin suoraan karsimaan ETL-prosessissa. Testiajojen yhteydessä katsottiin, että kohdetauluihin tuli suunnilleen oikea määrä rivejä ja että ajon suorittaminen kaksi kertaa peräkkäin ei tehnyt mitään muutoksia toisella kerralla. Usein faktatauluja täytettäessä puuttuvat avaimet eli viittaukset ulottuvuustauluihin myös paljastivat virheitä.

ETL-prosessin tehokkuutta pystyttiin selvittämään katsomalla kunkin vaiheen suoritusaikaa. Suoritusaikojen kasvaessa suuriksi prosessia saatiin useissa tapauksissa optimoitua nopeammaksi. Apuna käytettiin mm. taulujen indeksointia ja eri vaiheiden suoritusjärjestyksen vaihtamista. Osittain suoritusaikaan liittyen pieni osa järjestelmää asennettiin myös toiselle tietokoneelle, jotta järjestelmän siirrettävyydestä voitaisiin varmistua. Käytännössä siirto vaatii jonkin verran ylimääräistä työtä ja määritysten muokkaamista. Toinen testilaitteisto oli vanhempi ja hitaampi Macintosh, ja ETL-prosessin suoritusaika kasvoi tällöin moninkertaiseksi. Tässä tilanteessa vaatimus ETL-prosessin puolen tunnin suoritusajasta ei olisi täyttynyt, joten laitteistoon on myös kiinnitettävä huomiota.

Koska tietovarasto toteutettiin inkrementaalisesti osa kerrallaan, voitiin toteutuksen loppupuolella myös käyttäjien kokemuksia hyödyntää datan oikeellisuuden tarkastamiseen. Tätä kuitenkin haittasivat huomattavasti ongelmat päästä tietovarastoon käsiksi, mutta lyhyiden kokeilujen pohjalta käyttäjät eivät ilmoittaneet löytäneensä virheitä.

Myös joitakin muita testaustapoja pohdittiin, mutta ne osoittautuivat liian työläiksi. Samaa tietoa hakevia kyselyitä olisi voitu suorittaa sekä lähdetietokantoihin että tietovarastoon. Tulokset olisivat kuitenkin poikenneet toisistaan jonkin verran, koska dataa on siivottu ja muutettu eri muotoon välissä. Toinen mahdollisuus olisi ollut luoda käsin testidataa ja tarkistaa, käsitteleekö ETL-prosessi sen oikein. Keinotekoisen datan luominen olisi kuitenkin vienyt kohtuuttomasti aikaa.

Testauksesta voidaan lopputuloksena sanoa, että paljon selvästi sotkuista dataa saatiin korjattua ja testikäyttäjien kokeiluja poistettua. Tietovaraston dataan on saattanut jäädä lähinnä lähdejärjestelmien testikäyttäjiä, jotka käyttivät aidonnäköisiä tunnuksia. Eniten resursseja käytettiin TRAKLA2-datan siivoamiseen, sillä siinä oli paljon pieniä taulukoita ja tiedostoja, joita oli luontevaa ja tarpeellista selata läpi pelkästään ETL-prosessin toteuttamisen vuoksi. Lisäksi kyseinen data oli kokeilukäytön kannalta mielenkiintoisinta. Niinpä tietovaraston TRAKLA2-data on todennäköisesti virheettömämpää kuin muiden.

5.3 Käyttöönotto

Käyttöönotto suunniteltiin suoritettavaksi inkrementaalisesti osittain järjestelmän kehittämisen kanssa rinnakkain. Eri lähteet liitettiin tietovarastoon seuraavassa järjestyksessä:

- TRAKLA2 TKK kevät 2005
- TRAKLA2 TKK muut osat
- Goblin
- OSR

TRAKLA2-järjestelmässä oli järjestetty keväällä 2005 erityinen koeasetelma, ja kyseisen datan analysointi oli tietovarastoa luodessa ajankohtaista. Datan hyödyntämisessä tuli kuitenkin vastaan pieniä ongelmia, sillä käyttäjät eivät päässeet siihen käsiksi aivan riittävän helposti. Niinpä käytettävyyden jättäminen pienemmälle huomiolle projektissa kostautui hieman. Kaikki muukin TRAKLA2-data oli liitetty tietovarastoon, ennen kuin koekäyttö päästiin lopulta aloittamaan.

Alkuvaiheessa Goblinin tai OSR:n dataa ei ole käytetty juuri lainkaan. Niinpä tietovaraston käyttöönoton onnistuminen selviää lopullisesti vasta tulevaisuudessa. Tähän liittyen voidaan jatkaa käyttäjien tarpeiden selvittämistä ja monitoroida käyttöä tietovaraston kehittämiseksi. Aivan ensimmäiseksi

käyttäjille ja ylläpitäjille voidaan antaa lisää koulutusta tietovarastoon liittyen, mikä toivottavasti lisää kiinnostusta sen käyttämiseen.

5.4 Ylläpito

Tietovarasto vaatii jatkuvaa ylläpitoa pysyäkseen hyödyllisenä käyttäjilleen. Uutta kurssidataa valmistuu vähintään puolen vuoden välein, ja tuorein data on yleensä mielenkiintoisinta. Koska kurssien järjestäjät tietävät, milloin kurssien datavirta tyrehtyy, ei erillistä päivitysaikataulua tarvita, vaan tieto päivitystarpeesta voidaan välittää muille kiinnostuneille ja tietovaraston ylläpidolle palavereissa. Helpoimmillaan uuden lukukauden tietojen siirtäminen vaatii tiedostojen ja määritysten kopioinnin lisäksi vain pieniä muutoksia.

Kaikille lähdejärjestelmille on kuitenkin suunniteltu parannuksia, ja näitä toteutettaneen vähitellen tulevaisuudessa. Tämä tarkoittaa, että lähteiden tietokantamääritykset muuttuvat jonkin verran vuosien varrella. Käytännössä suurin ongelma aiheutunee siitä, että kutakin järjestelmää ylläpitää täysin erillinen taho, ja tieto muutoksista ei välttämättä kulje kaikkialle. Toinen riski on metadatan ja dokumentaation päivittäminen, mikä jää helposti tekemättä, jolloin ylläpitäjän vaihtuessa tai uusien käyttäjien aloittaessa tietovaraston käytön he saattavat joutua pulaan. Siksi ylläpitovastuu aiotaan keskittää yhdelle henkilölle, joka tekee tarvittavat päivitykset tietovarastoon ja dokumentaatioon sekä pitää yhteyttä lähdejärjestelmien vastuuhenkilöihin.

Työ ei ota kantaa lähdejärjestelmien vanhan datan säilyttämiseen. Vanhat tietokannat on helppo pitää uusien rinnalla tallessa tai poistaa, jos niille ei ole enää mitään käyttöä.

6 Tulosten analysointi

Vaatimusmäärittely on pääasiallinen keino projektin onnistumisen mittaamiseen. Tämän lisäksi projektin tuloksen yleistettävyyttä pohditaan lyhyesti, ja esitetään luonnollisesti vastaukset tutkimuskysymyksiin. Aluksi tuloksia arvioidaan kuitenkin käyttäjien tyytyväisyyden ja heiltä saadun palautteen kautta.

Yhteyden saaminen tietovarastoon vaati joissakin tapauksissa käyttäjien mielestä paljon vaivaa, sillä ohjelmia ja ajureita täytyy hakea ja asentaa internetistä. He eivät löytäneet aivan kaikkea haluamaansa dataa tietovarastosta, mutta tämä johtuu osittain siitä, että dataa ei ole ollut edes lähdejärjestelmissä tarjolla riittävästi. Tietovaraston skeema aiheutti alussa hämmennystä lähinnä sen vuoksi, että se poikkeaa tavallisesta tietokannasta. Tarkemmin sanottuna sijaisavainten merkitys ei ollut aivan selvää, ja kurssiulottuvuus käyttäytyi eri lailla eri faktataulujen suhteen. Lisäksi samoja kurssiin ja sen ajankohtaan liittyviä valintoja joutuu tekemään useasta taulusta eri kentistä. Kaiken kaikkiaan käyttäjien suurin mielenkiinto kohdistui TRAKLA2-dataan, sillä he ovat muutenkin sen kanssa eniten tekemisissä.

Käyttäjien palautteesta voidaan päätellä, että käytettävyyteen olisi kannattanut kiinnittää enemmän huomiota, kuten jo aikaisemmin todettiin. Myös koulutus tietovaraston käyttöön on tarpeellista, jotta sen poikkeavuudet tavallisista tietojärjestelmistä tulevat selviksi. Koekäytön yhteydessä kunnollista käyttöohjetta ei tosin vielä ollut valmiina.

6.1 Vaatimusten täyttyminen

Luvun 3.3 vaatimusmäärittelyn toteutuminen käydään tässä läpi kohta kohdalta. Asteikko on kolmiportainen siten, että hymyilevät kasvot tarkoittavat vaatimuksen toteutuneen ainakin pääosin, suorana viivana oleva suu osoittaa vaatimuksen täyttyneen vain osittain ja surulliset kasvot merkitsevät vaatimuksen jääneen toteutumatta. Tämän lisäksi kunkin vaatimuksen toteutumisesta esitetään tarkentava tekstimuotoinen arvio. Taulukossa 15 käydään läpi toiminnalliset vaatimukset ja taulukossa 16 ei-toiminnalliset.

Taulukko 15. Toiminnallisten vaatimusten täyttyminen

#	Vaatimus	Priori-teetti	Toteutuminen	
T1	ETL-prosessi määrityksineen kullekin lähdejärjestelmälle on käytettävissä uudestaan ja muokattavissa jälkeenpäin.	1	Kyllä. KETTLE tallentaa ETL-määritykset tietokantaan tai XML-tiedostoon, josta ne voi ajaa uudestaan ja niitä voi helposti muokata.	☺
T2	Tietovarastoon voi liittää uusia lähdejärjestelmiä.	1	Kyllä. Lähdejärjestelmiä voidaan liittää samalla tavoin kuin nykyisetkin järjestelmät on yksi kerrallaan lisätty, sillä lähteet ovat toisistaan riippumattomia.	☺
T3	Tietovarastosta voidaan hakea tietoa vakiintuneella kyselykielellä tai vastaavalla tavalla vapaasti.	1	Kyllä. Käyttäjille on luotu tunnukset, joilla he pääsevät tietovarastoon käsiksi hakemaan sieltä tietoa haluamallaan tavalla, esim. Excelin avulla.	☺
T4	Tietovarastosta voidaan hakea tietoa graafisella käyttöliittymällä.	3	Jotkut tarkastellut analysointityökalut tarjoavat graafisen käyttöliittymän, mutta niidenkin käytettävyydessä on parannettavaa.	😐
T5	Tietovarastolla voi olla useita samanaikaisia käyttäjiä.	3	Kyllä. PostgreSQL sallii hyvin paljon samanaikaisia käyttäjiä.	☺

Taulukoista nähdään, että vaatimukset ovat täyttyneet tärkeimmiltä osiltaan. Osa vaatimuksista on jouduttu arvioimaan melko kevyin perustein, mutta kaiken kaikkiaan vaatimusten toteutuminen on vähintään tyydyttävää. On syytä huomata, että vaatimusmäärittely ei juurikaan koskettanut käytettävyyspuolta, jossa ongelmia ilmeni. Teknisesti järjestelmän pitäisi kuitenkin olla hyvä.

Taulukko 16. Ei-toiminnallisten vaatimusten täyttyminen

#	Vaatimus	Priori-teetti	Toteutuminen	
E1	Järjestelmän palvelimen ja tietokannan voi asentaa sekä Linux- että Windows-ympäristöön.	1	Kyllä. PostgreSQL tukee molempia käyttöjärjestelmiä. Linuxin sijaan asentamista kokeiltiin Macintoshille, eikä merkittäviä ongelmia ilmennyt.	☺
E2	Tietovarasto on loppukäyttäjälle käytettävissä Linux-, Macintosh- ja Windows-ympäristöistä käsin.	1	Kyllä. Tietovarastoon saa yhteyden mistä tahansa käyttöjärjestelmästä yhteisten rajapintojen ansiosta. Mainittuja käyttöjärjestelmiä on myös kokeiltu.	☺
E3	Vastaus suurimpaan osaan kyselyistä saadaan alle 5 sekunnissa, nopeimpiin alle sekunnissa ja äärimmäisen harvoihin yli 20 sekunnissa.	1	Kyllä. Edellytyksenä on kuitenkin, että kysely on järkevä, ettei järjettömän suuria taululiitoksia synny vahingossa.	☺
E4	Tietovaraston ETL-prosessista luodaan loki, josta nähdään ongelmakohdat suorittamisen aikana.	1	Kyllä. KETTLE tallentaa tietokantaan lokia ETL-prosessin suorituksesta.	☺
E5	Järjestelmä pystyy käsittelemään riittävän suuren määrän dataa sovellusalueen tarpeisiin.	1	Kyllä. Tietokanta ei rajoita datamäärää, mutta ETL-prosessi saattaisi tulevaisuudessa hidastua huomattavasti, jos datamäärä moninkertaistuisi.	☺
E6	Järjestelmässä käytetään ainoastaan avoimen lähdekoodin tai muita ilmaisia sovelluksia.	2	Enimmäkseen kyllä. Järjestelmän ydin eli tietokanta ja ETL-sovellus ovat avoimen lähdekoodin sovelluksia. Tiedon hakuun ja analysointiin voidaan käyttää muitakin sovelluksia.	☺
E7	Järjestelmä on modulaarinen siten, että yksittäinen osa tai sovellus on helppo vaihtaa.	2	Enimmäkseen kyllä. Tietokannan vaihtaminen vaatisi jonkin verran muutoksia ETL-määrityksiin. ETL-sovelluksen vaihtaminen olisi hankalaa siinä mielessä, että koko ETL-prosessi pitäisi määrittää uudestaan. Sen sijaan uusissa järjestelmissä eri ETL-sovelluksen käyttäminen pitäisi olla helppoa.	☺
E8	Tietovaraston lataaminen valmiista ETL-määrityksestä kestää alle puoli tuntia.	2	Kyllä. Testilaitteistolla lataamiseen kului vajaat 30 minuuttia.	☺

E9	Tietovarastosta tehdään riittävästi metadataa helpottamaan järjestelmän käytön, toiminnan ja muokkaamisen ymmärtämistä.	2	Kyllä. Alustavista suunnitelmista poikettiin jonkin verran, mutta metadata sisältää vastaavat asiat.	☺
E10	Järjestelmän hyödyntäminen datan analysointiin on nopeasti opittavissa ja vähintään yhtä helppokäyttöistä kuin yksittäisissä lähdejärjestelmissä.	2	Kyllä. Datan hyödyntäminen on suunnilleen yhtä helppoa kuin ennenkin, mutta parantamisen varaakin on vielä paljon.	☺
E11	Järjestelmä on englanninkielinen.	2	Enimmäkseen kyllä. Tietokantataulut, lokit, ETL-prosessi ja osa dokumentaatiosta ovat englanninkielisiä. Sen sijaan käyttäjän käyttöohje ja tämä diplomityödokumentti ovat suomeksi.	😐
E12	Uusien järjestelmien liittäminen tietovarastoon ja ETL-prosessien muokkaaminen on opittavissa ja tehtävissä kohtuullisella vaivalla.	2	Epävarmaa. Uutta ylläpitäjää ei ole ehditty kouluttaa vielä, joten tätä vaatimusta ei päästy testaamaan.	😐
E13	Järjestelmä pystyy esittämään datan moniulotteisessa ja hierarkkisessa muodossa.	3	Osittain. Tähtimalli tukee moniulotteista esitystapaa ja Palo pystyy esittämään datan moniulotteisessa muodossa, mutta muut esitetyt tiedonhakutavat eivät tähän pysty.	😐

6.2 Vastaukset tutkimuskysymyksiin ja yleistettävyys

Ensimmäisessä luvussa esitetty tutkimuskysymys oli:

- **Miten voidaan rakentaa tietovarasto automaattisten tarkastusjärjestelmien keräämien tietojen yhdistämiseen?**

Tämä jaettiin kolmeen tarkentavaan kysymykseen:

- Millainen prosessi tiedon kerääminen lähdejärjestelmistä on?
- Millaisia valmiita tietovarastointisovelluksia on olemassa ja voitaisiinko niitä hyödyntää?
- Mikä tietokanta- ja muu teknologia soveltuu tehtävään parhaiten?

Tässä työssä käytetyt menetelmät ja periaatteet ovat olleet samankaltaisia kuin tietovarastoinnissa tavallisestikin. Tekniikka- ja sovellusvertailun perusteella

päädyttiin relaatiotietokantoihin, mikä on ollut jo pitkään vakiintunut ratkaisu. Niinpä sovellusalue, pieni koko, käyttötarkoitus ja muut tekijät eivät näyttäisi ratkaisevasti haittaavan tai muuttavan tietovaraston rakentamista.

Työn tärkeimpänä tuloksena voidaan pitää osoitusta siitä, että tietovarasto voidaan nykyään rakentaa avoimen lähdekoodin sovellusten varaan. Sovelluksissa alkaa jopa olla valinnan varaa tietokannan lisäksi myös ETL-prosessin sekä analysoinnin ja raportoinnin toteuttamiseen. Horstmanniin (2005) verrattuna tästä havaitaan, että työkalut kehittyvät hyvin nopealla vauhdilla tällä hetkellä. Samoin vertailu Mäkiseen (1999) osoittaa, että perusteoria on pysynyt melko samana ja lähinnä työkalut ovat kehittyneet.

Pitkälle meneviä johtopäätöksiä ei kuitenkaan voida vetää. Työssä oli lukuisia erityispiirteitä, erityisesti melko pieni koko. Suuremmassa projektissa esim. suorituskyky olisi saattanut laskea selvästi. Vaikka työtä voidaan pitää melko onnistuneena, eivät sen tulokset siis ole täysin yleistettävissä ilman muiden vastaavien projektien tulosten tukea.

7 Yhteenveto

Työn tavoitteena oli rakentaa tietovarasto tietotekniikan perusopetuksessa käytettävien automaattisten tarkastusjärjestelmien keräämälle datalle. Järjestelmät ovat vuosien varrella keränneet tietoja hajallaan oleviin tietokantoihin, joiden data haluttiin nyt yhdistää, jotta sitä voisi paremmin hyödyntää opetuksen kehittämisessä. Teoriaosassa käytiin läpi tietokantatekniikoita sekä tietovarastoinnin teknisiä perusasioita. Jälkimmäisestä esiteltiin mm. dimensiomalli ja ETL-prosessin osia. Teorian jälkeen siirryttiin työn erityispiirteisiin, joita olivat ennen kaikkea sovellusalue, käyttötarkoitus, pieni koko, avoimen lähdekoodin sovellusten käyttö sekä keskittyminen tekniseen luontivaiheeseen analysoinnin ja raportoinnin sijaan.

Tietovarastoon liitettiin kolme lähdejärjestelmää. Tietorakenteiden ja algoritmien opetuksessa käytettävä TRAKLA2 käsitti eniten ja vaihtelevinta dataa, ja siitä käyttäjät olivat myös eniten kiinnostuneita. Kaksi muuta järjestelmää olivat ohjelmoinnin peruskursseilla käytettävä Goblin sekä kaikki tietotekniikan kurssien osasuoritukset sisältävä OSR. Lisäksi tietovarastossa huomioitiin mahdollisuus liittää jatkossa erilaisia järjestelmiä. Tämä mahdollistuu ennen kaikkea dimensiomallin hienovaraisten muutosten vuoksi, sillä ne eivät vaadi muutoksia jo liitettyjen järjestelmien määrityksiin.

Tietokantatekniikoita ja -sovelluksia vertailtiin parhaan ratkaisun löytämiseksi. Tekniikoista todettiin, että oliotietokannat sopivat lähinnä erikoistapauksiin, XML-tietokannat eivät ole vielä tarpeeksi kehittyneitä ja ilman varsinaista tietokantaa voi selvitä vain hyvin helpoissa tai erikoisissa tapauksissa. Relaatiotietokannat ovat siis yhä varmin ratkaisu. Sovelluksista PostgreSQL ja MySQL ovat likimain yhtä hyviä, ja muitakin käyttökelpoisia vaihtoehtoja on olemassa. PostgreSQL valittiin tähän tietovarastoon monipuolisten ominaisuuksien ansiosta.

Myös ETL- ja OLAP-sovelluksissa alkaa olla avoimen lähdekoodin puolella valinnanvaraa, ja työkalut kehittyvät vauhdilla. ETL-sovellukseksi valittiin

KETTLE ennen kaikkea hyvän graafisen käyttöliittymän ansiosta. Erilaisia OLAP-sovelluksia sekä muita keinoja hakea tietoa tietovarastosta, kuten taulukkolaskentaohjelmia, katsottiin lähinnä kokeilumielessä jättäen lopullinen ratkaisu käyttäjälle tai jatkokehityksen huoleksi.

Arkkitehtuuriksi valittiin yhteen keskustietovarastoon perustuva malli, josta voidaan tarvittaessa johtaa paikallisvarastoja erityistarpeisiin. Eri datalähteet ja usein niiden sisäisetkin osat ovat riippumattomia toisistaan, mikä helpottaa järjestelmän käyttöä ja ylläpitoa. Samoin lähdetietokannat, ETL-prosessi, itse tietovarasto, mahdolliset paikallisvarastot ja OLAP-sovellus voidaan sijoittaa fyysisesti joko samalle tai eri koneille ja käyttöjärjestelmille.

Tietovarastoprojektin tekniset vaatimukset täyttyivät melko hyvin, mutta käytettävyyteen jäi hieman toivomisen varaa johtuen työn painotuksesta. Tästä johtuen käyttöönotossa esiintyi pientä takkuilua. Työn tärkein havainto oli, että tietovarasto voidaan nykyään rakentaa avoimen lähdekoodin sovellusten varaan, mutta johtuen projektin erityispiirteistä tätä ei voida suoraan yleistää.

8 Jatkokehittäminen

Tietovarastossa on monta osa-aluetta, joilla sitä voidaan jatkossa kehittää. Olemassa olevan järjestelmän käyttöä voidaan tarkkailla, ja sitä kautta tietovarastoa pystytään tarvittaessa korjaamaan ja parantamaan. Koska analysointi ja raportointi eivät kuuluneet tämän työn piiriin kovin laajasti, voisi OLAP-työkalun käyttöönotto jatkossa parantaa huomattavasti tietovaraston käytettävyyttä, joka nyt jäi melko vähälle huomiolle.

Tulevaisuudessa tietovarastoa voidaan helposti laajentaa uusiin järjestelmiin. Varsinkin vanhojen järjestelmien uudet dataerät saa pienellä vaivalla liitettyä mukaan. TAPAS-projektin tuotokset vaativat varmasti muutoksia tietokantaskeemaan, mutta hienovaraisten muutosten ansiosta ne eivät häiritse jo toteutettuja osia. Vastaavasti kuin tietovarastossa on otettu huomioon laajennettavuus uusiin lähdejärjestelmiin, on syytä huomioida tulevien järjestelmien suunnittelussa jo mahdollisuus liittää niiden dataa tietovarastoon. Tästä voidaan johtaa tarve korjata myös jo liitettyjä lähdejärjestelmiä paremmin tietovaraston kanssa yhteensopiviksi samalla, kun niitä muutenkin kehitetään.

Jos yhteistyö laajenee esim. muihin yliopistoihin, tulee kyseeseen uusien käyttäjien ottaminen mukaan tietovaraston käyttäjäkuntaan. Tällöin on kiinnitettävä erityistä huomiota tietosuoja-asioihin.

Lähteet

Balcilhon François 1992. A Classification of Object-Oriented Database Systems. *Proceedings of the third international workshop on Database programming languages: bulk types & persistent data*. Morgan Kaufmann Publishers Inc., San Francisco. s. 3–6.

BEE 2006. The BEE Project. http://bee.insightstrategy.cz/en/index.html. viitattu 6.10.2006.

Boehnlein Michael, Ulbrich-vom Ende Achim 1999. Deriving Initial Data Warehouse Structures from the Conceptual Data Models of the Underlying Operational Information Systems. *Proceedings of the 2nd ACM international workshop on Data warehousing and OLAP*. ACM Press, New York. s. 15–21.

Clover.ETL 2006. Clover.ETL – Open Source ETL Tool. http://cloveretl.berlios.de/. viitattu 6.10.2006.

Eisenberg Andrew, Melton Jim 2005. XQuery 1.0 is Nearing Completion. *ACM SIGMOD Record*, Vol. 34, No. 4. ACM Press, New York. s. 78–84.

Firebird 2006. Firebird – Relational Database for the New Millenium. http://www.firebirdsql.org/. viitattu 6.10.2006.

Haas Laura M., Hernández Mauricio A., Ho Howard, Popa Lucian, Roth Mary 2005. Clio Grows Up: From Research Prototype to Industrial Tool. *Proceedings of the 2005 ACM SIGMOD International Conference on Management of data*. ACM Press, New York. s. 805–810.

Henkilötietolaki 22.4.1999/523 1999. FINLEX - Valtion säädöstietopankki. http://www.finlex.fi/fi/laki/ajantasa/1999/19990523. viitattu 25.4.2006.

Hiisilä Aki 2005. *Kurssinhallintajärjestelmä ohjelmoinnin perusopetuksen avuksi*. Diplomityö. Teknillinen korkeakoulu.

Horstmann Jutta 2005. *Migration to Open Source Databases*. Diploma thesis. Technical University Berlin.

Hovi Ari 1997. *Data Warehousing – Tietovarastotekniikka*. Suomen Atk-kustannus, Espoo.

Hovi Ari, Ylinen Jari, Koistinen Heikki 2001. *Tietovarastot liiketoiminnan tukena*. Talentum Media Oy, Helsinki.

Ingres 2006. Ingres – Business open source database. http://www.ingres.com/. viitattu 6.10.2006.

Inmon William H. 2002. *Building the Data Warehouse*. 3. painos. Wiley Publishing Inc., Indianapolis.

Jones Mary Elizabeth, Song Il-Yeol 2005. Dimensional Modeling: Identifying, Classifying & Applying Patterns. *Proceedings of the 8th ACM international workshop on Data warehousing and OLAP*. ACM Press, New York. s. 29–38.

Kakkonen Tuomo 2003. *Esseetehtävien tietokoneavusteinen arviointi*. Pro gradu -tutkielma. Joensuun yliopisto.

Karhumäki Seppo 2001. *Liiketaloudellisten tapahtumien jalostaminen tiedoksi*. Diplomityö. Teknillinen korkeakoulu.

KETTLE 2006. KETTLE – Turning Data Into Business. http://www.kettle.be/. viitattu 6.10.2006.

Kimball Ralph, Caserta Joe 2004. *The Data Warehouse ETL Toolkit – Practical Techniques for Extracting, Cleaning, Conforming and Delivering Data*. Wiley Publishing Inc., Indianapolis.

Kostiainen Jukka 2003. *A Data Warehousing Solution to Paper Mill Production and Quality Reporting*. Diplomityö. Teknillinen korkeakoulu.

Malmi Lauri, Korhonen Ari 2004. Automatic Feedback and Resubmissions as Learning Aid. *Proceedings of 4th IEEE International Conference on Advanced Learning Technologies (ICALT'04)*. IEEE Computer Society, Washington DC. s. 186–190.

Martyn Tim 2004. Reconsidering Multi-Dimensional Schemas. *ACM SIGMOD Record*. Vol. 33, No. 1. ACM Press, New York. s. 83–88.

MaxDB 2006. MaxDB. http://www.mysql.com/products/maxdb/. viitattu 6.10.2006.

Mondrian 2006. Mondrian OLAP Server. http://mondrian.sourceforge.net/. viitattu 6.10.2006.

Moody Daniel L., Kortink Mark A. R. 2000. From Enterprise Models to Dimensional Models: A Methodology for Data Warehouse and Data Mart Design. *Proceedings of the Second International Workshop on Design and Management of Data Warehouses*. Jeusfeld M., Shu H., Staudt M. ja Vossen G. (toim.). s. 5-1–5-12. http://CEUR-WS.org/Vol-28/paper5.pdf. viitattu 1.6.2006.

MySQL 2006. MySQL – The world's most popular open source database. http://www.mysql.com/. viitattu 6.10.2006.

Mäkinen Tero 1999. *Data Warehouse -järjestelmällä toteutettu asiakasraportointi, Case: Teleoperaattorin puheluliikennetietojen raportointi*. Diplomityö. Teknillinen korkeakoulu.

Octopus 2006. Enhydra Octopus. JDBC Data Transformations. http://www.enhydra.org/tech/octopus/index.html. viitattu 6.10.2006.

Ollila Päivi 2005. *Management of Data and Information Quality in Enterprise Systems – Case Study*. Diplomityö. Teknillinen korkeakoulu.

Palo 2006. Palo – Open-Source MOLAP. http://www.palo.net/index2.php?show=. viitattu 6.10.2006.

Pendse Nigel 2005. What is OLAP? http://www.olapreport.com/fasmi.htm. viitattu 27.6.2006.

PostgreSQL 2006. PostgreSQL – The world's most advanced open source database. http://www.postgresql.org/. viitattu 6.10.2006.

Rontu Markku 2004a. *Visual Queries for a Student Information System*. Diplomityö. Teknillinen korkeakoulu.

Rontu Markku 2004b. SEEQ – System for Enhanced Exploration and Querying. http://www.cs.hut.fi/Research/SVG/SEEQ/. viitattu 6.10.2006.

Rontu Markku, Korhonen Ari, Malmi Lauri 2006. System for Enhanced Exploration and Querying. *Proceedings of the working conference on Advanced visual interfaces AVI '06*. ACM Press, New York. s. 508–511.

Silberschatz Abraham, Korth Henry F., Sudarshan S. 2006. *Database System Concepts*. McGraw-Hill, New York.

Silvasti Panu 2003. *Tilastollisen datan kerääminen algoritmisten harjoitustehtäväsovelmien käytöstä*. Diplomityö. Teknillinen korkeakoulu.

Thomas Gomer, Thompson Glenn R., Chung Chin-Wan, Barkmeyer Edward, Carter Fred, Templeton Marjorie, Fox Stephen, Hartman Berl 1990. Heterogeneous Distributed Database Systems for Production Use. *ACM Computing Surveys*. Vol. 22, No. 3. ACM Press, New York. s. 237–266.

Torvinen Petteri 2003. *Tilastollinen analyysi algoritmisten harjoitustehtäväsovelmien käytöstä*. Diplomityö. Teknillinen korkeakoulu.

Tryfona Nectaria, Busborg Frank, Christiansen Jens G. Borch 1999. starER: A Conceptual Model for Data Warehouse Design. *Proceedings of the 2nd ACM international workshop on Data warehousing and OLAP*. ACM Press, New York. s. 3–8.

Törmänen Arla 1999. *Tietovarastointi – strategiasta toteutukseen*. Suomen Atk-kustannus, Espoo.

Ullman Jeffrey D., Widom Jennifer 2001. *A First Course in Database Systems*. 2. painos. Prentice Hall, New Jersey.

Vassiliadis Panos, Simitsis Alkis, Skiadopoulos Spiros 2002. Modeling ETL Activities as Graphs. *Proceedings of the 4th International Workshop on Design and Management of Data Warehouses*. Lakshmanan Laks V.S. (toim.). s. 52–61. http://CEUR-WS.org/Vol-58/simitsis.pdf. viitattu 26.7.2006.

W3C 1999. XML Path Language (XPath). http://www.w3.org/TR/xpath. viitattu 7.8.2006.

W3C 2004. XML Schema Part 0: Primer Second Edition. http://www.w3.org/TR/xmlschema-0/. viitattu 7.8.2006.

W3C 2006. XQuery 1.0: An XML Query Language. http://www.w3.org/TR/xquery/. viitattu 7.8.2006.

Westermann Utz, Klas Wolfgang 2003. An analysis of XML database solutions for the management of MPEG-7 media descriptions. *ACM Computing Surveys*. Vol. 35, No. 4. ACM Press, New York. s. 331–373.

You Jane, Dillon Tharam, Liu James 2001. An Integration of Data Mining and Data Warehousing for Hierarchical Multimedia Information Retrieval. *Proceedings of 2001 International Symposium of Intelligent Multimedia, Video and Speech Processing*. IEEE Press, Piscataway. s. 373–376.