TIETOKANTAPOHJAISEN BLOGIN TUOTTAMINEN

LAHDEN AMMATTIKORKEAKOULU
Mediatekniikan koulutusohjelma
Teknisen visualisoinnin suuntautumisvaihtoehto
Opinnäytetyö
5.5.2008
Eeva Andrejeff

Lahden ammattikorkeakoulu
Mediatekniikan koulutusohjelma

ANDREJEFF, EEVA: Tietokantapohjaisen blogin tuottaminen

Teknisen visualisoinnin opinnäytetyö, 50 sivua

Kevät 2008

TIIVISTELMÄ

Tämä opinnäytetyö käsittelee tietokantapohjaisen blogin tuottamista Macromedia Flash 8 –ohjelmalla toteutetulle sivupohjalle. Yleisellä tasolla työssä pyritään tuomaan esille erilaisia tapoja, joilla Flash-toteutuksen tietovarastona voidaan käyttää relaatiotietokantaa ja niitä tekniikoita, joita sovelluksen toteuttaminen vaatii.

Työn alussa tuodaan esille yleisiä asioita blogeista ja niiden käytöstä sekä kerrotaan blogeihin liittyvästä RSS-tekniikasta. Tämän jälkeen siirrytään käsittelemään verkkosovellusten suunnittelua, jonka kautta päästään käsittelemään itse verkkosovellustekniikoita, eli Flashia, ActionScriptiä, PHP:tä ja relaatiotietokantoja. Flash ja ActionScript-luvun yhteydessä käsitellään myös XML:n käyttöä Flash-sovellusten tiedonvälitysdokumentteina sovelluksen ja tietokannan välillä.

Case-osiossa hyödynnetään tutkittuja tekniikoita ja luodaan Lahden Seurakuntayhtymän Oppilaitostyön internetsivuille blogi. Lopuksi yhteenvedossa käydään läpi ajatuksia työstä ja arvioidaan tämän tekniikan käyttömahdollisuuksia yleisellä tasolla.

Avainsanat: blogi, Flash, tietokanta, XML, PHP

Lahti University of Applied Sciences
Faculty of Technolgy

ANDREJEFF, EEVA: Production of database-founded blog

Bacheor's Thesis in Visualization Engineering, 50 pages

Spring 2008

ABSTRACT

This thesis deals with the production of a database-founded blog to a web page that has been carried out with Macromedia Flash 8 program. The thesis concentrates on the techniques that can be used when a Flash-based application uses a database as its storage for information.

The beginning of the thesis deals with blogs in general and how they can be used. The first part also deals with the RSS technique that is related to blogs. The theory of the designing of a web application is also presented. The rest of the theoretical part concentrates on Flash, ActionScript, PHP and databases. The part about Flash and ActionScript also discusses using XML within Flash applications as a way to transmit data to the database.

The case part tells about creating a blog for Lahden Seurakuntayhtymän Oppilaitostyö (an organization formed by the parishes of Lahti for doing youth work in schools) using the techniques that have been introduced during the thesis. In the end the summary goes through the thoughts about the thesis and estimates the possibilities to use these techniques generally.

Keywords: blog, Flash, database, XML, PHP

SISÄLLYS

TERMISTÖ

AMF (Data Message Format)

ActionScript on ohjelmointikieli Flash-ympäristöön.

Avainkenttä on taulukon sarake, jonka tietojen avulla tietueet voidaan erottaa toisistaan.

Blogosfääri on kaikkien blogien muodostama yhteisö.

DOM (Document Object Model)

ER-kaavio on tietokannan rakennetta ja suhteita kuvaava kaavio.

Flash Player on Macromedia Flashilla tuotettujen esiteysten katseluun tarkoitettu ohjelma.

Kenttä on tietokannan taulun sarake johon yksittäinen tieto tallennetaan.

Perusavain on kenttä, jonka tieto on varmasti yksilöllinen tietueessa.

PHP (Hypertext Preprocessor)

Plugin on ohjelmaan asennettava lisäosa, jonka avulla ohjelman käyttöön saadaan uusia ominaisuuksia.

Podcast on äänitiedostojen julkaisemista verkossa siten, että mukaan liitetään metadata, jonka avulla erityiset ohjelmistot osaavat ladata tiedostot automaattisesti.

Relaatio tarkoittaa tietokannan taulua.

RSS (Really Simple Syndication)

SQL (Structured Query Language)

Taulu Tietokannat koostuvat perinteisesti kaksiulotteisista tauluista, joihin syötetyt tiedot tallennetaan. Taulut on jaettu sarakkeisiin ja riveihin.

Tietue tarkoittaa yhdellä tietokannan taulun rivillä olevia tietoja.

Viiteavaimen avulla yhdistetään eri taulujen tietoja.

XML (Extensible Markup Language)

1 JOHDANTO

Blogien suosio web-palvelujen maailmassa on kasvanut valtavasti viime vuosien aikana, ja niistä onkin muodostunut 2000-luvun nopeimmin kasvava media. Sekä yksityiset ihmiset, yhteisöt että yritykset voivat tuoda ajatuksiaan julkisuuteen aivan uudenlaisen kanavan voimin.

Blogipalveluja tarjoavat useat sivustot, joista kuka tahansa voi rekisteröitymällä palveluun saada itselleen oman blogin. Blogien suosion myötä web-maailmaan on kehittynyt myös uudenlaisia blogien seuraamista ja selailua helpottavia tekniikoita ja palveluita. Tällaisia ovat muun muassa RSS-syöte, jonka avulla blogeja voi tilata omaan syötteenlukijaansa ja Blogilista.fi –tyyppiset sivustot, jotka seuraavat blogosfääriä ja tarjoavat kanavan löytää juuri itseä kiinnostavat blogit.

Blogien suosion myötä myös liiketoiminta on saanut uusia piirteitä. Monien eri alojen yritykset niin ulkomailla kuin Suomessakin ovat löytäneet blogit ja havainneet niiden olevan uusi kilpailuvaltti ja mediakanava. Ihmiset haluavat nähdä toistensa arkeen, ja mikä olisikaan parempi tapa tuoda yrityksen elämää julki kuin työntekijän kirjoittama blogi?

Tässä työssä tutkitaan mahdollisuuksia toteuttaa blogi Flash-pohjaiselle sivustolle. Työn sisältö käsittelee blogeja ilmiönä ja kertoo sen suunnittelusta ja toteutuksesta verkkosovelluksen näkökulmasta. Työssä esitellään myös blogin tuottamiseen käytettäviä verkkosovellustekniikoita.

Case-osiossa hyödynnetään tutkittuja tekniikoita ja luodaan Lahden Seurakuntayhtymän Oppilaitostyön internetsivuille blogi. Koska jo olemassa olevat sivut on luotu käyttämällä Flash-ohjelmaa, on luontevaa käyttää jo valmiina olevaa pohjaa, ja luoda blogi samalla ilmeellä.

2 BLOGIT

2.1 Määritelmä

Sana blogi on syntynyt englanninkielisestä sanasta weblog. Blogi itsessään on WWW-sivu, johon tehdyt merkinnät ovat kronologisessa järjestyksessä, ja sen sisältö voi koskea muun muassa sivun ylläpitäjän elämää tai jotain kirjoittajan mielestä kiintoisaa aihetta. Usein blogimerkintöjä voi myös kommentoida. (TSK:n termipankki, 2005.) Blogit voivat siis palvella monia erilaisia tarpeita, eivätkä ne välttämättä ole aina päiväkirjoja sanan varsinaisessa merkityksessä.

Blogit eivät ole jääneet pelkäksi muoti-ilmiöksi, vaan ne ovat myös merkittävä teknologinen, sosiaalinen ja taloudellinen muutosvoima. Blogosfäärissä eli kaikkien blogien muodostamassa yhteisössä sananvapaus saa uudenlaisen konkreettisen ilmenemismuodon, sillä blogien kautta kuka tahansa voi julkaista ajatuksensa ja mielipiteensä missä tahansa, milloin tahansa ja kenelle tahansa. Blogin tärkein ominaisuus onkin tiedon julkaisemisen helppous ja nopeus. (Kilpi 2006, 3.)

Voitaisiinkin sanoa, että blogit ovat ilmiönä nousseet 90-luvun omien kotisivujen tilalle. Aikaisemmin internetjulkaiseminen oli monille liian vaivalloista, ja internetistä löytyykin miljoonia hylättyjä kotisivuja, joiden tekijöiden osaaminen, into ja aika eivät riittäneet sivujen ylläpitoon. 2000-luvulla blogit toivat aivan uudenlaisen alustan internetnetjulkaisemiselle; oman blogin voi julkaista, jos osaa käyttää tietokonetta. (Kilpi 2006, 4.)

2.2 Blogit ja yrityselämä

2.2.1 Blogin tarjoamat hyödyt

Yrityksille blogit tarjoavat täysin uudenlaisen kommunikaatiokanavan, joiden kautta voidaan tavoittaa nykyisiä ja tulevia asiakkaita, omia työntekijöitä, yrityksen sidosryhmiä sekä poliittisia päättäjiä. Yritysmaailmassa bloge-

ja voidaan siis hyödyntää sekä yrityksen sisäisessä että ulkoisessa viestinnässä. Muun muassa Nokia, Finnair, McDonald's, Microsoft ja Sun Microsystems ovat monikansallisia yrityksiä, jotka käyttävät yhtenä viestintävälineenään blogeja. (Kilpi 2006, 3.)

Blogit voivat antaa yrityksen sisäiselle viestinnälle uudenlaista lisäarvoa, sillä siihen tehdyt merkinnät tavoittavat helposti koko henkilökunnan ja viestit kulkevat kaikille samanlaisina. Blogit antavat myös yrityksen henkilöstölle mahdollisuuden kommentoida merkintöjä ja lähettää välitöntä palautetta blogissa ilmoitetuista asioista. Blogit siis voivat olla tie vapaampaan ja nopeampaan keskusteluun yrityksen johdon ja työntekijöiden välillä. (Kilpi 2006, 45.)

Myös ulkoisen viestinnän saralla yritykset voivat luoda avoimempaa mielikuvaa itsestään julkaisemalla blogia nettisivuillaan. Tuomas Kilpi käyttää esimerkkinään Microsoftia, joka käyttää blogeja esimerkiksi tuotekehityksensä tukena. Näin Microsoft saa suoraa tietoa loppukäyttäjien tarpeista ja mielipiteistä, mutta myös enemmän yhtiöön ja tuotteisiinsa kohdistunutta kritiikkiä. Tämä esimerkki kuitenkin osoittaa, että jos yritys on todella halukas parantamaan tuotteitaan ja toimintatapojaan, voi blogeista olla arvaamatonta hyötyä. (Kilpi 2006, 45-46.)

Kun yrityksessä harkitaan blogiviestinnän käyttöönottoa, on kuitenkin otettava huomioon, ettei pelkkä ohjelmien ja laitteistojen hankkiminen välttämättä riitä. On muistettava, että parhaat hyödyt tavoitetaan, kun henkilökunta on myös koulutettu käyttämään hankittuja välineitä. Jotta välineet ja koulutus eivät puolestaan mene hukkaan, olisi yritysblogia kirjoittavia työntekijöitä muistettava kannustaa ja palkita. Muun muassa esimiehen esimerkki voi toimia hyvänä keinona kannustaa muuta henkilökuntaa kirjoittamaan blogeja yrityksen hyväksi. (Kilpi 2006, 85.)

2.2.2 Blogi vaihtoehtona viestintäkanavaksi

Sähköpostit ovat tällä hetkellä yritysten tärkein sisäisen viestinnän väline. Sähköposteja on helppo lukea, kirjoittaa sekä lähettää eteenpäin, viestit pysyvät tallessa ja ovat helposti tarkastettavissa. Blogeilla on kuitenkin paljon sähköpostia paremmat ominaisuudet ryhmien hallintaan ja sisäi-

seen kommunikointiin riippumatta ryhmien henkilömäärästä. (Kilpi 2006, 91.)

Blogien etuja ryhmäviestinnän mediana ovat muun muassa seuraavat:

- Kaikilla, joilla on pääsy blogiin, on myös mahdollisuus lukea blogiin tallennettu informaatio ja keskustelu.
- Blogista on helppo hakea merkintöjä esimerkiksi kirjoittajan, päivämäärän tai avainsanan perusteella.
- Kaikki henkilökunnan jäsenet saavat saman version jaetusta informaatiosta. Usein yrityksissä suurimpana ongelmana on se, ettei aina ole selvää kenellä on ajankohtaisin versio tietystä dokumentista.
- Tieto säilyy blogissa ja kaikki pääsevät siihen käsiksi jos on tarve tarkistaa jokin seikka jälkikäteen.
- Tiimin uusien jäsenten on helppo päästä perille projektin kulusta ja tiimin työskentelykulttuurista blogimerkintöjen ja kommenttien kautta.
- Työntekijöiden voi olla helpompi sitoutua projektiin ja kokea se omakseen yhteisen blogin kautta.
- Blogi vähentää päivittäistä sähköpostimäärää.
- Jos blogit ovat avoimia muillekin kuin tiimien jäsenille, voi esille nousta tuoreita näkökulmia ja tiimi voi saada arvokasta kritiikkiä.
- Vaikka blogille ei enää olisi projektin loppumisen jälkeen tarvetta, tieto säilyy tulevaisuutta varten arkistoissa ja on käytössä uusien projektien yhteydessä. (Kilpi 2006, 91-92.)

Yritys voi myös muokata mediatiedotustaan siten, että esimerkiksi lehdistötiedotteissa viitataan yrityksen blogiin tai ne käsittelevät blogia itseään. Media saattaa jopa oma-aloitteisesti alkaa seurata kyseistä blogia, ja näin ollen yritys saa lisää julkisuutta. Blogissa kirjoitettu teksti on usein persoonoidumpaa ja vapaampaa kuin perinteisen lehdistötiedotteen kieli. (Kilpi 2006, 93-94.)

Vaikka joidenkin bloggaajien mielestä blogit tulevat syrjäyttämään perinteisen joukkotiedotuksen, näin tuskin tulee käymään. Blogit käsittelevät harvoin päivän uutisia ja niiden kiinnostavuus piilee muissa seikoissa kuin journalistisuudessa. On kuitenkin hyvä muistaa, että blogit ovat saaneet suurta suosioita, ja niiden arvoa julkaisujärjestelmänä ei kannata vähek-

syä. Blogit toimivat yhteisömediana, joka antaa mahdollisuuden moniääniseen keskusteluun, ne voimistavat yhteisöllisyyttä ja auttavat yhteisön ohjaamisessa. (Kilpi 2006, 96-97.)

2.3 Blogit ja laki

Blogimerkintöjä kirjoittaessa on hyvä muistaa, että nettijulkaisuihin pätevät samat lait ja asetukset kuin muussakin julkaisemisessa. Bloggaajan on otettava huomioon ainakin tekijänoikeuslaki, laki sananvapauden käyttämisestä joukkoviestinnässä sekä rikoslait, jotka koskevat yksityiselämää loukkaavan tiedon levittämistä, kunnianloukkausta ja kiihottamista kansan ryhmää vastaan koskevat säädökset. (Kilpi 2006, 149-150.)

Tekijänoikeuslaissa on blogin kannalta kaksi puolta. Toisaalta blogin kirjoittajan ei tulisi julkaista blogissaan muiden tekemiä teoksia ilman tekijän lupaa lukuun ottamatta lyhyitä siteerauksia. Puolestaan yrityksen näkökulmasta on hyvä muistaa, että työntekijän kirjoittaman blogin sisältö kuuluu tekijänoikeuslain mukaan kirjoittajalle eikä yritykselle. Tekijänoikeudet voidaan kuitenkin työsopimuksessa siirtää työnantajalle. (Kilpi 2006, 153.)

2.4 Blogit ja RSS

RSS eli Really Simple Syndication nimellä tarkoitetaan XML-kielen mukaisia tiedostoja, joiden avulla välitetään usein päivittyvien sivustojen sisältöä RSS -syötettä lukevaan ohjelmaan. Useat blogipalvelut- ja alustat tarjoavat automaattisesti mahdollisuuden tilata ja julkaista RSS-syötetiedostoja. Lukijassa syötteet ovat samalla tavoin aikajärjestyksessä kuin blogeissakin siten, että uusin syöte on ensimmäisenä luettavissa. (Blogilista.fi, 2005.)

Yleisimmin syöte on jaettu otsikko-osaan ja sisältöosaan, jossa esitetään koko otsikon alla oleva asia tai osa siitä. Useimmat RSS-syötettä käyttävät sivustot lisäävät tiedon RSS-syötteen tilaamismahdollisuudesta HTML -tiedoston head -osioon. Esimerkiksi Mozilla Firefox ja Safari osaavat tulkita tiedon HTML-dokumentista, jolloin selainikkunaan ilmestyy RSS -syöttestä ilmoitava ikoni. (Blogilista.fi, 2005.)

Yleisimmät käytössä olevat versiot ovat RDF Site Summary (versiot RSS 0.9 ja RSS 1.0) ja Really Simple Syndication (versio RSS 2.0). Jälkimmäinen versio mahdollistaa tiedostoviittauksen lisäämisen syötteeseen. Tätä ominaisuutta käytetään muun muassa suositussa Podcasting-tekniikassa. (Blogilista.fi, 2005.)

RSS-tekniikan hyötynä on, että se tarjoaa tavan seurata useita kiinnostavia sivustoja vähällä vaivalla. RSS-syöteen tarjoavien sivustojen määrä on noussut nopeasti viime vuosina, ja sitä käytetään muillakin kuin blogisivustoilla. Syötettä voi lukea monilla eri ohjelmilla, ja myös monet selain- ja sähköpostiohjelmat tukevat RSS-syötettä. (Whatisrss.)

Sekä Windowsille, Linuxille ja Mac OS X:lle on saatavilla erilaisia RSS-syötteen lukijoita, jotka voi ladata internetistä ja asentaa tietokoneen kovalevylle. Tämän lisäksi lukijoita on saatavilla kännykkäversioina, täysin webissä toimivia lukijoita sekä erilaisina plugin-tyyppisinä ohjelmina muun muassa Mozilla Firefoxille. Lukijoita on saatavilla useita kymmeniä erilaisia, joiden joukosta jokainen voi löytää omansa. (Schroeder, 2007.)

Suosittuja syötteen lukijoita eri käyttöjärjestelmiin ovat ainakin Amphetadesk Windowsille, Linuxille ja Mac OS X:lle, Feedreader Windowsille ja Outlook-ohjelmaan integroituva NewsGator. Web-pohjaisia suosiossa olevia syötteenlukijoita ovat puolestaan My Yahoo, Bloglines sekä Google Reader. Jos sivusto tarjoaa mahdollisuuden tilata RSS-syötettä, on asia yleensä ilmaistu pienellä ikonilla, jossa on kirjainyhdistelmä RSS, XML tai RDF. (Whatisrss.)

3 VERKKOSOVELLUSTEN SUUNNITTELU

3.1 Flash-sovelluksista

Flash-sovellukset eivät enää palvele pelkkiä animaatiotarpeita, vaan ne ovat kasvaneet tietokantoja ja –verkkoa hyödyntäviksi verkko-ohjelmiksi. Muutos on johtunut pitkälti uusista komponenteista ja ActionScript-ohjelmointikielen tarjoamista mahdollisuuksista hyödyntää sovelluksen ulkopuolista dataa. (Manninen & Marttila 2006, 377.)

Flash-sovellus on aina verkko-ohjelman osa, joka näkyy käyttäjälle selaimessa. Sovelluksien taustalla toimii kuitenkin usein monia erilaisia verkkotekniikoita, jotka ovat käyttäjälle näkymättömiä. Näiden tekniikoiden käyttö yhdessä ActionScripitin kanssa mahdollistavaa sen, että Flash-sovelluksesta voidaan luoda dynaaminen kokonaisuus, joka pystyy automaattisesti muuttamaan sisältöään. (Manninen & Marttila 2006, 378.)

Etuna Flash-sovelluksissa verrattuina perinteisiin web-sivustoihin on se, että niillä on kyky muuttaa sisältöään ilman, että itse HTML-sivua on tarve ladata uudelleen palvelimelta. Tämä dynaamisuus on mahdollista toteuttaa Flash-sovelluskehittimen tarjoamilla mahdollisuuksilla ja ActionScript-ohjelmointikielen avulla. (Manninen & Marttila 2006, 378.)

3.2 Sovelluksen suunnittelusta

Sovelluksen suunnitteluun ei ole olemassa yhtä oikeaa tapaa, mutta projektin kannalta on hyvä tehdä aina jonkinlaista suunnittelutyötä turhien virheiden välttämiseksi. Suunnittelun alkuvaiheessa on hyvä tehdä dokumentaatiota, jossa vapaasti kuvaillaan sovellusta ja siihen liittyviä tärkeitä asioita. Kohderyhmän määrittäminen puolestaan auttaa käyttöliittymän ja sovelluksen ulkoasun suunnittelussa, ja näin itse sovellus voidaan toteuttaa sellaiseksi, että se parhaiten palvelee käyttäjiään. (Manninen & Marttila 2006, 378.)

Suunnitelman syvemmällä tasolla on hyvä tehdä yksityiskohtainen suunnitelma jokaisesta sovelluksen sivusta. Myös jokaisen näytöllä näkyvän objektin kaikki parametrit on määriteltävä. Verkkosovelluksen kohdalla on otettava huomioon sovelluksen tiedostokoko, sillä suurten tiedostojen lataaminen verkosta on hidasta. Paras lopputulos saadaan kun materiaalin tarkkuus määritellään niin, että latausajan ja laadun suhde on paras mahdollinen. Tämä tarkoittaa käytännössä kuvien värimäärän sekä video- ja äänitiedostojen kokojen hallintaa. (Manninen & Marttila 2006, 378.)

Käyttöjärjestelmällä ei Flash-sovellusten kannalta ole suurta merkitystä, koska Flash Player toistaa tiedoston sisällön. Kuitenkin varsinkin uusimmalla Fash 8-versiolla tehdyt sovellukset saattavat vaatia suhteellisen tehokkaan tietokoneen toimiakseen. Ohjelmointiin vaikuttavat tekijät sovelluksen kannalta ovat Flash Player ja sen versio, ja suunnittelun ja ohjelmoinnin kannalta onkin hyvä tietää ja ymmärtää eri Flash Player–versioiden erot ja mahdollisuudet. (Manninen & Marttila 2006, 378.)

Suurempia sovelluksia suunniteltaessa on otettava huomioon myös projektin resurssit ja kesto ja määritellä, miten projekti etenee näiden puitteissa. Projektiryhmältä vaaditaan monesti useiden eri sovellusten hallintaa, joten on hyvä määritellä myös sisällön ja tehtävien vastuualueet ja selvittää saadaanko osa materiaalista esimerkiksi työn tilaajalta. (Manninen & Marttila 2006, 378-379.)

3.3 Sovelluksen rakenteen suunnittelu

Sivustokartan luominen on hyvä tapa kuvata verkkosovellusta kokonaisuudessaan, sillä näin projekti on helppo hahmottaa sekä laajasta näkökulmasta että pienempien yksityiskohtien kannalta. Sivustokartta on kaavio sivuston rakenteesta, jäsentelystä sekä tiedon ja sisällön ryhmittelystä. Pieneissä projekteissa sivutokartan teko on suhteellisen helppoa ja nopeaa, mutta suurempien sovellusten kohdalla sivustokartan tekeminen vaatii enemmän aikaa ja ajatustyötä. (Goto & Cotler 2003, 92.)

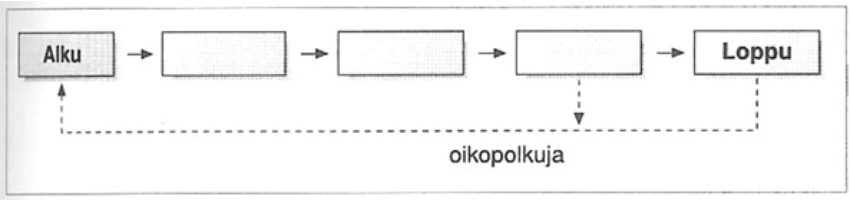
Sivustokartassa ei ole varsinaisesti kyse sivuston toiminnan kuvaamisesta vaan sen tarkoituksena on ainoastaan antaa kuva siitä, miten sivuston rakenne toimii. Sivustokartta toimii hyvänä runkona koko projektille, ja tämän vuoksi sitä on myös pidettävä ajan tasalla, kun sivuston rakenne muuttuu.

Sivustokartan pohjalta on helppo seurata myös projektin edistymistä, ja siihen on helppo tukeutua itse tuotantovaiheen aikana. Jokaiselle sivulle on mahdollisuuksien mukaan tehtävä oma laatikko, ja myös tärkeimmät linkit on hyvä kuvata sivustokartan yhteydessä. (Goto & Cotler 2003, 92-94.)

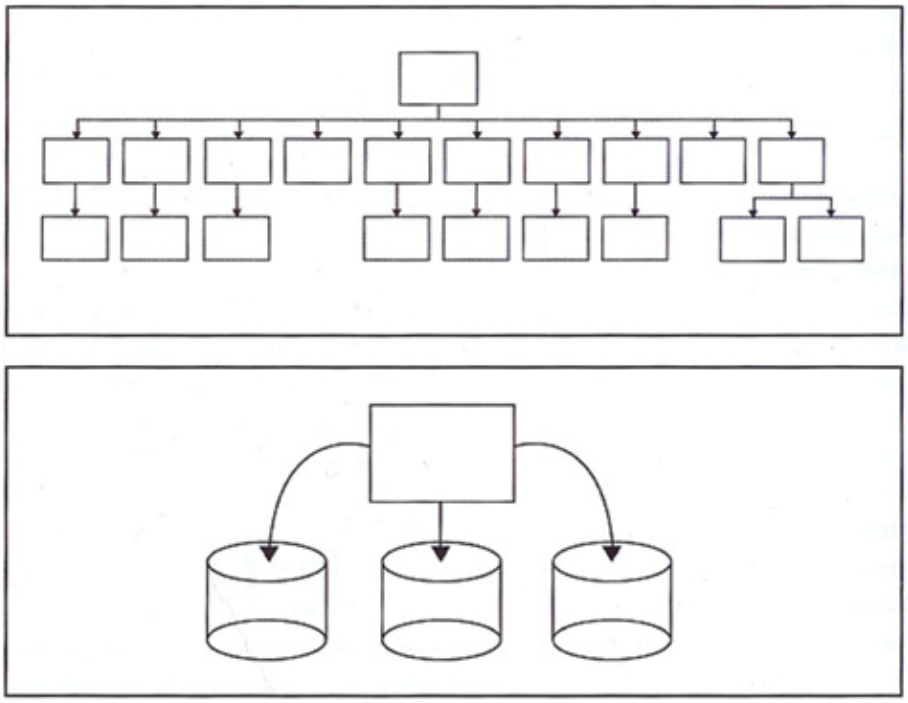
Sivustokartan pitäisi kuvata mahdollisimman yksityiskohtaisesti sovellusta, mutta myös yksinkertaisuus on hyvä pitää koko ajan mielessä sivustokarttaa kootessa. Sivustokartassa on hyvä tuoda myös selkeästi esille kohdat, jossa käyttäjä on vuorovaikutuksessa esimerkiksi tietokantojen kanssa. Esimerkkinä tästä esimerkiksi sivustolle sisään kirjautuminen, ja mitä onnistumis- ja epäonnistumistilanteissa tapahtuu. Sivustokartan yksityiskohtien määrä riippuu paljon suunnittelijan kokemuksesta. (Goto & Cotler 2003, 95.)

Sivustokartan rakenne voi olla monenlainen, mutta useimmiten se rakennetta kuvataan vasemmalta oikealle tai ylhäältä alaspäin luettavalla kuvaajalla. Koska sivustokartta tulee useissa tapauksissa muuttumaan, voidaan ensimmäinen kaavio tehdä esimerkiksi suoraan sivuston sisällön rungon mukaan. Näin päästään helposti myös hahmottamaan projektin organisaation liittyviä suunnitelmia. (Goto & Cotler 2003, 96.)

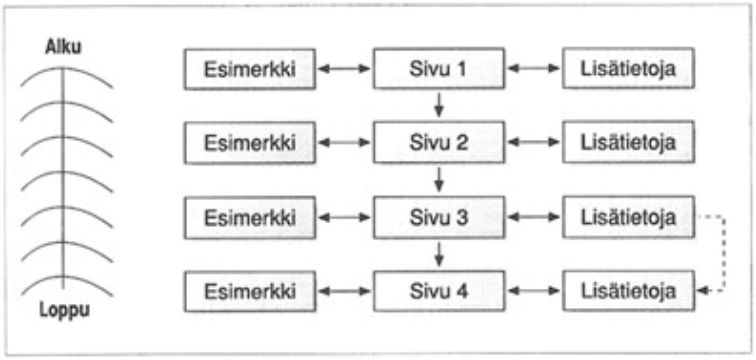
Sivustokartan yhteydessä kannattaa myös määritellä tiedostojen nimeämiskäytäntö, jolloin sivujen organisointi sujuu jouhevasti koko projektin ajan. Kun sivut on nimetty tietyn käytännön mukaan, kaikkien tiedostoja käsittelevien henkilöiden on helppo löytää tarvitsemansa tiedostot. (Goto & Cotler 2003, 96.)



KUVA 1. Lineaarinen tapa esittää sivustokartta. (Keränen, Lamberg, Penttinen 2001)



KUVA 2. Ylempi esimerkki on hierarkinen tapa esittää sivustokartta. Allemmasta kuvasta näkyy, miten tieto tallettuu tietokantoihin. (Goto & Cotler 2003)



KUVA 3. Sivustokartta kalnruotomaisella rakenteella. (Keränen, Lamberg, Penttinen 2001)

4 FLASH JA ACTIONSCRIPT

4.1 Yleistä

Flash –toimintaympäristö koostuu kolmesta peruselementistä, jotka ovat itse Flash sovellus, ActionScript –ohjelmointikieli ja Flash Player. Esitykset ja ohjelmat luodaan Flash sovelluksessa, jonka puitteissa voidaan käyttää myös ActionScriptiä. ActionScriptin käyttö esitysten luomisessa tarjoaa mahdollisuuden kehittyneempiin esitysmuotoihin. Flash Player on puolestaan esitysohjelma, joka osaa näyttää ainoastaan swf-tiedostoja. Useimmiten Flash Player toimii pluginina www-selaimen yhteydessä. (Manninen & Marttila 2006, 10.)

ActionScriptin järjestelmälliseen sijoittamiseen esityksessä on kolme vaihtoehtoa. Koodin voi sijoittaa aikajanan ensimmäisiin keyframeihin, jolloin koodin suoritetaan mahdollisimman lähellä esityksen alkua. Toinen vaihtoehto on kirjoittaa koodi esityksessä oleviin instansseihin. Kolmantena vaihtoehtona on ulkoinen ActionScript–tiedosto, josta koodia ajetaan esityksen aikana. (Manninen & Marttila 2006, 20.)

Koodin hajautuminen useaan eri paikkaan ja usealle eri tasolle esityksessä hankaloittaa ohjelmointia ja lisää virheitä. Esimerkiksi moninkertaiset MovieClip –rakenteet kasvattavat ohjelmointiongelmien määrää, sillä viittauksia eri tasoille ja eri instansseihin tulee paljon. Suositeltavin tapa tällä hetkellä on kirjoittaa koodia vain päätason aikajanan ensimmäisiin keyframeihin tai ulkoiseen ActionScript –tiedostoon. (Manninen & Marttila 2006, 20.)

Koodia ei ole aina mahdollista sijoittaa ainoastaan päätasolle. Tällöin on kuitenkin hyvä käyttää yhtenäistä ohjelmointitapaa läpi esityksen, jolloin ongelmanratkonta helpottuu, ja se auttaa muita esitystä käsitteleviä henkilöitä löytämään ohjelmaan sisällytetyn koodin. (Manninen & Marttila 2006, 20.)

Esityksen julkaisuversio on hyvä päättää jo ennen kuin esitystä aletaan käytännössä toteuttaa, sillä vanhemmat FlashPlayer versiot eivät tue kaikkia uusimmissa ohjelmaversiossa käytössä olevia ominaisuuksia. Julkai-

suasetukset voidaan säätää halutuksi heti esityksen teon alkaessa, jolloin esityksestä rajataan pois ominaisuuden, joita valittu julkaisuversio ei tue. Esimerkiksi ActionScript funktiot, joita vanhemmat julkaisuversiot eivät tue, värjäytyvät keltaiseksi Actions -ikkunan komentoluettelon listassa. (Manninen & Marttila 2006, 22.)

4.2 Flash ja XML

4.2.1 Yleistä XML:stä

XML on elementtipohjainen merkintäkieli, joka on riippumaton käytettävästä ohjelmistosta ja laitteistosta. Sen avulla voidaan siirtää tai tallentaa tietoa, ja tieto itsessään on kuvattu niin, että sen sukulaissuhteet on helppo selvittää. XML-merkintäkieli muistuttaa rakenteeltaan HTML-kieltä, mutta toisin kuin HTML-kielessä XML:n elementtejä ei ole määritelty etukäteen. Tämä antaa dokumenttien suunnittelijoille vapauden kuvata tietoa omilla elementeillä. (Manninen & Marttila 2006, 386.)

SVG ja RSS ovat kaksi laajalti tunnettua XML:n sovellutusta. SVG on tiedostomuoto, jolla voidaan kuvata vektorigrafiikkaa ja RSS käytetään Internet-ympäristössä esimerkiksi uutisten tai muun usein päivittyvän tiedon jakeluun. XML-tiedostoa muotoiltaessa on kuitenkin oltava tarkkana, sillä jos tiedosto ei ole oikeaoppisesti muotoiltu, ei tiedoston toimivuudestakaan ole takeita. Tarkan muotoilun vaatimus on kuitenkin selaimien kannalta hyvä asia, sillä silloin niiden ei tarvitse ottaa huomioon monia erikoistapauksia. (Heinisuo & Rauta 2007, 250-251.)

XML-kieleen kuuluu olennaisena osana DOM-määrittely. DOM eli Document Object Model on oliomalli, jolla kuvataan HTML- ja XML-tiedostojen rakennetta. Oliomallissa jokaisella dokumentin elementillä on oma olionsa, joka sisältää jäsenmuuttujia sekä jäsenfunktioita. Elementin sisältämät lapsielementit esimerkiksi ovat tämän elementin jäsenmuuttujia. Oliotyyppinen tiedonjäsentely puolestaan antaa mahdollisuuden käsitellä dokumentin sisältämää tietoa hyvin monilla eri tavoilla. (Heinisuo, Rauta 2007, 253-254.)

Flash-sovelluksessa XML-dokumenttien tietoa voidaan käyttää tiedon lataamiseen sovelluksen ulkopuolelta tai välittämään tietoa palvelimen ja Flash-sovelluksen välillä. Tietoa voidaan ladata käyttämällä XML-luokan load()-metodia. Vastaavasti tietoa voidaan lähettää saman luokan send() tai sendAndLoad()-metodeilla. (Manninen & Marttila 2006, 387-390.)

4.2.2 XML-objektin alustus ja tiedon lisääminen

Kutsumalla XML-luokan alustajaa Flash-sovelluksessa saadaan käyttöön uusi XML-objekti. Luokkaa on mahdollista kutsua ilman parametreja tai käyttämällä parametreja välittämään XML-tietoa merkkijonona. Parametrin tieto muutetaan automaattisesti vastaamaan rakenteeltaan XML-objektin sisältöä. (Manninen & Marttila 2006, 387.)

```
01 // tyhjän XML-objektin luominen
02 var my1_xml:XML = new XML();
03 //suoraa tietoa sisältävän XML-objektin luominen
04 var my2_xml:XML = new XML(”<hinta>2.30</hinta>”);
05 trace(my2_xml.firstChild.firstChild.nodeValue);

Tulostaa: 2.30
```

ActionScript-ohjelmointikieltä käyttämällä XML-objekteihin voidaan lisätä tietoa. Tietoa voidaan lisätä käyttämällä parseXML()-, createElemet()- ja createTextNode()-metodeja. Edellämainituista metodeista parseXML() toimii kuten XML-luokan alustaja eli se muuttaa syötetyn tiedon XML-rakenteeksi XML-objektiin. Kahta jälkimmäistä metodia käytetään DOM määrittelyjen mukaisesti lisäämällä uusia solmuja XML-objektiin. Uuden solmut liitetään toisiinsa appendChild()-metodilla. (Manninen & Marttila 2006, 387.)

4.2.3 Tiedon lähettäminen lataaminen

XML-luokan send()- ja sendAndLoad()-metodit mahdollistavat tiedon lähettämisen Flash-sovelluksesta XML-muotoisena. Näistä send()-metodi mahdollistaa ainoastaan tiedon lähettämisen parametrina määriteltyyn URL-osoitteeseen. Jos tiedon välittymisestä halutaan palautetta, on käytettävä sendAndLoad()-metodia. (Manninen & Marttila 2006, 390.)

Tiedon lähetys voidaan todentaa palauttamalla toisena parametrina määriteltyyn XML-objektiin lähetyksen kohteena olevasta osoitteesta tietoa. Molemmat edellä esitellyt metodit muuttavat XML-objektin sisältämän tiedon automaattisesti XML-dokumentiksi. (Manninen & Marttila 2006, 390.)

4.2.4 XMLNode

Flash-sovelluksessa olevien XML-objektien solmujen sisältämää tietoa voidaan käsitellä XMLnode-luokan ominaisuuksien avulla. XMLnode-luokan ominaisuuksia ovat firstChild, lastChild, nextSibling, parentNode ja previosSibling. Näiden ominaisuuksien avulla saadaan tietoa käsittelyssä olevasta solmusta katsoen tietoa muista solmuista. Näiden lisäksi nodeName ja nodeValue–ominaisuuksien avulla voidaan selvittää XML-solmun nimi tai solmun sisältämä arvo. (Manninen & Marttila 2006, 393.)

Jokaisen XML-solmun sisällön voi tyyppi voidaan määritellä nodeType-ominaiuuuden avulla. Tyyppi määritellään arvon avulla, jotka ovat 1(elementti), 2(attribuutti), 3(tekstisolmu). Edellä mainittujen ominaisuuksien lisäksi XMLnode-luokka sisältää vielä childNodes-taulukon ja attributes-ominaisuuden. ChildNodes sisältää käsiteltävän solmukohdan kaikki lapsisolmut taulukkomuodossa ja attributes puolestaan sisältää yhden solmun kaikki attribuutit. (Manninen & Marttila 2006, 393-394.)

4.3 AMFPHP

AMFPHP on ilmainen open source –projekti, joka on toteutettu PHP:llä ja se mahdollistaa Flash-sovelluksen ja tietokantojen välisen yhteyden. Flash –sovellus voi välittää ja vastaanottaa tietoa palvelimelta saumattomasti, sillä Flash Player itsessään sisältää Flash Remoting –tekniikan. Tekniikan avulla on mahdollista kutsua suoraan palvelimelta PHP:llä toteutettuja metodeja, ja näin sovellukselle voidaan monipuolisesti palauttaa tietoa. (Manninen & Marttila 2006, 397.)

AMFPHP:n käyttämän yhteyssillan yli voidaan siirtää myös monimutkaisia tietotyyppejä toisin kuin LoadVars- ja XML-objektien avulla. Käytännössä tieto välittyy Macromedian AMF -tiedostomuodossa binäärisesti http-protokollasta läpi. Näin on mahdollista välittää suoraan SQL-lausekkeen

palauttamaa tietoa yhteyssillan kautta Flash-sovelluksen komponentteihin eikä palvelimella tarvitse muodostaa muuttujia tai määritellä XML-muotoista tietoa. (Manninen & Marttila 2006, 397.)

AMFPHP on palvelimella toimiva ohjelmakokonaisuus, joka ladataan WWW-palvelun kotihakemistoon tai käyttäjän kotihakemiston www-kansioon. Flash 8 –version mukana ei automaattisesti tule Flash Remoting –komponentteja tai –luokkia, ja ne onkin ladattava erikseen Macromedian sivuilta. (Manninen & Marttila 2006, 398.)

5 PHP

5.1 PHP-ohjelmointi

PHP eli Hypertext Preprocessor oli alun perin kokoelma komentoja, jotka helpottivat WWW-pohjaisten sovellusten palvelinpohjaisten ohjelmien tekemistä. Nykyistä PHP-ohjelmointikieltä voidaan käyttää suoraan HTML-kielen lomassa, mutta PHP-koodia sisältävät tiedostot on aina ladattava WWW-palvelimelta, koska koodi suoritetaan palvelimella ennen kuin tieto lähetetään selaimeen. Jos PHP-tiedosto avataan suoraan tietokoneen kovalevyltä, sivut eivät toimi. (Heinisuo, Rauta 2007, 12.)

PHP:n perusrakenne muistuttaa hyvin paljon esimerkiksi C- tai Java-kieltä, ja mikäli näitä kieliä osaa jo entuudestaan PHP:tä on helppo oppia (Heinisuo, Rauta 2007, 63). PHP-koodi kirjoitetaan sivujen HTML-koodin lomaan ja se kirjoitetaan aina <?php -aloitusmerkin ja ?> -lopetusmerkin väliin. Edellä mainittuja merkkejä on aina käytettävä, jotta kirjoitettu PHP tulisi tunnistetuksi ja suoritetuksi. (Heinisuo, Rauta 2007, 64.)

PHP:ssä käytetään silmukoita, ehtolauseita ja muuttujia samalla tavoin kuin esimerkiksi C-kielessä. PHP:ssä muuttujia merkitään kuitenkin $-merkillä ja muuttujan nimi saa sisältää ainoastaan kirjaimia välillä a-z, numeroita ja alaviivoja. Skandien ja erikoismerkkien käyttö puolestaan saattaa aiheuttaa ongelmia. (Heinisuo, Rauta 2007, 66.)

Muuttujiin voidaan tallentaa muun muassa tekstiä, joka kirjoitetaan lainaus- tai heittomerkkien sisään riippuen siitä miten PHP:n halutaan syötettyä merkkijonoa käsittelevän. Heittomerkkien (’teksti’) sisällä oleva teksti esitetään täysin sellaisenaan selaimessa, kun taas lainausmerkkien (”teksti”) sisällä olevaan merkkijonoon lisätään mahdollisten muuttujien tilalle niiden arvot. Muuten teksti toimii samalla tavoin riippumatta kumpia merkkejä käytetään. (Heinisuo, Rauta 2007, 69.)

5.2 PHP ja tietoturva

5.2.1 Verkkosovellusten tietoturvasta

Verkkosovelluksien tietoturvasta puhuttaessa kyse ei ole niinkään viruksista ja haittaohjelmista vaan palvelimen ja sillä olevien tietojen käytön turvaamisesta asiattomalta käytöltä. Palvelimen tietoturva perustuu paljolti siihen, että käyttäjän toiminnot palvelimella rajataan mahdollisimman tarkasti vastaamaan palvelun ylläpitäjän toiveita. (Manninen & Marttila 2006, 412.)

Sovellukset ovat usein näkyvillä kaikille internetin käyttäjille, ja tietoturvauhka koskee sekä sovellusta että sen käyttäjiä ja heidän tallentamiaan tietoja. Usein sovelluksen väärinkäytön tavoitteena on päästä käsiksi oikeuksiin, jotka mahdollistavat enemmän toimenpiteitä kuin palvelun ylläpitäjä on määritellyt tavalliselle käyttäjälle. Taitava krakkeri voi esimerkiksi muuttaa tai poistaa tietoa palvelimelta, asettaa palvelimen levittämään viruksia tai roskapostia tai sulkea sovelluksen kokonaan. (Manninen & Marttila 2006, 412.)

PHP:tä käyttävissä sovelluksissa ohjelma saa syötettä, jonka perusteella se antaa jonkinlaisen palautteen tai se hakee itselleen tietoa jostain ulkoisesta lähteestä. PHP:ssä syötteeksi luettavia taulukoita ovat $_GET, $_POST ja $_COOKIE sekä kaikkien näiden tiedot sisältävä $_REQUEST. Ulkoisia lähteitä sovelluksessa ovat esimerkiksi tietokannat tai tiedostojärjestelmät. (Heinisuo & Rauta 2007, 399.)

5.2.2 Käyttäjän antama syöte

PHP:tä käytettäessä tärkein sääntö on, ettei käyttäjän syöttämään tietoon kannata luottaa sokeasti. Jos käyttäjän lähettämää tietoa ei mitenkään varmisteta, ennemmin tai myöhemmin joku käyttää sovellusta ei-toivotulla tavalla. Käyttäjän lähettämän tiedon seassa voi esimerkiksi olla PHP-komentoja, joiden avulla saadaan selville käyttäjien salasanat tai jotka antavat oikeudet lukea, poistaa tai muuttaa tietoja palvelimella. (Manninen & Marttila 2006, 412-413.)

Käyttäjän syöttämän tiedon muotoa voidaan rajata erilaisin keinoin. Esimerkiksi syötetystä tiedosta voidaan rajata pois merkkejä, joiden avulla on mahdollista suorittaa haitallisia komentoja. Jos syötettävän tiedon tulisi olla numeromuotoista, voidaan sarakkeesta lähtevän tiedon tyyppi rajata numeromuotoiseksi. Myöskään sivun nimeä ei kannata ottaa vastaan kokonaisuudessaan, vaan se voidaan esimerkiksi ottaa vastaan numerona, jota verrataan hyväksyttyjen sivujen listaan. PHP-funktioista esimerkiksi addslashes() -, strip_tags() -, ja htmlentities() ovat käyttökelpoisia tapoja varmistaa tiedon turvallisuutta. (Manninen & Marttila 2006, 413-414.)

5.2.3 Tietokantojen luotettavuus

Tietokannoista saatuun tietoon voidaan pääasiallisesti luottaa, mikäli tietokantaa käyttää vain itse sovellus. Tietokantaan on kuitenkin mahdollista syöttää haitallista tietoa, mutta yksi hyvä tapa ehkäistä tällaisen tiedon syöttämistä, on käyttää taulukon sarakkeissa oikeita tietotyyppejä. Esimerkkinä voidaan ajatella saraketta jonka tietotyyppi on int, jolloin se voi sisältää kokonaislukuarvoja. (Heinisuo & Rauta 2007, 402.)

Myös istuntotaulukoista saatava tieto on kohtuullisen luotettavaa, sillä sovellus on itse kirjoittanut sieltä saatavat tiedot. Käytännössä on kuitenkin hyvä tarkistaa kaikki saadut syötteet riippumatta siitä mistä lähteestä tieto on peräisin. (Heinisuo & Rauta 2007, 402.) Virhetilanteita varten on myös hyvä luoda virhesivu, joka näytetään käyttäjälle virheviestin sijaan. Virheviestit itsessään saattaa sisältää tietoja, joita ei haluta muiden käyttäjien näkyville. (Manninen & Marttila 2006, 414.)

Palvelimen www-hakemiston sisältö on julkinen kaikille internetin käyttäjille, joten palvelimella ei kannata säilyttää tiedostoja, joista selviä esimerkiksi käyttäjätunnuksia ja salasanoja. Toinen huomioon otettava seikka on se, että mikäli index-sivua ei löydy, muun muassa Apache-palvelin antaa käyttäjälle listan hakemistossa olevista tiedostoista. (Manninen & Marttila 2006, 414.)

5.2.4 Tuloste PHP-sovelluksessa

Lomakkeiden piilokentät mahdollistavat tiedon siirron kätevästi ilman istuntoja, mutta on myös hyvä harkita, millaista tietoa tulostat käyttäjälle näytettäville sivuille. Piilokentissä sivulta toiselle siirtyvät salasanat tai käyttäjän ID:t eivät esimerkiksi ole hyvä ajatus, sillä kaikki palvelimelta selaimelle lähetetty tieto käyttäjän luettavissa ja väärinkäytettävissä. Käyttäjää on mahdotonta estää lukemasta sivun lähdekoodia, joten jos käyttäjän ei haluta näkevän jotain tietoa, sitä ei kannata lähettää ollenkaan. (Heinisuo & Rauta 2007, 402-403.)

Esimerkiksi JavaScriptillä on mahdollista saada aikaan dynaamisuutta verkkosivulle, mutta tätä mahdollisuutta ei kannata hyödyntää olennaisen toiminnan luomiseen. Muun muassa lomakkeiden arvoja on mahdollista tarkistaa jo selaimessa ennen lomakkeen välittämistä palvelimelle, ja käytännössä tiedon välittäminen ilman sivun uudelleenlatausta parantaa käytettävyyttä. JavaScriptillä välitettyihin tietoihin ei kuitenkaan voi luottaa, ja tietojen tarkistus on aina tehtävä myös PHP-tiedostossa, jossa käytetään lomakkeen arvoja. (Heinisuo & Rauta 2007, 402-403.)

6 RELAATIOTIETOKANNAT

6.1 Relaatiotietokantojen suunnittelu

6.1.1 Suunnittelusta

Tietokantojen suunnittelu on yksi tärkeä osa-alue tietokantapohjaisen verkkosovelluksen tuottamista. On olennaista tietää, kuinka monta taulua tietokantaan tarvitaan ja millaisia sarakkeita kussakin taulussa tarvitaan. Relaatiotietokantojen etuna on tietojen helppo saatavuus, sillä taulujen välille luotavat yhteydet nopeuttavat ja tarkentavat tietojen päivittämistä ja tiedot on tallennettu vain yhteen paikkaan tietokannassa. (Sarja 2006.)

Suunnittelutapoja on monia erilaisia, mutta tunnetuin ja käytetyin tapa on ER-kaavion luominen tietokannan rakenteesta, ominaisuuksista ja suhteista (Lahtonen 2002, 18). Relaatiotietokannan suunnitteluun kuuluu neljä tärkeää käsitettä, jotka ovat käsiteanalyysi, tarveanalyysi käsitemallin luominen ja normalisointi (Asmala 2005.).

6.1.2 Käsiteanalyysi

Käsiteanalyysissa rajatun kohdealueen tietoja luokitellaan, yksilöidään, nimetään ja ryhmitellään. Näiden yhteydessä määritellään myös käsitteiden välisiä riippuvuuksia ja ominaisuuksia. Käsiteanalyysin lopputuote on käsitemalli, joka kuvaa työryhmän jäsenten yhteistä näkemystä kohdealueesta. Käsiteanalyysin tavoitteena on, että jokainen tieto tallennetaan tietokantaan vain kerran. (Asmala 2005.)

Käsiteanalyysin kuvauksen kohteita ovat kohdealueen käsitteet, tiedot ja yhteydet. Käsitteet ovat asioita, joista on tarve tallentaa tietoja tietokantaan, ja niitä kuvataan ER-kaaviossa laatikolla. Yleisiä käsiteluokkia ovat esimerkiksi esineet, henkilöt, tapahtumat, joista on tallennettava tietoa tai käsitteelliset yksilöt kuten esimerkiksi kustannuspaikka tai työsuhde. (Asmala 2005.)

Tiedot eli attribuutit ovat käsitteestä tallennettavia asioita, jotka on merkitty ER-kaavioon ovaalilla. Tällaisia asioita voivat olla esimerkiksi käyttäjän nimi, osoite tai vaikkapa asiakastunnus. Jokaisen käsitteen tietojoukosta yksi tieto on käsitteen yksilöivä tieto eli perusavain. Perusavain on tieto, joka ei voi olla sama millään toisella taulun jäsenellä. (Asmala 2005.)

Käsitteiden välillä on yhteyksiä, joita kuvataan käsitemallissa kärjellään olevana neliönä käsitteiden välillä. Yhteyksiä luodessa on tärkeää muistaa, ettei yhteyksiin voi liittää tietoa, joka halutaan tallettaa tietokantaan. Yhteydestä on tiedettävä niiden pakollisuus ja lukuisuus ja ne katsotaan kummastakin osapuolesta erikseen. Yhteyden tulee kuvata asioiden suhdetta niin kuin ne esiintyvät oikeassa elämässä. (Asmala 2005.)

6.1.3 Tarveanalyysi

Tarveanalyysin tarkoituksena on tarkentaa käsiteanalyysissa tuotettua käsitemallia testaamalla sitä tiedossa olevilla tietotarpeilla. Tarveanalyysin aikana käsitemalliin lisätään uusia tietoja ja tarpeen mukaan myös uusia käsitteitä ja yhteyksiä. (Asmala 2005.)

Tietotarpeet ovat sovelluksessa esiintyviä ikkunoita, raportteja, eräajoja tai muita ohjelmia, jotka käyttävät hyväkseen tietokannassa olevia tietoja. Tarveanalyysi selvittää myös tieto-, käyttäjä- ja tapahtumamäärät, joita käytetään muun muassa testaukseen ja hallintajärjestelmän valintaan. (Asmala 2005.)

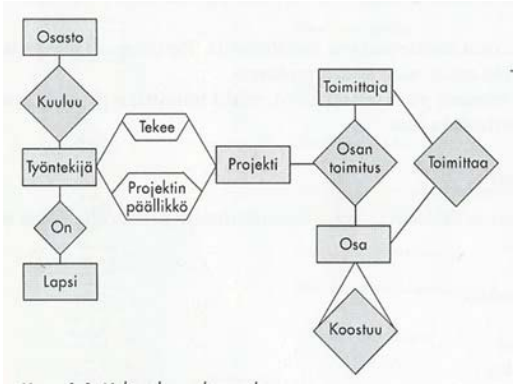
Suurten tietovarastojen raportointi tarpeet voivat vaatia lisäämään käsitemalliin esimerkiksi summatauluja, saldoja, viimeisiä tapahtumapäiviä tai historiatauluja. Myös tietokannan eheys on kokoajan pidettävä mielessä, ja on vahdittava tunnuslukujen ja summien ajantasaisuutta. (Asmala 2005.)

6.1.4 Käsitemalli

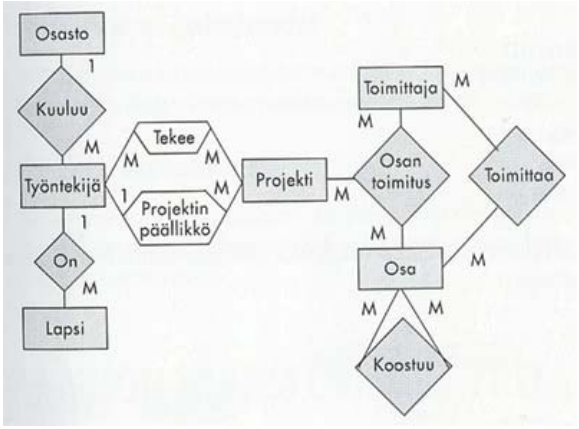
Käsitemalli itsessään on tuote ja tietokantariippumaton ja se kuvaa kohdealueen käsitteitä vain yleisellä tasolla. Tämä tarkoittaa sitä, ettei käsitemalli ota kantaa sovelluksen teknisiin puoliin kuten esimerkiksi suorituskykyasioihin tai fyysiseen rakenteeseen. Käsitemalli kuvataan useimmiten graafi-

sena ER-kaaviona, joka toimii tietokannan piirustuksina toteutusvaihetta varten. (Asmala 2005.)

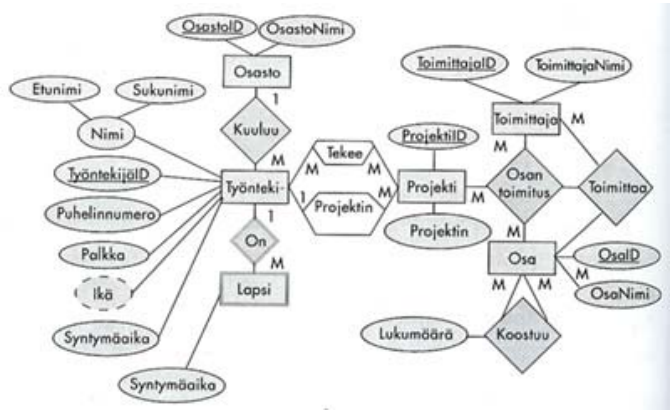
Käsitemallissa jokainen kohdealueen olennainen asia on määritelty vain yhdellä tavalla joko käsitteeksi, tiedoksi tai yhteydeksi. Käsitemallissa vain käsitteellä voi olla ominaisuuksia ja yhteys voi olla vain kahden käsitteen välillä. (Asmala 2005) Yhteyksiä voi kuitenkin olla eri tyyppisiä. Nämä tyypit ovat yhden suhde yhteen, yhden suhde moneen tai monen suhde yhteen ja monen suhde moneen. Suhteiden tyyppiä voidaan kuvata kaaviossa esimerkiksi numero yhdellä ja m-kirjaimella. (Lahtonen 2002, 19-20.)



KUVA 4. Kohteiden välisiä suhteita kuvaava ER-kaavio. (Lahtonen 2002.)



KUVA 5. Kuvassa merkitty kuinka moneen suhteeseen tietty kohde voi osallistua. (Lahtonen 2002.)



KUVA 6. Kuvaan merkitty eri kohdealueiden ominauudet. Avainominaisuudet on alleviivattu. (Lahtonen 2002.)

6.1.5 Normalisointi

Normalisoinnin tarkoituksena on vähentää tietojen toistumista tietokannassa ja tästä johtuvia ongelmia tietojen hallinnassa. Normalisointi lisää myös tietojen rakenteen selkeyttä, yhtenäisyyttä ja sekä laajennettavuutta. Normaalimuotoja on olemassa kuusi, joista kolmea ensimmäistä yleisimmin käytetään. (Lahtonen 2002, 30-31.)

Normaalimuodot ovat:

- ensimmäinen normaalimuoto (1NM)
- toinen normaalimuoto (2NM)
- kolmas normaalimuoto (3NM)
- Boyce-Codd normaalimuoto (BCNM)
- neljäs normaalimuoto (4NM)
- viides normaalimuoto (5NM)

Käytännössä normalisointi merkitsee sitä, että tietokannan tauluja jaetaan useammaksi tauluksi normalisointisääntöjen mukaan. Normalisoinnin hyödyt tulevat erityisesti esille tietokannoissa, joita päivitetään usein. Toisaalta, jos tietokannasta ainoastaan luetaan tietoa, saattaa normalisoimaton taulukanta olla toimivampi. Jos tiedot on jaettu useaan eri tauluun, tietojen hakemiseen tarvitaan useampia hakuja, jolloin tiedon hakeminen vaatii

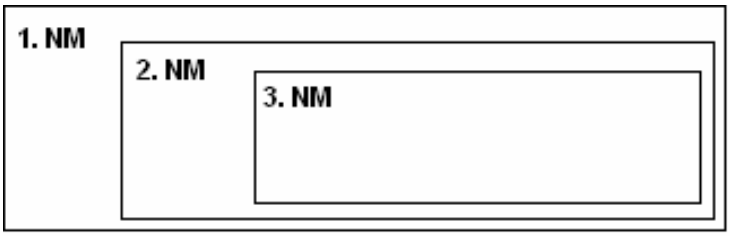
enemmän tehoa kuin normalisoimattomassa tietokannassa. (Asmala, 2006a.)

Useampaan tauluun jakaminen toteutetaan aina siten, ettei mikään tieto informaatio häviä normalisoinnin aikana. Kun yksi taulu on jaettu useaksi, on niiden välille myös luotava viiteavain, jonka avulla alun perin yhdessä taulussa olleet tiedot saadaan jälleen koottua yhteen. Normalisointi tapahtuu luontevasti noudattamalla kahta sääntöä:

1) Yhdessä taulussa on vain sellaista tietoa, joka välittömästi liittyy taulun käsitteeseen

2) Kunkin tiedon päivitys tehdään vain yhteen paikkaan (Lahtonen 2002, 31.)

Normalisointisäännöt tarkentuvat asteittain alkaen ensimmäisestä ja päättyen viidenteen normaalimuotoon. Normaalimuodot toimivat siis siten, että kolmannessa normaalimuodossa oleva taulu täyttää myös ensimmäisen ja toisen normaalimuodon säännöt. On myös hyvä muistaa, että normalisoidusta taulusta puhuttaessa tarkoitetaan kolmannessa normaalimuodossa olevaa taulua. (Asmala 2006a.)



KUVA 7. Normaalimuotojen hierarkia. (Asmala 2006.)

Ensimmäisessä normaalimuodossa olevista tauluista on poistettu kaikki toistuvat ryhmät, jolloin relaatioiden määrittelyjoukko on yksinkertainen. Toisessa normaalimuodossa poistetaan kaikki osittaiset funktionaaliset riippuvuudet ja kolmannessa muodossa kaikki transitiiviset riippuvuudet poistetaan. (Lahtonen 2002.)

Normalisoinnin lisäksi joudutaan joskus käyttämään denormalisointia, jossa vähennetään taulujen välisiä liitoksia ja lisätään ylimääräistä tietoa. Taulut siis eivät siis enää noudata kolmannen normaalimuodon sääntöjä, mutta tiedonhaku on nopeampaa. Denormalisoidut muodot nopeuttavat tiedonhakua, ja ovat käteviä varsinkin jos tietokannan tietoja ei tarvitse usein päivittää. (Asmala 2006a.)

6.2 SQL

SQL eli Structured Query Language pitkälle standardoitu kieli, jolla tehdään kyselyjä ja määrittelyjä relaatiotietokantoihin. Komentojasanoina kielessä ovat esimerkiksi select, update, delete, where ja order by. Taulukoiden luomisen ja poistamisen lisäksi kieli tarjoaa komennot tallennettujen tietojen käsittelyyn. Komennot jaetaan määrittely- ja käsittelykomentoihin sen pohjalta luodaanko tietokantaan uusia tietorakenteita vai käsitelläänkö jo olemassa olevia taulukoita. Yleisesti komentoja kutsutaan tietokantahauiksi tai vieläkin lyhyemmin pelkiksi hauksi. (Heinisuo, Rauta 2007, 97.)

PHP-ohjelmointikielellä voidaan hallita SQL-tietokantoja, sillä PHP tarjoaa rakenteet ja komennot, joiden avulla välitetään SQL-komentoja tietokannalle. PHP:n avulla tietokantojen sisältö voidaan lukea niin, että tieto näkyy WWW-sivuna. PHP:n ja SQL-kielten hallitseminen antaa valmiudet luoda tietokantapohjaisia WWW-sivuja ja –palveluja joustavalla ja yksinkertaisella tavalla. (Heinisuo, Rauta 2007, 97.)

Erilaisten kyselyiden teko on SQL-kielen yleisin käyttöalue, ja sen avulla tietokantaan tallennetuista tiedoista voidaan muodostaa erilaisia koosteita, raportteja ja yhteenvetoja. Tietojen lisäämistä, muuttamista ja poistamista tehdään harvemmin suoraan SQL-kielellä muuten kuin tietokantaohjelmistoja rakennettaessa. (Lahtonen 2002, 39-40.)

SQL:ää käytetään seuraavilla tavoilla eri yhteyksissä:

Vuorovaikutteinen SQL

Vuorovaikutteista SQL:ää käytetään suoraan omassa ikkunassaan, johon myös hakujen vastaukset suoraan saadaan. Tätä tapaa käytetään yleensä SQL-kielen opettelussa ja tietokannan testauksessa. (Lahtonen 2002, 39.)

Upotettu SQL

Upotettua SQL:ää käytettäessä käskyt on upotettu ohjelmointikieleen tai sovellus-/raporttikehittimeen. Vastaukset tietokannasta saadaan suoraan ohjelmointikielen muuttujiin, josta hakutuloksia voidaan käyttää itse ohjelman tarpeisiin. (Lahtonen 2002, 39.)

Dynaaminen SQL

Dynaamisesti käytettynä SQL-kielen käskyjä luodaan ohjelmassa sen tarpeiden mukaan. Tuotettu käsky lähetetään tietokantajärjestelmälle toteuttamista ja suorittamista varten. (Lahtonen 2002, 39.)

6.3 SQL-sovellukset ja WWW

Tietokantoja käytetään nykyään monien verkkosivujen ylläpitämiseen, sillä sivujen dynaamisen ja tiedon muokattavuus sivuilla on tärkeä ominaisuus. Koska tietojen keruu on helppo tietokantoja käyttämällä, voidaan niiden avulla toteuttaa esimerkiksi myös verkkokyselyitä tai kerätä sivuston käyttäjätietoja tietokantaan. (Lahtonen 2002, 164.)

Pelkkien dynaamisesti tietoa lataavien verkkosivujen tuottaminen ei ole vaikeaa, mutta jos tietokannan sisältöä on myös päivitettävä selaimen avulla, on toteuttaminen usein vaikeampaa. Useimmiten palvelimen ja selaimen välistä yhteyttä käytetään vain tarvittaessa, ja tällöin tietojen muokkaaminen, lisääminen ja poistaminen on ongelmallista. Jos tietokannan tietoja ei voi muokata kuin yksi henkilö kerrallaan ei suuria hankaluuksia

pitäisi ilmetä, mutta esimerkiksi tietojen päivittäminen ja poistaminen vaatii lukitusten määrittelemistä tietokantaan. (Lahtonen 2002, 164-165.)

WWW-palvelin hoitaa sivujen koodin suorittamisen ja yhteydenotot tietokantaan ja välittää saamansa tiedot selaimelle. Tämä tuo HTML-sivuihin verrattuna palvelimelle enemmän vastuuta verkkosivujen toimivuuden osalta. Dynaamiset sivut myös kuormittavat palvelinta enemmän kuin pelkkä HTML-dokumenttien välitys, ja palvelimen teho tulee aina ottaa huomioon ja miettiä, kuinka suuresta kohderyhmästä sivujen kohdalla on kyse. Käytännössä mitä suurempi käyttäjäryhmä on kyseessä, sitä tehokkaamman palvelimen sivusto vaatii. (Lahtonen 2002, 164-165.)

6.4 Tietokannan eheys

Tietokannan eheydellä tarkoitetaan tietokannan taulujen ja tietojen oikeellisuutta, yhdenmukaisuutta, ristiriidattomuutta ja samaa tuoreustasoa. Eheys jaetaan viite-eheyteen, avaineheyteen ja attribuuttieheyteen. (Sarja 2006.)

Viite-eheys

Viite-eheys on rakenteellinen piirre, jonka mukaan taulun kentistä voidaan viitata toisen taulun perusavaimeen tai toissijaiseen avaimeen. Viittaavaa kenttää kutsutaan viiteavaimeksi ja niiden arvojen on vastattava jotakin viitattavan taulun avainarvoista. Muita kuin viittauksen kohteena olevan taulun avainarvoja ei voi käyttää viiteavainkentässä. (Lahtonen 2002, 12.)

Tietokannan hallintajärjestelmissä on yleensä mahdollisuus valvoa viite-eheyden säilymistä. Viite-eheys ei kuitenkaan koske NULL-arvoja, joten yleensä on järkevää estää tyhjän kentän tallentaminen viiteavainkentissä. Viittauksen kohteena olevan taulun tietoja voidaan kuittenkin yrittää muuttaa tai poistaa, jolloin viite-eheys saattaa rikkoutua. (Lahtonen 2002, 12-13.)

Jos tietoja muutetaan tai poistetaan viittauksen alaisena olevasta kentästä, voidaan viite-eheyden rikkoutumista estää seuraavilla periaatteilla:

- Nollataan viittaavan taulun viiteavaimet, mikäli NULL-arvojen käyttämistä ei ole kielletty.
- Asetetaan oletusarvo kaikkiin muuttuvaan tietoon viittaaviin viiteavaimiin.
- Uusi perusavaimen arvo päivitetään automaattisesti siihen viittaavan viiteavaimen arvoksi johdannaismuutoksen avulla.
- Rajoitetaan muutoksia siten, että vain sellaisia perusavaimia voi muuttaa, joihin ei liity viiteavaimia. (Lahtonen 2002, 13.)

Avaineheys

Avaineheyssäännön mukaan kunkin perusavaimen on oltava uniikki eikä yksikään perusavaimeen kuuluvista kentistä voi saada tyhjää arvoa. Tietokantojen hallintajärjestelmissä on useimmiten mahdollisuus valvoa myös avaineheyden säilymistä. (Asmala 2006b.)

Attribuuttieheys

Attribuutti- eli arvoalue-eheys toteutuu kun kentän kaikki arvot kuuluvat kentälle määrättyyn arvojoukkoon ja muotoon tai tiedon arvo on nolla, jos se on kentälle sallittua. Esimerkiksi päivämäärä voidaan määrätä tallennettavaksi muotoon pp.kk.vvvv. (Asmala 2006b.)

6.5 Tietokannan määritely

6.5.1 tietokannan luominen

Yleisesti ottaen yhteen tietokantaan tallennetaan yhden sovelluksen tiedot ja niitä voi olla palvelimella useampia. Ne ovatkin suurimpia yksittäisiä kokonaisuuksia, joita palvelin sisältää. Tietokanta luodaan create database –komennolla, jonka yhteydessä yleensä määritellään myös käytettävä merkistö sekä aakkosjärjestys. (Heinisuo, Rauta 2007, 103.)

Tietokannan rakenne on tärkeää saada heti määritellyksi oikein, sillä rakenteen muuttaminen jälkikäteen on usein hankala. Rakenteen muuttumisen myötä myös kaikki hakulausekkeet voidaan joutua tekemään kokonaan uudestaan. Jos tietokannan rakennetta joudutaan myöhemmin

muokkaamaan, on se myös merkki huonosta tietokannan suunnittelusta. Rakenteen määrittelyyn liittyvät komennot kuuluvat SQL-kielen Data Definition Language –osaan. (Lahtonen 2002, 38.)

Yhdellä palvelimella sijaitsevien tietokantojen nimien on oltava yksilöllisiä, jotta palvelin erottaisi tietokannat toisistaan. Käytännössä tietokannan nimi kannattaa olla sellainen, jossa on vain kirjaimia aakkosten a-z väliltä, numeroita ja välilyönnin tilalla alaviiva. Sanoja, jotka kuuluvat SQL-kieleen ei sekaannusvaaran takia kannata käyttää. Tietokannan nimi kannattaa myös aina kirjoittaa täsmälleen samassa muodossa, kuin se on alun perin määritelty kirjoitettavaksi, sillä ominaisuus, jonka mukaan kirjainkoolla nimissä ei ole väliä, ei ole täysin toteutunut. (Heinisuo, Rauta 2007, 103)

Käytännössä tietokanta luodaan komennolla:

```
create database tietokannan_nimi character set merkistö collate aakkosjärjestys;
```

Tämä komento luo tietokannan, jonka nimi on tietokannan_nimi. Kohdat character set ja collate ovat vapaaehtoisia, mutta jos niitä ei määritellä, luotu tietokanta käyttää SQL:n asetustiedoissa määriteltyä oletusmerkistökoodausta, joka riippuu täysin palvelimesta. (Heinisuo, Rauta 2007, 104)

6.5.2 Merkistöt tietokannoissa

Tietokoneissa jokainen merkki vastaa tiettyä lukuarvoa. Merkkejä ovat kirjaimet, numerot ja erikoismerkit sekä välilyönnit ja rivinvaihdot. Myös kapitaalikirjaimet lasketaan omiksi merkeikseen pienten kirjainten rinnalla, ja jokaisella aksenttimerkillä varustetulla kirjaimella on oma lukuarvonsa. Koska maailmassa käytetään useita erilaisia kirjoitusmerkkejä, on ratkaisuksi kehitetty Unicode –standardi. Se on yleisin merkistökoodaus, ja nykyään web-käytössä suositellaan UTF-8 –koodausta, joka on yhteensopiva englanninkielisissä maissa käytetyn ASCII-merkistön kanssa, ja käytännössä se sisältää kaikki käytetyt kirjoitusmerkit maantieteellisestä sijainnista riippumatta. (Heinisuo, Rauta 2007, 104.)

Vaikka sovelluksen kannalta ei olisikaan välttämätöntä käyttää koko ajan samaa merkistöä, on kuitenkin hyvä pitää kiinni siitä, että käyttää johdon-

mukaisesti samaa merkistöä läpi koko sovelluksen. Tietokantaa luodessa kannattaa myös huomata, että esimerkiksi UTF-8-merkistökoodauksen nimi kirjoitetaan SQL:ssä nimellä utf8. (Heinisuo, Rauta 2007, 105.)

6.5.3 Aakkosjärjestys

Collate -komennolla määritellään, mitä aakkosjärjestystä tietokanta käyttää. Aakkosjärjestystä käytetään silloin, kun merkkien järjestystä ei voi päätellä yksiselitteisesti merkin numeroarvosta. Vaihtoehtoja aakkosjärjestykselle ovat muun muassa utf8_general_ci ja Latin1_swedish_ci. Collate-komennon määrittelemä aakkosjärjestys on aina kytköksissä johonkin määrättyyn merkistöön, ja jos tietokantaa luodessa määritellään vain käytettävä merkistö, on aakkosjärjestys merkistön oletusarvon mukainen. (Heinisuo, Rauta 2007, 105.)

Suomenkielisen sovelluksen kannalta tärkeimmät aakkosjärjestykset ovat utf8_swedish_ci ja Latin1_swedish_ci. Nämä aakkosjärjestykset tunnistavat suomalaiset ääkköset ja järjestävät kirjaimet meille tuttuun tapaan. Edellä mainittu generic-aakkosjärjestys puolestaan sijoittaisi aakkoset englanninkielelle tyypilliseen tapaan eikä tekisi eroa ä:n ja a:n välille. (Heinisuo, Rauta 2007, 106.)

6.5.4 Taulut ja niiden määrittely

Tietokantojen sisään luodaan tauluja, joihin tallennetaan sovelluksen käyttämät tiedot. Yksi tietokanta voi sisältää useita tauluja, ja näin myös lähes aina on. Tietokantataulujen sisältö järjestetään sarakkeisiin ja riveihin. Jokaiselle sarakkeelle annetaan oma otsikko, jonka alle tallennetaan riveille otsikon alle kuuluva tieto. (Heinisuo, Rauta 2007, 108.)

Päiväkirjasovelluksen tietokannassa tarvittavia tauluja ovat esimerkiksi käyttäjätietotaulu ja päiväkirjamerkintätaulu. Käyttäjätietotaulussa tarvittavia sarakkeita ovat muun muassa käyttäjätunnus, salasana ja käyttäjän oikea nimi. Päiväkirjamerkintöjä varten olevassa taulussa sarakeotsikoita ovat esimerkiksi id, jolla yksilöidään merkinnät; luotu-sarake, joka kertoo päivämäärän ja kellonajan, jolloin merkintä on kirjoitettu; muokattu-sarakkeen, joka kertoo, milloin merkintää on mahdollisesti muokattu; otsik-

ko-sarakkeen ja teksti-sarakkeen, johon itse merkintä tallennetaan. (Heinisuo, Rauta 2007, 108.)

Käytännössä taulu luodaan komennolla:

```
create table taulun_nimi (
sarake1    tietotyyppi lisämääritykset,
sarake2    tietotyppi lisämääritykset,
sarakeN    tietotyppi lisämääritykset
);
```

Ensimmäiseksi tietokannalle kerrotaan, että halutaan luoda taulu, jonka nimi on taulun nimi. Tämän jälkeen sulkeiden sisään määritellään taulun sarakkeet. Ensin sarakkeen nimi, siten tietotyyppi ja lopuksi muut mahdolliset lisäominaisuudet. Lisämääritykset ovat vapaaehtoisia, mutta usein tarpeellisia ominaisuuksia Sarakkeet määritellään komennon sisällä aina siinä järjestyksessä kuin niiden halutaan taulukossa olevan. (Heinisuo, Rauta 2007, 109.)

Samassa tietokannassa ei voi olla kahta samannimistä taulua, mutta eri tietokannoissa samaa taulun nimeä voi kuitenkin käyttää. Taulujen nimeämiseen pätevät samat säännöt kuin tietokannan nimeämiseen eli nimessä tulisi käyttää vain kirjaimia a-z, numeroita ja alaviivoja. Tietokannan tietotyypit karkeasti jaoteltuna ovat numeeriset tyypit, aikaan liittyvät tietotyypit ja tekstityypit sekä muutamat erikoisemmat tietotyypit. Lisämäärityksiä puolestaan käytetään rajaamaan tapoja, joilla saraketta voi käyttää. (Heinisuo, Rauta 2007, 111.)

6.6 Tietokannan sisällön käsittely

6.6.1 Tiedonhaku taulusta

Tietokannan sisällön käsittelyyn liittyvät komennot kuuluvat SQL-kielen Data Manipulation Language –osaan. Näillä komennoilla haetaan, muutetaan, lisätään ja poistetaan tietoja tietokannassa sekä hallitaan käyttäjien käyttöoikeuksia tietokannan suhteen. (Lahtonen 2002, 60.)

Yleisimmin käytetty hakukomento SQL:ssä on select. Esimerkiksi jos halutaan hakea jonkun taulun kaikkien sarakkeiden arvot, se tapahtuu komennolla:

```
select * from taulu;
```

Tämän lisäksi select -hakukomentoon liittyy määreitä, joiden avulla hakua voidaan tarkentaa koskemaan esimerkiksi vain tietynlaista tietoa sisältäviä rivejä. Selectiin liittyvät määreet ovat from, where, group by, having ja order by. Tietoja voidaan myös hakea useammasta kuin yhdestä taulusta kerralla ilmoittamalla, minkä taulun kentästä haussa on kyse. (Lahtonen 2002, 60-61.)

6.6.2 Tietojen lisääminen tauluun

Tietokantatauluun lisätään uusia rivejä insert –komennon avulla. Komennossa määritellään taulu, johon tieto lisätään, ja sarakelista, jonka avulla kerrotaan mihin sarakkeisiin tietoa lisätään. Sarakelista antaa mahdollisuuden lisätä tietoa muussakin järjestyksessä kuin missä järjestyksessä ne taulussa ovat, mutta tämä ei yleensä ole selkeyden kannalta järkevää. Lopuksi kerrotaan syötettävät arvot siinä järjestyksessä, jossa sarakkeet on aikaisemmin esitelty. (Heinisuo, Rauta 2007, 113.)

Mikäli jokainen taulun sarake saa arvon, sarakelistaa ei ole pakollista käyttää. Tätä tapaa voidaan käyttää kun tauluun syötetään manuaalisesti tietoja, mutta sen käyttöä ei suositella tilanteissa joissa tietoja lisätään jonkin sovelluksen kautta. Kun käytetyt sarakkeet on määritelty lausekkeessa, syötettävät arvot päätyvät varmasti omiin sarakkeisiinsa. Kaikkiin taulun sarakkeisiin ei ole myöskään pakko lisätä arvoja, jolloin niissä tullaan käyttämään niille määriteltyä oletusarvoa. (Heinisuo, Rauta 2007, 113-115.)

Yleisesti ottaen insert –lauseke muodostetaan seuraavasti:

```
insert into taulun_nimi (sarake1, …, sarakeN) values (arvo1, …, arvoN);
```

6.6.3 Tietojen päivittäminen

Tietokannan taulujen tietojen päivittäminen tapahtuu update-lausekkeen avulla. Lausekkeessa määritellään päivitettävät sarakkeet sekä uudet arvot määritellyille sarakkeille. Myös updat -lausekkeen yhteydessä käytetään where–määritettä, kun halutaan päivityksen koskevan tietynlaista tietoa sisältäviä taulun rivejä. Jos where-määrite unohtuu, saatetaan kaikki taulun rivi päivittää samanlaiseksi. Tiedonpäivitys kohdistuu aina siihen tauluun, joka määritellään lausekkeessa update–sanan jälkeen . (Heinisuo, Rauta 2007, 118-119.)

Käytännössä update–lauseke muodostetaan näin:

```
update taulun_nimi set sarake1 = arvo1, … , sarakeN = arvoN where
ehtolause;
```

Where –määritteen käyttö ei ole pakollista, mutta päivityksiä, joissa sitä ei käytetä, tapahtuu todella harvoin. (Heinisuo, Rauta 2007, 119.)

6.6.4 Rivien poistaminen

Rivien poistaminen tapahtuu delete–lausekkeen avulla, ja myös tämän lausekkeen yhteydessä käytetään where–määritettä rajaamaan taulusta vain halutut rivit. Where–ehdon jättäminen pois tuhoaa taulusta kaikki rivit ja huonosti muodostettu lauseke voi tuhota rivejä vääristä paikoista. Delete –lauseke muodostetaan näin:

```
delete from taulun_nimi where ehto;
```

Delete –lauseke poistaa aina kokonaisen rivin, ja tämän vuoksi siihen ei määritellä sarakkeita. Jos halutaan tyhjentää vain tietyn rivin tietyn sarake, kannattaa tähän käyttää update –lauseketta ja sijoittaa sarakkeen arvoksi null. (Heinisuo, Rauta 2007, 119-120.)

6.6.5 Käyttöoikeudet

Pääkäyttäjätunnuksilla voi tietokannassa tehdä minkälaisia muutoksia tahansa ja käyttää mitä tahansa komentoja missä tahansa. Tämän vuosi on tärkeää, että nämä tunnukset pidetään turvassa. Käyttäjä oikeuksia on kuitenkin mahdollista määritellä muille käyttäjille hyvinkin tarkasti palveli-

men, tietokantojen, taulujen ja jopa yksittäisten sarakkeiden tasolla. Myös taulujen tietojen päivittämiseen, poistamiseen ja lukemiseen voidaan määritellä erikseen. (Heinisuo, Rauta 2007, 120.)

Liian suuret käyttäjäoikeudet saattavat itsessään olla tietoturvariski, ja tämän vuoksi käyttäjätunnusten hallinta on hyvin olennainen osa tietokantapalvelimen ylläpitoa. Myös käyttäjätunnusten hallinalle on tämän vuoksi olemassa omat käyttöoikeusmäärityksensä, ja esimerkiksi on mahdollista määrittää, miltä tietokoneelta tiettyä käyttäjätunnusta voidaan käyttää. (Heinisuo, Rauta 2007, 120.)

7 CASE

7.1 Esittely kohdeyrityksestä

Lahden Seurakuntayhtymän Oppilaitostyö tarjoaa Lahden opiskelijoille palveluja, tapahtumia ja keskusteluapua. Oppilaitostyön kautta on mahdollista esimerkiksi suorittaa käymättä jäänyt rippikoulu, käydä tapaamassa oppilaitospastoria opiskelijaterveydenhuollon tiloissa tai osallistua oppilaitostyön järjestämiin iltatapahtumiin. Oppilaitostyöllä on myös vaihto-oppilastoimintaa, joka tarjoaa oppilaitostyön palvelut myös englantia puhuville opiskelijoille.

7.2 Tarkoitus ja tavoitteet

Kesällä 2007 Oppilaitostyön Internetsivujen ilme ja sisältö uudistettiin ja projektin yhteydessä toivottiin myös, että Jeesinet.net -sivujen yhteyteen saataisiin blogi. Aikaisemmilla sivuilla oli ollut käytössä vieraskirja, mutta sen käytöstä päätettiin luopua, sillä viestejä vieraskirjaan tuli vain vähän.

KUVA 8. Kuvakaappaus Oppilaitostyö Internetsivuista.

Blogin käyttöönottoa toivottiin, koska blogit ovat tällä hetkellä suuressa suosiossa, eikä niiden sisällön päivittyminen ole riippuvainen ulkopuolisista kävijöistä, kuten asia oli vieraskirjan kohdalla. Blogissa toivottiin myös olevan kommentointimahdollisuus, jotta blogin lukijoilla olisi myös mahdollisuus jättää viestinsä kirjoitukseen.

Tarkoituksena on, että oppilaitostyön työntekijät sekä mahdollisesti myös vapaaehtoiset toimintaan osallistuvat opiskelijat voivat kirjoittaa oppilaitostyön kuulumisista ja tapahtumista blogiin. Tällä tavoin sivuston toivotaan tarjoavan uudenlaista hyötyä ja saavan näin myös uusia lukijoita.

Koska nykyiset sivut on toteutettu Flash -pohjaisina, on luontevaa hyödyntää Flashin dynaamisuutta ja jo valmiina olemassa olevaa pohjaa blogin toteutuksessa. Tämän kautta lähdettiin siis myös tutkimaan mahdollisuuksia toteuttaa blogi hieman tavallisuudesta poikkeavalla tavalla.

Työstä rajattiin ulos esimerkiksi kuvien käyttäminen blogissa vaikka ulkoisten kuvien lataaminen onnistuukin hyvin Flashilla. Tämä siksi, että jo pelkästään tekstipohjaisen blogin tutkimisessa oli riittävästi aihetta tälle työlle.

7.3 Tekniikat ja työkalut

Blogin käyttäjälle näkyvä verkkosovellusosuus päätettiin toteuttaa Flash 8 –versiolla, jolla myös Internetsivut on toteutettu. Muita tarpeellisia työkaluja blogin kannalta ovat PHP ja SQL-tietokanta. Flash puolestaan ottaa vastaan tietokannoista tulevan tiedon XML-muotoisena tietona.

Sovelluksen kirjautumis-, merkintä ja poistamissivujen osalta toteutusvaihtoehtoja oli kaksi. Sivut voisi toteuttaa joko .swf-tiedostona ja PHP-tiedostoina tai pelkästään PHP-tiedostoina. Vaihtoehdoista valittiin pelkkien PHP-muotoisten tiedostojen käyttäminen, sillä se vähentää työn määrää eivätkä nämä sivut näy lukijalle, joten ulkonäön ei välttämättä tarvitse vastata muun sivuston ulkoasua.

Koska SWF -tiedosto tarvitsee avukseen PHP –tiedostoja saadakseen ja välittääkseen tietoa tietokannoille, joudutaan tiedostoja tekemään lähes

kaksinkertainen määrä verrattuna PHP –sovelluksen tekoon. Vaikka tämä ei sinällään ole ongelma, se kuitenkin lisää tehtävän työn määrää.

7.4 Sovelluksesta

7.4.1 Tiedosto- ja tietokantarakenne

Blogisovelluksessa tiedostot voidaan jakaa kahteen ryhmään siten, että toisilla esitetään tietokantaan tallennettua tietoa ja toisien kautta luodaan uusia merkintöjä sivustolle. Flash –pohjaisessa blogissa SWF-tiedosto sisältää kaiken mitä tiedostojen esittämiseen tarvitaan, joten useaa eri tiedostoa erilaisille bloginäkymille ei tarvita.

Blogisovelluksessa tarvittavat tiedostot:

blogi.swf	flashtiedosto, joka esittää blogimerkinnät
blogi.html	blogin pääsivu, joka näyttää blogi.swf tiedoston
artikkeli.php	lataa tietyn merkinnän tietokannasta id: perusteella
artikkelit.php	lataa uusimmat artikkelit tietokannasta
login.php	kirjautumissivu, jonka kautta blogeja pääsee kirjoittamaan
kirjoita.php	sivulla muokataan ja luodaan merkintöjä
poista.php	sivu merkintöjen poistamiseen
funktioita.php	tämä tiedosto sisältää PHP -tiedostojen käyttämiä funktioita

Tiedostoista Flash-soveluksella esitetään blogimerkinnät lukijoille, ja PHP-tiedostoilla hoidetaan uusien merkintöjen tekeminen ja poistaminen. XML-dokumentit eivät näy tiedostorakenteessa, sillä ne muodostuvat tarpeen mukaan PHP-tiedostossa, eivätkä ne jää mihinkään muistiin. Ne toimivat ainoastaan tiedon välittäjinä kahden eri tiedoston välillä.

Tietokannassa on taulut käyttäjiä ja merkintöjä varten. Merkintätaulun sarakkeet ovat id, päivämäärä, otsikko, teksti ja muokkauspäivämäärä. Käyttäjätaulussa olevat sarakkeet taas ovat id, nimimerkki ja salasana. Koska tietokannan tietoja ei tarvitse muokata usein eikä tauluja ole kuin kaksi, voidaan hyvin käyttää normalisoimattomia tauluja.

7.4.2 Toteutuksesta

Tietokantapohjaisissa sovelluksissa tietoa säilytetään tietokannoissa ja sitä haetaan tarpeen mukaan käyttäjälle näytettäväksi. Näin ollen itse Flash-sovellus ei sisällä mitään tekstiä, vaan kaikki sen kautta näytettävä tieto saadaan ulkoisesta lähteestä, tällä kertaa juuri tietokannasta. Flashiin voidaan tuoda monella tapaa dynaamista tietoa, mutta blogisovelluksessa kätevimmäksi osoittautui tietokannan käyttäminen.

Flash -sovelluksessa on määritetty tavalliselle lukijalle näkyvä ja esityksessä käytettävä ulkoasu ja fontit, ja siinä esitetään kirjoitetut merkinnät luettavaksi. PHP –tiedostoja puolestaan käytetään sisällön hallintaan eli hoidetaan sisäänkirjautuminen ja tietojen syöttäminen ja poistaminen.

Kun selain avaa Flash-sovelluksen, sovellus pyytää PHP:tä lähettämään sille halutut tiedot tietokannasta. PHP käsittelee pyynnön ja lähettää tietokantaan haun, jonka tulos palautetaan XML-dokumenttina Flash-sovellukselle. Tekstin muotoiluun ei voitu tässä työssä käyttää htmlText –muotoiluominaisuutta, sillä tällaisessa julkaisujärjestelmässä ulkoasu ja sisältö on hyvä pitää erillään. Näin ollen tietokannassa olevaa tekstiä voitaisiin myös käyttää muissa esityksissä tai vaikkapa HTML-sivuilla.

Koska työ muutenkin painottui lähes kokonaan pelkkään blogin tekniseen puoleen, ei ulkoasuasioihin tässä työssä tarvinnut juuri kiinnittää huomiota. Näin ollen myöskään Flashin graafisiin ominaisuuksiin ei tässä työssä ollut tarvetta kiinnittää huomiota.

7.4.3 Julkaiseminen

Flashilla toteutettu sovellustiedosto julkaistiin lopuksi .html ja.swf –tiedostoina. Julkaisuversioksi valittiin Flash 8 –version mukana tullut uusin julkaisuversio, vaikka myös vanhemman julkaisuversion käyttöä olisi voinut tässä tapauksessa harkita. Ongelmana kuitenkin oli, etteivät vanhemmat julkaisuversiot tue kaikkia sivulla käytettyjä ominaisuuksia, joten julkaisuversioksi piti valita uusin Flash 8 –version julkaisuversio.

7.5 Casen arviointi

Casen tekeminen oli haastavaa, sillä sen tekemiseen tarvittiin tietoa sekä ActionScriptistä, PHP:stä ja SQL:stä sekä jonkin verran tietoa myös XML:stä. Eniten tietoa ja kokemusta työn alkuvaiheessa oli lähinnä ActionScriptistä, muut kielet olivat enimmäkseen perusteiden tasolla tutuja. Alun alkaen aihe tuntui haastavalta, mutta loppua kohden tiedon lisääntymisen myötä työ kävi helpommaksi.

Työssä päästiin tutkimaan käytännössä, miten tietokantapohjaiset sovellukset yleensä toimivat ja miten näitä tekniikoita voidaan hyödyntää yhdessä Flashin kanssa. Vaihtoehtoja oli yllättävän monia, mutta lopulta päädyin kuitenkin käyttämään XML-pohjaista tiedonsiirtoa Flashista PHP:lle, josta tieto saatiin perinteidillä menetelmillä tallennettua tietokantoihin.

Lahden Seurakuntayhtymän Oppilaitostyölle blogi on hyvä työkalu, koska se tekee työtä opiskelijoiden keskuudessa. Oppilaitostyö saa uudenlaista näkyvyyttä opiskelijoiden keskuudessa, ja sivut saavat näin todennäköisesti myös enemmän lukijoita.

Koska blogisivustoja on toteutettu perinteisesti enemmänkin PHP:n, HTML:n ja CSS:n sekä tietokantojen avulla ja kehitystyötä on viety jo pitkälle näillä tekniikoilla, tuntui ajoittain hieman turhalta tuottaa blogia toisenlaiseen toteutusympäristöön. Pohjalla toimii kuitenkin samanlainen tekniikka kuin perinteisissä blogimalleissa, ja työ osoitti, että Flash -ympäristössäkin on mahdollista saada aikaan toimiva blogisovellus.

8 YHTEENVETO

Tietokantapohjaista blogia tai yleensä sovellusta suunnitellessa on huomioon kohteena monia erilaisia verkkotekniikoita. Kun perinteisiin tietokantoihin ja PHP:hen lisätään vielä Flash ja ActionScript, ohjelmointityön määrä kasvaa ja vaaditaan jo useamman erilaisen kielen hallintaa.

Vaihtoehtoja toteuttaa tietokantapohjaisia Flash-sovelluksia on monenlaisia, ja yhdenkin sovelluksen kohdalla on usein monta erilaista vaihtoehtoa toteuttaa käyttöliittymä ja yleinen toiminta. Koska Flash tarjoaa monia multimediaominaisuuksia, joita HTML:n tai PHP:n avulla on vaikeaa tai mahdotonta toteuttaa, on se varteen otettava vaihtoehto dynaamista verkkosovellusta suunniteltaessa.

Vaikka käyttäjille näkyvä sivusto toteutettaisiin Flash-sovelluksena, voidaan sen tiedonhallintajärjestelmä toteuttaa perinteisempään tapaan PHP:llä. Näin uusi tietokantaan tallennettava data kulkee harvempien välikäsien kautta tietokantaan, ja Flash-sovellukseen tarvitsee ainoastaan ladata tietoa tietokannasta. Jos sovelluksen toteuttamisessa kuitenkin päätetään käyttää työssä aikaisemmin esiteltyä AMFPHP–tietokantasovellusta, voidaan yhteys tietokantaan ottaa suoraan sovelluksesta. Näin ollen kaikki tähän tarvittava koodi voidaan sisällyttää .swf–tiedostoon, eikä PHP:tä ja XML:ää tarvita välikädeksi.

Flashin vahvuudeksi voidaan verkkosovellusten maailmassa lukea sen, että sillä voidaan tuottaa visuaalisesti mielenkiintoista ja näyttävää materiaalia. Myöskään sovellusten käytettävyys ja päivitettävyys eivät häviä muille tekniikoille. Flash-sovellusten haittapuolena voidaan kuitenkin pitää helposti suureksi nousevia tiedostokokoja, ja vaikka tietoa ja kuvia ladattaisiinkin ulkopuolelta sovellukseen, Flash-verkkosivujen lataaminen kestää lähes aina pidempään kuin perinteisesti toteutetun sivun lataaminen.

Tämä työ mielestäni tuo esille sen, että tietokannat ovat käyttökelpoinen tapa tuoda ja säilöä sovelluksen ulkopuolista tietoa myös Flash-sovellusten kohdalla. Koska tietokantapohjaisen Flash –sovelluksen tuot-

taminen vaatii monenlaisten verkkotekniikoiden hallintaa, tällaisia projekteja varten olisikin hyvä muodostaa ryhmä eri osa-alueet hallitsevista osaajista.

Blogi on verkkosovellusten joukossa tämän päivän hittituote. Vaikka blogin toteuttaminen onkin mahdollista Flashin avulla, saattaa sen toteuttaminen olla lopulta järkevämpää PHP:n, CSS:n ja tietokantojen avulla. Tästä huolimatta Flashin käyttö tietokantapohjaisessa verkkosovelluksessa on hyvä vaihtoehto, sillä se ohjelmana tarjoaa monia multimediaominaisuuksia, joita PHP tai CSS eivät itsessään tarjoa.

Flashin ja tietokantojen väliset datanvälitystekniikat kehittyvät jatkuvasti, ja varmasti jo lähitulevaisuudessa nähdään entistä enemmän tietokantoja tiedonlähteenä käyttäviä Flash-sovelluksia.

LÄHTEET

Asmala, H. , 2005. [verkkodokumentti] Relaatiotietokannan suunnittelu. [viitattu 28.3.2008]. Saatavissa: http://www.tp.spt.fi/~salabra/ha/Relaatiotietokannat/Kasiteanalyys.html

Asmala, H. , 2006a. [verkkodokumentti] Relaatiotietokannan normalisointi. [viitattu 28.3.2008]. Saatavissa: http://www.tp.spt.fi/~salabra/ha/Relaatiotietokannat/normalisointi.html

Asmala, H. , 2006b. [verkkodokumentti] Peruskäsitteitä. [viitattu 30.3.2008]. Saatavissa: http://www.tp.spt.fi/~salabra/ha/Relaatiotietokannat/peruskasitteita.html

Blogilista.fi, 2005. [verkkodokumentti] [viitattu 18.3.2008]. Saatavissa: http://www.blogilista.fi/wiki/RSS-syöte

Goto, K. & Cotler, E., 2003. Verkkopalveluprojekti. 1. painos. Helsinki: Edita Prima Oy.

Heinisuo, R. & Rauta, I., 2007. PHP ja MySQL Tietokantapohjaiset verkkopalvelut. 4. Uudistettu painos. Gummerus Kirjapaino Oy.

Kipli, T. 2006. Blogit ja bloggaaminen. 1. painos. Jyväsylä: Gummerus Kirjapaino Oy.

Lahtonen, T. 2002. SQL. 1.painos. Jyväskylä: Docendo Finland Oy.

Manninen, P. & Marttila, J., 2006. Flash 8 & ActionScript. 1. painos. Jyväskylä: Docendo Finland Oy.

Sarja, J. , 2006. [verkkodokumentti] Relaatiotietokanta. [viitattu 28.3.2008]. Saatavissa: http://sarja.internetix.fi/fi/sisalto/materiaalit/access2003/luku03?C:D=419702&selres=419702

Schroeder, S., 2007. [verkkodukumentti] The Ultimate RSS Toolbox - 120+ RSS Resources. [viitattu 29.3.2008]. Saatavissa: http://mashable.com/2007/06/11/rss-toolbox/

Tekniikan sanastokeskus (TSK), Termipankki, 2005. [verkkodokumentti] [viitattu 21.2.2008].
Saatavissa: http://www.tsk.fi/tepa/netmot.exe?UI=figr&height=141

What Is RSS? RSS Explained [viitattu 18.3.2008]. Saatavissa: http://www.whatisrss.com/

KUVALÄHTEET

Asmala, H., 2006. [verkkodokumentti] Relaatiotietokannan normalisointi. [viitattu 28.3.2008]. Saatavissa: http://www.tp.spt.fi/~salabra/ha/Relaatiotietokannat/normalisointi.html

Keränen, V., Lamberg, N. & Penttinen, J. 2001. Digitaalinen viestintä. 1. painos. Jyväskylä: Docendo Finlad Oy.

Lahtonen, T. 2002. SQL. 1.painos. Jyväskylä: Docendo Finland Oy.