Kauko Kähkönen

LABORATORIOIDEN KOMPONENTTIEN JA LAITTEIDEN HALLINTAJÄRJESTELMÄ

Insinöörityö
Kajaanin ammattikorkeakoulu
Tekniikan ja liikenteen ala
Tietotekniikan koulutusohjelma
Syksy 2005

Kajaanin ammattikorkeakoulu

INSINÖÖRITYÖ

TIIVISTELMÄ

Osasto	Koulutusohjelma
Tekniikka	Tietotekniikka
Tekijä(t) Kauko Kähkönen	
Työn nimi LABORATORIOIDEN KOMPONENTTIEN JA LAITTEIDEN HALLINTAJÄRJESTELMÄ	
Vaihtoehtoiset ammattiopinnot	Ohjaaja(t) Arto Partanen, Markku Karppinen
Aika Syksy 2005	Sivumäärä 48 + 3

Työn tavoitteena oli suunnitella ja toteuttaa sovellusjärjestelmä, jolla voidaan selata ja valvoa ammattikorkeakoulun testauslaboratorioiden komponentteja ja testauslaitteita sekä testauslaitteiden vuokrausta. Järjestelmä asennetaan koulun sisäiseen intranet-verkkoon siten, että ainoastaan koulun sisäiseen verkkoon kirjautuneet henkilöt pääsevät käyttöliittymän kautta selaamaan ja muokkaamaan tietokannan tietoja.

Käyttöliittymä on toteutettu internet-selain pohjaisesti ohjelmointikielenä PHP. WWW-palvelimena toimii Apachen www-palvelinohjelma ja tietokanta on toteutettu MySQL- relaatio-tietokannan hallintajärjestelmällä.

Insinöörityön lopputuloksena syntyi aiheen suunnittelun mukainen selaus- ja hallintajärjestelmä testauslaboratorioiden komponenteille ja testauslaitteille. Itsenäinen työskentely antoi lisää itsevarmuutta kursseilta saatuihin tietokantasuunnittelun ja ohjelmoinnin perusteisiin. Insinöörityön tekeminen oli hyödyllinen ja mielenkiintoinen kokemus ja työhön liittyvien ohjelmien valinnan syynä oli opetella uusien ohjelmien käyttö.

Luottamuksellinen	
Kyllä	
Ei	x

Hakusanat
MySQL, PHP, kompohaku, laitehaku
Säilytyspaikka
Kajaanin ammattikorkeakoulu

Kajaanin ammattikorkeakoulu

ABSTRACT

THESIS

Faculty	Degree programme
Faculty of Engineering	Information Technology
Author(s) Kauko Kähkönen	
Title A User Interface for a Test Laboratory's Database	
Optional professional studies	Instructor(s) / Supervisor(s) Arto Partanen, Markku Karppinen
Date Autumn 2005	Total number of pages 48 + 3

The purpose of this Bachelor's thesis was to implement a user interface and database where by the components and rentable testing equipment and supplies can be browsed.

The user interface should be installed to the intranet of Kajaani Polytechnic so that only the person who is signed into the network can search for components from data base through the user interface.

The person responsible for maintains the database and controls the renting of equipment. Each of them has an individual user identifier and password so they can modify the database, add or remove component and rent testing equipment. A student can only browse the database and search for components on the basis of the keyword.

The user interface was implemented with the PHP programming language using the Apache server. The database was implemented with the MySQL server's relation database software.

Confidential
Yes
No x
Keywords MySQL, PHP, kompohaku, laitehaku
Deposited at Kajaani Polytechnic Library

ALKUSANAT

Tämän insinöörityön tilaajana on Kajaanin ammattikorkeakoulu. Työn tarkoituksena on laatia hallintajärjestelmä, jolla voidaan selata, valvoa ja ylläpitää laboratorioiden komponentteja ja testauslaitteita.

Kiitokset ansaitsee tietotekniikan ohjelmoinnin opettaja Arto Partanen, joka auttoi mielenkiintoisen insinöörityön aiheen valinnassa ja toimi työn valvojana. Lisäksi kiitokset saamistani neuvoista ja työhön liittyvistä, tekemistä helpottavista vinkeistä edelleen opettaja Arto Partaselle, kehitysinsinööreille Markku Karppiselle ja Ari Pulkkiselle sekä kielellisestä ohjauksesta kielten opettajille Eero Soiniselle ja Kaisu Korhoselle.

Kajaanissa 1. syyskuuta 2005

Kauko Kähkönen

SISÄLLYSLUETTELO

TIIVISTELMÄ
ABSTRACT
ALKUSANAT
KÄYTETYT TERMIT JA LYHENTEET

1 JOHDANTO 9

2 TIETOKANTASUUNNITTELU 10

2.1 Lähtökohtia 10
2.2 Hyvän tietokantasuunnittelun merkitys 11
2.3 Tavoitteita tietokannan rakenteelle 12
2.3.1 Räätälöidyt tietokannat 13
2.3.2 Valmisohjelmiston tietokannat 13
2.3.3 Tietovarastokannat 13
2.4 Eheys rakenteen yhteisenä tavoitteena 14

3 KÄYTTÖLIITTYMÄN SUUNNITTELU 15

3.1 Internet toimintaympäristönä 15
3.2 WWW-sovellukset 16
3.3 Käyttöliittymän suunnittelu 17
3.4 Käyttöliittymän visuaalinen suunnittelu 18
3.5 Käyttöliittymän suunnittelun tavoitteet 19

4 TIETOKANTAPOHJAINEN SOVELLUS INTERNETISSÄ 20

4.1 Tietokannat 20
4.2 Relaatiotietokannat 21
4.2.1 Relaatiomalli 22
4.3 Normalisoinnin tarkoitus 24
4.4 SQL-kieli 24
4.5 Tietokantahallintajärjestelmät 25
4.5.1 Indeksit ja optimointiohjelma 26
4.6 Kolmikerrosmalli 27

4.7 Tietokantapalvelimena MySQL 29
4.8 WWW-sovellusten toteuttamistekniikat 30
4.8.1 Käyttöliittymä 30
4.8.2 Sovelluslogiikka 31
4.8.3 Käyttöliittymän ja sovelluslogiikan välinen viestintä 32
4.8.4 PHP-kieli 33
4.9 Zend Engine 34
4.10 Apachen www-palvelinohjelma 36
5 PROJEKTIN TOTEUTUKSEN KUVAUS 37
5.1 Tekninen ratkaisu 37
5.2 Tietokannan toteutus 38
5.3 Käyttöliittymän toteutus 40
6 TIETOKANNAN TESTAUS JA WWW-SIVUJEN TOIMINNOT 43
6.1 Tietokantaan lisääminen 43
6.2 Tietokannan listaus 43
6.3 Tietokannan muokkaaminen, päivitys ja tietokannasta poistaminen 43
6.4 Tietokannan tietojen seuranta 44
7 YHTEENVETO 45

LÄHDELUETTELO
LIITTEET

KÄYTETYT TERMIT JA LYHENTEET

CGI, *Common Gateway Interface*
Standardoitu tiedonsiirtorajapinta www-palvelimen ja sen suorittaman ohjelman (CGI-ohjelman) välillä.

HTML, *Hypertext Markup Language*
WWW-järjestelmässä hypertekstien kuvauskieli, jolla luodaan www-sivujen muoto ja sisältö.

HTTP, *HyperText Transfer Protocol*
WWW- palvelimen ja selaimen välisessä kommunikoinnissa käytetty protokolla.

Kolmikerrosmalli (*Three-Tier model*)
Kolmeen kerrokseen perustuva sovellusmalli kaksikerroksisen asiakaspalvelinmallin sijaan. WWW-sovelluksissa käytettävä toteutustapa, yleensä asiakaskerros – esitys, välikerros – käsittely ja pohjakerros – tietovarasto.

MySQL
Suosituin avoimeen lähdekoodiin perustuva relaatiotietokannan hallintajärjestelmä. Monipuolinen, joustava ja suorituskykyinen varsinkin www-sovelluksissa.

Palvelinsivu
WWW-dokumentti, johon on HTML:n sekaan erikoiselementtien sisälle upotettu palvelimessa suoritettavia *skriptejä*. Tunnetaan myös nimellä ”upotetut tekniikat”.

PHP, *PHP: Hypertext Preprocessor*
Useilla alustoilla toimiva avoin upotus- tai palvelinsivutekniikka. Käytetyn skriptikielen syntaksi on lainattu pääosin C-kielestä.

Skripti

Yleisesti käytetty nimitys tulkattavasta ohjelmasta.

SQL, *Structured Query Language*

Standardoitu (ANSI/ISO-standardi 1992) kyselykieli relaatiotietokantojen tiedonhakuun ja rakenteen muokkaamiseen.

UML, *Unified Modeling Language*

UML on oliosuunnittelun kuvauskieli, jota voidaan hyödyntää myös generatiivisessa ohjelmoinnissa.

URL, *Uniform Resource Locator*

Internet-verkoissa käytettävä osoite, mistä jokin tietty asia on ladattavissa.

1 JOHDANTO

Insinöörityön tavoitteena oli toteuttaa Kajaanin ammattikorkeakoulun testauslaboratorioiden ylläpitotietokanta ja sille käyttöliittymäohjelmisto, joka mahdollistaa komponenttien ja laitteiden tietojen selaamisen, tallentamisen, poistamisen ja muokkaamisen.

Koulun oppilaille ja opettajille sekä testauslaboratorioiden vastuuhenkilöille suunnattu Internet-perustainen tietojärjestelmä tarjoaa mahdollisuuden selata tietokantaa ja paikallistaa komponenttien ja laitteiden sijaintipaikan sekä mahdollisuuden lisätä, poistaa ja päivittää komponenttien ja laitteiden tietoja sekä tehdä niitä koskevia hakuja tietokannasta. Järjestelmä mahdollistaa tietokannan tietojen selaamisen ja ylläpidon ilman, että itse tietokantaan tarvitsee koskea.

Vastaavaa laboratorioiden valvontatietokantajärjestelmää ei ammattikorkeakoululla ole ollut aikaisemmin, vaan valvonta ja ylläpito on suoritettu Microsoft Officen Excel-taulukkolaskentaohjelmalla. Kyseiseen taulukkoon ei oppilailla ja opettajilla ole ollut mahdollisuutta päästä.

Järjestelmä toteutetaan tietokantana, johon tallennetaan laboratorioiden, vastuuhenkilöiden, komponenttien ja laitteiden tiedot sekä lisäksi laitteiden huolto-, vuokraus- ja vuokraajatiedot. Tietokannan sisältöä hallitaan www-selaimelle rakennetun käyttöliittymän avulla. Tietojärjestelmä asennetaan koulun sisäiseen intranet-verkkoon, johon pääsee kirjautumaan vain koulun määräämällä käyttäjätunnuksella ja salasanalla.

Tietokanta ja sen käyttöliittymä luodaan ilmaisilla *Open Source* -tekniikoilla; alustana käytetään Apachen www-palvelinohjelmistoa, toiminnallisuus luodaan PHP-ohjelmointikielen avulla HTML-tekniikkaa hyväksi käyttäen ja tietokantana käytetään MySQL-relaatiotietokantojen hallintajärjestelmäohjelmistoa.

2 TIETOKANTASUUNNITTELU

2.1 Lähtökohtia

Tietokantojen suunnittelu on tärkeää monestakin syystä. Ensinnäkin yhä suurempi osa tiedoista tallennetaan tietokantoihin. Toiseksi luonnollisestikin on tärkeää, että tarvittavat tiedot löytyvät myös tietokannoista ja tämä tieto voidaan esittää tiedon hakijalle nopeasti ja varmasti. Kolmanneksi tietokannan rakennetta pitää pystyä muuttamaan esim. lakisäädösten seurauksena. Siksi hyvä suunnittelu säästää sekä aikaa että rahaa. On kuitenkin hyvä muistaa, että toimivaksi osoittautunut ratkaisu ei välttämättä ole aina paras vaihtoehto.

Lähes kaikki uudet tietojärjestelmät tehdään relaatiotietokantapohjaisiksi. Markkinoilla on useita hyviä tietokantatuotteita, joista ehkä tunnetuimpia ovat Oracle, IBM DB2, Microsoft SQL Server, Informix, Solid Server, Adaptive SQL Server ja MySQL. Merkittävä tekniikan kehityssuunta on ollut levylaitteiden hintojen halpeneminen niin, että tietokannan suunnittelijan ei tarvitse välittää tiedon tilantarpeista. On tärkeää osata käsitellä myös menetelmiä, joilla parannetaan jo olemassa olevien tietokantojen suorituskykyä. Yksi näistä menetelmistä on tietokannan indeksointi.

Laajennettaessa tai tehostettaessa jo olemassa olevaa ratkaisua tai luotaessa uutta tietokantaa olisi hyvä varmistaa ennen varsinaista käyttöönottoa muutosten tai uusien piirteiden toimivuus. IT-ammattilaisen on työssään usein tutkittava jo olemassa olevien tietokantojen rakenteita. Tähän liittyy sekä yksittäisen henkilön tietokantakuvausten lukutaito että käytettävän suunnittelutyökalun kyky lukea tietokannan rakenne (nk. *reverse engineering* -toiminto). Ei riitä, että työkalu pystyy näyttämään taulut ja niiden väliset yhteydet. vaan tärkeämpää on, että tietokantasuunnittelija osaa tulkita kuvauksia ja tehdä perusteltuja päätöksiä esim. tietokannan laajentamiseksi tai virittämiseksi.

UML-menetelmällä yleistynyt oliosuunnittelu on lisääntynyt viime vuosina huomattavasti. Oliosuunnittelussa pohditaan sekä olioiden käyttäytymistä että niiden rakennetta, jota kuvataan yleensä luokkakaavioina. Tämän vuoksi onkin

hyvä ymmärtää, kuinka luokkakaaviosta päästään tehokkaaseen relaatiotietokanta ratkaisuun. [1.]

2.2 Hyvän tietokantasuunnittelun merkitys

Tietokanta on nykyaikaisten sovellusten perusta. Tietokannan suunnittelussa ja rakentamisessa on tärkeää tehdä se hyvin siitäkin huolimatta, että käyttäjille näkyvät vain lähinnä käyttöliittymä ja raportit. Huonosti suunnitellulla tietokannalla ei sovelluksesta tule koskaan onnistunut, koska sovellusohjelmilla joudutaan usein paikkaamaan hankaliakin tietorakenteita. Hyvin suunniteltu tietokanta helpottaa ohjelmointia ja muodostaa sovellukselle vankan perusrakenteen.

Nykyaikaisilla tietokantatuotteilla ja sovelluskehittimillä on helppo toteuttaa tietokanta ja sen taulut. Jos tauluja perustetaan nopeasti ilman huolellista suunnittelua, ottaa tietokannan perustaja tietyn riskin. Yksinkertainen, pieni tietokanta voi onnistua, kuten esimerkiksi koirankopin rakentaminen, mutta isomman asuin- tai teollisuusrakennuksen rakentaminen vaatii jo tarkempaa arkkitehturisuunnittelua niin sähkö-, lämmitys- kuin vesiputkijärjestelmänkin osalta. ”Mitä monimutkaisemmasta ja laajemmasta kokonaisuudesta on kyse, sitä tärkeämpää hyvä suunnittelu on.” [1.]

Tietokannan mallinnus (*database modelling*) on pääasiassa tietokannan kuvaamista jollakin kuvaustekniikalla, kun sitä vastoin tietokannan suunnittelu (*database design*) on jo laajempi käsite. Tietokannan suunnittelu käsittää paljon enemmän huomioonotettavia vaiheita vaatimusmäärittelystä tietokannan mallinnukseen ja fyysiseen suunnitteluun.

Tietojärjestelmän keskeiset osat kuvataan tietyllä sovitulla menetelmällä ja tekniikalla. Useimmat standarditekniikat perustuvat graafiseen esitystapaan, jossa kuvataan mm. käsitteet ja niiden väliset yhteydet. Nykyisen, jo olemassa olevan tietokannan tunteminen on tärkeää tietokannan laajentajalle onnistua tehtävässään. On siis hyvä oppia sekä lukemaan ja tulkitsemaan jo tehtyjä tietokantakuvauksia että itse tuottamaan niitä. Sama pätee talon laajentamista harkitsevan - on hyvä osata lukea talonsa piirustuksia.

Tietokannan tehokkaan toiminnan kannalta tietokannalla täytyy olla optimoituja suhteita - se täytyy olla normalisoitu. Kysymys ei ole ainoastaan suorituskyvystä, vaan myös ylläpidosta. Siksi toistuvaa dataa on hyvä olla rajoitettu määrä (tai ei ollenkaan). Jos tietokannassa on paljon toistuvaa dataa ja jokin tämän datan ilmentymä kokee muutoksen (esimerkiksi nimi), muutos täytyy tehdä kaikkiin datan esiintymiin.

Saman tiedon toistamisen poistamiseksi ja mahdollisuuksien parantamiseksi ylläpitää tietoa täytyy tehdä tauluja, jotka sisältävät mahdolliset arvot. Arvoon viittaamiseen käytetään avainta, joka huolehtii siitä, että arvon vaihtuessa muutos tapahtuu vain kerran, kuhunkin asiaan liittyvässä päätaulussa. Viittaus pysyy samana kaikissa muissa tauluissa. [1.]

Hyvin suunnitellun ja toteutetun tietokannan edut ovat lukemattomat. Siksi onkin järkevää, että mitä enemmän työskennellään etukäteen, sitä vähemmän joudutaan korjailemaan myöhemmin. Tietokantaa käyttävän sovelluksen julkistamisen jälkeen uudelleensuunnittelu käy aina kalliiksi. Kannattaa aina käyttää reilusti aikaa tietokannan suunnitteluun ennen sovelluksen koodauksen aloittamista. Suhteet (*relationships*) ja normalisointi (*normalisation*) ovat kaksi tärkeää palikkaa suunnittelun palapelissä. [2.]

2.3 Tavoitteita tietokannan rakenteelle

Ennen tietokannan suunnittelua tulisi pohtia, minkälainen olisi hyvä tietokannan rakenne eli mihin pyritään. Hyvän tietokannan rakenteen keskeisiä ominaisuuksia ovat *kattavuus*, joka sisältää kaikki järjestelmissä tai kyselyissä tarvittavat tiedot ja yhteydet. *Selkeys ja ymmärrettävyys* tarkoittavat yksinkertaista rakennetta, ilmaisuvoimaa ja helppoja kyselyjä. *Muutosjoustavuus* tarkoittaa tietokannan laajennettavuutta minimoiden nykyisten ohjelmien muutokset. *Yleiskäyttöisyys* tarkoittaa tietokannan soveltuvuutta erilaisiin ympäristöihin ja eri asiakkaille tarvitsematta muuttaa tietokannan rakennetta. *Eheys* sisältää toisteisuuden välttämisen, oikeellisuuden ja sisäisen ristiriidattomuuden. *Ohjelmointimukavuus* tarkoittaa selkeitä tietorakenteita ja sarakkeiden kiinteää merkitystä (sarakkeen

merkitys ei saisi riippua toisesta sarakkeesta). *Suorituskyky eli tehokkuus* käsittää riittävän vastausajan tapahtumille ja riittävän tehokkaat eräajot. Tietokantojen rakenteen tulee olla selkeä ja tarkasti tarpeisiin sovitettu. Taulujen ja tietojen nimet ovat omia, tuttuja termejä. Sarakkeet tarkoittavat yleensä yhtä asiaa. [1.]

Tietokannat voidaan jakaa kolmeen kategoriaan:

1. räätälöityihin operatiivisiin järjestelmiin
2. valmisohjelmistoihin ja
3. tietovarastoihin.

2.3.1 Räätälöidyt tietokannat

Tietokannan hyvä muutosjoustavuus tarkoittaa, että tietokannan ylläpitäjä voi lisätä uusia tauluja ja sarakkeita ilman, että olemassa oleviin ohjelmiin tarvitsee koskea. Jos esim. henkilötietoihin tarvitaan uusi sarake ”koulutus”, se lisätään henkilötauluun ja sitä on helppo kysellä. Tietokannan rakenteen suhteellinen yksinkertaisuus on eduksi suorituskyvylle (vähemmän liitoksia). [1.]

2.3.2 Valmisohjelmiston tietokannat

Valmisohjelmistojen tietokantojen tulee olla hyvin yleiskäyttöisiä, jotta sama tietokantarakenne voitaisiin monistaa kaikille asiakkaille. Valmisohjelma sovitetaan kuhunkin ympäristöön erilaisilla määriteltävillä parametreilla. [1.]

2.3.3 Tietovarastokannat

Tietovarastokantojen tulee olla hyvin selkeitä, helppokäyttöisiä ja ymmärrettäviä. Rakenteen tulee tukea helppoja kyselyjä ilman SQL-tyyppistä ohjelmointia. Tiedoilla pitää olla selkeät, kiinteät nimet. Tietovarastot rakennetaan yleensä räätälöidysti, jolloin muutosjoustavuus ja laajennettavuus ovat tärkeitä tavoiteltavia ominaisuuksia. Tiedon toisteisuutta ei vältetä, mutta kyselyjen nopeuttamista suositaan (toisteisuus pitää olla hallinnassa, koska tietojen ylläpito on keskitettyä). Tietovarastoissa säilytetään usein monen vuoden historia, joten verrattuna operatiivisiin kantoihin levytila täytyy ottaa tarkemmin huomioon. [1.]

2.4 Eheys rakenteen yhteisenä tavoitteena

Eheys (*integrity*) tarkoittaa mm. avain- ja viite-eheyttä. Viite-eheyssäännöillä ehkäistään orvoksi jäävien tietojen esiintyminen tietokannassa. Eheys sisältää tietokannan tietojen oikeellisuuden ja sisäisen ristiriidattomuuden, taulutasolla tiedon hallitsemattoman toistamisen.

Hyvän tietokannan muut ominaisuudet saattavat määräytyä tapauskohtaisesti:

- Yhteensopivuus nykyisten tietojärjestelmien tai tietokannan hallintajärjestelmien kanssa.
- Siirrettävyys (skaalautuvuus) laiteympäristöstä tai tietokannan hallintajärjestelmästä toiseen (joskus tärkeää valmisohjelmistoille).
- Turvallisuus - tietoihin pääsee käsiksi vain määriteltyjen ja myönnettyjen käyttöoikeuksien mukaisesti.

Suuntauksen ollessa kohti keskitettyjä tietokantoja operatiivisten tietokantojen hajautusta ei suositella. Sitä vastoin tietojen monistaminen (replikointi) kopiointina kannettaviin tietokoneisiin on yleistynyt. Hyvän tietokannan ominaisuuksista, kuten turvallisuudesta ja viite-eheydestä, huolehtii automaattisesti hyvä tietokannan hallintajärjestelmä. Tietokannan ominaisuudet pitää muistaa määritellä ja suunnitella huolellisesti. [1.]

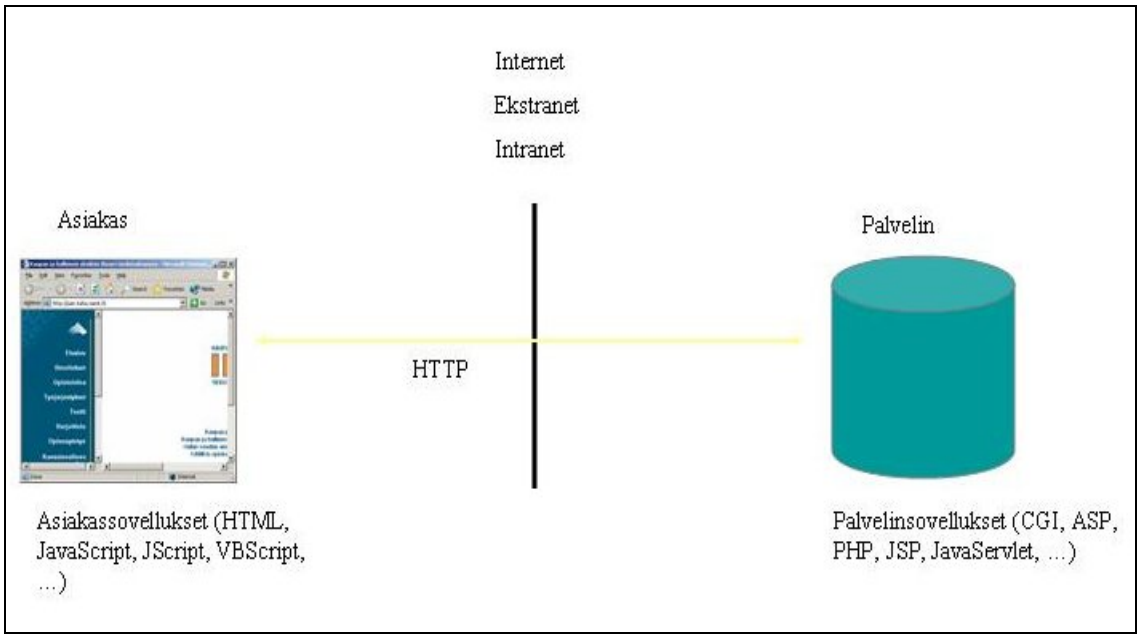
Tietokannan suunnittelussa on hyvä muistaa ainakin kolme tärkeää asiaa:

1. Määritellä tietokannassa käytettävät taulut.
2. Hahmotella taulukot kolmanteen normaalimuotoon - poistaa toisteisuus ja luoda erillisiä tauluja asiaan liittyvälle datalle.
3. Määritellä taulujen suhteet. [2.]

3 KÄYTTÖLIITTYMÄN SUUNNITTELU

3.1 Internet toimintaympäristönä

Internetin toiminta perustuu **asiakas-palvelin** -malliin (kuva 1). Käyttäjät ovat asiakkaita, jotka pyytävät www-palvelimilta haluamiaan sivuja URL (*Universal Resource Locator*) -osoitteella selaimen osoiterivillä. Pyyntö ohjautuu verkossa oikealle palvelimelle, jossa oleva www-palvelinohjelmisto suorittaa pyynnön mukaiset toiminnot ja palauttaa sivun käyttäjän selaimeen katseltavaksi.



Kuva 1. Internetin asiakas - palvelin malli. [3.]

Yhteys selaimen ja palvelimen välille muodostetaan käyttämällä HTTP (*Hyper-Text Transfer Protocol*) -protokollaa. Jokaisella www-palvelimella on yksiselitteinen nimi, jonka perusteella palvelupyyntö ohjautuu oikealle palvelimelle. Julkisella www-palvelimella täytyy siis olla nimi (kuten esimerkiksi kajak.fi), jonka perusteella sitä voidaan kutsua, ja jonka perusteella se voidaan erottaa muista palvelimista.

Tietoturva on noussut erittäin merkittäväksi seikaksi toteutettaessa web-palveluja. Palvelussa käytettävä tietoliikenne tulee salata, mikäli siinä käsitellään arkaluontoista tietoa. Salaaminen voidaan tehdä esimerkiksi käyttämällä

HTTPS-protokollaa. Tietoliikenteen salaamisen lisäksi verkkopalveluun vaaditaan sisään kirjautuminen, jonka perusteella käyttäjä voidaan tunnistaa.

Julkisen Internetin lisäksi www-sovellus voi toimia Intranet- tai Ekstranet-verkossa. Intranet-verkko on täysin suojattu ulkopuoliselta tietoliikenteeltä, jolloin tiettyyn sisäiseen palveluun päästään vain tietyn lähiverkon työasemilta. Ekstranet-verkko on osittain julkinen, jolloin esimerkiksi organisaation yhteistyökumppanit voivat organisaation sisäisen verkon ulkopuolelta käyttää joitakin www-palvelun toimintoja. Monet organisaatiot käyttävät www-tekniikoita erilaisten sisäisten tietojärjestelmien toteutukseen, koska tällöin ylläpitotoimet ovat yleensä helpompia. Asennukset ja käytön aikainen ylläpito voidaan tehdä keskitetysti www-palvelimella, eikä käyttäjien työasemiin tarvitse tehdä muutoksia mahdollisen www-selaimen asennuksen lisäksi. WWW-tekniikoita käyttäen ei kuitenkaan voida toteuttaa kaikkia sovelluksia, vaan pöytäsovelluksiakin tarvitaan. [1.][3.]

3.2 WWW-sovellukset

Yleensä www-dokumentti kuvataan HTML-kielellä. **Staattisen** www-dokumentin sisältö on tehty muuttumattomaksi, paitsi tietenkin silloin, kun sivun tekijä päivittää sivun sisältöä. Mikäli www-palveluun halutaan vuorovaikutteisuutta ja dynaamisuutta, täytyy www-sivusto tehdä lähes poikkeuksetta ohjelmoimalla. **Dynaamisen** www-sivun sisältö on muuttuvaa tietoa ja voi muuttua esimerkiksi jonkin tietokannan sisällön tai käyttäjän tekemien toimintojen perusteella. WWW-sovellukseksi kutsutaan siis www-palvelua, joka on dynaaminen ja toimii vuorovaikutuksessa (interaktiivisesti) käyttäjän kanssa. Tällainen palvelu on esimerkiksi Nordea-pankin verkkopalvelu osoitteessa;
<https://solo3.nordea.fi/cgi-bin/SOLO0001?A01Y_LANF=1&A01Y_P02=DemP>

WWW-sovellus voi olla toteutettu palvelin- ja/tai asiakastekniikalla. Palvelin ja asiakas -sanat määräytyvät www-sovelluksen toiminnan tapahtumapaikan perusteella. Asiakastekniikalla toteutetussa palvelussa sovelluksen suoritus tapahtuu käyttäjän koneella. Palvelintekniikalla toteutetussa sovelluksessa toiminta tapahtuu www-palvelimella. Käyttäjä pyytää siis sivua normaalisti kirjoittamalla

selaimeen osoitteen, jonka jälkeen pyyntö välittyy www-palvelimelle. Esimerkiksi PHP-ohjelmointikielellä toteutettu ohjelmakoodi, joka suoritetaan www-palvelimessa, palautetaan selaimeen - useimmiten HTML-kielellä.

Asiakastekniikoita käytettäessä täytyy tietää, millaisia selaimia sovelluksen käyttäjillä on, mutta palvelintekniikkaa käytettäessä käyttäjäkunnan selaimilla ei ole niin suurta merkitystä, koska kaikki selaimet tukevat standardia HTML-kieltä. Tosin selaimet saattavat näyttää HTML-kielellä koodatun sivun eri tavalla, mikä voi joskus aiheuttaa ongelmia. Asiakastekniikalla toteutettaessa voidaan käyttää mm. JavaScript-, JScript ja VBScript-ohjelmointikieliä. JavaScript on tuettu kaikista laajimmin erilaisissa selaimissa, kun taas esimerkiksi VBScript toimii vain Microsoft Internet Explorer -selaimissa.

Palvelinsovellus koostuu fyysisesti palvelintietokoneesta, johon asennetaan www-palvelin. WWW-palvelimen lisäksi tarvitaan käytettävän tekniikan mukaan erilaisia palvelinohjelmistoja. Esimerkiksi PHP:tä käytettäessä täytyy palvelimelle asentaa PHP-tulkki. Tietokantaa käytettäessä täytyy luonnollisesti asentaa tietokantaohjelmisto tai luoda www-palvelimelta yhteys johonkin tietokantapalvelimeen. WWW-palvelimia on saatavana useita monelta eri valmistajalta. Eräs suosittu ilmainen www-palvelin on Apache, jonka yhteyteen voidaan asentaa tuki PHP-tekniikalle, kuten myös monille muille tekniikoille ja ohjelmointikielille. Apachea käytetään enemmän Unix- tai Linux-käyttöjärjestelmällä varustetuissa koneissa, mutta soveltuu myös hyvin Windows-ympäristöön, jossa käytetyin on ollut IIS (*Internet Information Server*) -palvelin. Kun HTTP-palvelin ja muut tarvittavat ohjelmistot on asennettu, voidaan palvelimelle asentaa ohjelmistokehittäjän tekemiä sovelluksia. [1.][3.]

3.3 Käyttöliittymän suunnittelu

Suunnittelun alkuvaiheessa tulee ottaa huomioon käyttöliittymän tuleva käyttäjä, jolle ehkä tärkeimpiä ovat käytännöllisyys ja visuaalisuus. Ei pidä unohtaa myöskään käyttöliittymän käyttötarkoitusta ja kohderyhmää, jolle tuote tehdään. Koska käyttöliittymä mahdollistaa käyttäjän ja sovelluksen välisen vuorovaikutuksen (interaktiivisuuden), on tärkeää, että käyttöliittymän kautta kommunikoi-

minen koneen kanssa olisi mahdollisimman helppoa. Visuaalisesti hyvin suunniteltu käyttöliittymä luo mukavuutta tietokonetyöskentelyyn. Suunnitteluvaiheessa olisi suositeltavaa testaajaryhmien käyttö, koska ulkopuolinen käyttäjä huomaa mahdolliset virheet helpommin. [4.]

3.4 Käyttöliittymän visuaalinen suunnittelu

Käyttöliittymän visuaalinen suunnittelu on yksi osa-alue suunnitteluprosessin kokonaisuudesta ja siksi suunnittelun tavoitteena tulisikin olla johdonmukaisuus, selkeys ja miellyttävä ulkonäkö. Parhaassa tapauksessa ohjelma on käyttöominaisuuksiltaan ja sisällöltään toimiva esteettinen kokonaisuus, joten kannattaa kiinnittää huomiota myös konkreettiseen elementtien asetteluun ja sommitteluun kuvaruudulla. Kaikki elementit vaikuttavat tavalla tai toisella toisiinsa, (pohja, kuvat, värit, tekstityypit jne.) myös tyhjä tila ruudulla, ja muodostavat kokonaisuuden, käyttöliittymän sivun kokonaisilmeen.

Tyylisuunnittelussa kannattaa pitää kiinni aiheeseen valitusta, sopivasta linjasta, välttää liian monia erilaisia tekstityylejä ja liiallista värien käyttöä, koska liiallinen kikkailu saattaa ärsyttää käyttäjää ja juoruta tekijän tyylitajuttomuudesta. Värien käytössä tulisi ottaa huomioon käyttöliittymän tarkoitus ja kohderyhmä ja päämääränä tulisikin olla käyttötehokkuuden ja käyttöliittymän miellyttävyyden lisääminen, näin asiakas saadaan viipymään sivulla. Värien avulla helpotetaan näytöllä olevien elementtien erottuminen toisistaan ja saadaan luotua yhtenäinen, looginen sovellus. On kuitenkin hyvä välttää moninaista ja liiallista värienkin käyttöä, koska sillä voidaan saada aikaiseksi vastenmielinen, raskas käyttöinen ja sekava käyttöliittymä. Jokaisella värisävyllä on oma merkityksensä ja symboliikkansa.

Jäsentely rajatulle alueelle (koko, muoto, väri ja niiden kontrastisuus) on myös hyvä muistaa, koska niiden avulla voidaan ohjata silmän liikettä haluttuun suuntaan. Ei pidä myöskään unohtaa käyttöliittymän loogisuutta ja tarkoituksenmukaisuutta. Kuvien käyttö hillitysti on myös suositeltavaa. Kuvan rajauksella voidaan korostaa tiettyjä tai jättää pois joitakin asioita, sillä voidaan tehdä myös kuvakulma tarkoituksenmukaiseksi ja mielenkiintoiseksi. Kuvan suunnalla ja kuvassa tapahtuvalla liikkeellä on myös oma merkityksensä. [4.]

3.5 Käyttöliittymän suunnittelun tavoitteet

WWW-sivuston rakennetta suunniteltaessa tulisi ottaa huomioon, että ajan kuluessa sivuston sisältämä tietomäärä kasvaa ja sivujen sisältö muuttuu. Sivusto tulisi suunnitella siten, että sivujen sisältöä voidaan muuttaa ja uusia sivuja lisätä ilman, että sivuston rakenteen perusideaa joudutaan muuttamaan. Sivuston rakenne ei saisi perustua sellaisiin elementteihin, jotka muuttuvat jatkuvasti vaan rakenteen tulisi pysyä vakiona useita vuosia.

4 TIETOKANTAPOHJAINEN SOVELLUS INTERNETISSÄ

Kaikki tärkeät tietojärjestelmät käyttävät tietokantatekniikkaa tietojen tallentamiseen. Yrityksen kannalta tiedot ovat tärkeä resurssi ja monet yritykset ovatkin täysin riippuvaisia tietokannoistaan. Tiedot on tallennettava järkevässä muodossa siten, että tiedoista voi nopeasti saada yhdistelmiä erilaisiin tarpeisiin. Termi tiedonhallinta tarkoittaa kaikkea tallennetun tiedon määrittelemiseen ja käyttämiseen liittyvää toimintaa. Yhtenä esimerkkinä voisi mainita yrityksen materiaalit ja henkilöstön, jotka ovat tärkeitä resursseja yritykselle. Niinpä voidaankin puhua materiaalinhallinnasta ja henkilöstön hallinnasta - tietoresurssin hallinta on siis tiedonhallintaa. [1.]

4.1 Tietokannat

Tietokantana voidaan pitää mitä tahansa tietokokoelmaa, joka koostuu loogisesti yhteenkuuluvasta ja tallennettavasta tiedosta. Tiedot tarkoittavat kirjattavia tosiasioita, joilla on jokin merkitys. Tietokanta voi olla hyvinkin pieni ja yksinkertainen, joka voi olla esimerkiksi reseptikokoelma ja videorekisteri tai henkilökohtainen puhelinnumeromuistio, johon tallennettavaa tietoa ovat tavallisesti vain ihmisten nimet, puhelinnumerot ja osoitteet. Tietokanta siis esittää joitain reaalimaailman asioita, on loogisesti yhtenäinen kokoelma merkityksellistä tietoa, on suunniteltu, rakennettu ja täytetty jotain tiettyä tarkoitusta varten olevilla tiedoilla. Tietokannalla on jokin tietty käyttäjäryhmä ja joitain tiettyjä ohjelmia, joita käyttäjät käyttävät.

Lyhyesti; Tietokannan sisältämä tieto on peräisin jostakin tietolähteestä. Tietokannalla on jotain tekemistä reaalimaailman tapahtumien kanssa. Tietokannan käyttäjät ovat kiinnostuneita sen sisältämistä tiedoista. Tietokanta voi olla pieni ja yksinkertainen tai erittäin suuri ja monimutkainen. Esimerkiksi tavallisen Matti Meikäläisen puhelinnumeromuistio on hyvin pieni ja yksinkertainen tietokanta. Tallennettavaa tietoa on yleensä vain ihmisten nimet, puhelinnumerot ja mahdollisesti osoitteet. Tällainen tietokanta tarvitsee tallennustilaa vain joitakin kymmeniä kilotavuja, eikä vaadi erikoisia haku-, lisäys- tai poistotoimintoja.

Altavistan <URL: *http://www.altavista.com/*> hakukone, joka yrittää ylläpitää hakusanastoa koko internetin sisällöstä, on vaatimuksiltaan aivan eri kokoluokkaa. Pelkkää levytilaa kuluu yli 200 gigatavua puhumattakaan muista laitteistovaatimuksista ja varsinaisen tietokantaohjelmiston toiminnoista. Eikä se ole edes maailman suurin tietokanta, ei edes lähellä suurinta. Tässä työssä käytettävä tietokanta on pienehkö, se vie levytilaa tyhjänä vain joitakin satoja kilotavuja ja täytettynäkin puhutaan vain megatavuista.

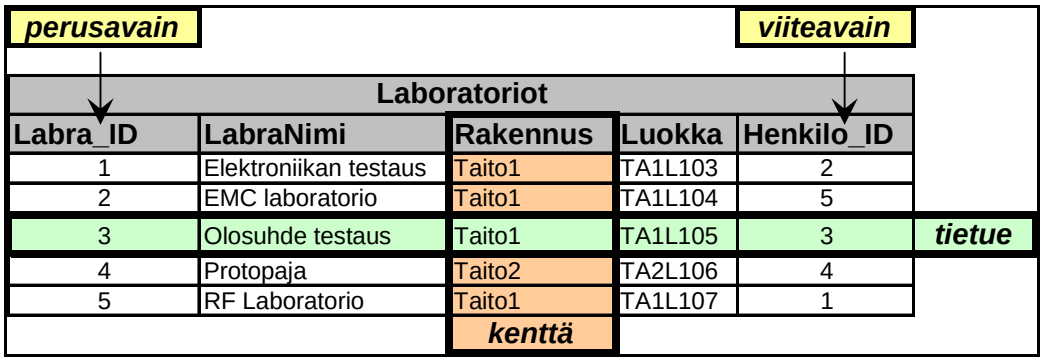
Tietokannalle asetettavia vaatimuksia ovat ainakin:
1) Kukin tieto tallennetaan kannassa vain yhteen paikkaan ilman turhaan esiintyvää toistoa (*redundancy*).
2) Tietojen haku erilaisin perustein sujuu joustavasti, myös perustein, jotka ovat jääneet tietokannan suunnitteluvaiheessa ennakoimatta.
3) Rakenteen muuttaminen käy helposti ja joustavasti.
4) Sovellusohjelmat ja kannan käyttö ovat tietojen fyysisestä tallennusrakenteesta riippumattomia, millä tarkoitetaan tietoriippumattomuutta. [1.][3.]

Tietomallit
Tietomalli (*data model*) on kuvausmenetelmä ja joukko sääntöjä tietorakenteiden ja niiden välisten yhteyksien kuvaamiseen. Tyypillisiä tietomalleja ovat käsitemalli, oliomalli ja relaatiomalli, jotka vielä jakaantuvat Coddin esittämään abstraktiin relaatiomalliin ja sen toteutuksena relaatiotietokantatuotteiden fyysiseen taulumalliin. [1.]

4.2 Relaatiotietokannat

Relaatiotietokannassa tiedot esitetään tauluina (*table*) eli relaatioina, yksi rivi on tietue (*record*) ja taulun jokaisella rivillä on yhtä monta tietoa eli kenttää (*field*). Jokaisella rivillä täytyy olla yksikäsitteinen perusavain, joka vastaa jotakin reaalimaailman kohdetta, mihin liitetään vain siihen välittömästi liittyvät ominaisuudet. Yksittäiset tiedot relaatiotietokannassa voidaan hakea ainakin taulun nimen, perusavaimen kentän nimen ja avaimen arvon sekä haettavan tiedon ken-

tän nimen perusteella. Tietoa siis haetaan vain tiedon nimien ja arvojen perusteella, ei tiedon sijainnin tai järjestyksen mukaan. [5.]



Kuva 2. Relaatiotietokannan taulu.

4.2.1 Relaatiomalli

Relaatiotietokannat perustuvat IBM:n tutkija E. F. Coddin v. 1970 julkaisemaan relaatiomalliin (*the relational model*), joka määrittelee relaatiotietokannan teoreettisen pohjan. Relaatiomalli perustuu joukko-oppiin, matematiikkaan ja predikaattilogiikkaan (SQL-lauseen WHERE-lausekkeessa oleva hakuehto). Coddin määrittelemä relaatiomalli on syrjäyttänyt aiemmin käytetyt hierarkkiset ja verkkomalliset tietokantatyypit. SQL on standardoitunut lähes ainoaksi tietokantakieleksi. Melkein kaikki uudet tietojärjestelmät rakennetaan relaatiokantatuotteiden avulla. Relaatiomalli voidaan jakaa kolmeen osaan: **rakenne**, **käsittely** ja **eheyssäännöt**.

Rakenne

Tietokannan rakenne on kerrottava tietokannan hallintajärjestelmälle (TKHJ) ja kanta on saatava tehokkaaksi. Tietokannan peruselementti on taulu. Taulussa on sarakkeita (*column*) eli kenttiä (*field*), joilla on jokaisella eri nimi, mutta kullakin yhteinen tietotyyppi joko numeerinen tai merkkimuotoinen.

Perus- ja viiteavain

Kussakin taulussa on tunnisteena perusavain (*primary key*, PK), jonka on oltava uniikki eli yksilöivä. Yksilöivyys määrittelee, että sarakkeessa ei voi olla kahdella (tai useammalla) eri rivillä samaa arvoa eli jokaisella rivillä on oma yksikäsitteinen

numeronsa. Perusavainten suunnittelu on tärkeä osa relaatiokannan suunnitteluprosessia ja se voi koostua useammastakin sarakkeesta. Relaatiomallin sääntöjen mukaan taulussa ei saa olla tuplarivejä eikä toistuvia, alisteisia ryhmiä. Tuplarivit estetään perusavaimen avulla ja toistuvat ryhmät tietorakenteiden normalisoimisella. Jos tietokantaan syötettäessä sarakkeella ei ole arvoa, tulee tuohon kohtaa tauluun erityinen merkintä, NULL- eli tyhjä arvo. NULL ei tarkoita välilyöntiä eikä nollaa, vaan tuntematonta arvoa. SQL-kielessä on omat operaationsa NULL-arvon haulle. Taulua perustettaessa voi erikseen määritellä, onko NULL-arvo sallittu vai ei.

Käsittely

Coddin nerokas oivallus relaatiomallissa oli tietojen käsittely joukko-opillisesti. Taulu muodostuu joukosta rivejä. Tähän joukkoon voidaan kohdistaa voimakkaita joukko-operaatioita, kuten ”hae kaikki espoolaiset asiakkaat (*valinta*), niistä nimi ja osoite (*projektio*)”. Joukko-operaatiolla voidaan käsitellä koko taulu tai useampiakin tauluja. Joukko-opillisuus koskee perinteisistä kyselykielistä poiketen myös päivityksiä. Yhdellä päivityskäskyllä voidaan esim. keskeltä Java-kielistä ohjelmaa päivittää iso joukko taulun rivejä ilman silmukoita, mistä johtuen tietokantakäsittelystä tulee hyvin tuottavaa.

Eheyssäännöt

Relaatiomalli määrittelee myös tietokannan eheyden (*integrity*). Tietokanta on eheä, kun sen tiedot ovat oikein, ristiriidattomia ja vastaavat reaalimaailmaa. Tietokanta ei ole eheä, jos esim. sama asiakas tallennetaan kahteen kertaan tai jos asiakkaasta löytyy kaksi eri osoitetta eikä tiedetä, kumpi on oikein. Codd määritteli relaatiomalliin tiettyjä eheysrajoitteita. Ensimmäinen Coddin eheyssääntö on avaineheys (*entity integrity*). Tämän säännön mukaan perusavaimen arvo ei saa olla tyhjä eli NULL-arvo, toisin sanoen perusavaimen arvo on pakollinen. Toinen eheyssääntö on viite-eheys (*referential integrity*). Isätaulusta ei saa poistaa tietoja, jos lapsitaulussa on ko. isään liittyviä lapsirivejä. Muutoin isätaulun rivit jäisivät ”orvoiksi” lapsitauluun. Useimmat relaatiokannan mahdollistavat nykyisin viite-eheyden valvomisen määrittelemällä kantaan eheysrajoitteet, joita ei voida ohittaa. [1.]

4.3 Normalisoinnin tarkoitus

Normalisoinnin (*normalisation*) avulla voidaan tietorakenteita jalostaa ”parempaan” tallennusmuotoon. Parempi tarkoittaa rakennetta, jossa on tietojen toistaminen (*redundanssi*) minimoitu. Rakenne on tehokas päivitystilanteissa ja helpompi yhdenmukaisena, sillä tiedot tarvitsee päivittää vain yhteen paikkaan ja on muutosjoustava. Normalisoinnin kehitti 1972 E. F. Codd eli sama mies, joka määritteli relaatiokantateorian. Sitä voidaan soveltaa kaikenlaisiin tietorakenteisiin. Normalisointiteoriassa tauluille määritellään useita eri normaalimuotoja (*normal form*, NF), joista yleisimmät ja alkuperäiset ovat ensimmäinen, toinen ja kolmas normaalimuoto. [1.]

Normalisointi on yksinkertaisesti joukko sääntöjä, jotka helpottavat huomattavasti tietokantojen ylläpitäjien elämää. Normalisoinnilla tietokanta voidaan organisoida niin, että taulujen suhteet ovat asianmukaiset ja tarpeeksi joustavat ajatellen tietokannan kasvua tulevaisuudessa. Normalisoinnissa käytettyjä sääntöjä kutsutaan normaalimuodoiksi (*normal forms*). Jos seurataan kolmea ensimmäistä normalisoinnin sääntöjoukkoa, tietokannan sanotaan olevan kolmatta normaalimuotoa. [2]

4.4 SQL-kieli

Relaatiotietokantatuotteissa joukko-opillisuus toteutetaan SQL- (*Structured Query Language*) kielellä, joka on syntynyt IBM:n laboratoriossa 70-luvulla ja sittemmin levinnyt kaikkiin merkittäviin relaatiotietokantatuotteisiin. Useimmat relaatiotietokannat ”ymmärtävät” yksinomaan SQL-kieltä. SQL-kielestä on laadittu useita kansainvälisiä standardeja, joista uusimpia on ANSIn SQL-99. SQL on monipuolinen ja voimakas tietokantakieli, niinsanottu ei-proseduraalinen kieli, mikä tarkoittaa, että mitä tietoa haetaan, mutta ei miten. SQL-kielellä voidaan lukea ja päivittää tietokantaa sekä lisätä rivejä. SQL-kieltä voidaan käyttää vuorovaikutteisesti joko päätteeltä tai työasemalta sekä myös upottaa Java-, Visual Basic-, C#- ja Cobol -ohjelmointikieliin.

SQL-kielellä voidaan tehdä liitosoperaatioita, joissa haetaan tiettyjen kenttien tietoja perus- ja viiteavaimien avulla eri tauluista. Liitoskyselyssä yhdistetään isätaulun perusavain lapsitaulun viiteavaimeen, nyt tietokannan hallintajärjestelmä (TKHJ) osaa yhdistää lapsi- ja isätaulun tiedot oikein. SQL-kielessä olevan ilmaisuvoiman esille saamiseksi on kohteena olevan tietokannan oltava hyvin suunniteltu. [1.]

SQL:n suurimpia etuja on se, että alkuperäisen tarkoituksensa mukaisesti se on monen alustan ja monen tuotteen kieli. Tietokantaohjelmien valmistajat ovat kuitenkin kehittäneet SQL:stä erilaisia variaatioita omia tuotteitaan varten, joten kannattaa aina varmistaa, mitä SQL:n ominaisuuksia kulloinkin käytettävä tuote tukee. Nimestään huolimatta SQL ei ole pelkästään kyselykieli, vaan sillä voidaan tietojen lisäämisen, muuttamisen ja poistamisen lisäksi myös määritellä ja muuttaa tietokannan rakennetta sekä valtuuksia tietokannan käyttäjille. Kyselyt ovat kuitenkin SQL:n yleisin käyttöalue. Suurin osa tietokantojen käsittelystä on kyselyjen, yhteenvetojen, raporttien yms. laatimista tietokannan sisältämistä tiedoista. Näissä asioissa SQL on vahvimmillaan. Seuraavalla hyvin tyypillisellä SQL-kyselyllä haetaan henkilöstö-tietokannasta Kajaanissa asuvat henkilöt.

```
SELECT * FROM Henkilot WHERE Kotikunta='Kajaani'
```

Relaatiotietokantojen ja SQL:n avulla tämä voidaan tehdä kiinnittämättä huomiota siihen, kuinka tietojen haku tai tallentaminen on teknisesti toteutettu. Esimerkissä esiintyvien komentosanojen lisäksi muita yleisiä komentosanoja ovat esimerkiksi INSERT, UPDATE, ORDER BY ja DELETE. Koodin lukemisen helpottamiseksi kannattaa kyselyrakenteessa komennot kirjoittaa isoilla kirjaimilla. [6.]

4.5 Tietokantahallintajärjestelmät

Tietokanta (*Database*, DB) voi olla tuotekohtainen, moniselitteinen käsite. Tietokanta on siis loogisesti yhteenkuuluvien, tallennettujen tietojen joukko, jota voidaan helposti käsitellä jollakin tietokantakielellä (esim. SQL). Tietokannassa olevia tietoja hallinnoi erityinen ohjelmisto, tietokannan hallintajärjestelmä TKHJ (*Da-*

tabase Management System, DBMS). Tunnettuja esimerkkejä näistä ovat Oracle, DB2, Microsoft SQL Server, MySQL ja Access.

Tietokannan hallintajärjestelmät ovat erittäin monimutkaisia ja isoja ohjelmistoja, jotka tarjoavat ohjelmoijille ja käyttäjille monenlaisia palveluja. Muutosjoustavuuden lisäämiseksi, tietoeheyden turvaamiseksi, suorituskyvyn parantamiseksi sekä sovellusohjelmoinnin helpottamiseksi käytetään tietokantoja tietojen tallennukseen. Jos TKHJ:ta ei olisi, käyttäisimme tiedostoja, jolloin varsinkin monimutkaisempien tietokokonaisuuksien ohjelmointi olisi työläämpää, tietokannan sisällön eheys olisi heikompi ja tietojen hakeminen olisi vaikeaa. Tietokannan hallintajärjestelmän avulla tietokanta voi olla: Yhteiskäyttöinen useille eri sovelluksille. Ajantasainen - yhdessä paikassa tehdyt päivitykset näkyvät heti kaikille muille. Ei toisteinen - tiedot on tallennettu kertaalleen. Eheä - tiedot ristiriidattomasti kuvaavat mahdollisimman tarkasti reaalimaailmaa. [1.]

Tietokannan hallintajärjestelmät ovat perustaltaan aivan samanlaisia. Kuitenkin niihin on rakennettu sellaisia ominaisuuksia kuin monipuoliset tietoturvaominaisuudet, elvytystekniikka ja transaktiot, joita ei henkilökohtaisissa tietokantaohjelmissa (*Access, Paradox*) välttämättä ole ainakaan samassa laajuudessa. Henkilökohtaisia tietokantaohjelmia ei ole myöskään suunniteltu yhtä suurien tietomäärien tallentamiseen ja käsittelyyn kuin "oikeita" tietokannanhallintajärjestelmiä.

Tietokannanhallintajärjestelmän perusvaatimuksia ovat:
1) Perusoperaatiot (tallennus, haku, päivitys) 2) Tietoriippumattomuus 3) Yhteiskäytön mahdollisuus 4) Ylimäärättömyys 5) Tiedon eheys 6) Tiedon turvaaminen 7) Tehokkuus ja hyvä suorituskyky 8) Yhteensopivuus 9) Skaalautuvuus. [5.]

4.5.1 Indeksit ja optimointiohjelma

Tiedot tallennetaan relaatiokantaan tauluihin, joiden lisäksi tietokantaan voidaan perustaa indeksejä (*index*). Indeksi, joka toimii kuten kirjan hakemisto, tarkoituksena on nopeuttaa hakuja relaatiokannan tauluista. Koska hakemisto on järjestyksessä, sieltä voi nopeasti hakea halutun asian ja hypätä sitten oikealle

sivulle. Ei tarvitse lukea kirjan joka sivua eikä lukea koko taulua läpi. On vain osattava suunnitella oikeat indeksit. [1.]

Tietokantojen indeksit siis auttavat löytämään asioita nopeasti. Jos määrittelee tietyn kentän taulussa perusavaimeksi, MySQL lisää automaattisesti tämän tiedon indeksiin. Indeksejä voidaan lisätä tauluun myös käsin, jos halutaan indeksoida muita kuin perusavaimena toimivia kenttiä, tai jos halutaan indeksoida indeksejä tarjoavat kentät, jotka ovat yhdistelmiä yhdestä tai useammasta kentästä. Valittaessa tietokannan indeksoimia tietueita, kysely toimii nopeammin ja antaa näin ollen tuloksen nopeammin kuin jos taulussa ei olisi indeksejä. Vastaavasti kun lisätään tietue tauluun, jonka täytyy indeksoida arvo, kysely on hieman hitaampi kuin jos indeksiä ei vaadittaisi. Siksi kannattaa miettiä, milloin lisätä indeksejä ja milloin olla lisäämättä.

Tauluihin, joita ”luetaan” enemmän kuin ”kirjoitetaan”, kannattaa lisätä indeksejä. Toisin sanoen lisätään indeksejä tauluihin, joiden tarkoitus on sisältää näytettävää dataa, eikä satunnaista käyttöä varten olevia tietoja. Peukalosääntönä indeksien lisäämisessä voidaan pitää, että kannattaa luoda yleisiä valintaan liittyviä kyselyjä jäljittelevä indeksi. Esimerkiksi; sovellus kutsuu kyselyä, joka valitsee kaikki tuotteet, jotka ovat sinisiä (väriltään) ja suuria (kooltaan). Lisää indeksi, joka on yhdistelmä molemmista kentistä, eli väristä ja koosta, tämä useamman sarakkeen yhdistävä indeksi varmasti vauhdittaisi asioita. [2.]

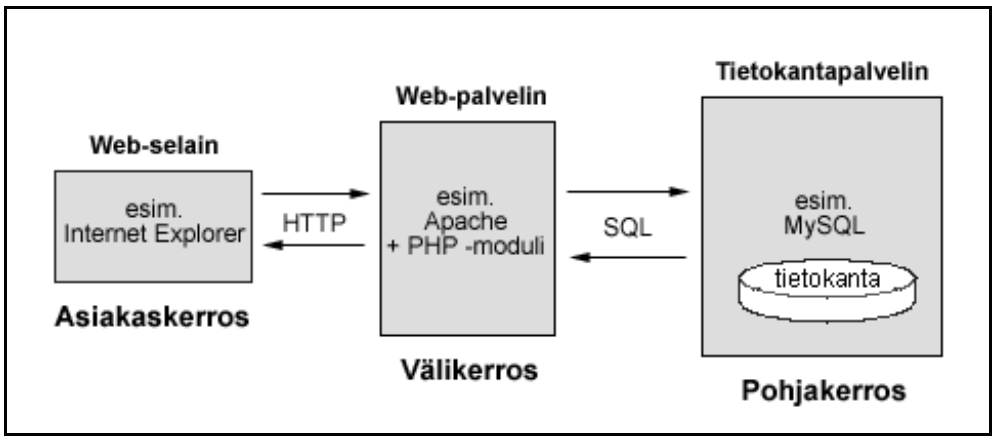
Kaikissa kunnon relaatiotietokantatuotteissa on optimointiohjelma (*optimizer*), joka ottaa vastaa SQL-kyselyn ja pyrkii löytämään sille nopeimman saantipolun (*access path*). Optimointiohjelma päättää, onko saantipolkuna taulun läpiluku vai indeksin kautta hakeminen. [1.]

4.6 Kolmikerrosmalli

Yksinkertaisin tapa käyttää relaatiotietokantaa on suoraan henkilökohtaisen työaseman tiedostojärjestelmän tai lähiverkon tiedostopalvelimen välityksellä. Asiakas-palvelinmallissa asiakasohjelma lähettää SQL-kyselyn tietokantapalvelimelle, joka suorittaa kyselyn ja palauttaa tulokset asiakkaalle. Tällä tavalla tietokanta-

palvelin pystyy palvelemaan samanaikaisesti useita asiakasohjelmia. Asiakas-palvelinmallin etuna on myös, että siinä tarvitsee kuljettaa ainoastaan kyselyt ja niiden tulokset. Kaikissa käyttöympäristöissä ei kuitenkaan voida käyttää tietokantojen asiakas-palvelinmallia. Web on tällainen ympäristö, koska www-selain ei yleensä pysty toimimaan suoraan tietokantapalvelimen asiakkaana. Silloin joudutaan turvautumaan niin sanottuun kolmikerrosjärjestelmään (*three-tier model*), jossa selaimen ja tietokantapalvelimen väliin lisätään välikerros, joka toimii palvelimena selaimelle ja asiakkaana tietokantapalvelimelle.

Kuvassa 3 on esitelty yleinen www-sovelluksissa käytetty kolmikerrosmalli. Se vastaa tämän työn kehitysympäristönä ollutta mallia. Asiakaskerroksena on www-selain, välikerroksena PHP:llä laajennettu www-palvelin ja pohjakerroksena MySQL-tietokantapalvelin ja tietokanta.



Kuva 3: Kehitysympäristöä vastaavan kolmikerrosmallin kuvaus. [6.]

Kolmikerrosmallilla on omat etunsa, koska jokainen kerros voi sijaita ja käytännössä usein sijaitseekin eri koneissa, tämä lisää suorituskykyä. Hyvin suunnitellussa kerrosrakenteessa yhtä kerrosta voidaan muuttaa tai jopa vaihtaa kerros ilman muutoksia muihin kerroksiin. Esimerkiksi jos havaitaan, että aiemmin käytetty Access-tietokanta ei enää vastaa tarpeisiin, voidaan tietokanta siirtää tehokkaampaan tietokantapalvelimeen, kuten Microsoftin SQL Serveriin tai MySQL relaatiotietokannan hallintajärjestelmään. Sitten vain ohjataan välikerroksen tietokantakyselyt uuteen palvelimeen, jos sovellus on järkevästi toteutettu. [6.]

4.7 Tietokantapalvelimena MySQL

MySQL on kehitetty alun perinkin käytettäväksi verkkosovelluksissa, missä tietokannoille asetettavat tärkeimmät vaatimukset ovat nopeus, laajennettavuus ja ylläpidon helppous. Näiden tavoitteiden saavuttamiseksi siitä on jätetty pois muutamia tavallisia tietokantojen ominaisuuksia, joihin kuuluvat muun muassa transaktioiden eli tapahtumien tuki, hyödylliset viiteavaimet ja alikyselyt kyselyjen sisällä. Kaikki on kuitenkin otettu mukaan MySQL 4.0:n ja 4.1:n kehityssuunnitelmiin. [2.]

MySQL on monipuolinen, joustava ja suorituskykyinen relaatiotietokanta, jota käytetään niin suurten kuin pientenkin www-palvelujen taustalla. Ruotsalainen MySQL Ab:n kehitti sen alun perin kyseisen konsultointiyrityksen sisäiseen käyttöön. MySQL noudattaa asiakas - palvelin -arkkitehtuuria, jossa sovellukset eivät käsittele tietokantaa suoraan vaan käsittely tapahtuu aina palvelinohjelman kautta. MySQL-palvelimeen voidaan ottaa yhteys ainakin ODBC- tai Java/JDBC-, PHP-, Perl-, Python- ja Tcl-tekniikoilla tai sitten sen oman C-ohjelmointirajapinnan (MySQL, C, API) tarjoamien käskyjen avulla. Myös oliopohjainen C++ -ohjelmointirajapinta on käytettävissä.

Yhdellä MySQL-palvelimella voi olla useita tietokantoja ja niissä kussakin useita tauluja. Kuten SQL-tietokannoissa yleensä, MySQL-tietokannan sisäiset käyttöoikeusasetukset mahdollistavat monimutkaistenkin sovellusten vaatimat käyttöoikeusmäärittelyt. Tietokantapalvelimelle voidaan luoda rajaton määrä käyttäjätunnuksia, joilla kullakin voi olla eritasoisia oikeuksia tietokantoihin ja tauluihin. MySQL-tietokannan kyselykielenä on SQL, joka on standardoitu kieli, mutta ei noudata standardia kovinkaan perusteellisesti muutoin kuin peruskomentojen kohdalla. Omalta osaltaan MySQL laajentaa SQL-komentokantaa, joten kyseessä ei ole karsittu versio standardi SQL-kielestä, ainoastaan erilainen versio.

MySQL on saatavilla ilmaiseksi GPL-lisenssillä. Kuitenkin, jos ohjelmistoa levitetään kaupallisesti, erillinen MySQL-lisenssi pitää aina ostaa. MySQL:n lisenssipolitiikka ja vaihtoehdot voi tarkistaa osoitteesta <*http://www.mysql,com*> en-

nen johtopäätöksien tekemistä. Ajantasaisen tiedon lisenssivaatimuksista saa samasta osoitteesta. [6.]

4.8 WWW-sovellusten toteuttamistekniikat

WWW-sovellusten alkuperäistä staattista luonnetta on paranneltu useilla erilaisilla tekniikoilla. Käytettävät tekniikat voidaan jakaa karkeasti kolmeen osaan:

1) WWW-selaimessa suoritettaviin asiakastekniikoihin

- HTML, CSS, JavaScript, DHTML, Java-sovelmat, ActiveX, Flash, VRML

2) WWW-palvelimessa suoritettaviin palvelintekniikoihin

- CGI (*Common Gateway Interface*)
- Java Servletit
- Upotetut (*Embedded*) tekniikat (SSI - *Server Side Includes*, PHP, ASP - *Active Server Pages*, JSP - *JavaServer Pages*)
- Sovelluspalvelimet (*Application servers*)

3) HTTP-yhteyskäytäntöön (protokollaan) liittyviin tekniikoihin

- URL:n rakenne ja koodaus, evästeet, autentikointimenetelmät, istunnon hallinta.

Kokonainen www-sovellus voi rakentua:

- käyttöliittymästä, joka sijaitsee www-selaimessa asiakaspuolella ja rakentuu minimissään sekä HTML-elementeistä (erityisesti lomakkeet) että www-selaimen ominaisuuksista
- sovelluslogiikasta, joka käyttää lähes poikkeuksetta palvelintekniikoita ja joka toteutetaan esim. PHP-kielellä
- käyttöliittymän ja sovelluslogiikan välisestä viestinnästä
- HTTP-protokollasta [6.]

4.8.1 Käyttöliittymä

Käyttöliittymä, joka toimii kolmikerrosmallin **asiakaskerroksena** (kuva 3), rakentuu www-sovelluksissa sekä HTML-elementeistä (erityisesti lomakkeet) että www-selaimen ominaisuuksista. Lisäksi voidaan käyttää erilaisia elävöittämistekniikoita, kuten esimerkiksi CSS (*Cascading Style Sheet*), JavaScript, Java-

sovelmat. DHTML (*Dynamic* HTML) on yleisnimitys tekniikoille, jotka koostuvat W3C:n DOM (*Document Ohject Model*) -määrittelystä, jossa HTML-dokumenttia kuvataan puumaisena oliorakenteena. Käytännössä se tarkoittaa HTML:n, CSS:n ja JavaScriptin integroimista saman käsitteen alle. Vaikka asiakastekniikoita käyttämällä voidaan saavuttaa tiettyä dynaamisuutta www-sovellukseen, on erittäin tärkeää ymmärtää, että se rajoittuu pääsääntöisesti käyttöliittymän hallintaan. [6.]

4.8.2 Sovelluslogiikka

Sovelluslogiikan, joka toimii kolmikerrosmallin **välikerroksena** (kuva 3), suoritus tapahtuu lähes poikkeuksetta palvelintekniikoilla. Organisaatioiden tietojärjestelmissä ja niiden hyödyntämisessä www-sovelluksessa tarvitaankin miltei poikkeuksetta palvelinpuolen tekniikkaa siitä syystä, että tietojärjestelmän tiedot eivät ole asiakkaiden selaimissa. Mikäli organisaation toiminta on riippuvainen tietojärjestelmän toiminnasta, on kyseessä tietokantaan perustuva tietokantapalvelin systeemi, joka toimii kolmikerrosmallin **pohjakerroksena** (kuva 3). Perinteisesti käytetään relaatiotietokantoja, joihin www-sovellukset joudutaan integroimaan ja joiden käsittely perustuu SQL-kieleen (*Structured Query Language*). Palvelinpuolen infrastruktuuria käyttöjärjestelmä-, www-palvelin- ja tietokantapalvelinratkaisuineen tarvitaan lähes kaikkien ratkaisujen pohjalle.

Upotetut tekniikat (mm. PHP, ASP, SSI, JSP) ovat tekniikoita, jotka oletusarvoisesti www-palvelimella tulkitsevat HTML-dokumenttien sisään upotetun ohjelmakoodin. Yleisesti näitä tekniikoita pidetään helppoina varsinkin aloitteleville www-ohjelmoijille, koska aluksi voidaan luoda HTML-dokumentin rakenne millä välineellä tahansa ja vasta sen jälkeen upottaa dokumenttiin haluttu määrä sovelluslogiikkaa eli ”älykkyyttä”. Seuraavassa lyhyt tiivistetty kuvaus käytetyistä tekniikoista:

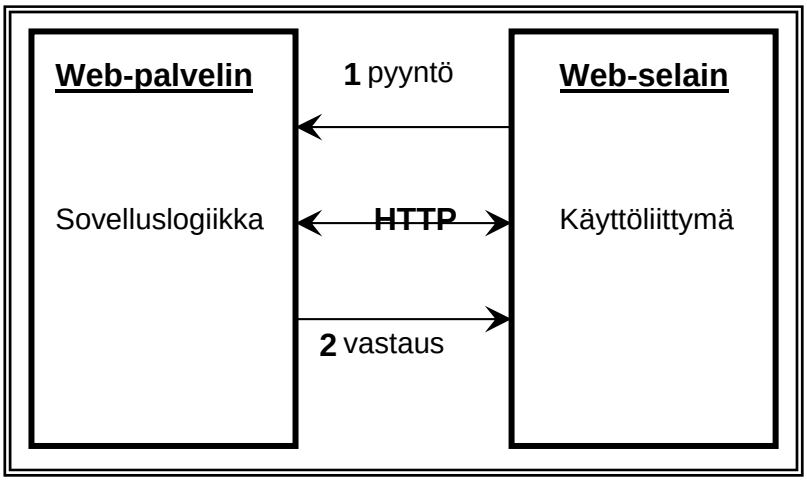
- **CGI** (*Common Gateway Interface*) on yleinen liityntä (ei ohjelmointikieli) www-palvelimen ja sen suorittamien ohjelmien välillä, jota voidaan hyödyntää useilla eri ohjelmointi- tai skriptikielillä (esim. Perl, C, PHP).
- **PHP** (*PHP: Hypertext Preprocessor*) on useilla alustoilla toimiva vapaa (ilmainen) upotustekniikka. PHP-kielen syntaksi on lainattu pääosin C-kielestä.

- **ASP** (*Active Server Pages*). Microsoftin www-palvelimien (kuten IIS) käyttämä upotustekniikka, jonka ohjelmointikieli voi olla mm. VBScript, Perl, jne.
- **SSI** (*Server Side Includes*), jonka avulla voidaan upottaa yksinkertaista palvelinpään toiminnallisuutta www-dokumentteihin. SSI mahdollistaa esimerkiksi kaikilla sivuilla toistuvan navigaatiopalkin upottamisen useaan dokumenttiin. SSI:llä generoidut dokumentit tunnistaa tiedoston .shtml -päätteestä.
- **JSP** (*JavaServer Pages*). Sunin kehittämä ASP:tä ja PHP:tä vastaava tekniikka, jonka dokumentit käännetään ennen suoritusta Java Servleteiksi.
- **Java Servletit**. WWW-palvelimella dynaamisesti toimiva tekniikka www-dokumenttien luomiseksi Java-sovelmien (applettien) avulla.
- **Sovelluspalvelin**. Tiettyä sovellusaluetta (esim. sähköinen kaupankäynti) varten koostuva kokonaisuus www-palvelimesta, tietokantapalvelimesta ja muista ohjelmistoista. [6.]

4.8.3 Käyttöliittymän ja sovelluslogiikan välinen viestintä

WWW-sovelluksen ohjelmointiympäristö saattaa tuntua aluksi vaikealta. Hallittavaa on paljon www-selaimessa muodostettavasta käyttöliittymästä palvelimella suoritettavaan sovelluslogiikkaan. Vaikeinta on kuitenkin omaksua yhdistäviä HTTP (*HyperText Transfer Protocol*) toimintamekanismeja. HTTP on yhteyskäytäntö, jolla www-sovelluksen käyttöliittymä ja palvelimella sijaitseva sovelluslogiikka viestivät keskenään (kuva 4). HTTP-protokollan tilattomuus vaatii tutustumista HTTP-viestien rakenteeseen otsakkeineen, joita hyödyntämällä voidaan käyttää evästeitä, autentikointimenetelmiä ja istunnon hallintaa. Viestien välityksessä käytetty URL- koodaustapa vaatii myös tarkempaa perehtymistä asiaan.

Tosiasia on, että käyttäjä käyttää www-sovellusta www-selaimen ja yleensä selaimen HTML-lomakkeella, joka on www-sivulla oleva alue. HTML-lomakkeen avulla käyttäjä voi syöttää tietoja lähettääkseen ne www-selaimella www-palvelimelle ja sitä kautta sovellukselle. [6.]



Kuva 4. WWW-sovelluksen rakenne: käyttöliittymä ja sovelluslogiikka. [6.]

4.8.4 PHP-kieli

PHP (*Hypertext Prosessor*) oli alun perin kokoelma www-pohjaisten sovellusten, eräänlainen palvelimella ajettavien CGI-ohjelmien tekemistä helpottava komentokokoelma. PHP:n kehitys on ollut nopeaa, ja versiosta neljä ollaan siirtymässä jo versioon viisi. PHP4 on täysiverinen ohjelmointikieli, mutta PHP5 tuo mukanaan paljon lisää tehoa uusia ominaisuuksia.

PHP-ohjelmointikielen komentoja voi kirjoittaa mihin tahansa kohtaan HTML-sivustojen sisään. Sivulla voi myös olla useita erillisiä lohkoja, joissa käytetään PHP-koodia. Muuttumattomana pysyvän HTML-koodin sisällä voidaan näin tulostaa dynaamisesti muuttuvia tietoja, kuten päivämääriä, käyttäjätietoja ja tietokannasta haettuja sisältöjä. Kun perinteisiä CGI-ohjelmia tehtiin siten, että ohjelmointikielen komentojen avulla tuotettiin myös sivulla tarvittavat HTML-merkkaukset, niin PHP toimii päinvastoin. PHP-kielen komentoja voidaan siis kirjoittaa mihin tahansa paikkaan HTML-tiedostoa ja www-palvelimella olevaa tekstimuotoista tiedostoa. Myös aktiivisten ja tietokantapohjaisten WAP-laitteissa (*Wireless Application Protocol*) näytettävien WML (*Wireless Markup Language*) -sivujen tekeminen on PHP:n avulla helppoa. Tulevaisuuden kannettavien päätelaitteiden XML-pohjaiset (*eXtended Markup Language*) formaatit ovat sellaisenaan PHP-yhteensopivia.

PHP on tulkattava kieli, eli www-sivun sisällä oleva PHP-koodi ajetaan joka kerta, kun www-palvelin lähettää sivun selaimelle. PHP voi toimia Apache-

palvelimen sisäisenä moduulina, jolloin se on kiinteä osa www-palvelinta. Tämän kytköksen ansiosta PHP-koodia sisältävien sivujen ajaminen on suhteellisen nopeaa, sillä erillisen tulkin käynnistämistä ulkoisten PERL-ohjelmien tapaan ei tarvita. PHP-koodi ajetaan aina palvelimella, juuri ennen kuin sivu lähetetään selaimeen. Windows CGI-versiossa PHP on erillinen php.exe -ohjelma, jonka kautta sivut suoritetaan. PHP on www-palvelimen laajennus, joka mahdollistaa monimutkaistenkin sovellusten toteutuksen palvelimella. Samasta syystä sivujen lukijat ja sovelluksen käyttäjät eivät näe PHP-koodia, kun he katsovat sivun HTML-koodia selaimessaan. PHP ei siis kilpaile selaimissa toimivan JavaScriptin kanssa. [7.]

Esimerkki 1 HTML-koodista (taulukko 1), johon on upotettu PHP-skripti (lopputuloksena WWW-sivu, jossa lukee; ”Moi, koodataan PHP-skriptiä!” ja ”Tänään on 30.9.2005”).

Taulukko 1. HTML-koodin esimerkki.

```
<HTML>
  <HEAD>
     <TITLE>Esimerkki</TITLE>
  </HEAD>
   <BODY>
    <H2>
            <?PHP
             echo ”Moi, koodataan PHP-skriptiä!”;
             ?>
    </H2>
   Tänään on
           <?PHP
           print date (”j.n.Y”);
           ?>
   </BODY>
</HTML>
```

4.9 Zend Engine

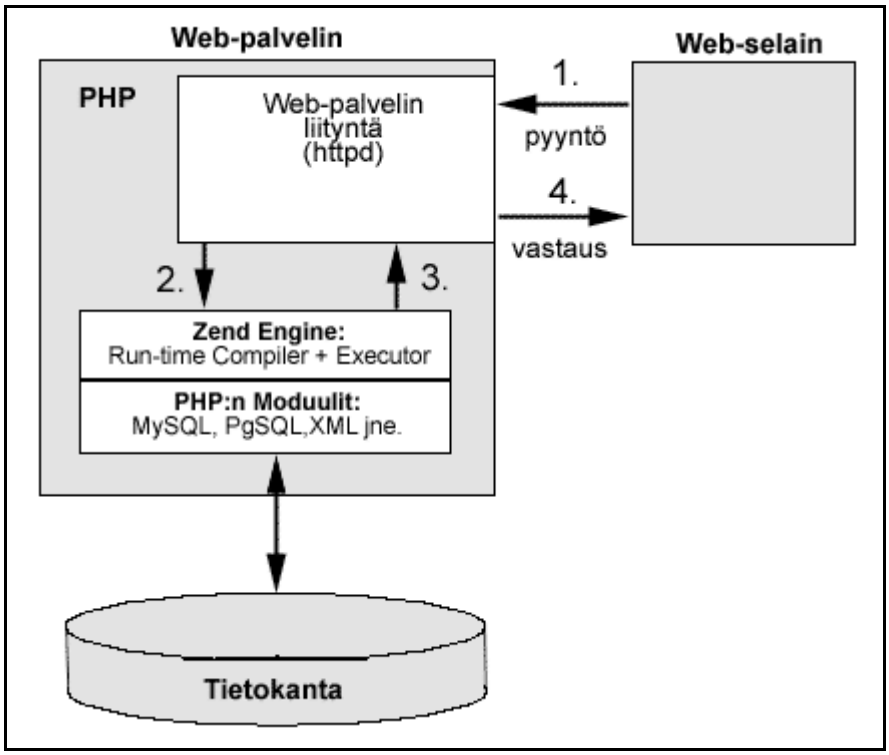
PHP-kieltä käytetään nykyisin yhä isommissa järjestelmissä. Versio 4 varten skaalautuvuuden varmistamiseksi PHP-tulkki kirjoitettiin kokonaan uudelleen. Tämän PHP:n keskeisimmän osan nimi on Zend Engine, jonka nimi tulee pääke-

hittäjiensä Zeev Suraskin ja Andy Gurmansin mukaan. Zendin keskeisimmät osat ovat ajonaikainen kääntäjä (*Run-Time Compiler*) ja suoritin (*Executor*). Zend on skriptimoottori, joka toimii PHP-pohjaisten moduulien alla ja on optimoitu parantamaan merkittävästi suorituskykyä. [1]

Kun taulukossa 1 oleva koodi suoritetaan, näkyy selaimessa ainoastaan ohjelman tulostus ”Moi, koodataan PHP-skriptiä! ja Tänään on 30.9.2005”. Jos käyttäjä, siis asiakas haluaa nähdä tulostuksen lähdekoodin, hän näkee ainoastaan HTML-koodilla kirjoitetun tekstin. **<?PHP, echo, ?>** ja **<?PHP, print date, ?>** eivät näy lähdekoodin näytössä, koska ne ovat PHP-koodia.

Zend Encoderin (kaupallinen) avulla PHP-koodi on mahdollista kääntää binääriseksi esitysmuodoksi ja näin tehdä PHP-kielellä toteutetuista ohjelmista lähdekoodiltaan suojattuja. Käyttäjä eli asiakas ei näin pääsekään enää käsiksi lähdekoodiin hankittuaan PHP:llä toteutetun verkkopalvelun. PHP-tulkki ei osaa automaattisesti suorittaa binäärimuotoon käännettyjä tiedostoja. Avuksi tarvitaan Zend Optimizer (kaupalinen), joka on saatavilla useimmille käyttöjärjestelmäalustoille. Zend Optimizer toimii tavallisten PHP-sivujen kanssa, nimensä mukaisesti optimoiden koodin suoritusta - käytännössä siis Optimizer kannattaa asentaa palvelimelle, vaikka binäärisiä PHP-tiedostoja ei olisikaan käytössä. PHP-koodin kääntäminen Zendin avulla on helppoa. Käännetyt PHP-tiedostot toimivat siis Zend Optimizerin avulla kuten normaalit PHP-tiedostot, niiden lähdekoodia ei vain enää pystytä lukemaan. [7.]

Kuvan 5 mukaisesti dynaamisen www-dokumentin muodostaminen PHP-tekniikalla alkaa selaimen pyynnöstä (kohta 1). WWW-palvelin havaitsee pyynnön tiedostotarkenteesta **(.php**), että dokumentti sisältää käännettäviä skriptejä. Se ohjaa dokumentin Zendin ajonaikaiselle kääntäjälle (kohta 2) käännettäväksi, minkä jälkeen suoritin suorittaa käännetyt skriptit käyttäen tarvittaessa moduuleja esimerkiksi yhteyden ottamiseksi tietokantaan. Zendin suoritin palauttaa luomansa HTML-koodin www-palvelimelle (kohta 3), joka palauttaa www-dokumentin vastauksena (kohta 4) selaimelle esitettäväksi. [6.]



Kuva 5. PHP:n toiminta. [6.]

4.10 Apachen www-palvelinohjelma

Apache on monipuolinen www-palvelinohjelmisto, jota käytetään pääasiassa Unix-käyttöjärjestelmissä. Apachen Unix- ja Windows-versiot ovat toiminnoiltaan hyvin lähellä toisiaan, vaikka Apachen Unix-versiot ovatkin tehokkaampia ja paremmin testattuja. Apachen Cross Platform on luotettava ja paljon käytetty palvelinohjelma, jonka Windows- ja Unix-pohjaiset versiot ovat toiminnoiltaan ja määrityksiltään melkein identtiset, mikä mahdollistaa palvelujen kehittämisen ja tuottamisen molemmissa ympäristöissä. [7.]

5 PROJEKTIN TOTEUTUKSEN KUVAUS

5.1 Tekninen ratkaisu

Kolmikerrosmallin mukaisessa ratkaisussa pohjakerroksena toimii MySQL- tietokantapalvelimessa (kehitysympäristössä versio 4.1.9) oleva tietokannan hallintajärjestelmä. Välikerros muodostuu www-palvelimesta ja sen PHP-laajennuksesta sekä PHP-MyAdmin tietokantarajapinnasta (kehitysympäristössä Apache versio 1.3.33, PHP versio 4.3.10 ja PHP-MyAdmin versio 2.6.1) sekä palvelimella ajettavista www-dokumentteihin upotetuista PHP-skripteistä. Asiakaskerroksen muodostavat järjestelmän käyttäjän oma selain.

Sovelluksen lopulliseksi alustaksi tulee Kajaanin ammattikorkeakoulun Windows-palvelin. Sitä laajennetaan tämän työn lopullisen version valmistuttua MySQL v4.1.9 -tietokantapalvelimella, PHP v4.3.10 -ympäristöllä, Apache v1.3.33 -palvelinohjelmalla sekä PHP-MyAdmin v2.6.1 -tietokantarajapinnalla. Kaikki edellä mainitut ohjelmistot on koottu yhdeksi asennusohjelmapaketiksi EasyPHP v1.8, joka on tarkoitettu käytettäväksi Windows-ympäristössä ja löytyy osoitteesta <*http://www.easyphp.org/*>. Sovelluksen siirtäminen kehitysympäristöstä lopulliseen sovellusympäristöön pitäisi sujua ongelmitta. Ohjelmaversiot voivat muuttua, mikäli työn tilaaja haluaa asentaa ohjelmiston Linux-palvelinympäristöön.

5.2 Tietokannan toteutus

Aluksi piti selvittää järjestelmältä vaadittavat ominaisuudet. Tämä tehtiin keskustelemalla Kajaanin ammattikorkeakoulun edustajan ja valvovan opettajan sekä laitteiden vuokrauksesta vastaavan opettajan kanssa ja tutustumalla aikaisempiin Excel-taulukkolaskentaohjelmalla tehtyihin taulukoihin. Tuloksena saatiin tietokannalle seuraavia vaatimuksia:

- Testauslaboratorioista tallennetaan nimi, sijainti ja vastuuhenkilö.
- Vastuuhenkilöistä tallennetaan nimi, työtehtävä, yhteystiedot ja varamies.
- Komponenteista tallennetaan nimi, tyyppinumero, selitys, kotelotyyppi, määrä, valmistaja, toimittaja ja datasivut sekä sijainti.
- Testauslaitteista tallennetaan nimi, tyyppi, sarjanumero, turvamerkintänumero, valmistaja, toimittaja ja onko laite vuokralla vai ei sekä sijainti.
- Jokainen komponentti ja testauslaite kuuluvat johonkin laboratorioon.
- Testauslaitteiden huollosta tallennetaan käyttöönottopäivämäärä, kalibrointiväli, -päivämäärä ja -luokka, tarkastusohje sekä toimenpiteet.
- Testauslaitteiden vuokrauksesta tallennetaan vuokraus- ja palautuspäivämäärä, vuokrapäivät yhteensä, vuorokausivuokra, vuokra yhteensä sekä laskutuspäivämäärä.
- Testauslaitteiden vuokraajasta tallennetaan vuokraava yritys, yrityksen yhteyshenkilö, puhelinnumero ja yrityksen osoitetiedot.

Komponenteille ja laitteille on molemmille tehty omat sijaintitaulunsa, jotka on yhdistetty sijaintitaulussa laboratorioihin viiteavaimen perusteella. Lisäksi komponenttien määrä on sijoitettu komponenttien sijaintitauluun, koska saman tyypin komponenttia voi esiintyä useammassa laboratoriossa. Tämä takaa sen, että komponenttien määrä kussakin laboratoriossa näkyy listauksessa oikein.

Seuraavaksi jokaiseen tauluun on määritelty kentät ja kenttien ominaisuudet, jotka ovat joko merkkijono-, numero- tai päivämäärätieto. Kaikille kentille ei ole pakko määritellä arvoa. Esimerkiksi taululla **Komponentit** olevat kentät ovat **Kompo_ID**, **Komponimi**, **TyyppiNro**, **Kuvausselite**, **Kotelotyyppi**, **Valmistaja**, **Toimittaja** ja **Datasivut**. Kaikille muille kentille on pakko määrittää arvo paitsi Kuvausselite, Kotelotyyppi ja Datasivut. Sitten on määritelty taulujen perus-

avaimina toimivat kentät. Esimerkiksi taululla **Komponentit** se on **Kompo_ID**. Avain voi koostua myös useammasta kuin yhdestä kentästä, jolloin sitä kutsutaan yhdistetyksi avaimeksi (*composite key*). Jokaisen taulun kentistä täytyy löytyä avaimeksi sopiva yksilöivä ominaisuus. Tässä tietokannassa uniikki perusavain on jokaisen taulun ID-numero, koska se on määritelty automaattisesti kasvavaksi, tietokanta huolehtii, että missään taulussa ei esiinny kahta kertaa samaa ID-numeroa.

Seuraavaksi selvitetään taulujen väliset suhteet. Tässä taulujen välillä tarvitaan esimerkiksi seuraavanlaisia suhteita:

- Jokaisella laboratoriolla on yksi sijaintipaikka ja yksi vastuuhenkilö.
- Yhdellä vastuuhenkilöllä voi olla useampi laboratorio samanaikaisesti.
- Komponentit voivat sijaita useammassa laboratoriossa.
- Kukin testauslaite voi sijaita vain yhdessä laboratoriossa.
- Kullakin testauslaitteella voi olla useampi huolto- ja vuokraustieto, mutta vain yksi vuokraaja kerrallaan.

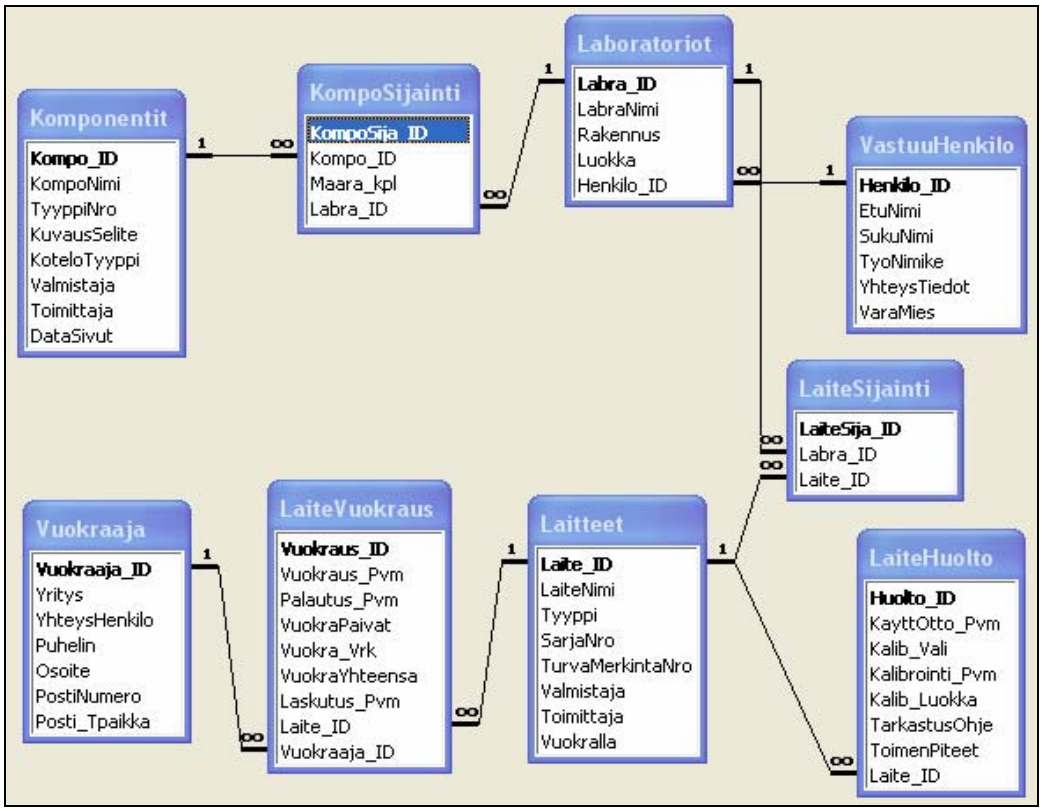
Taulujen välisiä suhteita ovat yhden suhde yhteen (1 to 1), yhden suhde moneen (1 to n) ja monen suhde moneen (n to n). Esimerkiksi tämän järjestelmän tietokannassa yhdessä laboratoriossa on monta laitetta, mutta yksi laite voi kuulua vain yhteen laboratorioon. Kyseessä on siis yhden suhde moneen (1 to n). Tai yhdessä laboratoriossa on monenlaista komponenttia ja yhdenlainen komponentti voi olla useammassa laboratoriossa, suhde on siis moni moneen (n to n). [2.]

Tietokannan suunnittelussa on tärkeää muistaa myös normalisointi. Sillä pyritään välttämään tietojen toistamista ja siitä seuraavia ongelmia tietokannan tietojen käsittelyssä. Normalisoinnilla pyritään myös lisäämään tietokannan rakenteen selkeyttä, yhtenäisyyttä ja laajennettavuutta. Tämä tietokanta on normalisoitu kolmanteen normaalimuotoon, joten toisteisuutta ei pitäisi esiintyä.

Tietokannanhallintajärjestelmäksi on valittu MySQL-relaatiotietokantaohjelma, versio 4.1.9. Kehitystyötä varten on suunniteltu testaustietokanta ja siihen tarvittavat taulut Access-tietokantaohjelmalla, joiden avulla testikanta on alustettu tes-

tidatalla ja testattu tietokannan relaatioyhteyksien toiminta. Testitietokannan hakujen onnistuttua ja yhteyksien toimiessa moitteettomasti luotiin varsinainen tietokanta MySQL-tietokantaohjelmalla.

Kuvassa 6 näkyvät testauslaboratorioiden tietokannan relaatiomalli, ylläpitoon tarvittavat taulut ja niiden väliset yhteydet. Tässä yhteydessä on myös käytetty Access-tietokantaohjelmaa, koska vastaavaa ominaisuutta ei käytettävässä MySQL-ohjelmassa ole. Yhteensä kannassa on 9 taulua. Liitteessä A on esitetty tietokannan ER-kaavio sekä siitä puretut relaatiotaulut.

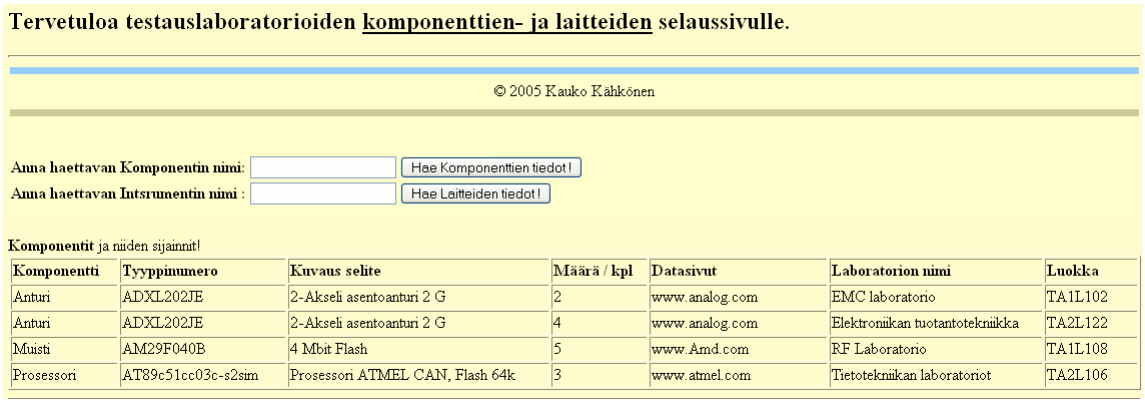


Kuva 6. Tietokannan relaatiomalli, taulujen sisältö sekä yhteydet.

5.3 Käyttöliittymän toteutus

Palvelinsivujen toteuttaminen on aloitettu selvittämällä, mitä kaikkea niiden avulla on tarkoitus tehdä. Tästä syntyi alustava käsitys tarvittavista sivuista ja sivuille tulostettavista tiedoista. Sivuja on kahta eri tyyppiä. Toinen on ”oppilaat” -sivu, jolle pääsevät kaikki Kajaanin ammattikorkeakoulun tietoverkkoon kirjautuneet

henkilöt. Sillä selataan tietokannan sisältöä, etsitään määriteltyjen kenttien tieto- ja joko komponenteista tai testauslaitteista hakusanan perusteella (kuva 7).



Kuva 7. Käyttöliittymän selaussivu.

Toiset sivut muodostavat Kajaanin ammattikorkeakoulun laboratorioiden vastuuhenkilöille tarkoitetun käyttöliittymän tietokannan hallintaan. Sen käyttöoikeutta on rajoitettu tietokantaan tallennettujen käyttäjätunnusten ja salasanojen avulla. Ylläpitosivuille päästään ainoastaan kirjoittamalla käyttäjätunnus ja salasana. Jos ne ovat oikein, avautuu toiminnon valintasivu (kuva 8), josta valitaan, mitä halutaan tehdä.

Kuva 8. Ylläpidon toiminnon valintasivu.

Käyttöliittymän sivut muodostuvat suurimmaksi osaksi erilaisista HTML-lomakkeista ja listaussivuista. Niillä voidaan lisätä, muokata ja poistaa laboratorioita, vastuuhenkilöitä, komponentteja, testauslaitteita sekä testauslaitteiden

huolto-, vuokraus- ja vuokraajatietoja. Tarkoitus on, että järjestelmän tietokantaa voidaan täysin ylläpitää näiden sivujen avulla eikä MySQL-tietokantaan tarvitse koskea suoraan. Näin voidaan varmistaa myös se, että tietokannan tietoja käsitellään oikein.

Sovelluslogiikka on toteutettu pääosin PHP-ohjelmointikielen versiolla 4.3.10, jonka avulla hoidetaan tarvittavat tietokantaoperaatiot ja istunnon hallinta. Käyttöliittymätaso on toteutettu HTML-kielellä, jota useimmat selaimet tukevat. WWW-palvelimeksi on valittu Apachen www-palvelinohjelmaversio 1.3.33, joka sijaitsee sovelluslogiikkatasolla ja joka tulkkaa PHP-koodin käyttöliittymätasolle selainyhteensopivaksi ainakin Mozilla- ja Internet Explorer selaimien ymmärtämään muotoon.

Sivuja toteutettaessa on pyritty selkeään, yksinkertaiseen ja yhtenäiseen ulkoasuun. Lisäyssivun väriksi on valittu vihreä eli aja, muokkaussivulla väri on sininen, ei suoraan myöntävä vihreä eikä kieltävä punainen, mutta siltä väliltä. Muilla sivuilla väri on valittu sattumanvaraisesti. Sivujen ulkoasut eivät ole lopullisia vaan työn tilaaja voi muokata sivujen ulkoasua vapaasti työn luovutuksen jälkeen.

6 TIETOKANNAN TESTAUS JA WWW-SIVUJEN TOIMINNOT

Tietokannan relaatiotaulut ja niiden väliset yhteydet ja toimivuus on luotu ja testattu Microsoft Officen Access-tietokantaohjelmalla. Hakujen ja muun toiminnallisuuden toiminnan jälkeen on tietokanta ja sen taulut luotu MySQL v4.3.10 -tietokannan hallintajärjestelmällä. Kuhunkin toimintoon tarkoitetun koodin kirjoittamisen jälkeen on toiminto käyttöliittymäsivulta testattu ja todettu toimivaksi.

6.1 Tietokantaan lisääminen

Tietokantaan lisäämissivuille tulostetaan kunkin taulun kaikki muut kentät paitsi ID-kenttää. Komponentteja ja testauslaitteita lisättäessä tulostetaan myös taulun nimi näkyviin, että tiedetään, mihin tauluun ollaan lisäämässä. Tietojen kirjoittamisen jälkeen tiedot tallennetaan tauluun ”Tallenna” -napin painalluksella.

6.2 Tietokannan listaus

Tietokannan listaussivulla listataan komponenttien tai testauslaitteiden tiedot sijainnin- ja ID-numeron mukaan joko annetun nimen perusteella tai jos hakuehtoa ei anneta, listataan kaikki taulun tiedot. Sijainnin listauksessa listataan myös laboratorion nimi, missä kukin komponentti tai testauslaite sijaitsee. Erillisessä ID-numeron listauksessa ID-numeron lisäksi - jota ei muissa listauksissa näytetä - listataan vain tärkeimmät tiedot komponentista tai testauslaitteesta, että lisäyksen jälkeen voidaan tuote sijoittaa oikeaan paikkaan.

6.3 Tietokannan muokkaaminen, päivitys ja tietokannasta poistaminen

Tietokannan muokkaussivulla tulostetaan muokattavan taulun, joko komponenttien tai testauslaitteiden, muokattavissa olevien kenttien tiedot kaikki tai hakuehdon mukaisesti. Mikäli tarkoitus on muuttaa jonkin kentän tietoa, se tapahtuu uuden tiedon kirjoittamisen jälkeen ”Päivitä” -napin painalluksella tai jos halutaan poistaa kaikki tiedot, painetaan ”Poista” -nappulaa, jolloin koko rivi ja sen tiedot häviävät, myös tauluista *Sijainti* ja *Laboratoriot* Sijainti-taulujen kautta.

6.4 Tietokannan tietojen seuranta

Tietokannan tietojen seurantasivut koskevat laboratorioita, vastuuhenkilöitä ja testauslaitteiden huolto- ja vuokraustietoja. Näille tiedoille ei ole tehty erikseen listaussivuja - oletuksena, että tietorivejä on sen verran vähän - mutta lisääminen, muokkaaminen, päivitys ja poistaminen tapahtuu saman sivuston sisällä.

7 YHTEENVETO

Insinöörityön lopputuloksena syntyi aiheen suunnittelun mukainen selaus- ja hallintajärjestelmä testauslaboratorioiden komponenteille ja testauslaitteille. Pohjatyö eli vaatimusten määrittely- ja projektisuunnitteluosuudet sekä tekninen ohjelmistosuunnitteluratkaisu on tehty koko sovelluksen osalta, ja niihin tuskin tulee enää suuria muutoksia. Sovelluksen alla toimiva tietokanta on saatu käyttökelpoiseen kuntoon tietokannan selausta ja ylläpitoa varten. Suurena apuna tietokannan suunnittelussa oli se, että oppitunneilla on käyty perusteet Access- ja MS SQL-Server 2000-tietokannoista. Tosin silloin on ollut kyse pienemmistä kannoista - tässä työssä tietokannan koko ja siihen tallennettavien asioiden ja ominaisuuksien määrä oli nyt laajempi - mutta periaate on selvinnyt.

Ongelmat työn tekemisessä liittyivät enemmänkin aikatauluihin. Ohjeita käyttämälleni versiolle oli hankala löytää, joten täytyi tyytyä vanhempien versioiden ohjeisiin ja soveltaa niitä kokeilun kautta. Tarkemmat ohjeet löytyvät vain ranskan kielellä. Toiseksi ongelmaksi meinasi osoittautua se, että sivut tulostuvat eri tavalla Internet Explorer- kuin Mozilla Firefox -ohjelmassa. Lisäksi Internet Explorer-ohjelmalla selattaessa tiedot tulevat näkyviin vasta sivun päivityksen jälkeen.

Aloitin työn koodaamisen perinteistä PHP-koodaustekniikkaa käyttäen, joten jatkoin vastaavalla tavalla loppuun saakka. Toinen mahdollisuus olisi ollut käyttää olioihin perustuvaa koodaustekniikkaa, jonka oivalsin vasta myöhäisessä vaiheessa työn edetessä enkä uskaltanut vaihtaa enää tekniikkaa työn ja opiskelun viivästymisen pelossa. Yksi järjestelmän kehittämissuunta olisi muuttaa järjestelmä kokonaan oliopohjaiseksi, joka ehkä parantaisi tiedostojen hallittavuutta. Myös sivujen ulkoasu voisi olla yksi tiedosto, jota kutsutaan aina kyseisen toiminnon sivun alussa.

Tietokannan toimintoja voisi vielä kehittää siltä osin, että kun *LaiteVuokraus* taulun *Vuokraus_pvm* kenttään lisätään päivämäärä, päivittyisi *Laitteet* taulun *Vuokralla* kenttä automaattisesti boolean arvolla *TRUE* ja päinvastoin, kun *Palautus_pvm* kenttään lisätään päivämäärä, päivittyisi *Vuokralla* kenttä automaattises-

ti boolean arvolla *FALSE*. Tätä automaattinen ominaisuutta ei vaatimuksissa ole määriteltykään, se puuttuu käyttämästäni tietokannan hallintajärjestelmästä ja lisäksi se vaatisi syvempää perehtymistä nimenomaan SQL-ohjelmoinnin koodaamisen syövereihin.

Hakutoimintoja voidaan muuttaa kooditasolla, kunhan vain ensin selvitetään, minkälaisista tiedoista on asiakkaille todellista hyötyä ja mikä on vain tiedon turhaa esittämistä. Sivujen ulkoasu on pyritty tekemään selkeäksi ja toimivaksi ottaen huomioon, että jos listataan esimerkiksi kaikki komponentit, listasta voi tulla hyvinkin pitkä ja sitä joudutaan rullaamaan sivun vierityspalkista. Toki listaaminen onnistuu myös hakusanan perusteella, missä listataan vain annetun hakusanan täyttävät ehdot. Käytettävyyttä ei ole ehditty vielä testata kovinkaan paljon, mutta todellinen käytettävyyden arviointi ja täydellisen toimivuuden varmistaminen pystytään tekemään kunnolla vasta, kun tietokanta on asennettu lopulliseen käyttöympäristöönsä. Kehitysympäristössä kaikki tasot sijaitsevat samalla kovalevyllä.

Tietokantasuunnittelun ja ohjelmoinnin kannalta insinöörityön tekeminen oli hyödyllinen ja mielenkiintoinen kokemus. Insinöörityöhön liittyvien ohjelmien valinnan syynä oli opetella täysin uudet ohjelmat. Opittuja taitoja ja ohjelmia voi hyödyntää myöhemminkin, mikäli vastaavanlaisia ohjelmistoja meinaa suunnitella ja toteuttaa kaupallisessa tarkoituksessa. Ei tarvitse heti alkuun satsata kalliisiin ohjelmistoihin, koska ainakin vielä toistaiseksi ohjelmistopaketti on imuroitavissa ilmaiseksi internetissä. Joihinkin osioihin on kyllä olemassa lisenssimaksu, jos ohjelmaa käytetään kaupallisesti, mutta se maksu ei ole suuri. Tästä on hyvä jatkaa.

LÄHDELUETTELO

1 Hovi, Huotari, Lahdenmäki. Tietokantojen suunnittelu & indeksointi. Ensimmäinen painos. Porvoo: WS Bookwell, 2003. ISBN 951-846-178-3.

2 Meloni, J. MySQL - Trainer Kit. Ensimmäinen painos. Helsinki: Edita Prima Oy, 2003. ISBN 951-826-698-0.

3 Software Business Competence. Web-sovellusten ohjelmointi. [WWW-sivusto] (3.9.2005) <http://www.oamk.fi/sbc/www/johdanto.php#websovellukset>

4 JYU, Revonkorpi, M. Käyttöliittymän visuaalinen suunnittelu. [WWW-sivusto] (3.9.2005) <http://www.mit.jyu.fi/~vesal/kurssit/winohj/kaytto/minja/luento/luentomatskua.htm>

5 JYU, Lahtonen, T. Tietokannat. [WWW-sivusto] (30.8.2005) <http://appro.mit.jyu.fi/doc/tiedonhallinta/tietokannat/index.html>

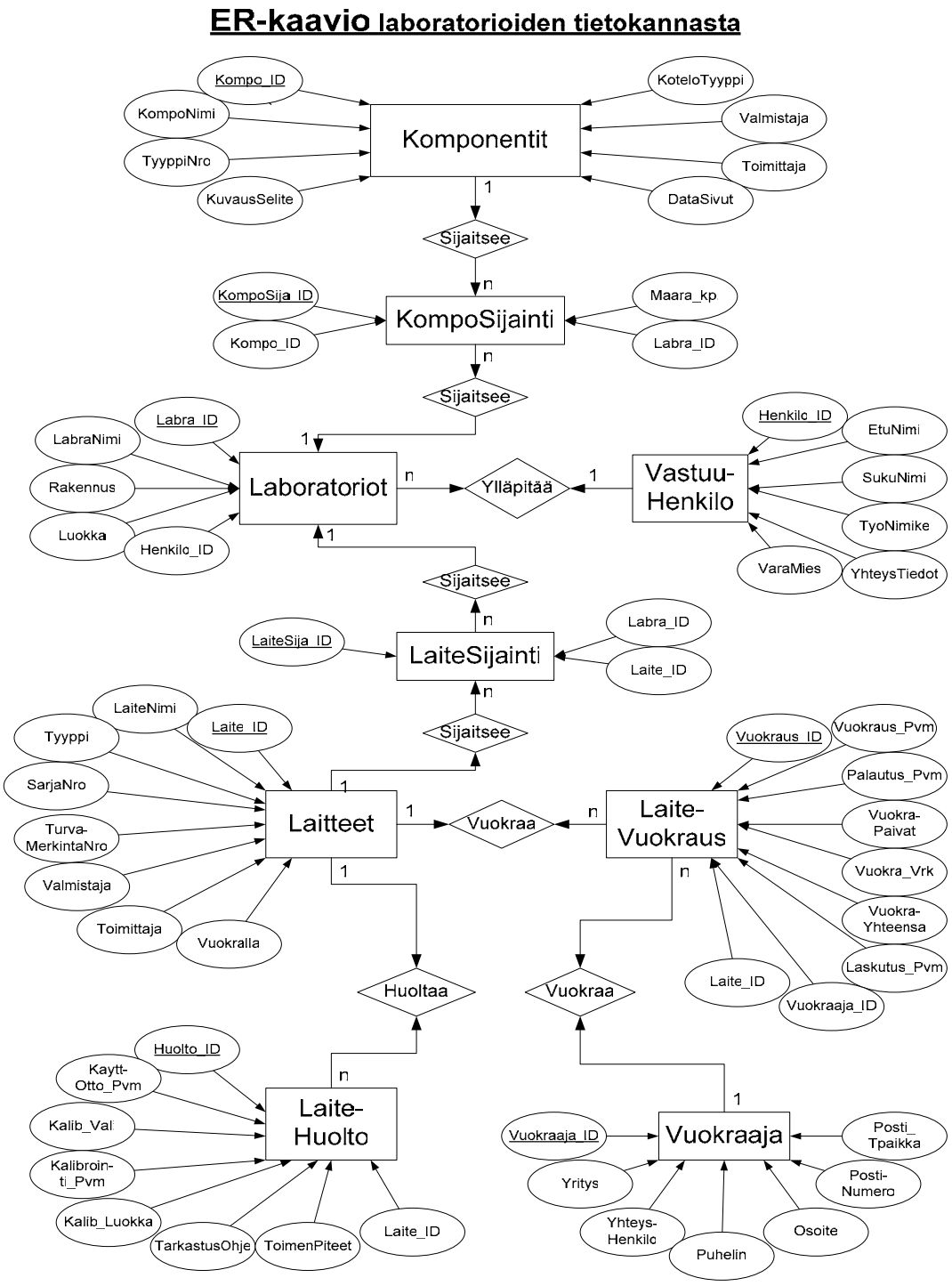
6 Rantala, A. PHP - Web-ohjelmoinnin peruskirja. Ensimmäinen painos. Porvoo: WS Bookwell, 2002. ISBN 951-846-147-3.

7 Heinisuo, R., Talentum Media Oy. PHP ja MySQL - Tietokantapohjaiset verkkopalvelut. Kolmas, uudistettu painos. Jyväskylä: Gummerus Kirjapaino Oy, 2004. ISBN 952-14-0847-2.

LIITTEET

Liite A ER-kaavio ja relaatiotaulut

Kuvassa 1 on kuvattu testauslaboratorioiden tietokantaa ER-kaaviolla.



Kuva 1. Testauslaboratorioiden tietokannan ER-kaavio.

Relaatiotaulut

Kuvassa 1 oleva ER-kaavio on purettu alla olevissa taulukoissa relaatiotauluiksi.

Taulukko1. Laboratoriotiedot.

Laboratoriot	Tietotyyppi	Pakollinen	Oletusarvo	Rajoitus
Labra_ID	luku	kyllä	ei	uniikki
LabraNimi	teksti	kyllä	ei	40 merkkiä
Rakennus	teksti	kyllä	ei	20 merkkiä
Luokka	teksti	kyllä	ei	20 merkkiä
Henkilo_ID	luku	kyllä	ei	uniikki

Taulukko2. Laboratorion vastuuhenkilön tiedot.

VastuuHenkilo	Tietotyyppi	Pakollinen	Oletusarvo	Rajoitus
Henkilo_ID	luku	kyllä	ei	uniikki
EtuNimi	teksti	kyllä	ei	20 merkkiä
SukuNimi	teksti	kyllä	ei	30 merkkiä
TyoNimike	teksti	kyllä	ei	30 merkkiä
YhteysTiedot	teksti	kyllä	ei	30 merkkiä
VaraMies	teksti	kyllä	ei	30 merkkiä

Taulukko3. Komponentit.

Komponentit	Tietotyyppi	Pakollinen	Oletusarvo	Rajoitus
Kompo_ID	luku	kyllä	ei	uniikki
KompoNimi	teksti	kyllä	ei	40 merkkiä
TyyppiNro	teksti	kyllä	ei	30 merkkiä
KuvausSelite	teksti	ei	ei	50 merkkiä
KoteloTyyppi	teksti	ei	ei	20 merkkiä
Maara_kpl	luku	kyllä	ei	6 merkkiä
Valmistaja	teksti	kyllä	ei	40 merkkiä
Toimittaja	teksti	kyllä	ei	40 merkkiä
DataSivut	teksti	ei	ei	50 merkkiä

Taulukko4. Komponenttien sijaintitiedot.

KompoSijainti	Tietotyyppi	Pakollinen	Oletusarvo	Rajoitus
KompoSija_ID	luku	kyllä	ei	uniikki
Kompo_ID	luku	kyllä	ei	uniikki
Labra_ID	luku	kyllä	ei	uniikki

Taulukko5. Laitetiedot.

Laitteet	Tietotyyppi	Pakollinen	Oletusarvo	Rajoitus
Laite_ID	luku	kyllä	ei	uniikki
LaiteNimi	teksti	kyllä	ei	30 merkkiä
Tyyppi	teksti	kyllä	ei	20 merkkiä
SarjaNro	teksti	kyllä	ei	30 merkkiä
TurvaMerkintaNro	teksti	kyllä	ei	20 merkkiä
Valmistaja	teksti	kyllä	ei	40 merkkiä
Toimittaja	teksti	kyllä	ei	40 merkkiä
Vuokralla	boolean	kyllä	false	true / false

Taulukko6. Laitteiden sijaintitiedot.

LaiteSijainti	Tietotyyppi	Pakollinen	Oletusarvo	Rajoitus
LaiteSija_ID	luku	kyllä	ei	uniikki
Labra_ID	luku	kyllä	ei	uniikki
Laite_ID	luku	kyllä	ei	uniikki

Taulukko7. Laitteiden huoltotiedot.

LaiteHuolto	Tietotyyppi	Pakollinen	Oletusarvo	Rajoitus
Huolto_ID	luku	kyllä	ei	uniikki
KayttOtto_Pvm	päivämäärä	kyllä	ei	10 merkkiä
Kalib_Vali	teksti	ei	ei	10 merkkiä
Kalibrointi_Pvm	päivämäärä	kyllä	ei	50 merkkiä
Kalib_Luokka	boolean	kyllä	ei	true / false
TarkastusOhje	teksti	ei	ei	30 merkkiä
ToimenPiteet	teksti	kyllä	ei	60 merkkiä
Laite_ID	luku	kyllä	ei	uniikki

Taulukko8. Laitteiden vuokraustiedot.

LaiteVuokraus	Tietotyyppi	Pakollinen	Oletusarvo	Rajoitus
Vuokraus_ID	luku	kyllä	ei	uniikki
Vuokraus_Pvm	päivämäärä	kyllä	ei	30 merkkiä
Palautus_Pvm	päivämäärä	kyllä	ei	30 merkkiä
VuokraPaivat	luku	kyllä	ei	5 merkkiä
Vuokra_Vrk	luku	kyllä	ei	6 merkkiä
VuokraYhteensa	luku	kyllä	ei	6 merkkiä
Laskutus_Pvm	päivämäärä	kyllä	ei	30 merkkiä
Laite_ID	luku	kyllä	ei	uniikki
Vuokraaja_ID	luku	kyllä	ei	uniikki

Taulukko9. Laitteita vuokraavan yrityksen tiedot.

Vuokraaja	Tietotyyppi	Pakollinen	Oletusarvo	Rajoitus
Vuokraaja_ID	luku	kyllä	ei	uniikki
Yritys	teksti	kyllä	ei	40 merkkiä
YhteysHenkilo	teksti	kyllä	ei	40 merkkiä
Puhelin	luku	ei	ei	30 merkkiä
Osoite	teksti	kyllä	ei	30 merkkiä
PostiNumero	luku	kyllä	ei	5 merkkiä
Posti_Tpaikka	teksti	kyllä	ei	30 merkkiä