Opinnäytetyö (AMK)

Tietojenkäsittelyn koulutusohjelma

Tietojärjestelmät

2010

Janina Puistovaara

RELAATIOTIETOKANNAN SUUNNITTELU JA MALLINTAMINEN KÄSITEANALYYSI-MENETELMÄLLÄ

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietojenkäsittelyn koulutusohjelma | Tietojärjestelmät

12.5.2010 | Sivumäärä 48 + 2 liitettä

Ohjaaja Anne Jumppanen

Tekijä Janina Puistovaara

RELAATIOTIETOKANNAN SUUNNITTELU JA MALLINTAMINEN KÄSITEANALYYSIMENETELMÄLLÄ

Opinnäytetyön tarkoituksena on toteuttaa Turun seudun lihastautiyhdistys ry:lle tietokanta, joka tukee yhdistyksen toimintoja. Tarkempana tutkimuskohteena ovat relaatiotietokannat sekä niiden suunnittelu ja mallintaminen.

Opinnäytetyössä esitellään relaatiomalli, joka antaa mallin relaatiotietokantojen rakenteelle, käsittelylle ja eheyssäännöille, ja jonka toiminta-ajatus perustuu joukko-oppiin. Samalla tarkastellaan myös relaatiotietokantojen kehitystyötä.

Relaatiotietokannan suunnittelu aloitetaan opinnäytetyössä käsiteanalyysimenetelmällä, jonka tarkoituksena on etsiä kohdealueen keskeisimmät käsitteet ja ominaisuudet sekä tarkastella käsitteiden välisiä yhteyksiä. Käsiteanalyysiä laadittaessa tietokannan taulut myös normalisoidaan. Suunnittelun tuloksena syntyy ER-kaavio, jota voidaan pitää tietokannan pohjapiirustuksena. Opinnäytetyössä ER-kaavio mallinnetaan UML-kielellä, ja samalla tarkastellaan myös kyseisen notaation soveltuvuutta tietokantojen mallintamiseen.

Syntyneen ER-kaavion perusteella tietokannalle luodaan taulut, joiden sarakkeille asetetaan soveltuvat tietotyypit ja niille luodaan indeksit. Opinnäytetyön tuloksena syntyy looginen ja joustava tietokanta, joka soveltuu hyvin Turun seudun lihastautiyhdistys ry:n käyttöön.

ASIASANAT:

Relaatiomalli, tietokanta, relaatiotietokanta, UML, käsiteanalyysi, ER-kaavio, SQL, MySQL

BACHELOR´S THESIS | ABSTRACT

UNIVERSITY OF APPLIED SCIENCES

Information Technology | Information systems

12 May 2010 | Total number of pages 48 and 2 attachments

Instructor Anne Jumppanen

Author Janina Puistovaara

DESIGN AND MODELLING OF RELATIONAL DATABASES

The purpose of this thesis was to design a database for Turun seudun lihastautiyhdistys ry (the Muscular Disease Association of Turku) and to study the designing and modelling of relational databases.

This thesis introduces databases which are based on the relational model of data. The relational model describes the structure, usage and integrity of databases and it is based on the mathematical concept of relation.

The first step of database design is to name the most important entity sets and attributes and to set the relationships among two or more entity sets. This results in an E/R diagram which is a graph representing those element types. This thesis uses UML, Unified Modeling Language, for modelling E/R diagrams.

After the E/R diagram has been completed, the entity sets, attributes and relationships are converted into tables and columns, and the data types and indexes for the columns are set. The final result of this thesis is a logical and flexible database which is suitable for the Turun seudun lihastautiyhdistys ry.

KEYWORDS:

Relational model, database, relational database, UML, entity set, E/R diagram, SQL, MySQL

SISÄLTÖ

KUVAT

LIITTEET

1 JOHDANTO

1.1 Toimeksiantajana Turun seudun lihastautiyhdistys ry

Tekniikan kehittyessä monella organisaatiolla ja yrityksellä on ollut vaikeuksia päivittää tietojenkäsittelyään mahdollisimman tuottavalle tasolle. Vanhanaikaisista tavoista, kuten paperiarkistoista, muistilapuista ja muistilehtiöistä on yllättävän vaikea päästä eroon, jos resursseja ei ole riittävästi tietojenkäsittelyn uudistamiseksi. Tämän ongelman kohtasi myös Turun seudun lihastautiyhdistys ry.

Tämän opinnäytetyön aiheena on tehdä tietokanta Turun seudun lihastautiyhdistys ry:n (myöhemmin Lihastautiyhdistys) tulevaan tietojärjestelmään. Kyseessä on yhdistys, joka tukee ja auttaa erilaisia lihastauteja sairastavia ihmisiä. Tämän lisäksi heidän kohderyhmänään ovat myös kehitys-, näkö- ja kuulovammaiset, jotka tarvitsevat apua erilaisiin arjen tilanteisiin.

Lihastautiyhdistys on perustettu vuonna 1981. Aluksi heillä oli kirjoilla vain muutama jäsen ja heidän toimintansa keskittyi pelkästään Turkuun, mutta vuosien aikana he ovat laajentaneet toimintaansa myös muualle Varsinais-Suomeen, Satakuntaan ja Ahvenanmaahan.

Yhdistyksen toiminnan tärkein osa-alue on Avustajakeskus, joka välittää ja kouluttaa avustajia apua tarvitseville ihmisille erilaisiin arjen tilanteisiin. Avustajakeskus toimii yhteistyössä kuntien vammaisjärjestöjen, sosiaali-, terveys- ja liikuntatoimien, seurakuntien sekä muiden julkisen sektorin tahojen kanssa.

Avustajakeskuksen listoilla on sekä palkallisia että palkattomia avustajia. Avustajien toimeenkuvaan kuuluu tukea avustettavia heidän jokapäiväisessä elämässään osallistumalla muun muassa harrastuksiin ja päivittäisten

ruokatarpeiden hankkimiseen. Kenellä tahansa apua tarvitsevalla on varaa avustajaan, sillä vapaaehtoisen avustajan saa korvaamalla ainoastaan matkakustannukset. Heidän asiakasmääränsä onkin kasvanut näiden vuosien aikana jo moninkertaisesti.

Asiakasmäärän kasvaminen on tuonut mukanaan kuitenkin omat ongelmansa, sillä heidän nykyiset tietojärjestelmänsä ovat auttamatta vanhentuneet. Avustajavälitys toimii lähes kokonaan pelkkien puhelinsoittojen, sähköpostien, paperikalenterien ja jäsenten oman panostuksen varassa. Nämä toimintamallit eivät enää riitä kattamaan tätä jatkuvasti laajentuvaa toimintaa. Jotta he pystyisivät jatkossakin täydentämään avustettaviensa tarpeet, heidän on päivitettävä tietojärjestelmänsä vastaamaan nykyisiin ja tuleviin vaatimuksiin. Koska avustajakeskus on keskeisin osa yhdistyksen toimintaa, tämän opinnäytetyön tuloksena syntyvä tietokanta on suunniteltu täyttämään nimenomaan avustajakeskuksen tarpeet.

1.2 Projektin jakaminen

Turun seudun lihastautiyhdistys ry:n tarvitsema tietojärjestelmä on niin laaja, ettei yksi opiskelija pysty toteuttamaan sitä opinnäytetyönään. Tämän vuoksi tietojärjestelmäprojekti jakautuu useampaan osaan.

Niko Skogström teki keväällä 2009 omana opinnäytetyönään selvityksen siitä, minkälaisen järjestelmän Lihastautiyhdistys tarvitsisi nykypäivänä. Hänen opinnäytetyössään on monia hyviä huomioita sekä kattava määrittely yhdistyksen tarpeista ja toiminnoista. Tämä opinnäytetyö jatkaa Skogströmin työtä, ja myöhemmin myös yksi tai useampi muu opiskelija tulee ottamaan osaa yhdistyksen tietojärjestelmän suunnittelu- ja toteutustyöhön.

Tämän projektiosa-alueen ja siitä syntyvän opinnäytetyön teoreettinen viitekehys rajautuu relaatiotietokantoihin sekä niiden suunnitteluun ja toteutukseen. Tarkoituksena on tutkia, kuinka rakennetaan selkeä, looginen ja

kaikin tavoin luotettava tietokanta. Empiirisen osan tarkoituksena on toteuttaa suunnittelun tuloksena syntynyt tietokanta, joka soveltuu parhaalla mahdollisella tavalla juuri Lihastautiyhdistyksen käyttöön.

1.3 Ongelmia järjestelmän suunnittelussa

Uuden järjestelmän toteuttamisessa on kuitenkin yksi suuri ongelma: yhdistys on vielä tälläkin hetkellä niin suuressa kehitysvaiheessa, että uuden järjestelmän toteuttaminen on vaikeaa jatkuvasti muuttuvien vaatimusten vuoksi. Niko Skogström teki pohjan järjestelmän suunnittelulle ja toteutukselle, mutta jo näiden kahden opinnäytetyön välisinä kuukausina yhdistyksen toiminnassa on ehtinyt tapahtua niin monia muutoksia, että Skogströmin määrittelyyn tulee tämän opinnäytetyön kautta useita tarkennuksia ja jopa selviä muutoksia. Tämä tuo mukanaan selviä haasteita tietojärjestelmän suunnitteluun sekä sen taustalla olevan tietokannan toteutukseen. Visio yhdistyksen tulevaisuudesta on kuitenkin jo selvillä, joten suunnitteleminen ei ole mahdotonta.

2 PROJEKTIN TAUSTA

2.1 Toiminnallinen ja tekninen määrittely

Järjestelmän toiminnallinen ja tekninen määrittely on tärkeä osa ohjelmistotuotantoa. Toiminnallisella määrittelyllä tarkoitetaan dokumenttia, joka syntyy ohjelmistotuotannon määrittelyvaiheessa. Dokumentissa analysoidaan asiakasvaatimukset, jotka on tarkoitus muuttaa täsmällisiksi ohjelmistovaatimuksiksi. Siinä kuvataan järjestelmän toiminnot, kuten ohjelmistolla toteutettavat ominaisuudet, käyttöliittymät ja kommunikointi muiden järjestelmien kanssa. Lisäksi toiminnallisessa määrittelyssä kuvataan ei-toiminnalliset vaatimukset, kuten vasteajat, käytettävyys ja rajoitukset. Rajoituksia ovat muun muassa tietyn ohjelmointikielen valitseminen ja muistin määrä. (Haikala & Märijärvi 2004, 38-39.)

Suunnitteluvaiheessa määrittelyn kuvaamien toimintojen toteutus suunnitellaan. Aluksi järjestelmä jaetaan itsenäisiin toisistaan riippumattomiin osiin, joita kutsutaan moduuleiksi. Tätä vaihetta kutsutaan arkkitehtuurisuunnitteluksi ja siitä syntyvää dokumenttia tekniseksi määrittelyksi. (Haikala & Märijärvi 2004, 40.) Teknisessä määrittelyssä käsitellään pääasiassa moduulien välisen työnjaon ja rajapintojen suunnittelua sellaisiksi, ettei yksittäisen moduulin sisällä tehtävät muutokset vaikuta toisiin moduuleihin. (Haikala & Märijärvi 2004, 80.)

Lyhyesti voidaan sanoa, että toiminnallisessa määrittelyssä kuvataan, *mitä* järjestelmä tekee. Teknisessä määrittelyssä puolestaan kerrotaan, *miten* järjestelmä sille annetut tehtävänsä suorittaa. (Haikala & Märijärvi 2004, 40.)

Niko Skogström laati omana opinnäytetyönään IEEE 830 -standardin mukaisen määrittelydokumentin Lihastautiyhdityksen tietojärjestelmästä. Teknistä määrittelyä hän ei laatinut, koska tarkoituksena oli jättää teknisen määrittelyn laatiminen opinnäytetyön jatkajalle.

2.2 Muuttuneet asiakasvaatimukset

Niko Skogström laatimassa määrittelydokumentissa määriteltiin, että yhdistys tarvitsee järjestelmän, joka sisältää henkilörekisteriosion avustajien ja asiakkaiden tietoja varten, ajanvarausosion, johon yhdistyksen jäsenet voivat käydä kirjaamassa avustuskeikkojen tietoja, ohje- ja lomakepankkiosion lomakkeiden hallintaan, yhteistyökumppanit-osion, koulutusosion sekä hallintaosion käyttäjätunnusten hallintaan (Skogström 2009, 9-13). Lisäksi Skogström määritteli, että ohjelmiston käyttäminen vaatisi Windows XP:n tai uudemman käyttöjärjestelmän sekä ohjelman asentamisen jokaiselle tietokoneelle, jolla ohjelmistoa haluttaisiin käyttää. Ohjelmistoa voitaisiin käyttää paikallisesti tai internetin kautta. (Skogström 2009, 7.)

Yhdistyksen asiakasvaatimukset ovat kuitenkin tässä välissä jo muuttuneet.

Suurin muutos Skogströmin laatimaan määrittelyyn on se, että ajatus Windows-pohjaisesta sovelluksesta on nyt muuttunut selainpohjaiseen intranet-ratkaisuun, joka toimii yhdistyksen omalla tai vaihtoehtoisesti jonkin palveluntarjoajan palvelimella. Kyseinen ratkaisu on ihanteellinen moneltakin kannalta, sillä intranet-ratkaisussa avustajien ei tarvitse asentaa tietokoneilleen ylimääräisiä sovelluksia, mutta jokainen avustaja pääsee omilla käyttäjätunnuksillaan katsomaan esimerkiksi omien keikkojensa tietoja. Lisäksi intranet-ratkaisu on laajennettavissa yhdistyksen kotisivuihin: järjestelmän ajanvarausosioon voidaan esimerkiksi lisätä ominaisuus, jonka kautta asiakkaat voivat itse käydä varaamassa keikan helposti Internetin kautta.

Koska määrittelyn sisältö on muuttunut, standardin IEEE 830 mukaisesta määrittelydokumentista on laadittu uusi versio, joka on tämän opinnäytetyön liitteenä 2. Intranet-ratkaisun lisäksi määrittelyyn on lisätty tietokannan tiedot sekä järjestelmän lomakekäyttöliittymien suunnitelmat.

3 RELAATIOTIETOKANTOJEN ESITTELY

3.1 Relaatiomalli tietokantojen perustana

Tieto on yritysten ja organisaatioiden yksi tärkeimmistä resursseista. Tietokannat ovat tärkeä osa tiedon hallitsemista. (Hovi ym. 2005, 4.)

Tietokanta (database) voidaan määritellä loogisesti yhteenkuuluvien tallennettujen tietojen joukoksi. Hyvässä tietokannassa tiedot on tallennettu loogisessa muodossa siten, että ne on helposti tavoitettavissa ilman ristiriitaisuuksia. (Hovi ym. 2005, 4.)

Tietomalli (data model) on kuvausmenetelmä, jolla kuvataan tietorakenteita ja niiden välisiä yhteyksiä (Hovi ym. 2005, 4). Yksi tunnetuimmista tietomalleista on IBM:n tutkija E. F. Coddin vuonna 1970 kehittämä relaatiomalli (the relational model), johon kaikki relaatiotietokannat (relational database) perustuvat. Coddin relaatiomalli oli suuri edistysaskel tietokantojen maailmassa, sillä se syrjäytti

aikaisemmat hierarkkiset ja verkkomalliset tietokantatyypit. Relaatiotietokanta onkin nykyään käytetyin tietokantatyyppi. (Hovi 2004, 5.)

Relaatiomalli perustuu joukko-oppiin, matematiikkaan ja predikaattilogiikkaan. Se on perusta nykyisille relaatiotietokannoille ja antaa mallin rakenteelle, käsittelylle ja eheysäännöille ottamatta kantaa fyysiseen toteutukseen. (Hovi ym. 2005, 7-8.)

3.2 Tietokantojen hallintajärjestelmät

Tietokannassa olevia tietoja voidaan hallinnoida tietokannan hallintajärjestelmän (Database Management System) avulla. Nykyiset tietokantojen hallintajärjestelmät ovat pääosin SQL-pohjaisia relaatiotietokantoja. Ne ovat monimutkaisia sovelluksia, jotka auttavat tietokantojen ylläpitämisessä ja käyttämisessä. (Hovi ym. 2005, 4-5.)

Tietokannan hallintajärjestelmiä ovat muun muassa MySQL, DB2, Oracle ja SQL Server. Lisäksi on olemassa niin kutsuttuja henkilökohtaisia tietokantaohjelmia, kuten Microsoft Access, joita ei ole suunniteltu yhtä suurien tietomäärien tallentamiseen. (Lahtonen 2002, 8.)

Hallintajärjestelmät mahdollistavat, ettei tietokannassa ole toistoa tai ristiriitaisuuksia, toisin sanoen järjestelmä huolehtii tiedon eheydestä. Lisäksi hallintajärjestelmien avulla tietokanta voi olla useamman sovelluksen käytettävissä ja tietojen muutokset sekä lisäykset näkyvät heti kaikilla käyttäjillä. (Hovi ym. 2005, 4-5.) Muita tietokannanhallintajärjestelmien perusominaisuuksia ovat muun muassa tallennus-, haku- ja päivitysominaisuudet, tietoturvallisuus, tehokkuus ja skaalautuvuus. (Lahtonen 2002, 8.)

3.3 MySQL

MySQL-hallintajärjestelmää alettiin kehittää vuonna 1995 suomalaisen Michael Wideniuksen ja ruotsalaisen David Axmarkin toimesta. Vuonna 2008 yhdysvaltainen Sun osti MySQL AB:n saaden haltuunsa myös MySQL-

ohjelmiston. (Lehtinen 2008.) Nykyään MySQL on kuitenkin Oraclen omistuksessa, sillä huhtikuussa 2009 Oracle ilmoitti ostavansa Sunin (Moisio 2009).

MySQL on saatavilla ilmaiseksi GPL-lisenssin alaisuudessa (Moisio 2008). GPL-lisenssillä tarkoitetaan lisenssiä, joka on suunniteltu takamaan käyttäjän vapaudet jakaa ja muunnella ohjelmistoa. Se turvaa käyttäjien oikeuksia kahdella tavalla: (1) ohjelma suojataan tekijänoikeudella, ja (2) käyttäjille tarjotaan lisenssi, joka antaa luvan kopioida, levittää ja muokata ohjelmaa. (Free Software Foundation, Inc 2007.)

MySQL-sovellus on kuitenkin saatavissa myös maksullisena versiona niille yrityksille, joiden liiketoimintamalliin GPL ei sovellu (Moisio 2008). Maksullinen MySQL-lisenssi on hankittava, jos ohjelmistoa levitetään kaupallisesti (Heinisuo & Rauta 2007, 38).

Turun seudun lihastautiyhdistys ry:n tietokanta toteutetaan MySQL-ohjelmistolla lähinnä taloudellisista, mutta myös teknisistä ja ylläpidollisista syistä. MySQL:n valintaa puoltaa muun muassa se, että sovellusta on menestyksekkäästi käytetty erilaisissa selainpohjaisissa järjestelmissä jo vuosikymmenien ajan. Lisäksi MySQL on helppo asentaa ja se vaatii vain vähän ylläpitoa verrattuna kaupallisiin järjestelmiin. Ohjelmisto on myös todella suorituskykyinen, ja sitä voidaan käyttää useista ohjelmointikielistä, kuten C, C++ ja PHP-kielistä. (Heinisuo & Rauta 2007, 38.)

4 RELAATIOKANNAN PERUSELEMENTIT

4.1 Rakenne, käsittely ja eheys

4.1.1 Taulut

Relaatiomallin mukaisesti tietokannan perusta voidaan jakaa rakenteeseen,

käsittelyyn ja eheyssääntöihin. Tietokannan rakenteen peruselementtinä toimivat taulut (table), joista tietokannan rakenne koostuu. Taulut kuvaavat tietokannoissa loogisesti yhteenkuuluvia asiakokonaisuuksia, ja ne nimetään niiden mukaisesti. Taulujen tietosisältö koostuu sarakkeista ja riveistä, joita kutsutaan joskus myös tietueiksi. (Hovi ym. 2005, 8).

Taulujen sarakkeet (column) nimetään taulujen sisällä toisistaan poikkeavasti. Yksi sarake sisältää samaan arvojoukkoon kuuluvia tietoja. Tämä tarkoittaa, että sarakkeen arvoilla on yhteinen tietotyyppi, jolle on varattu yhteiset ominaisuudet. (Hovi ym. 2005, 8-9).

Rivit puolestaan koostuvat yhdestä tai useammasta sarakkeesta. (Lahtonen 2002, 7.) Yksi rivi sisältää yhden kohteen tiedot siitä asiakokonaisuudesta, jota taulu kuvaa.

4.1.2 Perus- ja viiteavaimet

Perusavain on yksi tai useampi taulun sarake, jonka avulla tietokannan hallintajärjestelmä pitää taulun järjestyksessä. Perusavaimen tarkoituksena on yksilöidä taulun sisältämät tietueet, joten sen on aina oltava yksilöivä eli uniikki. (Lahtonen 2002, 7.) Tämä tarkoittaa, ettei sarakkeessa saa olla kahdella tai useammalla rivillä samaa arvoa (Hovi ym. 2005, 9). Arvo ei saa myöskään olla tyhjä- eli NULL-arvo (Hovi ym. 2005, 11).

Viiteavain tarvitaan, kun kahden taulun välille halutaan luoda yhteys: liitos tehdään perusavaimen ja viiteavaimen välille. Käytännössä viiteavainsarake sisältää samat tiedot kuin perusavainsarake, mutta viiteavainsarakkeessa tiedot saavat toistua (Hovi ym. 2005, 9).

4.1.3 Eheyssäännöt

Perus- ja viiteavaimet ovat yhteydessä tietokannan eheyteen. Tietokanta on eheä, kun sen tiedot ovat oikein, ne vastaavat reaalimaailmaa eikä tiedoissa ole

ristiriitoja. Eheys voidaan osittain turvata eheyssäännöillä. Avaineheys tarkoittaa sitä, että perusavaimen arvon on oltava pakollinen. Toinen eheyssääntö on viite-eheys, jonka mukaan taulusta ei saa poistaa tietoja, joihin viitataan viiteavaimella jostakin toisesta taulusta. Muuten viite-eheys särkyisi (Hovi ym. 2005, 11).

4.1.4 Joukko-opillisuus ja SQL

Relaatiomallin perusidea on se, että tietoja käsitellään joukko-opillisesti: Taulu muodostuu rivijoukosta, johon voidaan kohdistaa erilaisia joukko-operaatioita. (Hovi ym. 2005, 10.)

Joukko-opillisuus parantaa tietokantojen tietoriippumattomuutta, sillä uusia tauluja voidaan lisätä tietokantaan ilman vaikutusta jo toteutettuihin sovelluksiin. Lisäksi rivien järjestyksellä tauluissa ei ole vaikutusta ohjelmien logiikkaan: rivit ovat käytännössä täysin mielivaltaisessa järjestyksessä. (Hovi ym. 2005, 12).

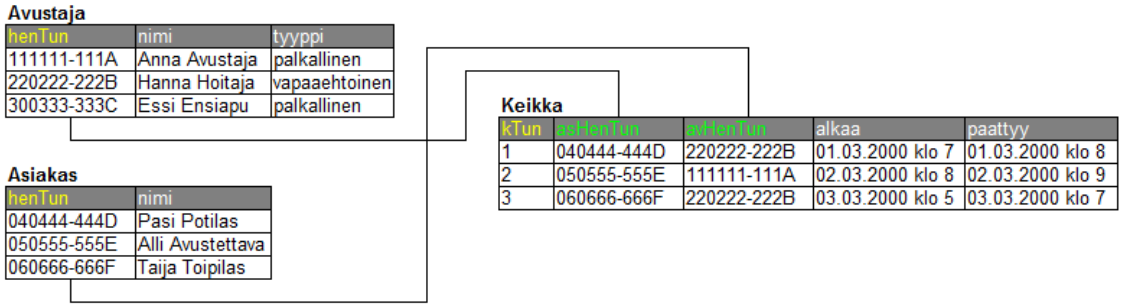
Relaatiotietokannoissa tietojen käsittely toteutetaan SQL-kielellä, joka on voimakas ja monipuolinen tietokantakieli. Se on niin kutsutusti ei-proseduraalinen kieli, mikä tarkoittaa sitä, että kielellä kerrotaan *mitä* tietoa haetaan, ei *miten*. Tietojen hakemisen lisäksi SQL-kielellä voidaan päivittää, poistaa ja lisätä tietokannan tietoja (Hovi ym. 2005, 10).

4.2 Relaatiotietokannan rakenne käytännössä

Kuvassa 1 on esimerkki yksinkertaisesta relaatiotietokannan rakenteesta. Kuva on tehty Turun seudun lihastautiyhdistys ry:n toimintamallia ajatellen, mutta sen tarkoituksena on toimia ainoastaan esimerkkinä, eikä se ota tässä vaiheessa vielä kantaa siihen, onko tietokannan rakenne mielekäs yhdistyksen toimintaa ajatellen.

Kuvan tietokannassa on taulut Asiakas, Avustaja ja Keikka, jotka kuvaavat

reaalimaailmaa. Jokaisessa taulussa on sarakkeita (column), kuten henTun, nimi ja tyyppi Avustaja-taulussa. Tauluissa on myös rivejä (row), joista jokainen sisältää aina yhden avustajan, keikan tai asiakkaan tiedot.



Avustaja

henTun	nimi	tyyppi
111111-111A	Anna Avustaja	palkallinen
220222-222B	Hanna Hoitaja	vapaaehtoinen
300333-333C	Essi Ensiapu	palkallinen

Keikka

kTun	asHenTun	avHenTun	alkaa	paattyy
1	040444-444D	220222-222B	01.03.2000 klo 7	01.03.2000 klo 8
2	050555-555E	111111-111A	02.03.2000 klo 8	02.03.2000 klo 9
3	060666-666F	220222-222B	03.03.2000 klo 5	03.03.2000 klo 7

Asiakas

henTun	nimi
040444-444D	Pasi Potilas
050555-555E	Alli Avustettava
060666-666F	Taija Toipilas

Kuva 1. Esimerkki yksinkertaisesta tietokantarakenteesta, jossa on kolme taulua

Keltaisella merkityt sarakkeet ovat taulunsa *perusavaimia* (primary key). Perusavaimen on aina oltava uniikki eli taulussa perusavaimen sarakkeessa ei saa olla useammalla rivillä samaa arvoa.

Kuvassa 1 Avustaja-taulussa sarake henTun (henkilötunnus) toimii perusavaimena. Henkilötunnus yksilöi avustajan, joten sama avustaja ei voi olla taulussa kuin yhden kerran. Kahdella avustajalla ei voi myöskään olla samaa henkilötunnusta. Myös Asiakas-taulussa on sarake henTun, joka toimii Asiakas-taulun perusavaimena ja vastaavasti yksilöi asiakkaat. Keikka-taulun perusavaimena on kTun, joka yksilöi keikat yksilöllisellä keikan numerolla.

Kuvassa vihreällä merkityt sarakkeet ovat taulujen *viiteavaimia* (foreign key). Keikka-taulussa viiteavaimia on kaksi, avHenTun ja asHenTun. AvHenTun-sarakkeeseen tulee keikan avustajan henkilötunnus, joka vastaa Avustaja-taulun perusavaimen arvoa kyseisen avustajan rivillä. Tällä tavalla taulujen Avustaja ja Keikka välillä on riippuvuus, joka on toteutettu perusavaimen ja viiteavaimen liitoksella. Samanlainen liitos on tehty myös Asiakas-taulun perusavaimen henTun ja Keikka-taulun viiteavaimen asHenTun välille.

5 KÄSITEANALYYSI TIETOKANTASUUNNITTELUN PERUSTANA

5.1 Suunniteltavan tietokannan perusasetelma

Koska tässä opinnäytetyössä suunnitellaan tietokantaa Turun seudun lihastautiyhdistys ry:n avustajakeskukselle, on hyvä ensin käydä läpi keskuksen toimintaa ja keskeisimpiä käsitteitä yhdistyksen toiminnassa.

Avustajakeskus välittää sekä vapaaehtoisia että palkallisia avustajia. Avustajan ja asiakkaan välisiä avustustapahtumia kutsutaan keikoiksi. Tämän lisäksi yhdistys järjestää myös erilaisia kursseja, kuten vammais- ja uintiavustajakoulutuksia.

Yhdistyksellä on myös toimipisteitä eri kunnissa. Toimipisteet tekevät yhteistyötä eri kunnissa olevien yhteistyökumppaneiden kanssa, kuten esimerkiksi kuntien vammaisjärjestöjen sekä eri julkisen sektorin tahojen kanssa.

Käsiteanalyysiä laatiessa suunnittelu pohjautuu nimenomaan Lihastautiyhdistyksen keskeisimpiin käsitteisiin: asiakkaisiin, palkallisiin ja palkattomiin avustajiin, keikkoihin, kursseihin, toimipisteisiin ja yhteistyökumppaneihin.

5.2 Käsiteanalyysi ja käsitemalli

Tietokannan suunnittelutyö alkaa käsiteanalyysillä, jossa suunnitellaan tietokantaa loogisella tasolla. Käsiteanalyysin tarkoituksena on kuvata sitä reaalimaailman osaa, kohdealuetta, jota kuvataan tietokannassa (Hovi ym. 2005, 32-33). Käsiteanalyysin tuloksena syntyy käsitemalli, joka kuvataan yleensä graafisena käsitekaaviona eli ER-kaaviona. (Hovi ym. 2005, 33).

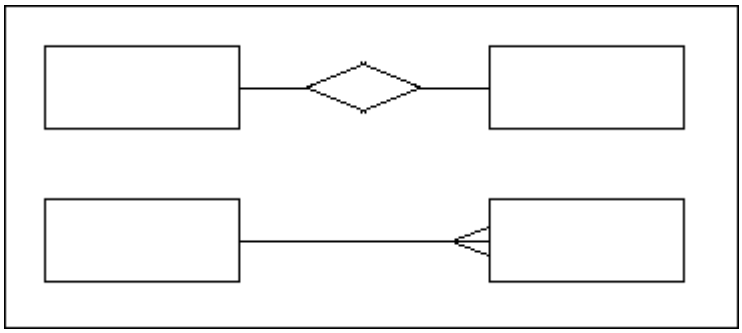
Käsitemallin tarkoituksena on määritellä pohja tietokannan fyysiselle

rakenteelle: voidaankin sanoa, että kyseessä on arkkitehtipiirustus tietokannan perustasta. Lisäksi käsitemalli toimii usein tekijöidensä yhteisenä sopimuksena siitä, miten mallinnettavana oleva kohdealue nähdään. (Hovi ym. 2005, 32-33).

Käsiteanalyysillä mallinnetaan kohde ensin melko karkealla tasolla, jota tarkennetaan suunnittelun edetessä (Hovi ym. 2005, 24). Teknisiä ominaisuuksia, kuten suorituskykyä, ei käsiteanalyysiä tehtäessä tarvitse vielä miettiä, sillä niihin kiinnitetään huomiota vasta suunnittelun myöhemmissä vaiheissa (Hovi ym. 2005, 32).

5.3 ER-kaavion mallintaminen UML-notaatiolla

Aikaisempina vuosina käsiteanalyysistä syntyvää ER-kaaviota ollaan mallinnettu perinteisesti joko Chenin notaatiolla tai Crowsin Crow’s Feet-notaatiolla. Chenin notaatio on kuvausmenetelmä, jossa suorakulmat kuvaavat käsitteitä ja viivat sekä niiden keskellä olevat timantit yhteyksiä käsitteiden välillä (esimerkki kuvassa 2 ylhäällä). Crownin notaatiossa (esimerkki kuvassa 2 alhaalla) käsitteet on kuvattu samalla tavalla, mutta yhteydet on kuvattu pelkillä viivoilla ja viivojen päissä on harakanvarpaat kuvaamassa yksi-moneen –yhteyttä. (Connolly & Begg 2002, xxxvi.)



Kuva 2. Chenin notaatio ja Crowsin notaatio (Crow's Feet)

Oliopohjainen suunnittelu ja toteutus ovat kuitenkin yleistyneet selvästi viime

vuosina (Hovi ym. 2005, 118). Sitä kautta tunnetuksi on tullut myös UML, Unified Modeling Language, joka on yksi yleisimmistä oliomallinnuksen kuvausmenetelmistä (Hovi ym. 2005, 120).

UML on notaatio, joka on saanut vaikutteita Rumbaughin, Boochn ja Jacobsonin oliomallinnusmenetelmistä. Tavallisesti sitä käytetään oliopohjaisten järjestelmien suunnittelussa, mutta sen käyttö on laajentunut myös tietokantojen suunnitteluun (Garcia-Molina ym. 2009, 171).

Koska UML on kehitetty oliomallinnukseen, sen perinteisillä oliosuunnittelumenetelmillä ei aukoitta pystytä mallintamaan relaatiotietokantaa. UML:ää voidaan kuitenkin hyödyntää notaationa eli kuvaustekniikkana. Käytännössä tämä tarkoittaa sitä, että relaatiotietokannan suunnittelu aloitetaan laatimalla käsiteanalyysimenetelmällä käsitemalli, mutta kuvaustekniikkana ei käytetä perinteisiä ER-notaatioita vaan UML:ää. (Hovi ym. 2005, 119.)

Käsiteanalyysissä hyödynnetään UML:n luokkakaaviota, joka perinteisesti koostuu luokista (class), attribuuteista (attributes) sekä metodeista (methods). Toisin kuin oliopohjaisessa suunnittelussa, mallinnuksessa ei kuitenkaan hyödynnetä metodeja. (Garcia-Molina ym. 2009, 172).

Tämän opinnäytetyön käsitemalli on laadittu UML-kuvaustekniikalla. Mallinnuksessa on hyödynnetty Objecteering Modeler –sovellusta, joka on ilmainen UML 2.0 –versiota tukeva mallinnustyökalu (Objecteering Software 2008).

5.4 Käsitteet ja niiden ominaisuudet

Käsitemalli sisältää ja siinä kuvataan käsitteitä (entity set). Käsitteellä tarkoitetaan jotakin konkreettista tai abstraktia asiaa, josta halutaan säilyttää tietokannassa dataa.

Käsitteisiin liittyy yleensä myös ominaisuuksia, joita kutsutaan tiedoiksi eli attribuuteiksi. Osalla näistä tiedoista on perusavaimen rooli, joten niin kuin tietokannan tauluillekin, myös käsitteille määritellään avaineheyssääntöä noudattava perusavain. (Hovi ym. 2005, 35-36.) Kuvassa 3 on esimerkki käsitteestä asiakas, jolla on attribuutteina asiakasnumero, etunimi, sukunimi, katuosoite, diagnoosi, apuvälineet, postinumero, postitoimipaikka, kotikunta ja puhelinnumero. UML:ssä perusavain voidaan merkitä lisäämällä attribuutin perään kirjaimet PK (Garcia-Molina ym. 2009, 173).

Kuva 3. Käsite Asiakas sekä sen ominaisuudet

Käsitettä ominaisuuksineen voidaan pitää eräänlaisena muottina, jonka avulla luodaan muottia noudattavia esiintymiä. Nämä esiintymät eli ilmentymät sisältävät tietojen varsinaiset arvot. (Hovi ym. 2005, 35-36).

5.5 Käsitteiden väliset yhteydet ja niiden pakollisuus

Käsitemallissa kuvataan myös yhteyksiä eli suhteita (relationship), joita luodaan eri käsitteiden välille. Yhteyksiä on kolmea eri tyyppiä: yksi-yhteen, yksi-moneen sekä moni-moneen. (Hovi ym. 2005, 37.)

5.5.1 Yksi-yhteen-yhteydet

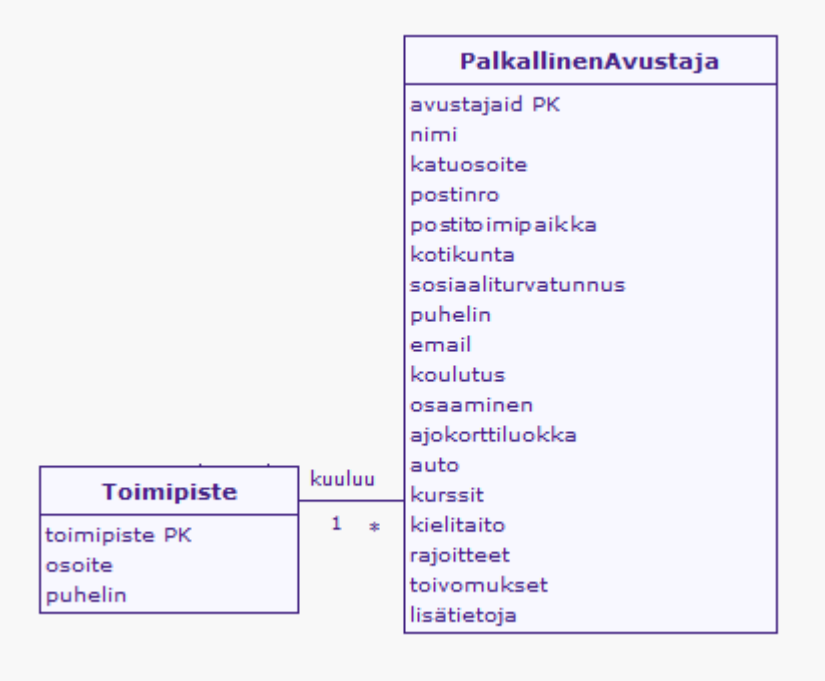
Kuvassa 4 on esitetty esimerkki yksi-yhteen-yhteydestä. Kuvasta voidaan

tulkita, että yksi toimipiste sijaitsee aina yhdessä kunnassa ja yhdessä kunnassa voi olla vain yksi toimipiste. Yhteystyyppiä havainnollistetaan UML:ssä numerolla 1, joka on sijoitettu yhteyden kumpaankin päähän (Garcia-Molina ym. 2009, 174).

Yksi-yhteen-yhteydet ovat hyvin harvinaisia ja yleensä merkkejä huonosta tai aikansa eläneestä suunnittelusta (Hovi ym. 2005, 37). Kuvankaan esimerkki ei ole mielekäs, sillä vaikka yhdessä kunnassa olisikin tällä hetkellä vain yksi Lihastautiyhdistyksen toimipiste, ei ole kuitenkaan mahdotonta, etteikö joku päivä toimipisteitä perustettaisi lisää.

Kuva 4. Yksi-yhteen-yhteys

5.5.2 Yksi-moneen-yhteydet

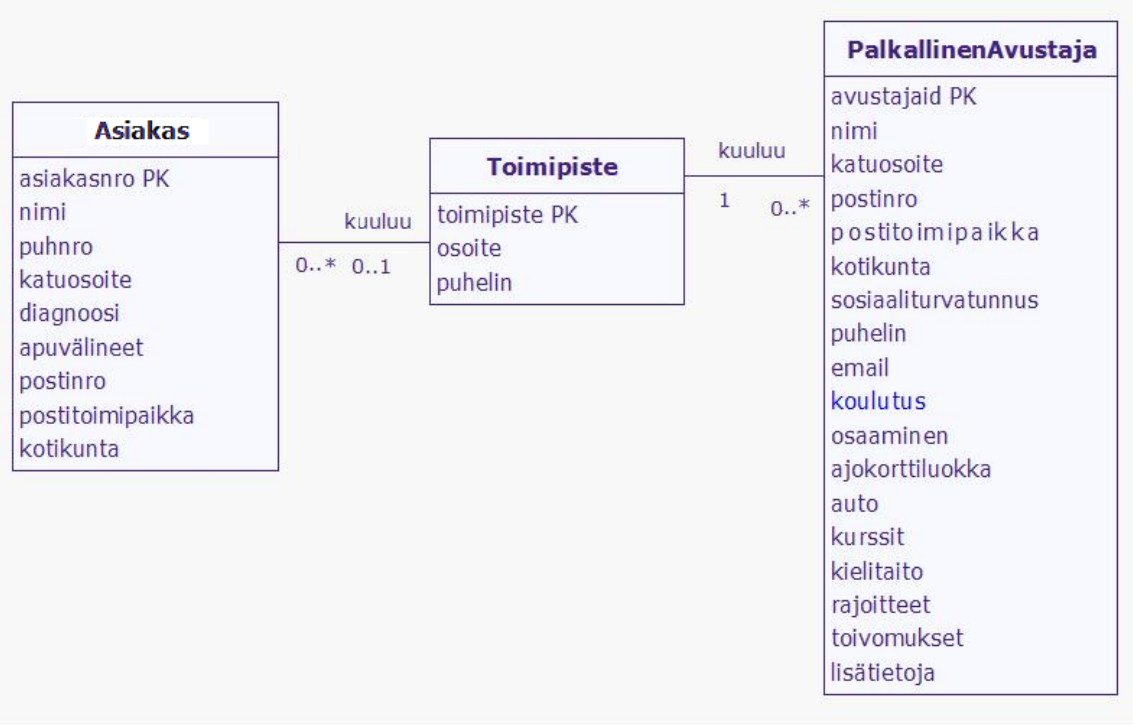


Kuva 5. Yksi-moneen-yhteys

Kuvassa 5 on esitetty esimerkki yleisimmästä yhteystyypistä, yksi-moneen-yhteydestä, jota voidaan kutsua myös isä-lapsi-yhteydeksi (Hovi ym. 2005, 37). Kuvassa on määritelty, että palkallinen avustaja ei voi kuulua kuin yhteen toimipisteeseen. Tämä merkitään UML:ssä numerolla 1, aivan niin kuin yksi-yhteen-yhteydessäkin (Garcia-Molina ym. 2009, 174). Toimipisteessä voi kuitenkin olla useampia avustajia. Tämä merkitään UML:ssä merkillä *, joka tarkoittaa sitä, ettei avustajien määrää ole rajoitettu (Garcia-Molina ym. 2009, 174).

Kun käsitteiden välille luodaan yhteyksiä, niitä voidaan kuvata sanoilla, jotka kertovat yhteyden laadusta (Garcia-Molina ym. 2009, 173). Kuvassa 5 yhteys on nimetty sanalla ”kuuluu”. Käsitteiden nimeäminen on tärkeää varsinkin silloin, kun kahden käsitteen välillä on useampi kuin yksi yhteys (Hovi ym. 2005, 55).

Yhteyksille voidaan määritellä myös yhteyden pakollisuus: yhteys voi olla pakollinen tai ehdollinen. (Hovi ym. 2005, 42).



Kuva 6. Yhteyksien pakollisuus

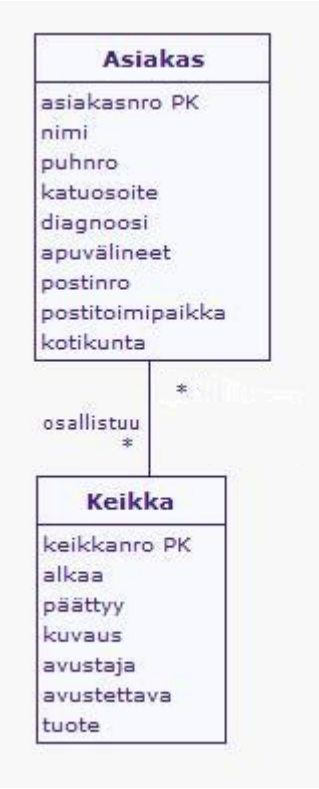
Kuvassa 6 on esitetty yhteyksiä, joille on määritelty eri pakollisuudet. Käsitteiden Asiakas ja Toimipiste välillä on yhteys, jossa asiakkaan päässä on merkintä 0..* ja toimipisteen päässä 0..1. Nämä ovat UML:n merkintätapoja ja ne voidaan tiivistää muotoon n..m, eli käsitteellä on oltava vähintään n ilmentymää, mutta enintään m. Merkintä 0..1 toimipisteen päässä tarkoittaa, että asiakkaan ei ole pakko kuulua toimipisteeseen, mutta asiakas voi kuulua yhteen toimipisteeseen. Yhteys ei siis ole pakollinen. (Garcia-Molina ym. 2009, 174) Merkintää 0..1 voitaisiin käyttää myös kuvan 4 yksi-yhteen-yhteydessä, eli myös yksi-yhteen-yhteyksille voidaan määrittää pakollisuus.

Merkintä 0..* asiakkaan päässä tarkoittaa, että toimipisteellä ei ole pakko olla asiakkaita, eikä asiakkaiden määrää ei ole rajoitettu. Merkintätapa 0..* voidaan myös merkitä ainoastaan merkillä *. (Garcia-Molina ym. 2009, 174.) Jos sen

sijaan merkintä olisikin muotoa 1..*, yhteys olisi pakollinen, eli toisin sanoen toimipisteen tietoja ei pystyttäisi tallentamaan tietokantaan, ellei toimipisteelle annettaisi vähintään yhtä asiakasta (Hovi ym. 2005, 42).

Käsitteiden Toimipiste ja PalkallinenAvustaja välillä on yhteys, jossa käsitteen Toimipiste päässä on merkintä 1 ja käsitteen PalkallinenAvustaja päässä 0..*. Merkintä 1 tarkoittaa samaa kuin 1..1, eli palkallisen avustajan on kuuluttava yhteen toimipisteeseen, toisin sanoen yhteys on pakollinen (Garcia-Molina ym. 2009, 174). Palkallista avustajaa ei siis voida tallentaa tietokantaan, ellei avustajalle ole määritelty toimipistettä. Toimipisteeseen voi sen sijaan kuulua monta palkallista avustajaa, mutta yhdenkään palkallisen avustajan tallentaminen ei ole pakollista.

5.5.3 Moni-moneen-yhteydet



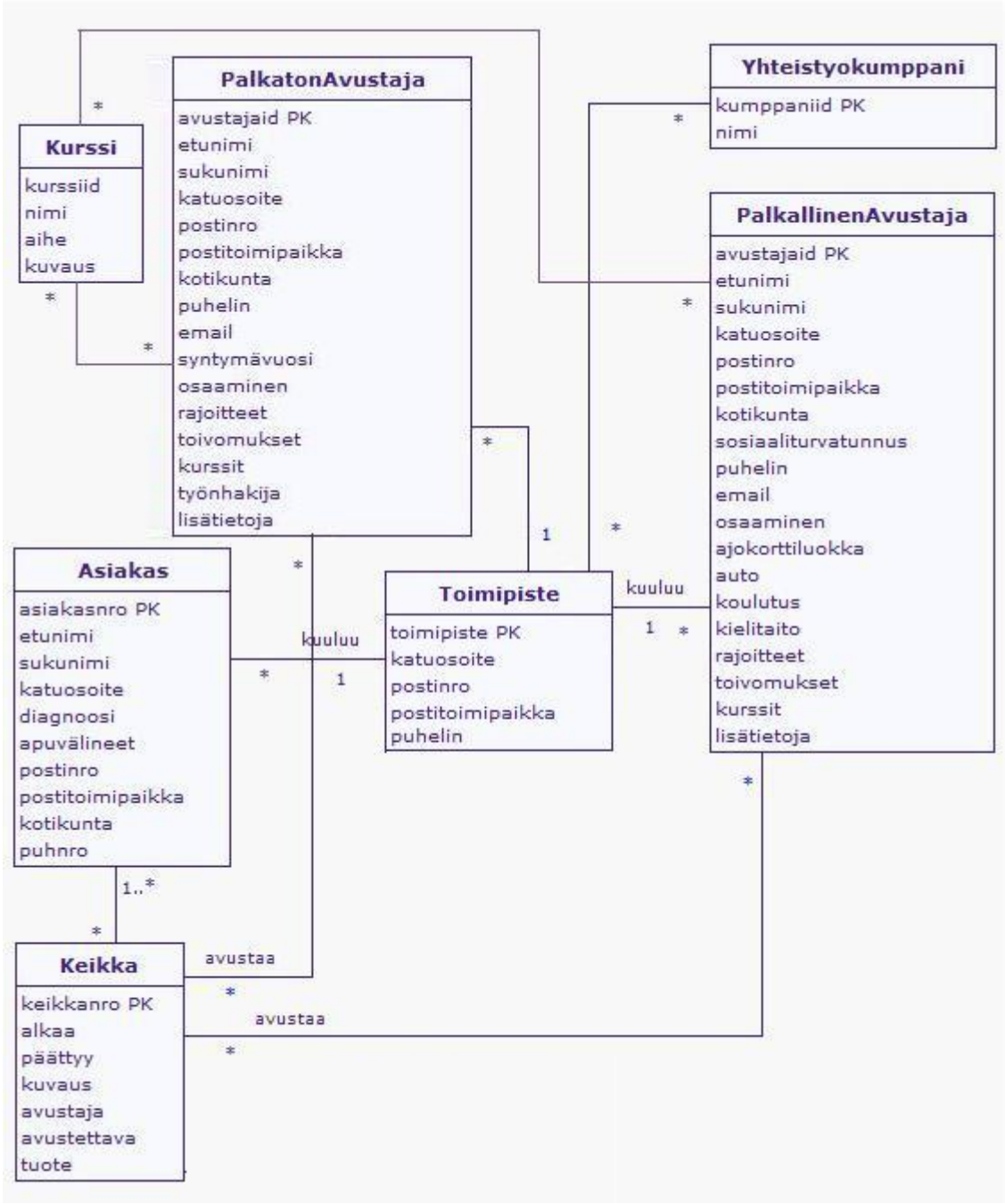
Kuva 7. Moni-moneen-yhteys

Kuvassa 7 on esitetty esimerkki moni-moneen-yhteydestä. Yhteystyyppi on perusteltu, koska asiakas voi olla monella keikalla ja keikalla voi olla monta

asiakasta.

Kuten kuvasta 7 näkyy, myös moni-moneen-yhteyksille voidaan asettaa pakollisuus samalla tavalla kuin yksi-moneen-yhteyksillekin. * tarkoittaa samaa kuin 0..*, eli asiakkaalla voi olla monta keikkaa ja keikalla voi olla monta asiakasta.

Moni-moneen-yhteydet kuitenkin eroavat selvästi yksi-moneen-yhteyksistä, sillä ne puretaan auki suunnittelun edetessä (Hovi ym. 2005, 44). Kuvassa 8 on käsitekaavio Turun seudun lihastautiyhdistys ry:n tietokannasta, jossa moni-moneen-yhteyksiä ei ole vielä purettu auki.



Kuva 8. ER-kaavio, jossa moni-moneen-yhteyksiä ei ole vielä purettu auki

Käytännössä moni-moneen-yhteyksien auki purkaminen tarkoittaa sitä, että yhteyksien väliin luodaan jokin välikäsite eli assosiatiivinen käsite, jonka nimi saadaan usein yhteyden kuvauksesta (Hovi ym. 2005, 44). Syntynyt välikäsite on lapsikäsite ja sen voidaan ajatella olevan yksi-moneen-yhteydessä isäkäsitteisiinsä, eli käytännössä moni-moneen-yhteydet puretaan kahdeksi

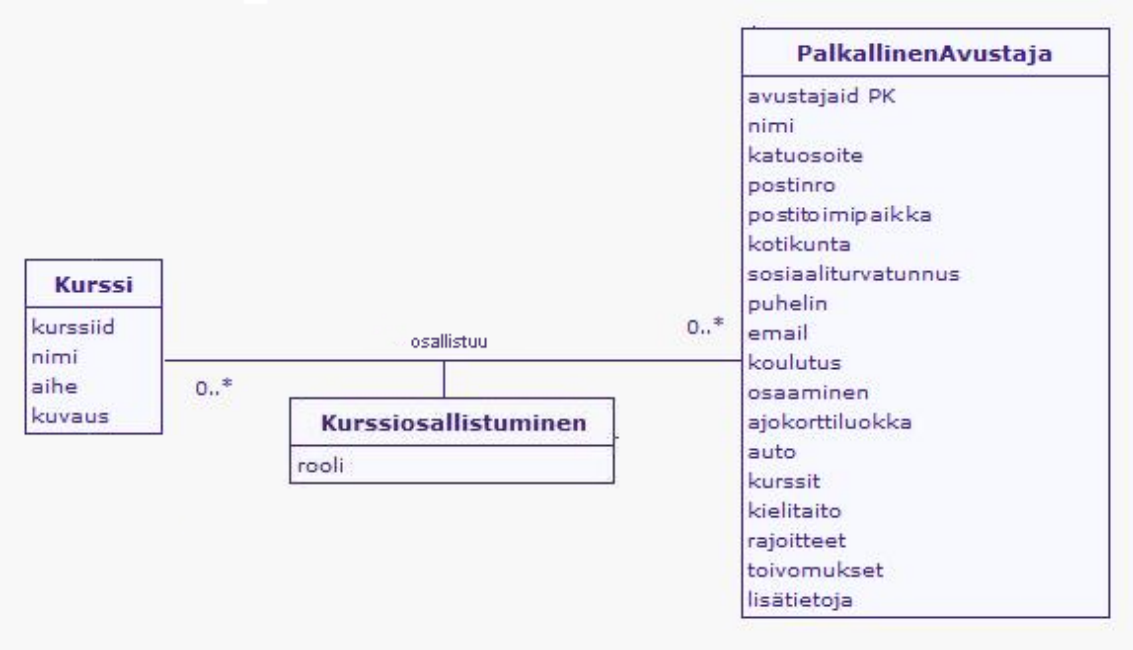
yksi-moneen-yhteydeksi (Hovi ym. 2005, 45).

Moni-moneen-yhteyksien auki purkaminen ei ole pelkkä muotoseikka, vaan yleensä välikäsitteeseen pystytään lisäämään todella tärkeitä tietoja, joiden lisääminen muualle olisi vaikeaa (Hovi ym. 2005, 44). Mietitään esimerkiksi kursseja, joissa avustajat voivat olla sekä osallistujia että kouluttajia: minne heidän roolinsa tulisi kirjata? Käsitteisiin PalkallinenAvustaja tai PalkatonAvustaja tietoa ei voida kirjata, koska tauluun pitäisi lisätä uusi sarake aina, kun avustaja osallistuisi kurssille. Kurssi-käsitteeseen roolia ei voida myöskään merkitä, koska jos kouluttajia onkin useampia samalla kurssilla, sarakemäärää olisi taas lisättävä.

Kuvassa 9 ongelmaan on esitetty ratkaisu, sillä käsitteiden PalkallinenAvustaja ja Kurssi välille on lisätty välikäsite Kurssiosallistuminen, jolle on annettu rooli-attribuutti. Tässä ratkaisussa samalle riville saadaan mahdutettua tiedot kurssista, avustajasta sekä kyseisen avustajan roolista. Jos kursseja tai avustajia on useampia, myös näistä tehdään omat rivinsä tauluun.

Kurssiosallistuminen-käsitteen perusavaimena toimii yhdistelmäavain käsitteiden Kurssi ja PalkallinenAvustaja perusavaimista. UML:ssä moni-moneen-yhteyksien väliin purettuun välikäsitteeseen ei kuitenkaan merkitä perusavainta, koska se voidaan päätellä muista käsitteistä (Garcia-Molina ym. 2009, 176).

Kuvassa 11 on kuvattu käsitekaavio, jossa moni-moneen-yhteydet on purettu auki. Kaavioon luotiin neljä uutta käsitettä: Yhteistyö, Osallistuminen, Avustaminen ja Kurssiosallistuminen.



Kuva 9. Assosiatiivinen käsite kahden moni-moneen-yhteydessä olevan käsitteen välissä

5.6 Käsitetyypit

Yhteyksiä ja niiden pakollisuutta määriteltäessä voidaan tarkastella myös eri käsitetyyppejä. Käsitteitä on kolmea eri tyyppiä: ydinkäsitteitä eli riippumattomia käsitteitä, riippuvia käsitteitä eli karakteristisiä käsitteitä ja moni-moneen-käsitteitä eli assosiatiivisia käsitteitä. (Hovi ym. 2005, 48.)

Ydinkäsitteet ovat keskeisimpiä käsitteitä. Niiden mukaisia esiintymiä voi tallentaa muista käsitteistä riippumatta, eikä niissä ole muiden käsitteiden tietoja**.** Ydinkäsitteiden perusavaimet ovat yleensä yksiosaisia. Esimerkiksi kuvassa 8 Asiakas on ydinkäsite. (Hovi ym. 2005, 48.)

Riippuvat käsittet ovat lapsikäsitteitä ja ne kuvaavat jotakin muuta käsitettä. Muiden käsitteiden ja riippuvien käsitteiden välillä on yksi-moneen-yhteys, ja perusavain on yleensä moniosainen. Riippuvan käsitteen mukaisia tietoja ei voida tallentaa kantaan ennen isäkäsitteen esiintymien tallentamista. (Hovi ym. 2005, 48.)

Turun seudun lihastautiyhdistys ry järjestää kursseja. Jokin kursseista saattaa kestää useamman päivän ja kurssikertoja on monta. Tällöin käsiteanalyysiin on lisättävä kuvan 10 mukainen käsite Kurssikerta, jonka perusavain koostuu kurssin tunnistenumerosta sekä päivämäärästä, jolloin kurssikerta pidetään. Kurssikerta on tällöin riippuva käsite, sillä se sisältää tietoja kurssista, eikä sen ilmentymiä voida tallentaa kantaan ilman kurssin ilmentymiä.

Kuva 10. Riippuva käsite Kurssikerta

Moni-moneen-käsitteet eli assosiatiiviset käsitteet puolestaan syntyvät nimensä mukaisesti siitä, kun moni-moneen-käsitteet puretaan auki (Hovi ym. 2005, 48). Toisin sanoen kahden toistensa kanssa moni-moneen-yhteydessä olevien käsitteiden väliin syntyvää välikäsitettä kutsutaan moni-moneen-käsitteeksi. Tälläinen käsite on muun muassa kuvan 9 käsite Kurssiosallistuminen.

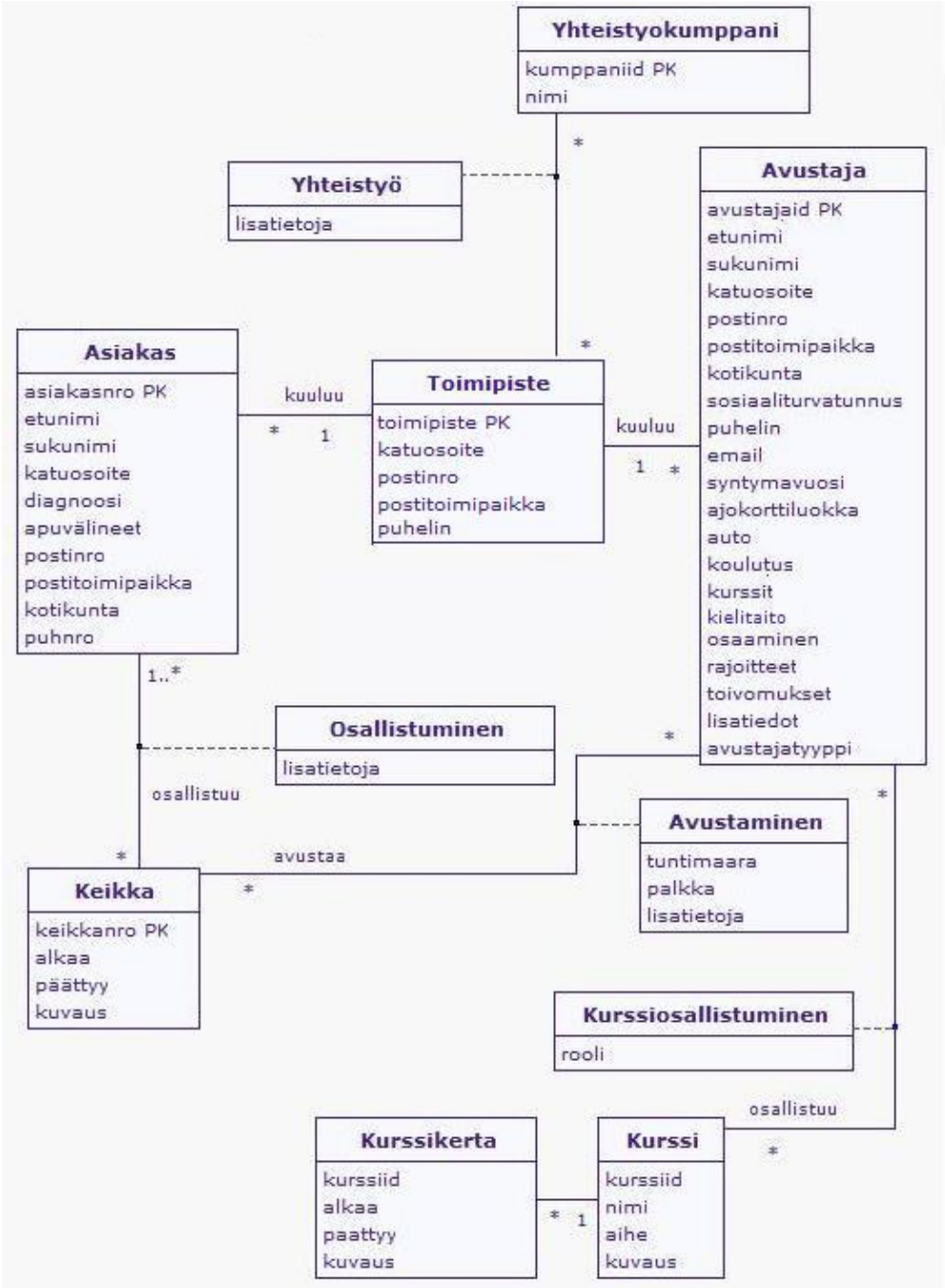
5.7 Käsitteiden yhdistäminen

Tietojen toistamisen välttäminen on yksi keskeisimmistä asioista tietokannan suunnittelun eri vaiheissa. Tämän vuoksi monissa tapauksissa käsitteitä voidaan yhdistää. (Hovi ym. 2005, 51.)

Kuvassa 8 on esitetty Turun seudun lihastautiyhdistys ry:n käsitekaavio, jossa moni-moneen-yhteyksiä ei ole vielä purettu auki. Kuvasta näkee, että kaaviossa on kaksi hyvin suurta käsitettä: PalkatonAvustaja ja PalkallinenAvustaja. Kun käsitteiden tietoja tutkitaan tarkemmin, huomataan, että näillä käsitteillä on monia samoja tietoja. Tämä viittaa siihen, että käsitteitä voi olla syytä yhdistää. (Hovi ym. 2005, 52.)

On myös muitakin syitä, jotka perustelevat käsitteiden PalkallinenAvustaja ja

PalkatonAvustaja yhdistämistä. Yksi syy on se, että näillä kahdella käsitteellä on sama perusavain, avustajaId. Lisäksi käsitteillä on lähes samanlaiset yhteydet. Suurin merkki käsitteiden yhdistämistarpeesta on kuitenkin se, että on olemassa suuri vaara, että sama esiintymä tulisi tallennetuksi tietokantaan kahteen kertaan. (Hovi ym. 2005, 52.) Palkattomasta avustajastahan voi tulla palkallinen avustaja.



Kuva 11. Käsitekaavion moni-moneen-yhteydet on purettu ja käsitteitä yhdistetty

Kuvassa 11 on esitetty kuva käsitemallista, jossa käsitteet PalkallinenAvustaja ja PalkatonAvustaja on yhdistetty käsitteeksi Avustaja. Molempien käsitteiden attribuutit on yhdistetty samaan käsitteeseen, ja lisäksi niiden joukkoon on lisätty uusi attribuutti, avustajatyyppi. Samasta syystä PalkatonAvustaja-käsitteen työnhakija-attribuutti on poistettu, sillä palkattoman ja vapaaehtoisen avustajan lisäksi myös työnhakija voi olla yksi avustajatyypeistä.

Avustaja-käsitteestä muodostettuun tauluun tulee jäämään tyhjiä sarakkeita, jotka vaihtelevat sen mukaan, onko kyseessä palkallinen vai palkaton avustaja. Tyhjät rivit eivät kuitenkaan haittaa relaatiotietokannassa, joten käsitteitä voi siinä mielessä huoletta yhdistää (Hovi ym. 2005, 52).

5.8 Tarveanalyysilla kohti tarkempaa käsiteanalyysiä

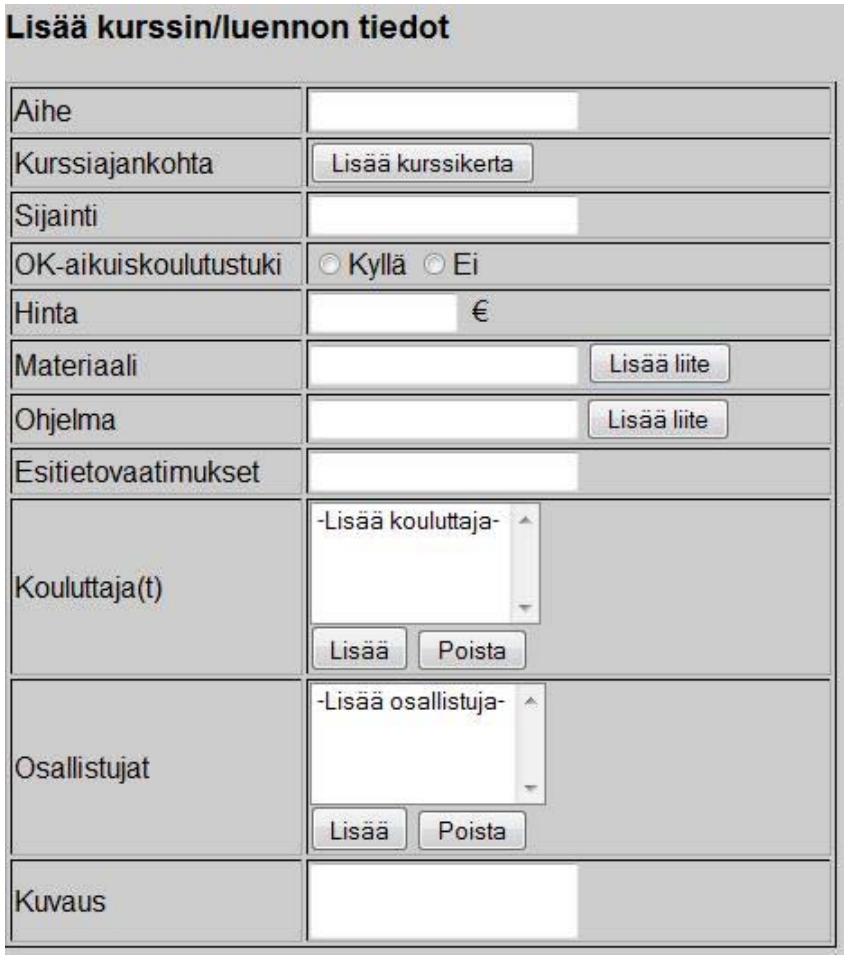
Käsiteanalyysi voi jäädä puutteelliseksi, ellei sen toimivuutta kokeilla tiedossa olevilla tietotarpeilla ja samalla lisätä mahdollisia uusia käsitteitä ja tietoja. Tätä toimenpidettä kutsutaan tarveanalyysiksi. (Hovi ym. 2005, 80.)

Tietotarpeilla tarkoitetaan muun muassa sovelluksen käyttöliittymiä ja raportteja, joita sovelluksesta pitäisi pystyä tulostamaan (Hovi ym. 2005, 80). Turun seudun lihastautiyhdistys ry:n sovellusta ei vielä kehitetä, joten tässä kohtaa tarveanalyysi jää hieman puutteelliseksi. Järjestelmästä on kuitenkin jo suunniteltu eri lomakkeiden käyttöliittymiä (liite 2, päivitetty määrittelydokumentti), joten tarveanalyysi tehdään niiden perusteella.

Käytännössä tarveanalyysi aloitetaan ottamalla jokin tietotarve, tässä tapauksessa eri lomakkeiden käyttöliittymät, ja verrataan sitä olemassa olevaan käsiteanalyysiin. Jos käsitemallista puuttuu joitakin tietoja, joita tietotarpeessa on esitetty, ne lisätään käsitteiden yhteyteen. Kun kaikki tietotarpeet on käyty läpi, käsitemallin pitäisi olla aiempaa yksityiskohtaisempi ja lisäksi se on osoitettu toimivaksi. (Hovi ym. 2005, 81.)

Esimerkiksi kuvassa 12 on esitetty järjestelmän koulutusosioa varten suunniteltu käyttölittymä, jossa on esitetty tiedot, joita kysytään järjestelmään lisättävästä keikasta. Kun verrataan kuvan 11 ER-kaaviota, huomataan, että käsitteestä puuttuu kokonaan muun muassa kurssin hinta ja koulutuksen sijainti. Lisäksi kuvassa 11 käsitteessä Kurssi näyttäisi olevan yksi ylimääräinen tieto, nimittäin kurssin nimi. Nämä asiat on korjattava käsitekaavioon.

Kuvan 12 käyttöliittymästä voidaan havaita myös eräs toinen varsin merkittävä asia, nimittäin mahdollisuus lisätä liitetiedostoja. Käsiteanalyysiin on lisättävä uusi käsite Lomake, joka sisältää tiedot järjestelmään liitettävistä liitetiedostoista. Lomake-käsite ja muut tarveanalyysissä huomatut puutteet on lisätty kuvaan 16, jossa on päivitetty käsitekaavio.



Kuva 12. Osa Koulutus-osion käyttöliittymää, jossa kurssin tiedot kirjataan järjestelmään

5.9 Normalisointi

Normalisointi on E. F. Coddin 1972 kehittämä menetelmä, jolla yritetään vähentää tietojen toistamista eli redundanssia. Hyvin normalisoitu tietokanta on muutosjoustava, tehokas päivittää ja se on helpompi pitää yhdenmukaisena, koska tiedot päivitetään vain yhteen paikkaan. (Hovi ym. 2005, 86.)

Yleisesti puhutaan lähinnä taulujen normalisoinnista, mutta normalisointi voidaan suunnitteluprosessia ajatellen tehdä jo käsiteanalyysivaiheessa soveltamalla normaalimuotojen sääntöjä taulujen sijasta käsitteisiin (Hovi ym. 2005, 97). Tässä opinnäytetyössä normalisointi suoritetaan jo käsiteanalyysivaiheessa.

Normalisointiteoriassa tauluille määritellään normaalimuotoja, joiden avulla tarkastellaan yksittäisiä tietoja ja niiden välisiä riippuvuuksia. (Hovi ym. 2005, 90.) Normaalimuodoista yleisimmät ovat ensimmäinen, toinen ja kolmas normaalimuoto. (Hovi ym. 2005, 86.)

5.9.1 Ensimmäinen normaalimuoto

Ensimmäisen normaalimuodon mukaisesti relaatiotietokannasta on poistettava toistuvat ryhmät ja moniarvoiset sarakkeet. Käytännössä tämä tarkoittaa sitä, että normalisoitavat taulut jaetaan kahteen tai useampaan osaan (Hovi ym. 2005, 88).

Kuvassa 11 on esitetty käsite Avustaja, josta muodostetun taulun ilmentymä voisi olla kuvan 13 mukainen. Kun tarkastellaan kuvaa 13, voidaan heti huomata, että ilmentymässä on moniarvoinen sarake kielitaito.

Ensimmäisenä ajatuksena saattaa olla, että esimerkiksi kielitaitosarakkeita voisi olla kaksi: kielitaito1 ja kielitaito2. Ongelmia tulee kuitenkin silloin, kun avustaja osaa useampaa kuin kahta kieltä: tällöin olisi taas luotava uusi sarake.

avustajaid	etunimi	sukunimi ...	... ajokortti	auto ...	... kielitaito ...
1	Anna	Järvinen	Ei	Ei	englanti, ruotsi
2	Matti	Virtanen	Ei	Ei	englanti
3	Esko	Jokinen	Kyllä	Kyllä	saksa

Kuva 13. Avustaja-käsitteen ilmentymä

Tietokannan rakenteen on oltava sellainen, ettei sitä tarvitse koko ajan muuttaa. Moniarvoiset sarakkeet puolestaan vaikeuttavat ylläpitoa sekä erilaisten laskutoimitusten tekemistä. Tämän vuoksi ongelma ratkaistaan luomalla uusi käsite. (Hovi ym. 2005, 88.)

Kuvassa 14 on esimerkki uuden Kielitaito-käsitteen ilmentymästä. Kuvasta näkee, että avustaja voi osata vaikka kuinka montaa eri kieltä, ja kaikki voidaan lisätä helposti ja tietokantaa muuttamatta tietokantaan.

avustajaid	kieli
1	englanti
1	ruotsi
2	englanti
3	saksa

Kuva 14. Kielitaito-käsitteen ilmentymä

5.9.2 Toinen normaalimuoto

Toista ja kolmatta normaalimuotoa tarkasteltaessa on ensin määriteltävä funktionaalinen riippuvuus. Sanotaan, että sarake A on funktionaalisesti riippuvainen sarakkeesta B, jos jokaista B:tä kohti on yksi tai ei yhtään A:n arvoa kunakin ajanhetkenä (Hovi ym. 2005, 91). Esimerkiksi kuvassa 11 käsitteessä Asiakas sukunimi-attribuutti on funktionaalisesti riippuvainen käsitteestä asiakasnro. Asiakasnro ei kuitenkaan ole funktionaalisesti riippuvainen sukunimestä, koska monella asiakkaalla voi olla sama sukunimi, eli kutakin sukunimeä kohti voi olla monta asiakasnumeroa.

Toisen normaalimuodon sääntö on, että jos taulussa on moniosainen perusavain, tulee kaikkien attribuuttien olla funktionaalisesti riippuvia koko perusavaimesta (Hovi ym. 2005, 91). Kuvassa 11 käsitteessä Yhteistyö on kaksiosainen perusavain kumppani-id ja toimipiste. Jos käsitteeseen lisättäisiin tieto toimipisteosoite, toista normaalimuotoa rikottaisiin, koska tieto olisi funktionaalisesti riippuvainen toimipisteestä, muttei kumppani-id:stä.

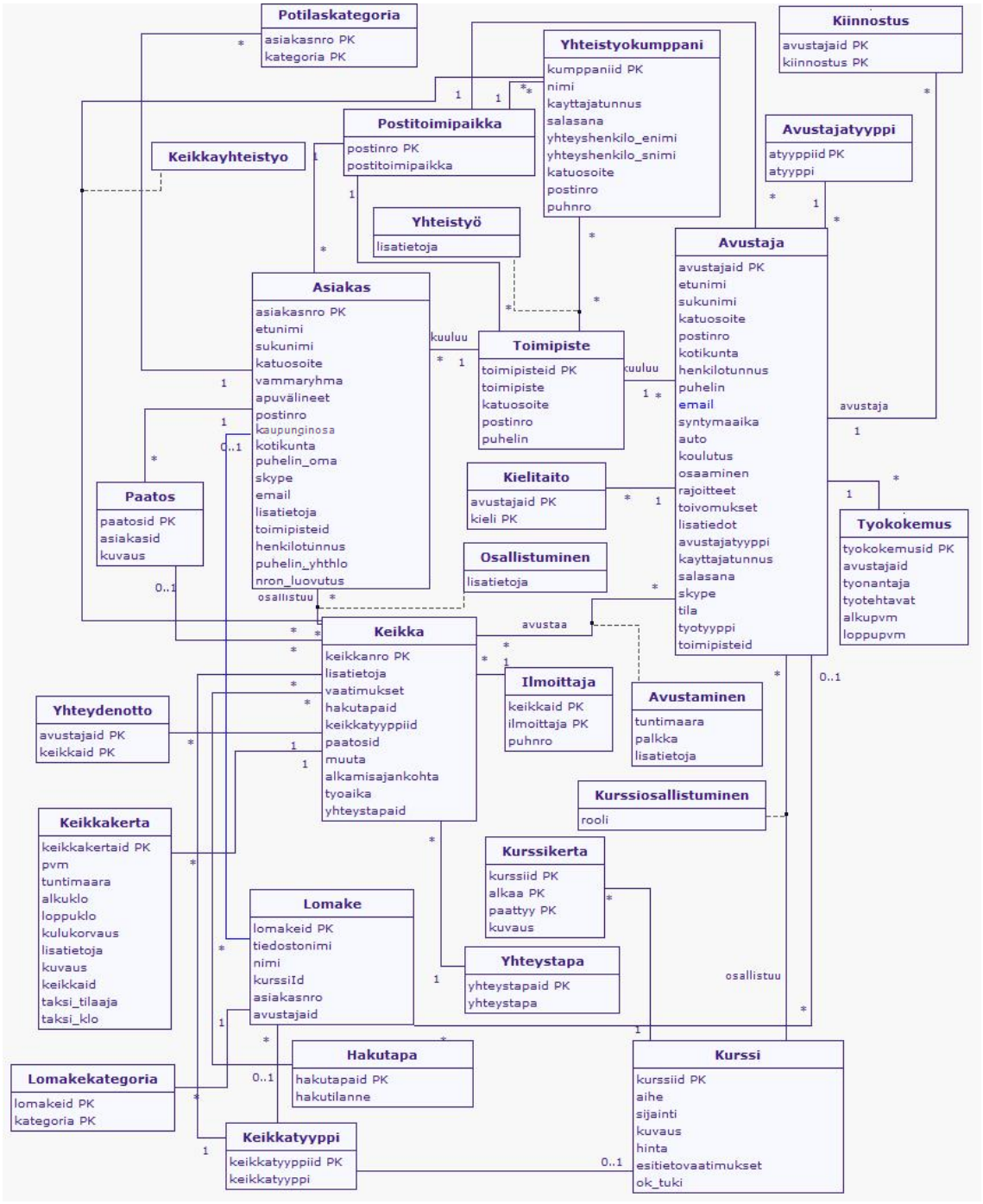
5.9.3 Kolmas normaalimuoto

Toisessa normaalimuodossa tarkastellaan saman taulun sarakkeiden riippuvuutta moniosaiseen perusavaimeen. Kolmas normaalimuoto sen sijaan tarkastelee sarakkeiden funktionaalisia riippuvuuksia toisiin sarakkeisiin, jotka eivät ole perusavaimia. Kolmannen normaalimuodon säännön mukaan jokaisen sarakkeen tulee olla funktionaalisesti riippuvainen vain perusavaimesta. (Hovi ym. 2005, 93.)

Kuvassa 11 on käsite Asiakas, jonka jokainen tieto on funktionaalisesti riippuvainen perusavaimesta asiakasnro. Kun käsitettä tutkitaan tarkemmin, voidaan kuitenkin huomata, että myös postinron ja postitoimipaikan välillä on funktionaalinen riippuvuus, sillä jokaista postinumeroa kohti on yksi postitoimipaikan nimi. Tämä johtaa siihen, että postitoimipaikkojen nimet toistuvat tarpeettoman monta kertaa, eikä niitä voida tallentaa etukäteen tietokantaan (Hovi ym. 2005, 94). Ongelma voidaan ratkaista lisäämällä käsitemalliin uusi kuvan 15 mukainen käsite Postitoimipaikka.

Kuva 15. Uusi käsite postitoimipaikka, johon tallennetaan postinumerot ja postitoimipaikat

Kuvassa 16 on esitetty ER-kaavio, jossa käsitteet on normalisoitu. Lisäksi käsitteitä on testattu tarveanalyysillä.



Kuva 16. Valmis käsiteanalyysi, joka on normalisoitu ja testattu tarveanalyysillä

5.9.4 Denormalisointi

Kun tietokanta normalisoidaan kolmanteen normaalimuotoon, tietojen toisto minimoituu ja avaintietoja lukuun ottamatta tiedot tallennetaan vain kertaalleen tietokantaan. Normalisoinnista saattaa kuitenkin olla haittaakin, sillä taulujen määrän kasvaessa myös liitosten määrä kasvaa. Tämä saattaa hidastaa

tietokantahakuja tarpeettoman paljon. (Hovi ym. 2005, 95.)

Hakujen nopeuttamiseksi tauluja voidaan denormalisoida eli toisin sanoen useamman taulun tietoja voidaan yhdistää samaan tauluun. Esimerkiksi kuvassa 16 taulu Avustajatyyppi voitaisiin denormalisoida tauluun Avustaja, jos hakujen katsotaan toimivan liian hitaasti. Kuitenkin jos avustajatyyppejä on tarve päivittää usein, denormalisoitu ratkaisu on huono, koska normalisointi edistää nimenomaan päivittämistä. (Hovi ym. 2005, 94-97.)

6 KÄSITEANALYYSISTÄ TAULUJEN MUODOSTAMISEEN

6.1 Käsitemallista relaatiomalliin

Käsiteanalyysiä laadittaessa ei vielä tarvinnut huolehtia käsitemallin soveltuvuudesta eri tietokantatuotteisiin. Tauluja muodostaessa on kuitenkin jo otettava huomioon eri tietokantatuotteiden eroavaisuudet. (Hovi ym. 2005, 104.) Tässä opinnäytetyössä taulujen luominen on toteutettu MySQL-sovellusta ajatellen.

Relaatiotietokannan muodostaminen alkaa käsitteiden muuttamisesta tauluiksi. Jokaisesta käsitteestä tulee oma taulunsa relaatiotietokantaan. Käsitteiden tiedoista tulee taulujen tietoja eli sarakkeita, ja käsitteisiin merkityistä perusavaimista tulee taulujen perusavaimet. (Hovi ym. 2005, 105.)

6.2 Tietotyypit

Tauluja luotaessa on kiinnitettävä huomiota sarakkeiden tietotyyppeihin. Merkkimuotoisia tietotyyppejä ovat muun muassa char ja varchar, jotka eroavat toisistaan siinä mielessä, että char on kiinteämittainen ja varchar on vaihtuvamittainen tietotyyppi. Numeerisiin tietotyyppeihin kuuluvat muun muassa int ja decimal, joista int on suuri kokonaisluku ja decimal pakattu

desimaaliluku. (Hovi ym. 2005, 111.)

Kuvassa 17 on käsitteestä Asiakas muodostettu taulu. Suurimmalle osalle taulun sarakkeista on valittu tietotyypiksi vaihtuvamittainen varchar, jonka perään on merkitty sallittujen merkkien määrä sulkeisiin (esim. varchar 30). Varchar-tietotyypin valitseminen on perusteltua, koska jos esimerkiksi sarakkeen sukunimi tietotyypiksi olisi valittu char(30), olisi 10 merkkiä pitkä sukunimi vienyt 30 merkin tilan. Sen sijaan muutaman merkin pituisille tiedoille tietotyypiksi kannattaa valita char, koska vaihtuvamittaisella sarakkeella tilaa vie myös erillisenä tallennettu tiedon pituus. Tämän vuoksi postinro-sarakkeiden tietotyypiksi on valittu char. (Hovi ym. 2005, 112.)

Asiakas

Sarake	Tyyppi	Aakkosjärjestys	Attribuutit	Tyhjä	Oletusarvo	Lisätiedot
asiakasnro	int(10)		UNSIGNED	Ei	*Ei mitään*	auto_increment
etunimi	varchar(20)	utf8_general_ci		Ei	*Ei mitään*	
sukunimi	varchar(30)	utf8_general_ci		Ei	*Ei mitään*	
katuosoite	varchar(50)	utf8_general_ci		Ei	*Ei mitään*	
postinro	char(5)	utf8_general_ci		Ei	*Ei mitään*	
kotikunta	varchar(15)	utf8_general_ci		Ei	*Ei mitään*	
kaupunginosa	varchar(30)	utf8_general_ci		Ei	*Ei mitään*	
puhnro_oma	varchar(15)	utf8_general_ci		Ei	*Ei mitään*	
yhthlo_oma	varchar(15)	utf8_general_ci		Ei	*Ei mitään*	
nronluovutus	tinyint(1)			Kyllä	*NULL*	
email	varchar(40)	utf8_general_ci		Kyllä	*NULL*	
skype	varchar(40)	utf8_general_ci		Kyllä	*NULL*	
henkilotunnus	varchar(11)	utf8_general_ci		Kyllä	*NULL*	
vammaryhma	text	utf8_general_ci		Kyllä	*NULL*	
apuvalineet	text	utf8_general_ci		Kyllä	*NULL*	
lisatietoja	text	utf8_general_ci		Kyllä	*NULL*	
toimipisteid	int(10)		UNSIGNED	Ei	*Ei mitään*	
perustaja	varchar(20)	utf8_general_ci		Ei	*Ei mitään*	
luontiaika	datetime			Ei	*Ei mitään*	
paivittaja	varchar(20)	utf8_general_ci		Kyllä	*NULL*	
paivitysaika	datetime			Kyllä	*NULL*	

Kuva 17. Käsitteestä Asiakas muodostettu taulu

Osalle tiedoista on valittu tietotyypiksi TEXT. TEXT on vaihtuvamittainen tietotyyppi, jota voidaan hyödyntää vapaatekstikenttänä (Oracle and/or its affiliates 2008d). TEXT on hyödyllinen, kun sarakkeeseen voi tulla paljon tekstiä, eikä pituutta osata arvioida etukäteen.

DATE, TIME ja DATETIME ovat päivämäärän ja ajan tietotyyppejä, joista nimensä mukaisesti DATE kuvaa päivämäärää, TIME aikaa ja DATETIME molempia (Oracle and/or its affiliates 2008b). Kuvassa 18 on Lihastautiyhdistyksen Keikka-taulu, jossa on käytetty tietotyyppejä DATE ja TIME. Tämä on perusteltua, koska kaikkien keikkojen kohdalle ei kirjata kellonaikaa.

Kuvien 17, 18 ja 19 tauluihin on kuvan 16 käsitekaaviosta poiketen lisätty muutamia teknisiä sarakkeita, jotka käsittelevät taulujen lokitietoja. Tällaisia tietoja ovat muun muassa perustaja, luontiaika, paivittaja ja paivitysaika. Käytännössä tämä tarkoittaa sitä, että kun tauluun lisätään ensimmäinen rivi, perustaja-sarakkeeseen tulee tieto perustajan käyttäjätunnuksesta sekä luontiaika-sarakkeeseen aika, jolloin rivi lisättiin. Samoin kun riviä muutetaan, paivittaja-sarakkeeseen tulee päivittäjän käyttäjätunnus ja paivitysaika-sarakkeeseen tieto päivitysajasta. Nämä tiedot auttavat erilaisten virhetilanteiden jäljittämisessä. (Hovi ym. 2005, 116.) Lihastautiyhdistyksen tietokannassa luontiaika- ja paivitysaika-sarakkeet ovat tyyppiä DATETIME. Perustaja ja paivittaja ovat tyyppiä VARCHAR.

Keikka

Sarake	Tyyppi	Aakkosjärjestys	Attribuutit	Tyhjä	Oletusarvo	Lisätiedot
keikkanro	int(10)		UNSIGNED	Ei	*Ei mitään*	auto_increment
alkamisajankohta	date			Kyllä	*NULL*	
hakutapaid	int(10)		UNSIGNED	Kyllä	*NULL*	
lisatietoja	text	utf8_general_ci		Kyllä	*NULL*	
muuta	text	utf8_general_ci		Kyllä	*NULL*	
paatosid	int(10)		UNSIGNED	Ei	*Ei mitään*	
tyoaika	varchar(30)	utf8_general_ci		Ei	*Ei mitään*	
vaatimukset	text	utf8_general_ci		Kyllä	*NULL*	
keikkatyyppiid	int(10)		UNSIGNED	Ei	*Ei mitään*	
yhteystapaid	int(10)		UNSIGNED	Ei	*Ei mitään*	
perustaja	varchar(20)	utf8_general_ci		Ei	*Ei mitään*	
luontiaika	datetime			Ei	*Ei mitään*	
paivittaja	varchar(20)	utf8_general_ci		Kyllä	*NULL*	
paivitysaika	datetime			Kyllä	*NULL*	

Kuva 18. Käsitteestä Keikka muodostettu taulu

Kuvassa 19 on esitetty tietotyyppi TINYINT. MySQL-sovelluksessa ei ole erillistä boolean-tietotyyppiä (arvo voi olla joko true tai false), joten sen korvaamiseksi käytetään tinyint(1)-tietotyyppiä. Tällöin 0 tarkoittaa arvoa epätosi (false) ja muut luvut (yleensä 1) tarkoittavat arvoa tosi (true). (Oracle and/or its affiliates 2008c.) Kyseinen tietotyyppi sopii hyvin kuvaamaan kuvan 19 Avustaja-taulussa, onko avustajalla auto käytössä vai ei, sekä sitä, onko avustaja aktiivinen vai ei (eli ottaako avustaja keikkoja vastaan).

Kuvan 19 Avustaja-taulun rakenteessa on myös esitetty tietotyyppi ENUM, jota on käytetty muun muassa ajokorttiluokka-sarakkeen arvovaihtoehtojen kuvaamiseen. Nyt sarake voi saada arvoikseen vain ENUM-sarakkeessa luetellut arvot (Oracle and/or its affiliates 2008g).

Avustaja

Sarake	Tyyppi	Aakkosjärjestys	Attribuutit	Tyhjä	Oletusarvo	Lisätiedot
avustajaid	int(10)		UNSIGNED	Ei	*Ei mitään*	auto_increment
etunimi	varchar(20)	utf8_general_ci		Ei	*Ei mitään*	
sukunimi	varchar(30)	utf8_general_ci		Ei	*Ei mitään*	
katuosoite	varchar(50)	utf8_general_ci		Ei	*Ei mitään*	
postinro	char(5)	utf8_general_ci		Ei	*Ei mitään*	
kotikunta	varchar(15)	utf8_general_ci		Ei	*Ei mitään*	
puhnro	varchar(15)	utf8_general_ci		Ei	*Ei mitään*	
email	varchar(40)	utf8_general_ci		Kyllä	*NULL*	
skype	varchar(40)	utf8_general_ci		Kyllä	*NULL*	
syntymaaika	date			Kyllä	*NULL*	
henkilotunnus	varchar(11)	utf8_general_ci		Kyllä	*NULL*	
ajokorttiluokka	enum('A1','A','B','C1','C','D1','BE','C1E','CE','D1E','DE','M','T')	utf8_general_ci		Kyllä	*NULL*	
auto	tinyint(1)			Kyllä	*NULL*	
koulutus	text	utf8_general_ci		Kyllä	*NULL*	
osaaminen	text	utf8_general_ci		Kyllä	*NULL*	
rajoitteet	text	utf8_general_ci		Kyllä	*NULL*	
toivomukset	text	utf8_general_ci		Kyllä	*NULL*	
lisatiedot	text	utf8_general_ci		Kyllä	*NULL*	
tyokokemus	text	utf8_general_ci		Kyllä	*NULL*	
tyotyyppi	enum('kokopäivätyö','osa-aikatyö')	utf8_general_ci		Kyllä	*NULL*	
kayttajatunnus	varchar(20)	utf8_general_ci		Kyllä	*NULL*	
salasana	varchar(32)	utf8_general_ci		Kyllä	*NULL*	
aktiivinen	tinyint(1)			Kyllä	*NULL*	
atyyppi	int(10)		UNSIGNED	Ei	*Ei mitään*	
toimipisteid	int(10)		UNSIGNED	Ei	*Ei mitään*	
perustaja	varchar(20)	utf8_general_ci		Ei	*Ei mitään*	
luontiaika	datetime			Ei	*Ei mitään*	
paivittaja	varchar(20)	utf8_general_ci		Kyllä	*NULL*	
paivitysaika	datetime			Kyllä	*NULL*	

Kuva 19. Käsitteestä Avustaja muodostettu taulu

6.3 Perusavaimen ominaisuudet

Kuvissa 17, 18 ja 19 taulujen perusavaimet on alleviivattu ja tietotyypeiksi on valittu numeerinen INT. Lisäksi sarakkeiden ominaisuudeksi on valittu auto_increment. Tämä tarkoittaa, että tietokanta generoi automaattisesti jokaiselle riville uuden uniikin arvon (Oracle and/or its affiliates 2008a). Tällaista perusavainta kutsutaan surrogaatiksi eli keinoavaimeksi (Hovi ym. 2005, 63).

Surrogaatin perusavaimen sijasta tauluille voidaan antaa luonnollinen perusavain. Tällainen voisi olla esimerkiksi henkilötunnus Asiakas- ja Avustaja-tauluilla. On kuitenkin tilanteita, jolloin henkilötunnus voi muuttua: tietokantaan on saatettu syöttää esimerkiksi virheellinen tieto. Tämän vuoksi perusavaimeksi

on hyvä valita surrogaatti, sillä se ei koskaan muutu. (Hovi ym. 2005, 63.)

Perusavainsarakkeille on asetettu myös UNSIGNED-ominaisuus. Kyseinen tieto määrittelee INT-tietotyypin yhteydessä, että sarake voi saada arvokseen vain positiivisia kokonaislukuja.

6.4 Viite-avaimet

Suurin ero käsitemallin ja relaatiomalliin perustuvan tietokannan taulujen välillä on se, että käsitemalli muodostuu kolmenlaisista objekteista: käsitteistä, tiedoista ja yhteyksistä. Relaatiomalli muodostuu ainoastaan tauluista ja tiedoista. (Hovi ym. 2005, 104.)

Koska relaatiomallissa ei ole enää yhteysobjekteja, yksi-moneen-yhteyksistä syntyy viiteavaimia: lapsitauluun tulee isätaulun perusavain eli toisin sanoen viiteavain. (Hovi ym. 2005, 105.)

Viiteavaimia merkittäessä on huomioitava, onko yhteys isä- ja lapsitaulun välillä pakollinen vai ehdollinen. Jos yhteys on pakollinen, viiteavainsarakkeelle on annettava määritys NOT NULL, koska tarkoituksena on estää lapsitaulun mukaisen ilmentymän tallentaminen tietokantaan isätaulun ilmentymän puuttuessa. Vastaavasti NULL-arvot sallitaan, jos yhteys ei ole pakollinen – tällöinhän viiteavaimella ei ole pakko olla arvoa. (Hovi ym. 2005, 107.)

6.5 Taulujen luonti SQL-kielellä

Kun taulujen toteuttamiseen käytetään SQL-kieltä eikä esimerkiksi MySQL-sovelluksen graafista käyttöliittymää, taulut luodaan CREATE TABLE -komennolla. (Lahtonen 2002, 43.) Kuvan 17 mukainen Asiakas-taulu voidaan luoda seuraavasti:

```
CREATE TABLE Asiakas (asiakasnro INT UNSIGNED NOT NULL
AUTO_INCREMENT,
```

```
etunimi VARCHAR(20) NOT NULL,
sukunimi VARCHAR(30) NOT NULL,
katuosoite VARCHAR(50) NOT NULL,
postinro CHAR(5) NOT NULL,
kotikunta VARCHAR(15) NOT NULL,
kaupunginosa VARCHAR(30) NOT NULL,
puhnro_oma VARCHAR(15) NOT NULL,
yhthlo_oma VARCHAR(15) NOT NULL,
nronluovutus TINYINT(1),
email VARCHAR(40),
skype VARCHAR(40),
henkilotunnus VARCHAR(11),
vammaryhma TEXT,
apuvalineet TEXT,
lisatietoja TEXT,
toimipisteid INT UNSIGNED NOT NULL,
perustaja VARCHAR(20) NOT NULL,
luontiaika DATETIME NOT NULL,
paivittaja VARCHAR(20),
paivitysaika DATETIME,
CONSTRAINT Asiakas_PK PRIMARY KEY (asiakasnro),
CONSTRAINT Asiakas_FK_PTP FOREIGN KEY (postinro) REFERENCES
Postitoimipaikka (postinro) ON DELETE RESTRICT ON UPDATE CASCADE,
CONSTRAINT Asiakas_FK_TP FOREIGN KEY (toimipisteid) REFERENCES
Toimipiste (toimipisteid) ON DELETE RESTRICT ON UPDATE CASCADE)
ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Tauluja luotaessa SQL-kielellä komentoon kirjoitetaan sarakkeet, niiden tietotyypit, tieto perus- ja viiteavaimesta sekä muut mahdolliset tiedot (Lahtonen 2002, 43). Lihastautiyhdistyksen kaikille tauluille on kirjoitettu SQL-komennot valmiiksi ja ne löytyvät liitteestä 1.

6.6 Viite-eheyden toteuttamistavat MySQL-sovelluksessa

Tietokannan tauluille voidaan määrittää erilaisia viite-eheyssääntöjä, jotka ovat aina yhteyskohtaisia (Hovi ym. 2005, 50). Viite-eheyssääntöjä ovat muun muassa seuraavat:

Restrict, estosääntö: isätaulun riviä ei voida poistaa, jos lapsitaulussa on rivejä, jotka on liitetty kyseessä olevaan isätauluun. (Hovi ym. 2005, 50.)

Cascade, vyörytyssääntö: jos isätaulusta poistetaan rivi, myös lapsitaulut poistuvat automaattisesti. (Hovi ym. 2005, 50.)

Set null, tyhjäyssääntö: jos isätaulusta poistetaan rivi, siihen liitettyjen lapsitaulujen rivit saavat viiteavaimen arvoksi NULL-arvon. (Hovi ym. 2005, 50.)

Set default, oletusarvosääntö: jos isätaulusta poistetaan rivi, siihen liitettyjen lapsirivien viiteavaimen arvoksi annetaan jokin oletusarvo, joka annetaan taulua perustaessa. (Hovi ym. 2005, 50.)

MySQL-sovelluksessa voidaan laatia erityyppisiä tauluja (storage engines), joista kaikille ei voida määrittää viite-eheyssääntöjä. Oletuksena MySQL-sovelluksen taulujen tyyppi on MyISAM, joka tarjoaa suorituskykyistä varastointia ja hakuja. Jos kuitenkin halutaan luoda tauluja, jotka tukevat viiteavaimille asetettuja viite-eheyssääntöjä, taulujen tyypiksi on valittava InnoDB. (Oracle and/or its affiliates 2008e.) Tämän opinnäytetyön taulujen ajatellaan olevan InnoDB-tyyppiä, jotta tietokannasta saataisiin kaikki mahdollinen hyöty irti. Samasta syystä jokaisen SQL-kyselyn loppuun on määritelty taulun tyypiksi ENGINE-komennolla InnoDB. Komennon voisi kuitenkin jättää pois, jos oletustaulutyyppi vaihdettaisiin MySQL-sovelluksen asetuksista (Oracle and/or its affiliates 2008e).

Otetaan viite-eheyden esimerkiksi SQL-kysely, jolla luodaan taulu Toimipiste. SQL-kielessä viite-eheyssäännöt toteutetaan ON DELETE ja ON UPDATE –

komennoilla, jotka tulevat viiteavainten määritysten yhteen (Oracle and/or its affiliates 2008F). Esimerkin mukaisessa Toimipiste-taulussa postinumeron ja postitoimipaikan poistaminen on estetty Postitoimipaikka-taulusta, jos Toimipiste-taulussa on Postitoimipaikka-tauluun liittyviä arvoja. Lisäksi jos Postitoimipaikka-taulun tiedot muuttuvat, myös Toimipiste-taulun kyseiset arvot muuttuvat. Myös liitteen 1 SQL-kyselyissä viite-eheyssäännöt on huomioitu.

```
CREATE TABLE Toimipiste (toimipisteid INT UNSIGNED NOT NULL AUTO_INCREMENT,
nimi VARCHAR(30) NOT NULL,
katuosoite VARCHAR(50) NOT NULL,
postinro CHAR(5) NOT NULL,
puhnro VARCHAR(15) NOT NULL,
perustaja VARCHAR(20) NOT NULL,
luontiaika DATETIME NOT NULL,
paivittaja VARCHAR(20),
paivitysaika DATETIME,
CONSTRAINT Toimipiste_PK PRIMARY KEY (toimipisteid),
CONSTRAINT Toimipiste_FK_PTP FOREIGN KEY (postinro) REFERENCES Postitoimipaikka (postinro) ON DELETE RESTRICT ON UPDATE CASCADE)
ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

6.7 Indeksointi

Indeksejä voidaan ajatella sarakkeiden hakemistoina. Niiden avulla hakutoimintoja pystytään nopeuttamaan määrittelemällä indeksit koskemaan sellaisia taulujen sarakkeita, joita käytetään usein hakuehtoina tai joiden perusteella hakutuloksia järjestetään. (Lahtonen 2002, 54.) Muun muassa perusavain- ja viiteavainsarakkeista luodaan indeksit (Hovi ym. 2005, 105.).

SQL-kielellä indeksi luodaan CREATE INDEX -komennolla. (Lahtonen 2002, 55.) Lihastautiyhdistyksen tietokantaan kannattaa luoda indeksit taulujen

Asiakas ja Avustaja sukunimi-sarakkeille, koska tietoja haetaan usein sukunimen perusteella järjestettyinä.

CREATE INDEX Avustaja_sukunimi ON Avustaja(sukunimi)
CREATE INDEX Asiakas_sukunimi ON Asiakas(sukunimi)

Muut indeksit on esitetty liitteessä 1.

7 POHDINTA

Tietokannan suunnittelu on monivaiheinen prosessi, jota mielestäni pidetään liian helppona ohjelmistotuotannon vaiheena: taulut laaditaan miten sattuu sitä mukaan tietokantaan, kun sovelluskoodi etenee, ja rakennetta ajatellaan ohjelmakoodin, ei tietokannan perusteella. Suunnittelutyössä ei osata ottaa huomioon, että tietokanta saattaisi toimia myös paljon joustavammin ja suorituskykyisemmin. Tämä saattaa aiheuttaa ristiriitoja tietokantoja ammatikseen suunnittelevien sekä ohjelmoijien välillä: ohjelmoijat eivät ymmärrä tietokantojen suunnittelutyötä ja tietokantasuunnittelijat eivät opi mallintamaan tietokantoja ohjelmointikoodin näkökulmasta.

Tätä opinnäytetyötä tehdessä on tullut tutkittua, miten relaatiotietokannat ovat kehittyneet 2000-luvun aikana. Lähdeteoksista on voinut päätellä, että relaatiotietokantojen perusta on pysynyt lähes täysin samana näiden vuosien aikana: perusasiat, kuten tauluista koostuva rakenne, viite-eheyteen vaikuttavat säännöt ja joukko-opillisuuteen perustuva käsittely ovat säilyneet muuttumattomina.

Myöskään SQL-kieli ei ole kokenut suuria mullistuksia: Lähdeteoksista voi huomata, että tämän opinnäytetyön kappaleen 6.5, Taulujen luonti SQL-kielellä, SQL-lause on laadittu samalla tavalla jo useamman vuoden ajan.

Tietokannan mallinnustyökään ei ole kokenut 2000-luvun aikana suuria mullistuksia: Lähes kaikissa tietokantasuunnittelua käsittelevissä lähdeteoksissa

ollaan sitä mieltä, että ER-kaavion laatimisella on paras aloittaa tietokannan suunnittelutyö. UML-mallinnuskielellä sen sijaan tuntuu olevan kaikkein ristiriitaisin asema tietokantojen suunnittelussa: teokset korostavat, että UML on oliomallinnuskieli, eikä sitä ole alun perin tarkoitettu tietokantojen mallintamiseen. Kuitenkin lähes kaikissa teoksissa esitellään tapoja, joilla tietokantoja voidaan mallintaa myös UML-kielellä. Mielestäni tämä kertoo siitä, että UML:n saaman suuren suosion vuoksi kieltä halutaan oppia käyttämään myös tietokantojen mallinnukseen. Kenties UML:stä halutaan mallinnuksen valtakieli, jota kaikki ohjelmistokehityksen parissa työskentelevät osaisivat käyttää? Tämä saattaisi ratkaista omalta osaltaan tietokantoja työkseen tekevien ja ohjelmoijien väliset ristiriidat: Olisi olemassa yksi mallinnuskieli, jota molemmat osaisivat käyttää ja jota molemmat ymmärtäisivät.

Todennäköisesti samasta syystä UML elää edelleen suurimmassa kehitysvaiheessa: 2000-luvun aikana UML on esimerkiksi saanut uusia ominaisuuksia, jotka tukevat nimenomaan tietokantojen mallinnustyötä. On kuitenkin mielenkiintoista, että tätä ei heti pystyisi näkemään relaatiotietokantojen mallinnusta käsittelevistä teoksista: Vuoden 2002 teoksessa mallinnettu kaavio näyttää ainakin päällisin puolin hyvin samanlaiselta, kuin vuoden 2009 teoksessa: käsitekaavio on tehty UML:n luokkakaavion pohjalle, yhteystyypit on merkitty samalla tavalla ja metodeita ei ole käytetty mallinnustyössä.

Yhdestä asiasta voidaan kuitenkin olla varmoja: Hyvin suunniteltu tietokanta ennaltaehkäisee monia ongelmia sitä hyödyntävien järjestelmien elinkaaressa. Kun tietokanta on hyvin suunniteltu, sen rakennetta ei tarvitse koko ajan muuttaa, ja näin ollen myös tarve järjestelmän ohjelmakoodin muokkaamiseen minimoituu. Tämä puolestaan vähentää myös mahdollisten virhetilanteiden syntymistä. Tietokannan laatiminen, niin kuin muutkin ohjelmistoprojektit, kannattaa siis aloittaa perusteellisella suunnittelutyöllä. Yhteenvetona voidaankin todeta, että huonosti suunniteltu tietokanta on kuin uppoava laiva: sitä on turha lastata.

LÄHTEET

Connolly, T. & Begg C. 2002. Database Systems A Practical Approach to Design, Implementation, and Management. 3. painos. Harlow : Addison-Wesley.

Garcia-Molina, H.; Ullman, J. & Widom, J. 2009. Database systems The complete book. 2. painos. London: Prentice Hall/Pearson Education International.

Free Software Foundation, Inc 2007. GNU GENERAL PUBLIC LICENSE. Viitattu 12.5.2010 http://www.gnu.org/copyleft/gpl.html

Haikala, I. & Märijärvi, J. 2004. Ohjelmistotuotanto. 10. painos. Helsinki: Talentum Media Oy.

Heinisuo, R. & Rauta, I. 2007. PHP ja MySQL Tietokantapohjaiset verkkopalvelut. 4. painos. Helsinki: Talentum Media Oy.

Hovi, A.; Huotari, J & Lähdenmäki, T. 2005. Tietokantojen suunnittelu & indeksointi. 1. painos. Jyväskylä: Docendo Finland Oy.

Hovi, A. 2004. SQL-opas. 1. painos. Jyväskylä: Docendo Finland Oy.

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

Lehtinen, J. 2008. Sun ostaa MySQL:n miljardilla dollarilla. Digitoday. Viitattu 20.10.2009 http://www.digitoday.fi/data/2008/01/16/sun-ostaa-mysqln-miljardilla-dollarilla/20081458/66

Moisio, A. 2008. MySQL – suomalainen menestystarina. Digitoday. Viitattu 20.10.2009 http://www.digitoday.fi/bisnes/2008/01/16/mysql--suomalainen-menestystarina/20081468/66

Moisio, A. 2009. Suomalainen isä: Oraclen pitäisi myydä MySQL. Digitoday. Viitattu 20.10.2009 http://www.digitoday.fi/bisnes/2009/10/19/suomalainen-isa-oraclen-pitaisi-myyda-mysql/200922239/66

Objecteering Software 2008. Download Objecteering UML Free Edition. Viitattu 29.3.2010 http://www.objecteering.com/downloads_uml_free_edition.php

Oracle and/or its affiliates 2008a. MySQL 5.1 Reference Manual. Viitattu 13.4.2010 http://dev.mysql.com/doc/refman/5.1/en/example-auto-increment.html

Oracle and/or its affiliates 2008b. MySQL 5.1 Reference Manual. Viitattu 13.4.2010 http://dev.mysql.com/doc/refman/5.1/en/date-and-time-type-overview.html

Oracle and/or its affiliates 2008c. MySQL 5.1 Reference Manual. Viitattu 13.4.2010 http://dev.mysql.com/doc/refman/5.1/en/numeric-type-overview.html

Oracle and/or its affiliates 2008d. MySQL 5.1 Reference Manual. Viitattu 26.4.2010 http://dev.mysql.com/doc/refman/5.1/en/blob.html

Oracle and/or its affiliates 2008e. MySQL 5.1 Reference Manual. Viitattu 26.4.2010 http://dev.mysql.com/doc/refman/5.1/en/storage-engines.html

Oracle and/or its affiliates 2008f. MySQL 5.1 Reference Manual. Viitattu 26.4.2010
http://dev.mysql.com/doc/refman/5.1/en/innodb-foreign-key-constraints.html

Oracle and/or its affiliates 2008g. MySQL 5.1 Reference Manual. Viitattu 26.4.2010
http://dev.mysql.com/doc/refman/5.1/en/enum.html

LIITE 1 TAULUJEN SQL-KOMENNOT

Taulu Asiakas

```
CREATE TABLE Asiakas (asiakasnro INT UNSIGNED NOT NULL AUTO_INCREMENT,
etunimi VARCHAR(20) NOT NULL,
sukunimi VARCHAR(30) NOT NULL,
katuosoite VARCHAR(50) NOT NULL,
postinro CHAR(5) NOT NULL,
kotikunta VARCHAR(15) NOT NULL,
kaupunginosa VARCHAR(30) NOT NULL,
puhnro_oma VARCHAR(15) NOT NULL,
yhthlo_oma VARCHAR(15) NOT NULL,
nronluovutus TINYINT(1),
email VARCHAR(40),
skype VARCHAR(40),
henkilotunnus VARCHAR(11),
vammaryhma TEXT,
apuvalineet TEXT,
lisatietoja TEXT,
toimipisteid INT UNSIGNED NOT NULL,
perustaja VARCHAR(20) NOT NULL,
luontiaika DATETIME NOT NULL,
paivittaja VARCHAR(20),
paivitysaika DATETIME,
CONSTRAINT Asiakas_PK PRIMARY KEY (asiakasnro),
CONSTRAINT Asiakas_FK_PTP FOREIGN KEY (postinro) REFERENCES Postitoimipaikka (postinro) ON DELETE RESTRICT ON UPDATE CASCADE,
CONSTRAINT Asiakas_FK_TP FOREIGN KEY (toimipisteid) REFERENCES Toimipiste (toimipisteid) ON DELETE RESTRICT ON UPDATE CASCADE)
ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
CREATE INDEX Asiakas_sukunimi ON Asiakas(sukunimi)
```

Taulu Avustaja

```
CREATE TABLE Avustaja (avustajaid INT UNSIGNED NOT NULL AUTO_INCREMENT,
etunimi VARCHAR(20) NOT NULL,
sukunimi VARCHAR(30) NOT NULL,
katuosoite VARCHAR(50) NOT NULL,
postinro CHAR(5) NOT NULL,
kotikunta VARCHAR(15) NOT NULL,
puhnro VARCHAR(15) NOT NULL,
email VARCHAR(40),
skype VARCHAR(40),
syntymaaika DATE,
henkilotunnus VARCHAR(11),
ajokorttiluokka ENUM('A1', 'A', 'B', 'C1', 'C','D1', 'BE', 'C1E', 'CE', 'D1E', 'DE', 'M', 'T'),
auto TINYINT(1),
koulutus TEXT,
osaaminen TEXT,
rajoitteet TEXT,
toivomukset TEXT,
lisatiedot TEXT,
tyokokemus TEXT,
tyotyyppi ENUM('kokopäivätyö', 'osa-aikatyö'),
kayttajatunnus VARCHAR(20),
salasana VARCHAR(32),
aktiivinen TINYINT(1),
atyyppi INT UNSIGNED NOT NULL,
toimipisteid INT UNSIGNED NOT NULL,
```

```
perustaja VARCHAR(20) NOT NULL,
luontiaika DATETIME NOT NULL,
paivittaja VARCHAR(20),
paivitysaika DATETIME,
CONSTRAINT Avustaja_PK PRIMARY KEY (avustajaid),
CONSTRAINT Avustaja_FK_PTP FOREIGN KEY (postinro) REFERENCES
Postitoimipaikka (postinro) ON DELETE RESTRICT ON UPDATE CASCADE,
CONSTRAINT Avustaja_FK_TP FOREIGN KEY (toimipisteid) REFERENCES
Toimipiste (toimipisteid) ON DELETE RESTRICT ON UPDATE CASCADE,
CONSTRAINT Avustaja_FK_Atyyppi FOREIGN KEY (atyyppi) REFERENCES
Avustajatyyppi (atyyppiid) ON DELETE RESTRICT ON UPDATE CASCADE)
ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE INDEX Avustaja_sukunimi ON Avustaja(sukunimi)
```

Taulu Avustajatyyppi

```
CREATE TABLE Avustajatyyppi(atyyppiid INT UNSIGNED NOT NULL
AUTO_INCREMENT,
atyyppi VARCHAR(30) NOT NULL,
CONSTRAINT Avustajatyyppi_PK PRIMARY KEY (atyyppiid))
ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Taulu Avustaminen

```
CREATE TABLE Avustaminen(avustajaid INT UNSIGNED NOT NULL,
keikkanro INT UNSIGNED NOT NULL,
tuntimaara DECIMAL,
palkka DECIMAL,
lisatietoja TEXT,
CONSTRAINT Avustaminen_PK PRIMARY KEY (avustajaid, keikkanro),
CONSTRAINT Avustaminen_FK_Avu FOREIGN KEY (avustajaid)
```

```
REFERENCES Avustaja(avustajaid) ON DELETE RESTRICT ON UPDATE
CASCADE,
CONSTRAINT Avustaminen_FK_Kei FOREIGN KEY (keikkanro)
REFERENCES Keikka(keikkanro) ON DELETE RESTRICT ON UPDATE
CASCADE)
ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Taulu Avustamiskategoria

```
CREATE TABLE Avustamiskategoria(avustajaid INT UNSIGNED NOT NULL,
kategoriaid INT UNSIGNED NOT NULL,
CONSTRAINT Avustamiskategoria_PK PRIMARY KEY (avustajaid,
kategoriaid),
CONSTRAINT Avustamiskategoria_FK_Avu FOREIGN KEY (avustajaid)
REFERENCES Avustaja(avustajaid) ON DELETE RESTRICT ON UPDATE
CASCADE,
CONSTRAINT Avustamiskategoria_FK_Kat FOREIGN KEY (kategoriaid)
REFERENCES Potilaskategoria(kategoriaid) ON DELETE RESTRICT ON
UPDATE CASCADE)
ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Taulu Hakutapa

```
CREATE TABLE Hakutapa (hakutapaid INT UNSIGNED NOT NULL
AUTO_INCREMENT,
hakutapa VARCHAR(30) NOT NULL,
CONSTRAINT Hakutapa_PK PRIMARY KEY (hakutapaid))
ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Taulu Ilmoittaja

```
CREATE TABLE Ilmoittaja(keikkanro INT UNSIGNED NOT NULL,
```

```
ilmoittaja VARCHAR(10) NOT NULL,
puhnro VARCHAR(10) NOT NULL,
CONSTRAINT Ilmoittaja_PK PRIMARY KEY (keikkanro, ilmoittaja),
CONSTRAINT Ilmoittaja_FK_Kei FOREIGN KEY (keikkanro) REFERENCES
Keikka (keikkanro) ON DELETE RESTRICT ON UPDATE CASCADE)
ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Taulu Keikka

```
CREATE TABLE Keikka (keikkanro INT UNSIGNED NOT NULL
AUTO_INCREMENT,
alkamisajankohta DATE,
hakutapaid INT UNSIGNED,
lisatietoja TEXT,
muuta TEXT,
paatosid INT UNSIGNED NOT NULL,
tyoaika VARCHAR(30) NOT NULL,
vaatimukset TEXT,
keikkatyyppiid INT UNSIGNED NOT NULL,
yhteystapaid INT UNSIGNED NOT NULL,
perustaja VARCHAR(20) NOT NULL,
luontiaika DATETIME NOT NULL,
paivittaja VARCHAR(20),
paivitysaika DATETIME,
CONSTRAINT Keikka_PK PRIMARY KEY (keikkanro),
CONSTRAINT Keikka_FK_HakuT FOREIGN KEY (hakutapaid) REFERENCES
Hakutapa (hakutapaid) ON DELETE RESTRICT ON UPDATE CASCADE,
CONSTRAINT Keikka_FK_Ktyyppi FOREIGN KEY (keikkatyyppiid)
REFERENCES Keikkatyyppi(keikkatyyppiid) ON DELETE RESTRICT ON
UPDATE CASCADE,
CONSTRAINT Keikka_FK_yhteystapa FOREIGN KEY (yhteystapaid)
REFERENCES Yhteystapa(yhteystapaid) ON DELETE RESTRICT ON
```

```
UPDATE CASCADE,
CONSTRAINT Keikka_FK_paatos FOREIGN KEY (paatosid) REFERENCES Paatos(paatosid) ON DELETE RESTRICT ON UPDATE CASCADE)
ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE INDEX Keikka_alkupvm ON Keikka(alkupvm)
CREATE INDEX Keikka_loppupvm ON Keikka(loppupvm)
```

Taulu Keikkakerta

```
CREATE TABLE Keikkakerta (keikkakertaid INT UNSIGNED NOT NULL,
pvm DATE NOT NULL,
tuntimaara DECIMAL NOT NULL,
alkuklo TIME NOT NULL,
loppuklo TIME NOT NULL,
kulukorvaus DECIMAL,
lisatietoja TEXT,
kuvaus TEXT,
keikkanro INT UNSIGNED NOT NULL,
taksi_tilaaja VARCHAR(10) NOT NULL,
taksi_klo TIME,
perustaja VARCHAR(20) NOT NULL,
luontiaika DATETIME NOT NULL,
paivittaja VARCHAR(20),
paivitysaika DATETIME,
CONSTRAINT Keikkakerta_PK PRIMARY KEY (keikkakertaid),
CONSTRAINT Keikkakerta _FK_Kei FOREIGN KEY (keikkanro) REFERENCES Keikka (keikkanro) ON DELETE RESTRICT ON UPDATE CASCADE)
ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Taulu Keikkatyyppi

```
CREATE TABLE Keikkatyyppi(keikkatyyppiid INT UNSIGNED NOT NULL AUTO_INCREMENT,
keikkatyyppi VARCHAR(30) NOT NULL,
CONSTRAINT Keikkatyyppi_PK PRIMARY KEY (keikkatyyppiid))
ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Taulu Keikkayhteistyo

```
CREATE TABLE Keikkayhteistyo(kumppaniid INT UNSIGNED NOT NULL,
keikkanro INT UNSIGNED NOT NULL,
CONSTRAINT Keikkahteistyo_PK PRIMARY KEY (kumppaniid, keikkanro),
CONSTRAINT Keikkahteistyo_FK_Yht FOREIGN KEY (kumppaniid) REFERENCES Yhteistyokumppani(kumppaniid) ON DELETE RESTRICT ON UPDATE CASCADE,
CONSTRAINT Keikkahteistyo_FK_Keik FOREIGN KEY (keikkanro) REFERENCES Keikka(keikkanro) ON DELETE RESTRICT ON UPDATE CASCADE)
ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Taulu Kielitaito

```
CREATE TABLE Kielitaito(avustajaid INT UNSIGNED NOT NULL,
kieli VARCHAR(10) NOT NULL,
CONSTRAINT Kielitaito_PK PRIMARY KEY (avustajaid, kieli),
CONSTRAINT Kielitaito_FK_Avu FOREIGN KEY (avustajaid) REFERENCES Avustaja(avustajaid) ON DELETE RESTRICT ON UPDATE CASCADE)
ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE INDEX Kielitaito_kieli ON Kielitaito(kieli)
```

Taulu Kiinnostus

```
CREATE TABLE Kiinnostus (avustajaid INT UNSIGNED NOT NULL,
kiinnostus VARCHAR(50) NOT NULL,
CONSTRAINT Kiinnostus_PK PRIMARY KEY (avustajaid, kiinnostus),
CONSTRAINT Kiinnostus _FK_AS FOREIGN KEY (avustajaid) REFERENCES
Avustaja(avustajaid) ON DELETE RESTRICT ON UPDATE CASCADE))
ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Taulu Kurssi

```
CREATE TABLE Kurssi (kurssiid INT UNSIGNED NOT NULL
AUTO_INCREMENT,
aihe VARCHAR(50),
sijainti VARCHAR (50),
kuvaus TEXT,
hinta INT,
esitietovaatimukset VARCHAR (50),
ok_tuki TINYINT(1),
perustaja VARCHAR(20) NOT NULL,
luontiaika DATETIME NOT NULL,
paivittaja VARCHAR(20),
paivitysaika DATETIME,
CONSTRAINT Kurssi_PK PRIMARY KEY (kurssiid))
ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE INDEX Kurssi_aihe ON Kurssi(aihe)
```

Taulu Kurssikerta

```
CREATE TABLE Kurssikerta (kurssiid INT UNSIGNED NOT NULL,
alkaa DATETIME NOT NULL,
```

```
paattyy DATETIME NOT NULL,
perustaja VARCHAR(20) NOT NULL,
luontiaika DATETIME NOT NULL,
paivittaja VARCHAR(20),
paivitysaika DATETIME,
CONSTRAINT Kurssikerta_PK PRIMARY KEY (kurssiid, alkaa, paattyy),
CONSTRAINT Kurssikerta_FK_PTP FOREIGN KEY (kurssiid) REFERENCES
Kurssi(kurssiid) ON DELETE RESTRICT ON UPDATE CASCADE)
ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE INDEX Kurssikerta_alkaa ON Kurssikerta(alkaa)
CREATE INDEX Kurssikerta_paattyy ON Kurssikerta(paattyy)
```

Taulu Kurssiosallistuminen

```
CREATE TABLE Kurssiosallistuminen (kurssiid INT UNSIGNED NOT NULL,
avustajaid INT UNSIGNED NOT NULL,
rooli VARCHAR(30),
CONSTRAINT Kurssiosallistuminen_PK PRIMARY KEY (kurssiid, avustajaid),
CONSTRAINT Kurssiosallistuminen_FK_Kur FOREIGN KEY (kurssiid)
REFERENCES Kurssi(kurssiid) ON DELETE RESTRICT ON UPDATE
CASCADE,
CONSTRAINT Kurssiosallistuminen_FK_Avu FOREIGN KEY (avustajaid)
REFERENCES Avustaja(avustajaid) ON DELETE RESTRICT ON UPDATE
CASCADE)
ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Taulu Lomake

```
CREATE TABLE Lomake (lomakeid INT UNSIGNED NOT NULL
AUTO_INCREMENT,
tiedostonimi VARCHAR(50),
```

```
nimi VARCHAR (50),
kurssiid INT UNSIGNED,
asiakasnro INT UNSIGNED,
avustajaid INT UNSIGNED,
perustaja VARCHAR(20) NOT NULL,
luontiaika DATETIME NOT NULL,
paivittaja VARCHAR(20),
paivitysaika DATETIME,
CONSTRAINT Lomake_PK PRIMARY KEY (lomakeid),
CONSTRAINT Lomake_FK_kurs FOREIGN KEY (kurssiid) REFERENCES Kurssi (kurssiid) ON DELETE RESTRICT ON UPDATE CASCADE,
CONSTRAINT Lomake_FK_As FOREIGN KEY (asiakasnro) REFERENCES Asiakas (asiakasnro) ON DELETE RESTRICT ON UPDATE CASCADE,
CONSTRAINT Lomake_FK_Av FOREIGN KEY (avustajaid) REFERENCES Avustaja (avustajaid) ON DELETE RESTRICT ON UPDATE CASCADE)
ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE INDEX Lomake_tiedostonimi ON Lomake(tiedostonimi)
CREATE INDEX Lomake_nimi ON Lomake(nimi)
```

Taulu Lomakekategoria

```
CREATE TABLE Lomakekategoria(lomakeid INT UNSIGNED NOT NULL,
kategoria VARCHAR(20) NOT NULL,
CONSTRAINT Lomakekategoria_PK PRIMARY KEY (lomakeid, kategoria),
CONSTRAINT Lomakekategoria_FK_Lo FOREIGN KEY (lomakeid) REFERENCES Lomake (lomakeid) ON DELETE RESTRICT ON UPDATE CASCADE)
ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Taulu Osallistuminen

```
CREATE TABLE Osallistuminen(asiakasnro INT UNSIGNED NOT NULL,
keikkanro INT UNSIGNED NOT NULL,
lisatietoja TEXT,
CONSTRAINT Osallistuminen_PK PRIMARY KEY (asiakasnro, keikkanro),
CONSTRAINT Osallistuminen_FK_Asi FOREIGN KEY (asiakasnro) REFERENCES Asiakas(asiakasnro) ON DELETE RESTRICT ON UPDATE CASCADE,
CONSTRAINT Osallistuminen_FK_Kei FOREIGN KEY (keikkanro) REFERENCES Keikka(keikkanro) ON DELETE RESTRICT ON UPDATE CASCADE)
ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Taulu Paatos

```
CREATE TABLE Paatos (paatosid INT UNSIGNED NOT NULL AUTO_INCREMENT,
asiakasnro INT UNSIGNED NOT NULL,
kuvaus TEXT,
CONSTRAINT Paatos_PK PRIMARY KEY (paatosid),
CONSTRAINT Paatos_FK_Asi FOREIGN KEY (asiakasnro) REFERENCES Asiakas (asiakasnro) ON DELETE RESTRICT ON UPDATE CASCADE)
ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Taulu Postitoimipaikka

```
CREATE TABLE Postitoimipaikka (postinro CHAR(5) NOT NULL,
postitoimipaikka VARCHAR(15) NOT NULL,
CONSTRAINT Postitp_PK PRIMARY KEY (postinro))
ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

CREATE INDEX Postitoimipaikka_postitoimipaikka ON Postitoimipaikka(postitoimipaikka)

Taulu Potilaskategoria

CREATE TABLE Potilaskategoria (asiakasnro INT UNSIGNED NOT NULL,
kategoria VARCHAR(50) NOT NULL,
CONSTRAINT Potilaskategoria_PK PRIMARY KEY (asiakasnro, kategoria),
CONSTRAINT Potilaskategoria_FK_AS FOREIGN KEY (asiakasnro) REFERENCES Asiakas(asiakasnro) ON DELETE RESTRICT ON UPDATE CASCADE))
ENGINE=InnoDB DEFAULT CHARSET=utf8;

Taulu Toimipiste

CREATE TABLE Toimipiste (toimipisteid INT UNSIGNED NOT NULL AUTO_INCREMENT,
nimi VARCHAR(30) NOT NULL,
katuosoite VARCHAR(50) NOT NULL,
postinro CHAR(5) NOT NULL,
puhnro VARCHAR(15) NOT NULL,
perustaja VARCHAR(20) NOT NULL,
luontiaika DATETIME NOT NULL,
paivittaja VARCHAR(20),
paivitysaika DATETIME,
CONSTRAINT Toimipiste_PK PRIMARY KEY (toimipisteid),
CONSTRAINT Toimipiste_FK_PTP FOREIGN KEY (postinro) REFERENCES Postitoimipaikka (postinro) ON DELETE RESTRICT ON UPDATE CASCADE)
ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE INDEX Toimipiste_nimi ON Toimipiste(nimi)

Taulu Tyokokemus

CREATE TABLE Tyokokemus (tyokokemusid INT UNSIGNED NOT NULL,
avustajaid INT UNSIGNED NOT NULL,
tyonantaja VARCHAR(40) NOT NULL,
alkupvm DATE NOT NULL,
loppupvm DATE NOT NULL,
tyotehtavat TEXT,
CONSTRAINT Tyokokemus _PK PRIMARY KEY (Tyokokemusid),
CONSTRAINT Tyokokemus_FK_Av FOREIGN KEY (avustajaid) REFERENCES Avustaja (avustajaid) ON DELETE RESTRICT ON UPDATE CASCADE)
ENGINE=InnoDB DEFAULT CHARSET=utf8;

Taulu Yhteistyö

CREATE TABLE Yhteistyo(kumppaniid INT UNSIGNED NOT NULL,
toimipisteid INT UNSIGNED NOT NULL,
lisatietoja TEXT,
CONSTRAINT Yhteistyo_PK PRIMARY KEY (kumppaniid, toimipisteid),
CONSTRAINT Yhteistyo_FK_Yht FOREIGN KEY (kumppaniid) REFERENCES Yhteistyokumppani(kumppaniid) ON DELETE RESTRICT ON UPDATE CASCADE,
CONSTRAINT Yhteistyo_FK_Tp FOREIGN KEY (toimipisteid) REFERENCES Toimipiste(toimipisteid) ON DELETE RESTRICT ON UPDATE CASCADE)
ENGINE=InnoDB DEFAULT CHARSET=utf8;

Taulu Yhteistyökumppani

CREATE TABLE Yhteistyokumppani (kumppaniid INT UNSIGNED NOT NULL AUTO_INCREMENT,
nimi VARCHAR(30),

katuosoite VARCHAR(50) NOT NULL,
postinro CHAR(5) NOT NULL,
puhnro VARCHAR(15) NOT NULL,
yhteyshenkilo_enimi VARCHAR(20),
yhteyshenkilo_snimi VARCHAR(20),
kayttajatunnus VARCHAR(20),
salasana VARCHAR(32),
perustaja VARCHAR(20) NOT NULL,
luontiaika DATETIME NOT NULL,
paivittaja VARCHAR(20),
paivitysaika DATETIME,
CONSTRAINT Yhteistyokumppani_PK PRIMARY KEY (kumppaniid),
CONSTRAINT Yhteist_FK_PTP FOREIGN KEY (postinro) REFERENCES Postitoimipaikka (postinro) ON DELETE RESTRICT ON UPDATE CASCADE)
ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE INDEX Yhteistyokumppani_nimi ON Yhteistyokumppani(nimi)
CREATE INDEX Yhteistyokumppani_yhteyshenkilo_snimi ON Yhteistyokumppani(yhteyshenkilo_snimi)

Taulu Yhteydenotto

CREATE TABLE Yhteydenotto(avustajaid INT UNSIGNED NOT NULL,
keikkaid INT UNSIGNED NOT NULL,
CONSTRAINT Ilmoittaja_PK PRIMARY KEY (avustajaid, keikkaid),
CONSTRAINT Yhteydenotto_FK_avu FOREIGN KEY (avustajaid) REFERENCES Avustaja (avustajaid) ON DELETE RESTRICT ON UPDATE CASCADE,
CONSTRAINT Yhteydenotto_FK_Kei FOREIGN KEY (keikkanro) REFERENCES Keikka (keikkanro) ON DELETE RESTRICT ON UPDATE CASCADE)
ENGINE=InnoDB DEFAULT CHARSET=utf8;

Taulu Yhteystapa

```
CREATE TABLE Yhteystapa(yhteystapaid INT UNSIGNED NOT NULL,
yhteystapa VARCHAR(40) NOT NULL,
CONSTRAINT Yhteystapa_PK PRIMARY KEY (yhteystapaid))
ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

YHDISTYS INTRANET 2.0

– Määrittelydokumentti IEEE 830

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

VERSIOHISTORIA

Versio	Tekijä/päivittäjä	Päivitysaika	Selite
1.0	Niko Skogström	Kevät 2009	Ensimmäinen versio
2.0	Janina Puistovaara	Kevät 2010	Lisätty muun muassa käyttöliittymäkuvat, tietokannan tiedot, toiminnon kuvaukset päivitettiin, järjestelmä määriteltiin selainpohjaiseksi

SISÄLTÖ

KUVAT

LIITTEET

1 JOHDANTO

1.1 Tarkoitus

Tämä dokumentti sisältää toiminnallisen määrittelyn Turun seudun lihastautiyhdistys ry:lle (myöhemmin Lihastautiyhdistys) rakennettavasta järjestelmästä, jota tullaan käyttämään yhdistyksen toimintojen ohjaukseen. Dokumentissa esitellään järjestelmän toiminnot, kuten eri osiot, käyttöliittymä, ympäristö sekä muut vaatimukset.

Suunnittelu tullaan tekemään tämän dokumentin pohjalta. Lisäksi dokumentti on toimittajan ja tilaajan välinen sopimus siitä, minkälainen järjestelmä tullaan toteuttamaan.

1.2 Tuote

Tuotteena toimii selainpohjainen intranet-järjestelmä, jolla pystytään helposti graafisen WWW−käyttöliittymän avulla ohjaamaan Lihastautiyhdistyksen toimintoja ja tallentamaan tarpeellisia tietoja tietokantaan. Tuote korvaa yhdistyksen entiset toimintatavat ja yhdistää Lihastautiyhdistyksen tiedot yhden järjestelmän alle.

1.3 Dokumentin kattavuus

Tässä dokumentissä kuvataan toiminnallinen määrittely kokonaisuudessaan.

1.4 Määritelmät, termit, lyhenteet

IEEE-standardi: Kansainvälisen standardoimisjärjestön luoma standardi.

1.5 Viitteet ja muut liittyvät dokumentit

Tämä dokumentti perustuu Niko Skogströmin laatimaan esitutkimukseen Lihastautiyhdistyksen asiakasvaatimuksista. Lisäksi tämä dokumentti on päivitetty versio Niko Skogströmin laatimasta IEEE 830-standardin mukaisesta määrittelydokumentista.

Määrittely tehdään IEEE830 ja suunnittelu IEEE1016 standardin mukaan. Testauksessa käytetään IEEE829 standardia. Standardeista saatetaan poiketa silloin, kun se on ohjelmistosuunnittelun kannalta suositeltavaa. Poikkeukset eivät kuitenkaan vaikuta dokumenttien laatuun tai kattavuuteen.

1.6 Yleiskatsaus dokumenttiin

Luku 1 käsittelee määrittelyprosessia ja sen dokumentointia yleisesti.
Luvussa 2 käydään tarkemmin läpi ohjelmiston kuvaus.
Luku 3 sisältää kuvauksen ohjelmiston tietokannasta ja sen käsittelemistä tiedoista.
Luku 4 listaa kaikkien päätoimintojen kuvaukset.
Luku 5 kuvaa ohjelmiston ulkoisen toiminnan.
Luku 6 esittelee kaikki muut ominaisuudet, mitkä vaikuttavat ohjelmiston toimintaan.
Luvussa 7 käydään läpi ohjelmiston suunnittelun rajoitukset.

2 YLEISKUVAUS

2.1 Ympäristö

Ohjelmisto tulee ensin koekäyttöön Turun toimipisteeseen, jossa tapahtuu testaus ja ensimmäinen käyttöönotto. Käyttö laajenee myöhemmin myös muihin Lihastautiyhdistyksen toimipisteisiin.

Tuotteen sovellusympäristönä toimii WWW−selain. Pääasiallisena ajoympäristönä järjestelmällä on WWW−palvelin, jonka tarkemmat määritykset selviävät ohjelmiston suunnitteluvaiheessa.

2.2 Toiminta

Tuotetta käytetään graafisella selainkäyttöliittymällä. Tiedot tallennetaan tietokantaan.

2.3 Käyttäjät

Käyttäjinä toimivat Turun seudun lihastautiyhdistys ry:n työntekijät, asiakkaat ja yhdistyksen hyväksymät yhteistyökumppanit. Työntekijät käyttävät ohjelmistoa täyspainoisesti kaikissa työtehtävissä ja kaikissa työpisteissä.

Yhteistyökumppanit käyttävät ympäristöä rajatusti ja saavat vain rajoitetun pääsyn ohjelmistoon. Kumppanit pääsevät kirjautumaan järjestelmään.

Asiakkaat eivät ainakaan alustavien määritysten mukaan kirjaudu järjestelmään, mutta he voivat käyttää sovellusta täyttämällä ja lähettämällä internetin kautta erilaisia sähköisiä lomakkeita, jotka yhdistyksen työntekijät voivat käydä hyväksymässä järjestelmään. Tällainen on esimerkiksi henkilötietolomake.

2.4 Yleiset rajoitteet

Lihastautiyhdistyksen työntekijöillä ja yhteistyökumppaneilla on henkilökohtaiset käyttäjätunnukset ja salasanat järjestelmään. Asiakkaille ei ainakaan näillä näkymin rekisteröidä tunnuksia järjestelmään, mutta jos Lihastautiyhdistys katsoo myöhemmin asian tarpeelliseksi, asiakkaille voidaan antaa omat käyttäjätunnukset järjestelmään rajoitetuilla oikeuksilla.

Järjestelmän käyttöliittymän on oltava yksinkertainen ja helppo käyttää. Käyttäjältä ei vaadita kuin perustietojenkäsittelytaitoja.

Järjestelmän on oltava tietoturvallinen, jotta ulkopuoliset käyttäjät eivät pääse näkemään tai muuttamaan niitä sivustojen tietoja, joihin näillä ei ole oikeutta. Lisäksi useamman käyttäjän on pystyttävä käyttämään järjestelmää samanaikaisesti.

2.5 Oletukset ja riippuvuudet

Ohjelmiston käyttöön vaaditaan internet-yhteys.

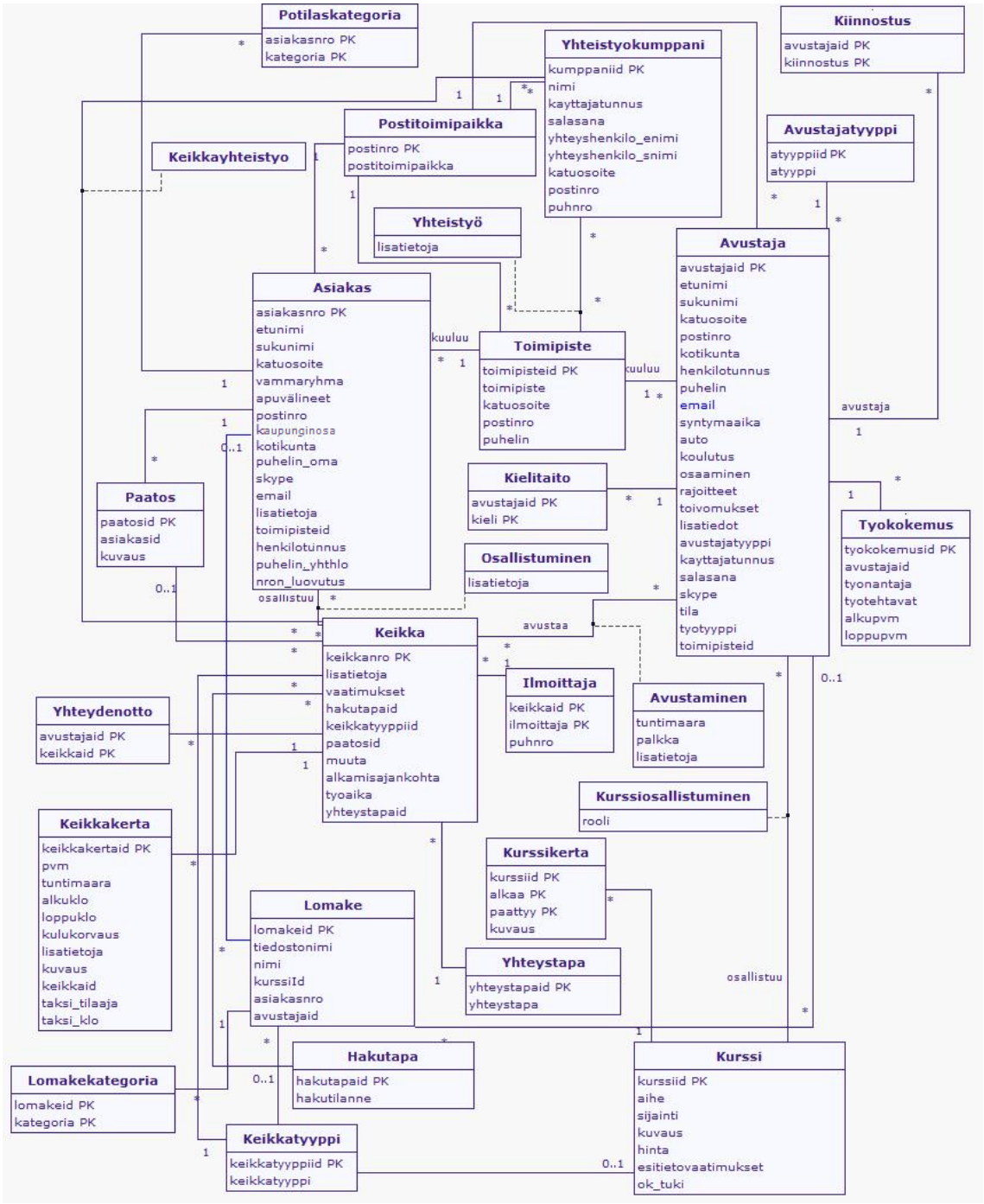
3 TIEDOT JA TIETOKANTA

3.1 Tietokanta

Tietokantaan tallennetaan tieto, jonka halutaan jäävän järjestelmään. Tietokantana toimii relaatiotietokanta ja tietokannan hallintajärjestelmänä käytetään MySQL-sovellusta. Tietokannasta mallinnettu ER-kaavio on esitetty kuvassa 1.

Osa tietokannan tiedoista on käyttäjien syöttämää tietoa, mutta jotkut tiedoista (esim. asiakasnumero) järjestelmä generoi automaattisesti.

Taulujen SQL-komennot on esitetty liitteessä 1, josta käyvät ilmi myös sarakkeiden tietotyypit.



Kuva 1. Tietokannan ER-kaavio

3.2 Taulut

Avustaja: Sisältää tiedot tietokantaan tallennettavista avustajista ja avustajiksi hakevista. Tähän tauluun on hyvä lisätä myös tiedot perustaja, luontiaika, paivittaja ja paivitysaika, joiden avulla voidaan seurata lokitietoja ja esimerkiksi jäljittää virheitä.

Avustajatyyppi: Sisältää tiedon avustajan tyypistä. Alustavasti avustajatyyppejä on neljä: Vapaaehtoinen avustaja, palkallinen avustaja, työtä hakeva vapaaehtoinen avustaja ja työtä hakevia.

Kiinnostus: Sisältää tiedon siitä, minkä tyyppisistä keikoista avustaja on kiinnostunut.

Työkokemus: Sisältää tiedon avustajan (tai työtä hakevan) työkokemuksesta.

Kielitaito: Sisältää tiedon avustajan kielitaidosta.

Postitoimipaikka: Sisältää eri postitoimipaikkojen tiedot.

Asiakas: Pitää sisällään tiedot yhdistyksen asiakkaista.

Päätös: Sisältää tiedot palkallisia avustajia hakevien asiakkaiden avustajapäätöksistä (eli asiakkaalla on todettu tarve esim. henkilökohtaiseen tai vapaa-ajan avustajaan).

Potilaskategoria: Sisältää tiedon siitä, mihin potilaskategoriaan asiakas kuuluu.

Toimipiste: Pitää sisällään Lihastautiyhdistyksen eri toimipisteiden tiedot.

Yhteistyökumppani: Sisältää yhdistyksen yhteistyökumppaneiden tiedot. On yhteydessä tauluun yhteistyö, joka sisältää tiedon siitä, mitkä toimipisteet ovat yhteydessä minkäkin yhteistyökumppanin kanssa.

Keikka: Sisältää tiedot järjestelmään tallennettavista keikoista. On yhteydessä tauluun osallistuminen, johon tulee tiedot keikalle osallistuvista asiakkaista, tauluun avustaminen, johon tulee tiedot keikan avustajista, sekä tauluun Keikkayhteistyö, johon tulee tiedot keikkaan liittyvistä yhteistyökumppaneista.

Keikkatyyppi: Sisältää tiedon siitä, onko keikka palkallisen vai vapaaehtoisen avustajan keikka.

Keikkakerta: Sisältää yhden keikkakerran tiedot (yksi keikka voi sisältää useamman tapaamisen).

Ilmoittaja: Sisältää tiedon siitä, kuka on ilmoittanut keikan tiedot (esim. asiakas, kotipalvelu, jokin muu taho).

Yhteydenotto: Sisältää tiedot avustajista, joita on jo pyydetty jollekin keikalle.

Hakutapa: Pitää sisällään tiedon palkallisten avustajien keikan hakutavasta, esim. avustajakeskus tekee ilmoituksen, ilmoitus tehdään avustettavan puolesta, mutta avustettava hoitaa itse rekrytoinnin, avustettava hoitaa itse koko rekrytoinnin. Voidaan tarvittaessa hakujen nopeuttamiseksi denormalisoida tauluun Keikka.

Yhteystapa: Sisältää tiedon siitä, miten asiakas ja avustaja ovat yhteydessä toisiinsa

Lomake: Sisältää tiedot lomakepankkiin tallennettavista lomakkeista.

Lomakekategoria: Sisältää tiedon kunkin lomakkeen kategoriasta

Kurssi: Sisältää tiedot Lihastautiyhdistyksen järjestämistä kursseista, luennoista ja koulutuksista. On yhteydessä tauluun Kurssiosallistuminen, joka sisältää tiedon kursseille osallistuvista avustajista sekä heidän rooleistaan (esim. onko kyseessä kouluttaja).

Kurssikerta: Sisältää yhden kurssikerran tiedot (yksi kurssi voi sisältää monta kurssikertaa).

3.3 Muut tiedot

Tauluihin on hyvä lisätä myös tiedot perustaja, luontiaika, paivittaja ja paivitysaika, joiden avulla voidaan seurata lokitietoja ja esimerkiksi jäljittää virheitä. Näiden tietojen avulla voidaan selvittää, kuka on lisännyt esim. tietyn asiakkaan tiedot järjestelmään tai milloin ja kenen toimesta kyseisen asiakkaan tietoja on viimeksi päivitetty. Lokitiedoille voidaan tarvittaessa myös perustaa oma taulunsa.

3.4 Käyttöintensiteetti

Oletuksena on, että sovellusta käytetään pääosin arkipäiväisin kello 8 ja 16 välillä.

3.5 Kapasiteettivaatimukset

Järjestelmän on kyettävä suorittamaan eri käyttäjien lähettämät yhtäaikaiset pyynnöt. Tietokannan arvioidaan tarvitsevan noin 100 Mt levytilaa.

4 TOIMINNOT

4.1 Henkilörekisteriosio: henkilön lisääminen, muokkaaminen ja poistaminen

Henkilörekisteri-osassa voidaan tallentaa, muokata ja poistaa henkilöitä. Henkilöt voidaan määritellä asiakkaiksi ja avustajiksi. Avustajia on neljää eri tyyppiä: palkallisia avustajia, vapaaehtoisia avustajia, palkalliseksi avustajaksi hakevia vapaaehtoisia avustajia ja työnhakijoita.

Palkallisista ja vapaaehtoisista avustajista tallennettavat tiedot ovat kuvan 2 mukaiset. Jokaisesta henkilöstä tallennetaan myös henkilönumero. Tämä tulee kuitenkin automaattisesti ohjelmistolta, joten käyttäjien ei tarvitse kirjoittaa sitä erikseen.

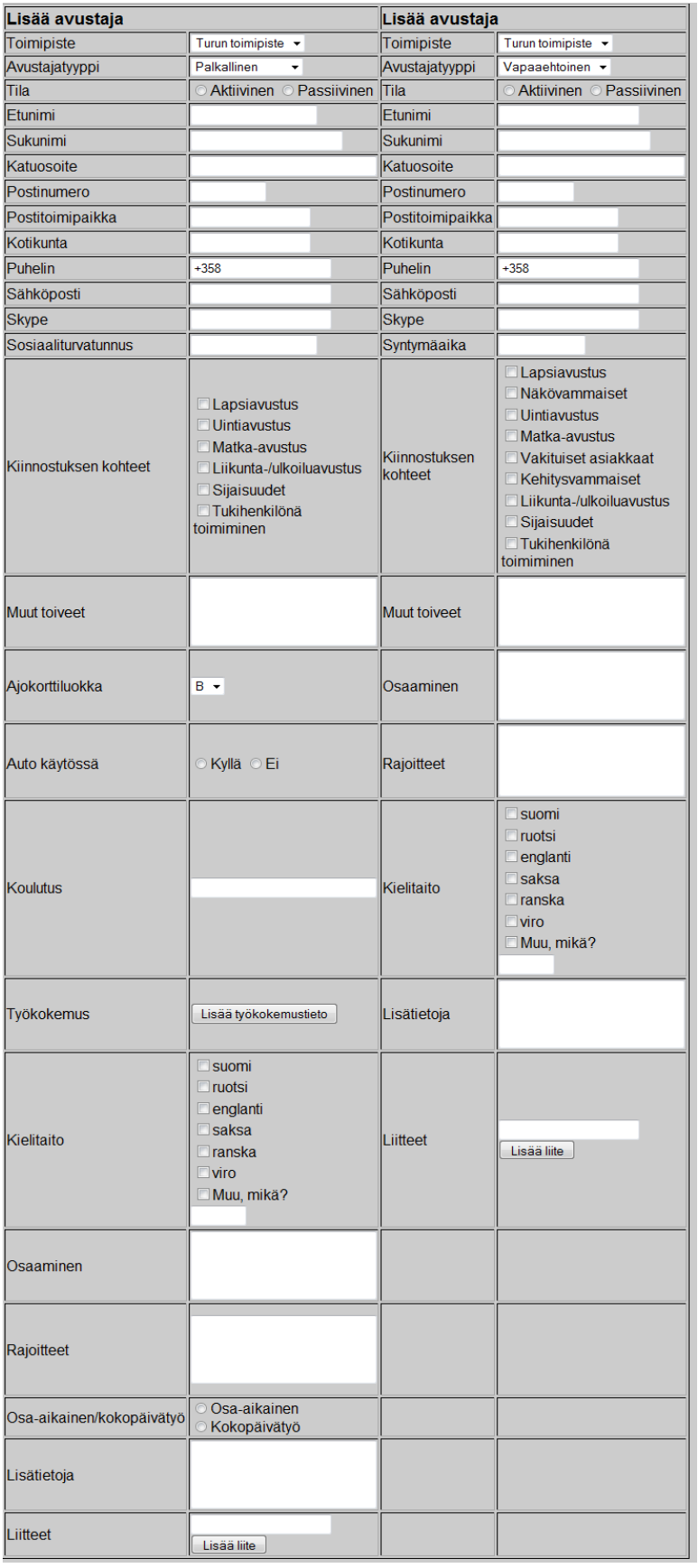
Työnhakijoiden tiedot täytetään vaihtamalla avustajatyyppi työnhakijaksi tai työtä hakevaksi vapaaehtoiseksi avustajaksi. Tiedot ovat samat kuin palkallisella avustajalla, sillä rekrytointi suoritetaan samojen tietojen perusteella.

Lisää työkokemustieto –napista päästään syöttämään työnhakijoiden tai palkallisten avustajien työkokemustiedot. Jokaisesta työkokemuksesta kysytään työnantajan nimeä, työtehtäviä sekä työn alkamis- ja päättymisajankohtaa.

Avustajista tallentuu myös tieto kaikista heidän käymistään Lihastautiyhdistyksen järjestämistä kursseista ja koulutuksista. Nämä merkinnät tulevat automaattisesti ohjelmiston koulutusosiosta.

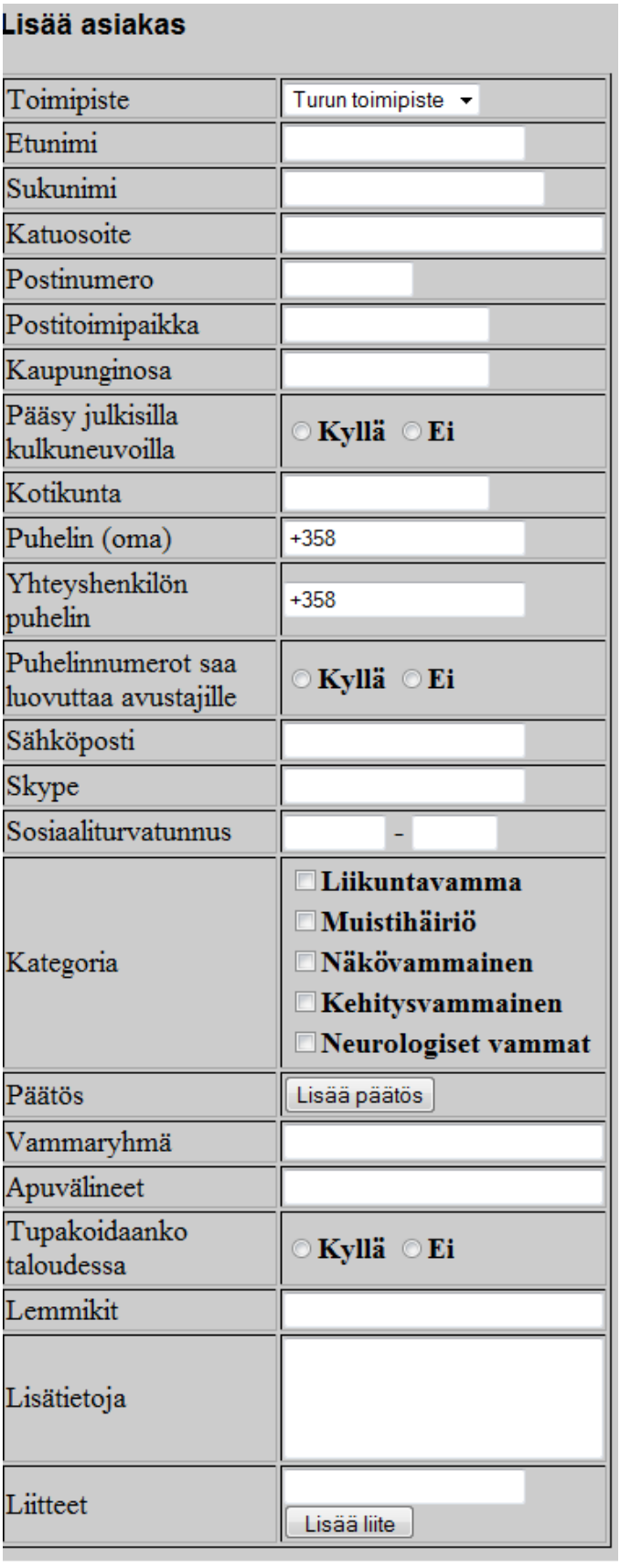
Asiakkaista tallennetaan järjestelmään kuvan 3 mukaiset tiedot. Lisää päätös –nappia painamalla päästään täyttämään avustuspäätöksen tiedot.

Jokaiseen henkilöön voidaan liittää liitetiedostoja, jotka voidaan avata erillisellä ohjelmalla. Näitä tiedostoja pääsee selaamaan myös lomakepankki-osiossa.



Lisää avustaja		Lisää avustaja	
Toimipiste	Turun toimipiste	Toimipiste	Turun toimipiste
Avustajatyyppi	Palkallinen	Avustajatyyppi	Vapaaehtoinen
Tila	Aktiivinen Passiivinen	Tila	Aktiivinen Passiivinen
Etunimi		Etunimi	
Sukunimi		Sukunimi	
Katuosoite		Katuosoite	
Postinumero		Postinumero	
Postitoimipaikka		Postitoimipaikka	
Kotikunta		Kotikunta	
Puhelin	+358	Puhelin	+358
Sähköposti		Sähköposti	
Skype		Skype	
Sosiaaliturvatunnus		Syntymäaika	
Kiinnostuksen kohteet	Lapsiavustus Uintiavustus Matka-avustus Liikunta-/ulkoiluavustus Sijaisuudet Tukihenkilönä toimiminen	Kiinnostuksen kohteet	Lapsiavustus Näkövammaiset Uintiavustus Matka-avustus Vakituiset asiakkaat Kehitysvammaiset Liikunta-/ulkoiluavustus Sijaisuudet Tukihenkilönä toimiminen
Muut toiveet		Muut toiveet	
Ajokorttiluokka	B	Osaaminen	
Auto käytössä	Kyllä Ei	Rajoitteet	
Koulutus		Kielitaito	suomi ruotsi englanti saksa ranska viro Muu, mikä?
Työkokemus	Lisää työkokemustieto	Lisätietoja	
Kielitaito	suomi ruotsi englanti saksa ranska viro Muu, mikä?	Liitteet	Lisää liite
Osaaminen			
Rajoitteet			
Osa-aikainen/kokopäivätyö	Osa-aikainen Kokopäivätyö		
Lisätietoja			
Liitteet	Lisää liite		

Kuva 2. Lisää avustaja -lomakkeiden käyttöliittymät



Kuva 3. Lisää asiakas -lomakkeen käyttöliittymä

4.1.1 Lisääminen, poistaminen ja muokkaaminen

Lisäämis-, muokkaus- ja poisto-oikeuksia ei anneta kuin ohjelman pääkäyttäjille. Muut kuin pääkäyttäjinä toimivat työntekijät voivat kuitenkin muokata omia tietojaan.

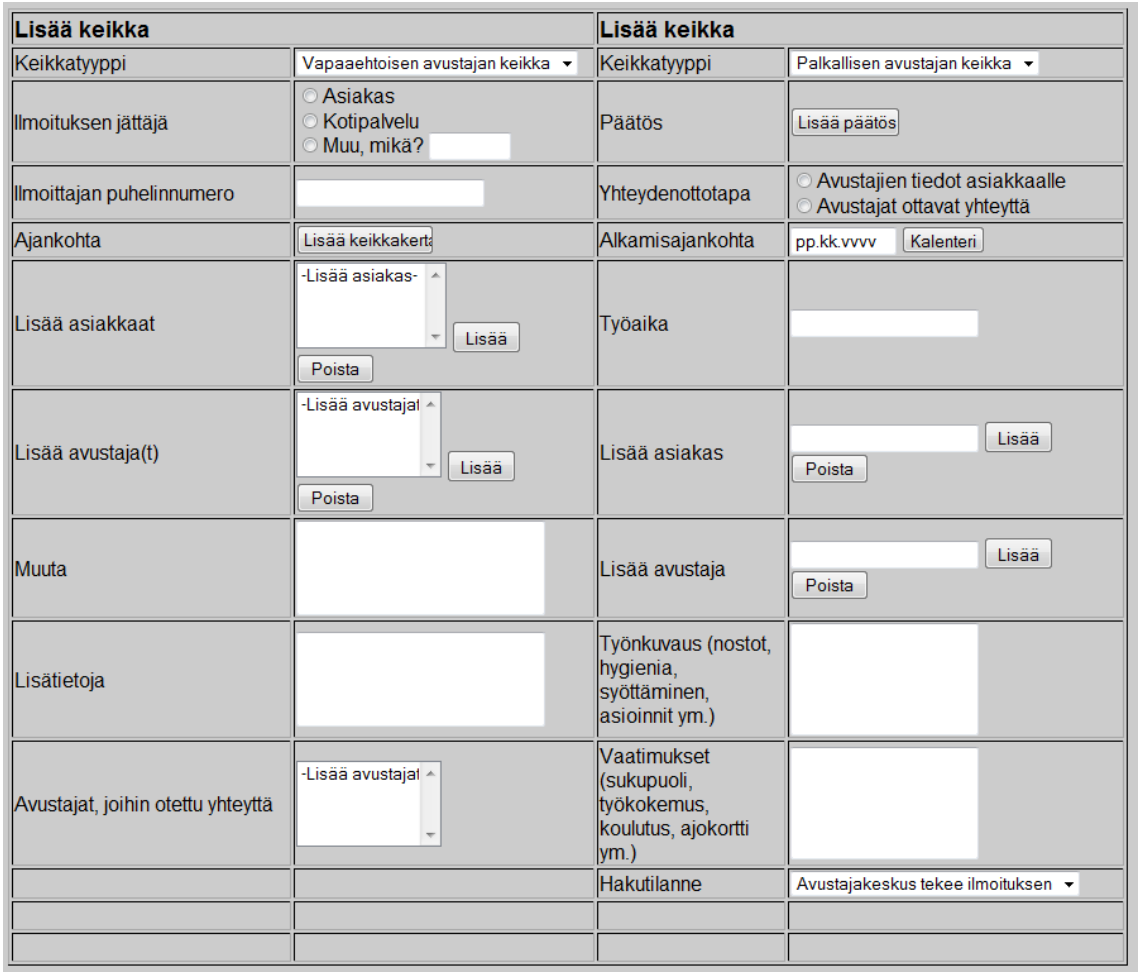
Oletuksena on, ettei avustajia monissakaan tapauksissa poisteta lopullisesti järjestelmästä, vaan nämä lisätään passiivisten joukkoon. Tämä tapahtuu myös automaattisesti kolmen kuukauden päästä työnhakijan syöttämisestä järjestelmään, jos häntä ei ole yhdistetty kesken olevaan avustuskeikkaan.

Asiakkaaksi ja avustajaksi haluavat voivat myös itse käydä täyttämässä omat tietonsa sähköiselle lomakkeelle internetissä ja lähettää ne sitten Lihastautiyhdistykselle. Yhdistyksen pääkäyttäjät voivat täydentää näitä tietoja ja lisätä ne sen jälkeen järjestelmään. Kun avustaja on hyväksytty, järjestelmä generoi tälle käyttäjätunnuksen ja salasanan. Käyttäjä voi myöhemmin käydä vaihtamassa salasanansa.

4.2 Ajanvarausosio

Ajanvarausosio on ohjelmiston sydän, jonka ympärille toiminta keskittyy. Ajanvarauksen avulla toteutetaan avustuskeikat, jolloin yhdistetään keikan tiedot, avustaja ja avustettava. Ajanvarausosiossa voidaan tehdä uusi keikka, muuttaa käynnissä olevan keikan tietoja ja katsella jo päättyneen keikan tietoja. Käynnissä olevia ja jo päättyneitä keikkoja voidaan hakea eri kriteerien avulla.

Avustuskeikkoja voi olla kahta tyyppiä, vapaaehtoisten keikka tai palkallisten keikka. Keikoista syötetään kuvan 4 mukaiset tiedot.



Kuva 4 Lisää keikka -lomakkeiden käyttöliittymät

Vapaaehtoisen avustajien keikoille voidaan lisätä useampia keikkakertoja Lisää keikkakerta –nappia painamalla (yksi vapaaehtoisen avustajan keikka voi sisältää useampia tapaamisia). Tätä kautta päästään täyttämään kellonajat, tiedot tilattavasta taksista, kulukorvaukset sekä muut lisätiedot.

Palkallisten keikan päätöstietoja ei tarvitse tässä kohtaa enää syöttää, sillä ne löytyvät asiakkaan tiedoista (olettaen, että asiakas on jo syötetty järjestelmään). Päätös on kuitenkin käytävä valitsemassa asiakkaan tiedoista, jos samalla asiakkaalla on useampia päätöksiä.

Palkallisten ja vapaaehtoisten avustajien keikkoja voidaan myös syöttää jälkikäteen järjestelmään. Tällöin järjestelmä kysyy varmistuksen, haluaako käyttäjä todella syöttää keikan jo menneeseen ajankohtaan.

4.2.1 Käyttöoikeudet

Jokaisella avustuskeikalla on oikeuslista, johon voidaan lisätä tarvittavien yhteistyökumppanien tunnukset. Listan avulla voidaan varmistaa, että vain ne ihmiset näkevät tiedot, joilla on siihen oikeus.

Ajanvarausosiota voidaan tarpeen vaatiessa laajentaa asiakkaiden käyttöön, jolloin asiakkaat voivat kirjautumalla palveluun käydä itse internetin kautta tekemässä keikkapyynnön täyttämällä sähköisen lomakkeen keikan tiedoista. Keikkapyyntö lähetetään yhdistykselle, jossa työntekijät voivat käydä täydentämässä keikan tiedot ja ilmoittavat asiakkaalle, kun sopiva avustaja on löytynyt. Tällöin asiakas pystyisi myös itse kirjautumaan järjestelmään ja seuraamaan ehdottamansa keikan tilaa.

4.3 Ohje- ja lomakepankkiosio

Tähän osioon tulee helposti hallittava ja selkeästi aseteltu ohje- ja lomakepankki. Ohjeet ja lomakkeet voidaan jaotella eri kategorioihin. Kategorioita voidaan tehdä lisää asetuksista ja niiden määrää ei ole rajoitettu. Yksi lomake voi kuulua useampaan kategoriaan.

Kaikki tähän osioon lisättävät ohjeet tai lomakkeet ovat omia tiedostojaan. Tässä osiossa ei ole tarkoitus päästä tuottamaan tekstejä, vaan ainoastaan hallitsemaan niitä.

Kategoriat on sijoitettu selkeästi ruudulle, jolloin niiden selaaminen on vaivatonta ja nopeaa. Ohje- ja lomakepankin ensimmäisellä näkymällä on myös pikavalintalista käytetyimmistä artikkeleista.

4.4 Yhteistyökumppanit osio

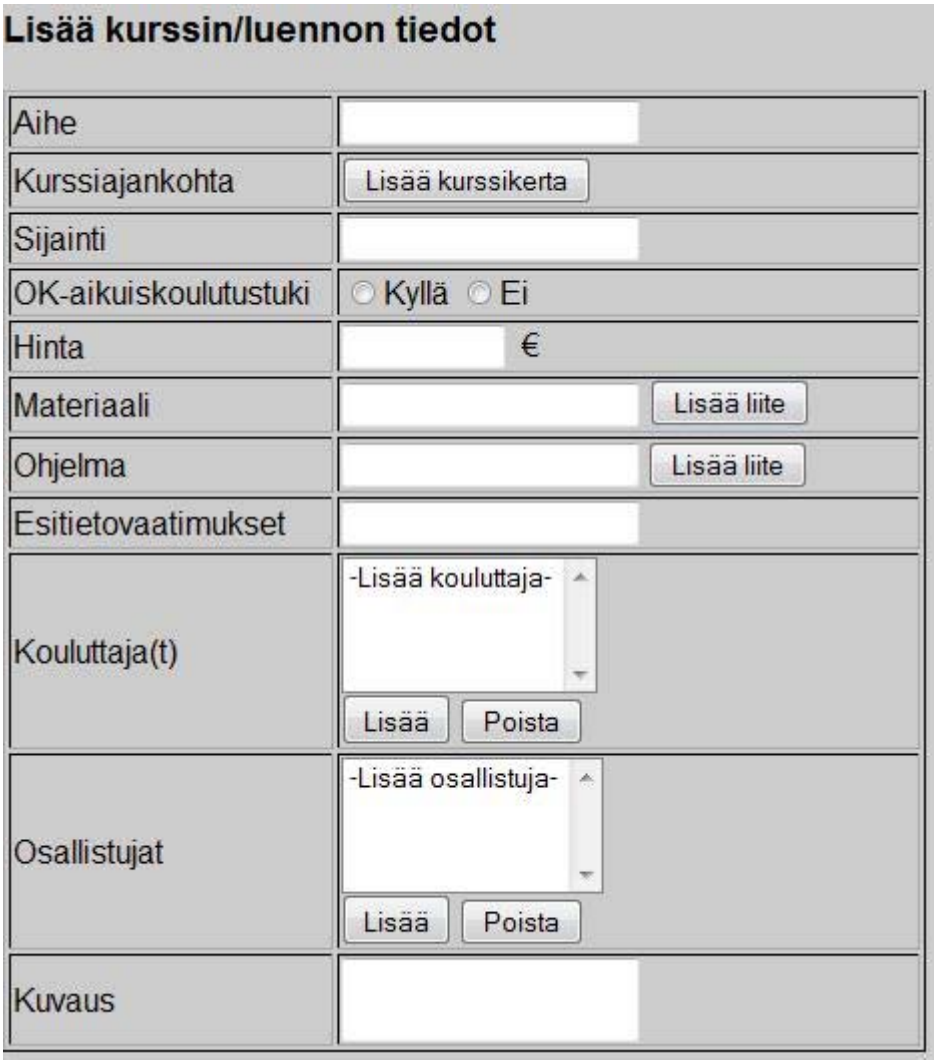
Yhteistyöosio on tärkeä osa ohjelmistoa. Tämän avulla ohjelmiston tietoja voidaan jakaa helposti kolmansille osapuolille. Esimerkiksi avustusten tietoja voidaan jakaa suoraan sosiaalitoimelle, mikä hoitaa taholtaan asiaa. Näin kaikilla osapuolilla on suora ja reaaliaikainen pääsy tietoihin.

Tämän osion käyttäjiä ovat pääsääntöisesti muut kuin yhdistyksen omat käyttäjät. Tämän takia osioon pääsyä on rajoitettu huomattavasti. Osioon pääsevät käyttäjät eivät pääse käsiksi muihin ohjelmiston tietoihin tai osioihin. Käyttäjät eivät voi myöskään muuttaa tai poistaa tietoja. Käyttäjillä on oikeus kirjoittaa osiossa oleviin muistiokenttiin kommentteja, jotka näkyvät kaikille muille käyttäjille.

Yhdistyksen käyttäjät voivat määrittää avustuskeikan näkymään yhteistyökumppanit osiossa. Jokainen yhteistyökumppanin käyttäjätunnus on lisättävä listalle, minkä mukaan voidaan määrittää näkyvät tiedot. Esimerkiksi Turun sosiaalitoimen tulee nähdä vain heille tarkoitetut tiedot.

4.5 Koulutusosio

Koulutusosiossa hallitaan yhdistyksen järjestämiä tai hallinnoimia kursseja ja koulutuksia. Osiossa voidaan luoda, muokata ja poistaa koulutusten tietoja. Koulutustiedot saadaan suoraan yhdistettyä henkilörekisterissä olevaan henkilöön, jolloin koulutus näkyy suoraan henkilön tiedoissa.



Kuva 5. Koulutusosion lomakekäyttöliittymä

Koulutuksista kerätään kuvan 5 mukaiset tiedot ylös. Lisää kurssikerta –nappia painamalla työntekijät, joilla on koulutusosioon oikeudet, pääsevät lisäämään kunkin kurssikerran tiedot (koulutus saattaa sisältää useamman kuin yhden tapaamiskerran). Tällaisia tietoja ovat kurssikerran alkamis- ja päättymisajankohta sekä kuvaus.

4.6 Hallintaosio

Hallintaosista tehdään kaikki ohjelmiston toimintaan ja turvallisuuteen vaikuttavat asiat. Täältä löytyvät käyttäjätunnusten hallinta, oikeuslistojen teko, lokitietojen katselu ja tallennus, tietokannan huoltotoiminnot ja raporttien

tulostus. Raporttien lopullinen muoto ja määrä varmistuu vasta ensimmäisten testien jälkeen.

5 ULKOISET LIITYNNÄT

5.1 Laitteistoliittymät

Ohjelmistoa voidaan käyttää millä tahansa laitteella, jossa on verkkoyhteys.

Tulostamiseen vaaditaan tulostin, jonka avulla voidaan tulostaa tietoja ohjelmistosta normaalille paperille ja tarra-arkeille.

5.2 Ohjelmistoliittymät

Sovellus käyttää MySQL-tietokannan hallintajärjestelmää tietojen varastoimiseen.

5.3 Tietoliikenneliitynnät

Ohjelmisto vaatii toimiakseen Internet-yhteyden.

6 MUUT OMINAISUUDET

6.1 Suorituskyky

Yksittäisen käyttäjän tietokoneen suorituskyky saattaa vaihdella laitteistosta ja verkkoyhteydestä riippuen, mutta tietokannan suorituskyky taataan < 1 sekunti per kysely vasteajalle. Vasteajaksi määritellään tietokannan aloittaman toimenpiteen ja kyselyn lähettämisen välinen aika.

6.2 Käytettävyys, toipuminen, turvallisuus ja suojaukset

Käytettävyys saadaan hyväksi suunnittelun ja testauksen avulla. Ohjelmisto toipuu kaikista käyttäjän tekemistä virheistä. Hyvä ohjelmointitavan avulla estetään ohjelmiston sisällä syntyvät virhetilanteet. Turvallisuus taataan käyttäjien tunnistamisella, verkkoyhteyksien tunnistamisella, tietokannan eheyssäännöillä ja säännöllisillä varmuuskopioilla.

Ohjelmisto suojataan käyttäjätunnusten väärinkäytöltä lukittuvilla tunnuksilla. Ohjelmiston ylläpitotyökalut vaativat eri oikeudet kuin tavalliset toiminnot. Verkkoyhteydet suojataan SSL-tekniikalla.

Käytettävyyden tueksi tehdään ohjelmistosta kattavat pikaohjeet ja varsinaiset ohjelmisto-ohjekirja. Ohjekirjassa on kerrottu ohjelmiston kaikki toiminnot ja asetukset yksityiskohtaisesti ja havainnollisesti.

Ohjelmistoon tallennettavat potilastiedot suojataan erikseen. Tietoihin pääsee käsiksi vain ne henkilöt, jotka on määritelty potilastietolistalle. Suojauksen tarkoitus on estää potilastietojen luvaton katselu ja muuttaminen.

6.3 Ylläpidettävyys

Dokumentit tehdään riittävän laajoiksi ja tarkoiksi ja ne pidetään ajantasaisina. Ohjelmoitaessa ohjelmistoa koodia kommentoidaan ja se muotoillaan hyvien ohjelmointitapojen mukaiseksi.

Järjestelmä pyritään tekemään mahdollisimman helposti ylläpidettäväksi, jottei erillistä ylläpitäjää järjestelmälle tarvita. Ylläpitotyökalut tehdään helpoiksi käyttää. Ylläpidon tueksi laaditaan ylläpito-ohje. Ohjeet sijoitetaan ohjelmiston ohjekirjaan.

Ohjelmiston käyttäjien tekemistä toimenpiteistä tehdään jatkuvana prosessina lokitiedosto.

Lokitiedoista tulee selvitä tehty toimenpide, käyttäjä ja ajankohta.

6.4 Siirrettävyys, yhteensopivuus

Ohjelmisto toimii mistä tahansa, kunhan verkkoyhteys on saatavilla.

Järjestelmän toteutetaan oletuksena PHP-kielellä, jolloin se on periaatteessa käytettävissä missä tahansa palvelinympäristössä, johon on asennettavissa PHP.

6.5 Operointi

Projektin aikana tuotetaan ylläpitosuunnitelma, joka kuvaa ohjelmiston operointitarpeet.

7 SUUNNITTELURAJOITTEET

7.1 Standardit

Määrittelyyn käytetään standardia IEEE 830. Muut käytettävät standardit täsmentyvät suunnittelun aikana.

7.2 Laitteistorajoitteet

Toimintaympäristönä toimivan laitteiston on kyettävä pyörittämään WWW−palvelinta. Käyttöympäristön on tuettava selainta, jolla järjestelmää käytetään.

7.3 Ohjelmistorajoitteet

Käytettävän WWW-selaimen on tuettava järjestelmän ominaisuuksia ja sen on sisällettävä uusimmat tietoturvapäivitykset, esimerkiksi tämän dokumentin päivityshetkellä Firefox 3.6.3.

7.4 Muut rajoitteet

Ohjelmisto vaatii tietoliikenneyhteyden.