Jimmy Valkama

TIETOKANNAN SUUNNITTELU HARJOITTELUNOHJAUSHANK-KEESEEN

Opinnäytetyö
CENTRIA-AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Elokuu 2017

TIIVISTELMÄ OPINNÄYTETYÖSTÄ

Centria-ammattikorkeakoulu	**Aika** Elokuu 2017	**Tekijä/tekijät** Jimmy Valkama
Koulutusohjelma Tietotekniikka		
Työn nimi TIETOKANNAN SUUNNITTELU HARJOITTELUNOHJAUSHANKKEESEEN		
Työn ohjaaja Sakari Männistö		**Sivumäärä** 37 + 1
Työelämäohjaaja Helena Åkerlund		

Tämän työn tavoitteena oli mallintaa tietokanta harjoittelunohjauksen tarpeisiin. Toimeksianto tuli Centria-ammattikorkeakoulun hallinnoimalta hankkeelta Ydinosaajat.

Opinnäytetyön teoriaosa alkoi tietokantojen peruskäsitteillä, ja sitten tutustuttiin syvällisemmin tässä työssä käytettävään tietokantatyyppiin eli relaatiotietokantaan. Muutamia muitakin vaihtoehtoja katseltiin lyhyesti. Tämän jälkeen valittiin työssä käytettävät työkalut. Teoriaosan lopussa tutustuttiin relaatiotietokantojen suunnitteluputkeen. Suunnitteluputki koostui vaatimusmäärittelystä, käsiteanalyysistä, tarveanalyysistä, normalisoinnista, taulujen muodostuksesta ja indeksoinnista.

Työ toteutettiin tekemällä hankkeen laatimat vaatimukset täyttävä suunnitelma. Se tehtiin edellisessä luvussa kuvatun suunnitteluputken mukaisesti. Viimeisessä luvussa pohdittiin työn tuloksia.

Asiasanat
Tietokanta, tietokannan suunnittelu, relaatiotietokanta, relaatiotietokannan suunnittelu, käsitemallinnus, ER-kaavio, UML

ABSTRACT

Centria University of Applied Sciences	Date	Author
	August 2017	Jimmy Valkama
Degree programme Information Technology		
Name of thesis DESIGNING A DATABASE FOR AN INTERNSHIP GUIDANCE PROJECT		
Instructor Sakari Männistö		**Pages** 37 + 1
Supervisor Helena Åkerlund		

The goal of this thesis was to model a database for the needs of internship guidance. The commission originated from the project Ydinosaajat which is supervised by Centria University of Applied Sciences.

The theory part starts by defining basic database concepts, and then it discusses the relational database more in depth since it is the chosen database type for this assignment. The thesis also looks briefly into some other database alternatives. After that the thesis introduces the chosen software tools. The final theoretical chapter describes the relational database design process. The design process consisted of requirement specification, conceptual modelling, requirement analysis, normalization, forming the database tables, and finally indexing.

The assignment was implemented by creating a plan that fulfills the requirements of the project. It was accomplished using the same design process as described in the previous chapter. In the final chapter the results of the assignment were reflected.

Key words
Database, database design, relational database, relational database design, conceptual modelling, ER Diagram, UML

KÄSITTEIDEN MÄÄRITTELY

ERD	Entity-relationship diagram eli ER-kaavio tarkoittaa kaaviomuotoista käsitemallia. Se on tietokantasuunnittelussa käytettävä malli, jolla kuvataan tietokannassa esiintyvät käsitteet ja niiden väliset yhteydet.
JSON	JavaScript Object Notation, JavaScript-kielestä alkunsa saanut merkintätapa, joka on datan esittämisessä vaihtoehto XML-kielelle.
MySQL	Yksi suosituimmista TKHJ-vaihtoehdoista. Käytetään erityisesti www-palvelujen tietokannoissa.
NoSQL	Käsite, joka kattaa muut kuin perinteiset relaatiomalliset, SQL-kielellä halittavat tietokannat.
SQL	Structured Query Language on yleisimmin relaatiotietokantojen yhteydessä käytetty kieli.
TKHJ	Tietokannan hallintajärjestelmä on laaja tietokantojen toiminnasta vastaava ohjelmisto. Englanninkielinen vastine termille on Database Management System, DBMS.
UML	Unified Modeling Language – kokoelma erilaisia, enimmäkseen ohjelmistotekniikan alalla käytettäviä diagrammityyppejä. Tämän työn kannalta keskeisimmät kaaviot ovat luokkakaavio ja käyttötapauskaavio. Luokkakaaviota käytetään yleensä oliopohjaisessa suunnittelussa, mutta se soveltuu hyvin myös käsitekaavion luomiseen.
XML	Extensible Markup Language on hyvin yleiskäyttöinen merkintäkieli. Se on käytössä joissakin dokumenttiorientoituneissa tietokannoissa.

TIIVISTELMÄ
ABSTRACT
KÄSITTEIDEN MÄÄRITTELY
SISÄLLYS

1 JOHDANTO

Opinnäytetyön tarkoituksena on suunnitella relaatiotietokanta, joka tulee käytettäväksi harjoittelunohjaushankkeeseen. Tavoitteena on selkeä ja helppotajuinen tavoitteet täyttävä tietokantaratkaisu, joka kuitenkin on samalla eheä ja riittävän tehokas. Suunnitelman pohjalta luodaan myöhemmin sovellus, joka toimii apuna organisaatioiden välisten yhteistyökumppanuuksien sekä yhteystietojen, tapahtumien ja toimenpiteiden rekisteröinnissä.

Opinnäytetyön toimeksianto tulee hankkeelta Ydinosaajat. Kyseisessä hankkeessa on tarkoituksena muun muassa kehittää opetussisältöjä ja yrityksiä palvelevaa toimintaa sekä muodostaa opetus- ja koulutusresursseista asiantuntijarekisteri. Hankkeen hallinnoitsija on Centria-ammattikorkeakoulu Oy. Taustalla hankkeessa on Pyhäjoelle rakennettava ydinvoimala Fennovoima Oy ja sen rakentamisen vaatima korkea osaamistaso. Ammattialojen osaamista ja nykyisen osaamisen kartoittamista ja jakamista voisi kehittää luomalla organisaatioiden välisiä verkostoja.

Harjoittelunohjaushankeprojekti on osa hanketta, jossa on mukana eri ammattikorkeakouluja, ammattiopistoja sekä joitakin yrityksiä. Yhteistyön pohjalta luodaan tietokanta, johon kaikkien osapuolten yhteystiedot ja yhteyshenkilöt sekä kumppanuuksien osa-alueet tallennetaan. Jatkossa tietokantaan tallennetaan esimerkiksi toteutuneet harjoittelujaksot ja opettajien yritysjaksot sekä koulutukset. Tämä opinnäytetyö määrittelee kuvauksen tietokannasta ja sen toiminnallisuudesta. Suunnitelman pohjalta voi hanke tai sen jatkohanke kehittää sovelluksen.

Opinnäytetyö esittelee aluksi tietokantojen teoriataustan sekä tässä työssä käytettävät työkalut. Teoriatausta esittelee relaatiotietokannan perusajatukset kohtuullisen tarkalla tasolla ja muutamia muita vaihtoehtoja lyhyesti. Tämän jälkeen käydään läpi relaatiotietokantojen suunnitteluputken rakenne, jota sitten sovelletaan käytännön osuudessa. Laajuuden vuoksi varsinaista toteutusta ei tehdä, mutta tietokanta pyritään mallintamaan ja kuvaamaan riittävän tarkalla tasolla jatkokehitystä varten. Viimeisessä luvussa pohditaan työn tuloksia.

2 TIETOKANNAT JA TYÖKALUT

Tässä luvussa tutkitaan aluksi tietokantojen teoriataustaa ja avataan samalla keskeisimpiä termejä. Keskitytään pääasiassa relaatiotietokantoihin, sillä se on tämän työn pääaihe. Käydään kuitenkin myös läpi muutaman muun tietokantatyypin perusajatuksia.

2.1 Mikä on tietokanta?

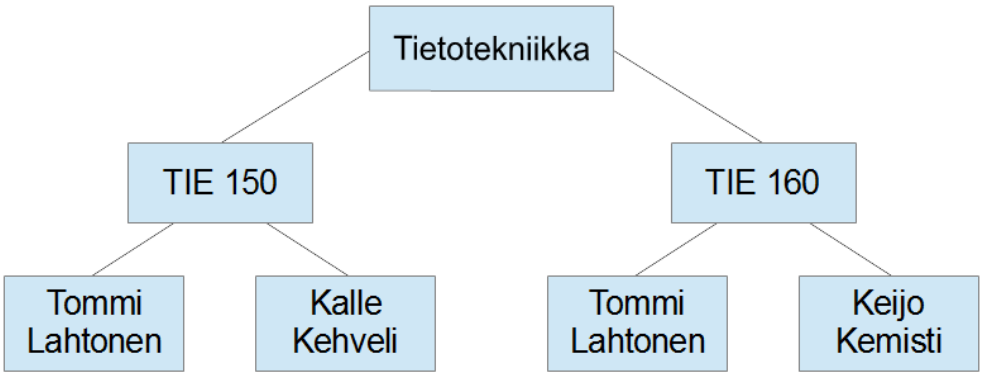
Tietokantaa käytetään tietojärjestelmissä, jotka varastoivat toisiinsa liittyviä tietoja. Usein tietokannat keräävät tietoa jostain tietystä reaaliasiasta, kuten esimerkiksi yrityksen keräämiä tietoja asiakkaistaan. Tietokannat ovat tärkeitä juuri erityisesti yrityksille, sillä niiden sisältämä tieto on keskeisin osa tiedonhallintaa. Jotta yrityksen hallinta sujuisi helpommin mm. investointien ja henkilöstön osalta, tarvitaan selkeä tietokantajärjestelmä. (Hovi, Huotari & Lahdenmäki 2005, 4.)

Yleensä tietokanta on loogisesti tallennettu eri tietojen joukko, jota voidaan lukea ja käsitellä jollain tietokantakielellä, useimmiten SQL-kielellä. Tietokannan hallintajärjestelmät (Database Management Systems) ovat isoja ja monimutkaisia ohjelmistoja. Ilman tällaisia hallintajärjestelemiä kaikki tieto olisi tiedostoina, mikä tekisi tietojen käyttämisestä ja hallinnoimisesta erittäin monimutkaista. Hallintajärjestelmään kuuluu myös monia tärkeitä työkaluja kuten monen käyttäjän rinnakkainen käyttö, virhetilanteista palautuminen ja turvallisuusominaisuuksia. Tunnettuja hallintajärjestelmiä ovat mm. MySQL, Oracle ja Microsoft SQL Server ja Access. (Hovi ym. 2005, 4; Teorey, Lightstone, Nadeau & Jagadish 2011, 2.)

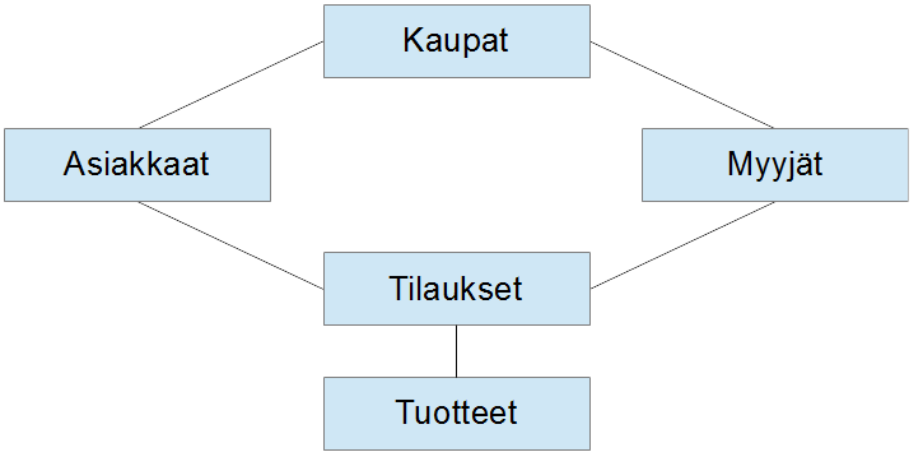
Tietokannat voidaan käyttötarkoituksensa mukaan jakaa operatiivisiin tietokantoihin ja tietovarastoihin. Operatiivisille tietokannoille on ominaista reaaliaikaisuus sekä enemmän tai vähemmän yleinen tietojen päivitys ja lisäys. Tietovarastot on tarkoitettu analysointiin ja historiatietojen keräämiseen. Tietovarastokannat voivat olla erittäin laajoja ja niiden sisältöä saatetaan muodostaa monesta erillisestä tietokannasta eräajoina esimerkiksi öisin. (Hakkarainen 2011, 59; Hovi ym. 15–16.)

2.2 Historiallisia tietokantoja

Tietokantojen varhaisin historia sijoittuu 1960-luvulle, jolloin käytettiin ns. hierarkkisia tietokantoja. Ideana oli, että tiedot ovat hierarkkisessa eli puurakenteessa (vertaa: sukupuu). Rakenne toki ajaa asiansa yksinkertaisissa tapauksissa, mutta kovin monimutkaisiin sovelluksiin se ei sovi. Hieman myöhemmin hierarkkisista tietokannoista kehitettiin verkkotietokanta mahdollistamalla lapsitaululle useampi äiti-taulu. Muuten se on toiminnaltaan hyvin samanlainen kuin verkkotietokanta, eikä sekään relaatiotietokantojen ilmestymisen jälkeen, 1970-luvulta alkaen, ole ollut kovin suuressa suosiossa. (Lahtonen 2002, 3–4.)



KUVIO 1. Esimerkki hierarkkisesta tietokannasta (Lahtonen 2002, 3)

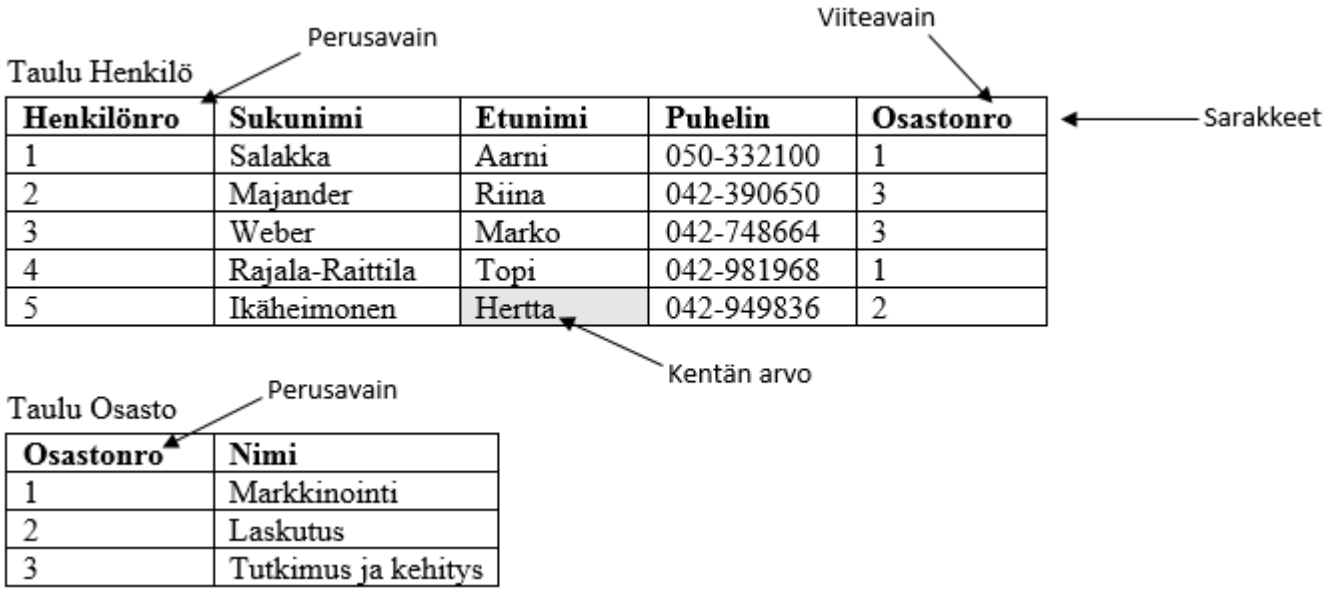


KUVIO 2. Esimerkki verkkotietokannasta (Lahtonen 2002, 4)

2.3 Relaatiotietokanta

Nykyään useimmiten käytetään relaatiotietokantaa, jonka avulla toteutetaan niin operatiivisia sovelluksia kuin tietovarastojakin. Sen pohjana on tutkija E. F. Coddin kehittämä relaatiomalli vuodelta 1970, jossa käytetään mallintamiseen joukko-oppia, predikaattilogiikkaa sekä matematiikkaa. Relaatiomalli voidaan jakaa abstraktiin relaatiomalliin ja fyysiseen taulumalliin. Coddin ajatukset kuuluvat näistä ensimmäiseen, ja fyysinen taulumalli on pitkälti tuotekohtainen asia. (Hovi ym. 2005, 6–7.)

Relaatiotietokanta rakentuu tauluista (table) ja niiden välisistä yhteyksistä. Taulu taas koostuu sarakkeista eli tiedoista (attribute) sekä perus- (primary-) ja viiteavaimista (foreign key). Taulut voisivat sisältää esimerkiksi tiedot yrityksen henkilökunnasta ja osastoista (KUVIO 3). Henkilö-taulu sitten sisältäisi henkilöihin liittyviä tietoja, kuten muun muassa nimen ja puhelinnumeron. Taulun varsinaisille riveille tulee itse ilmentymät eli näiden tietojen arvot. Kunkin asiakkaan tiedot ovat siten omalla rivillään. (Hovi ym. 2005, 7–8.)



KUVIO 3. Relaatiomallin peruskäsitteet (mukaillen Ollikainen, 18)

2.3.1 Attribuutit ja niiden tietotyypit

Taulua luotaessa on attribuuteille eli tiedoille annettava jokin tietotyyppi eli kerrottava minkälaista esitettävä tieto on. Tietotyypit ovat hieman ongelmallisia siksi, että niiden saatavuus ja joidenkin tietotyyppien kohdalla jopa merkityskin ovat tuotekohtaisia. Tämän takia käydään ne läpi melko yleisellä tasolla.

TAULUKKO 1. Osa standardeista SQL-tietotyypeistä (mukaillen Lahtonen 2002, 41, 118–119)

Tietotyyppi	Lyhenteet / synonyymit	Merkitys
CHARACTER	CHAR	Kiinteämittainen merkkijono
CHARACTER VARYING	VARCHAR, CHAR VARYING	Vaihtuvamittainen merkkijono
CLOB		Pituusrajoitukseton merkkijono
NUMERIC(pituus, desimaalit)		Tarkka desimaaliluku
DECIMAL(pituus, desimaalit)		Tarkka desimaaliluku
FLOAT(pituus)	DOUBLE, REAL	Liukuluku
INTEGER	INT	Kokonaisluku
SMALLINT		Kokonaisluku
DATE		Päivämäärä
TIME [WITH TIME ZONE]		Aika
TIMESTAMP [WITH TIME ZONE]		Aikaleima
BOOLEAN		Totuusarvo (TRUE / FALSE)

Char- ja varchar- (Oraclessa vastaava on varchar2) tyypeille annetaan määrittelyn yhteydessä pituus. Kiinteä- ja vaihtuvamittaisen ero on siinä, että kiinteämittaiset merkkijonot ovat aina annetun pituuden mittaisia. Jos pituudeksi on annettu esimerkiksi 16 ja annettu merkkijono on vain kahdeksan merkkiä pitkä, niin loput merkit tallennetaan välilyönteinä. Vaihtuvamittaisessa merkkijonossa näin ei tapahdu, vaan sen sijaan tallennetaan varsinaisen merkkijonon lisäksi sen todellinen pituus. Annettu pituus on tässä tapauksessa suurin sallittu pituus. Käyttäytyminen pituuden ylittyessä riippuu tietokantajärjestelmästä: loppuosa saatetaan katkaista pois tai sitten näytetään virheilmoitus. Clob (joissakin tuotteissa text) on merkkijono ilman pituusrajoitusta. Se ei tue kaikkia merkkijono-operaatioita, mutta voisi olla hyvä vaihtoehto suurempien tekstien tallennukseen. (Hovi ym. 2005, 111–112; Lahtonen 2012, 119, 214.)

Numeraalisia tietotyyppejä on erikseen kokonais- ja desimaaliluvuille. Kokonaislukutyypeistä tyypillisin on integer eli int. Sen koko on neljä tavua, mikä käytännössä tarkoittaa, että se voi sisältää yhden luvun väliltä noin ±2 miljardia. Kokonaisluvuille on olemassa myös pienempiä tai suurempia vaihtoehtoja, kuten taulukossa 1 mainittu smallint. Tuotteet voivat myös tarjota etumerkittömiä vaihtoehtoja, kuten unsigned integer. Desimaalilukutyypit voivat olla joko tarkkoja (decimal, numeric) tai liukulukuja (float, double, real). SQL-standardin mukaan numeric on täsmälleen halutun tarkkuuden mukainen, kun taas decimal vähintään yhtä tarkka. Tämä on kuitenkin hyvin tuotekohtaista, sillä esimerkiksi MySQL ymmärtää molemmat tyypit identtisiksi. Liukuluvut ovat likimääräisiä, eivätkä siksi sovi kovin hyvin esimerkiksi rahallisten arvojen käsittelyyn. (Lahtonen 2012, 42; MySQL 2017.)

Ajallisen informaation käsittelyyn on olemassa mm. taulukon 1 tyypit date pelkälle päivämäärälle, time pelkälle ajalle ja timestamp molempien tallennukselle (Lahtonen 2012, 41–43). Valitun tietokannan hallintajärjestelmän tarjoamat tyypit kannattaa tarkistaa, sillä aikatyypeissä voi olla paljonkin tuotekohtaisia eroavaisuuksia.

Muita tämän työn kannalta keskeisiä tietotyyppejä ovat boolean ja enum. Boolean voi saada vain tosi-/epätosi-arvoja, joten se sopii hyvin kyllä-/ei-tilanteisiin. Enum, joka ei itse asiassa ole standardi SQL-tietotyyppi, voi saada yhden arvon sille määritellystä joukosta. Enum-tietotyypin mukaisen toiminnan voi myös toteuttaa käyttämällä ns. CHECK-lausetta. (Hovi ym. 2005, 113–114; Lahtonen 2002, 118; MySQL 2017.)

2.3.2 SQL-kieli

Structured Query Language (SQL) eli rakenteellinen kyselykieli on IBM Researchin 1970-luvulla kehittämä ja kaikista laajimmin käytössä oleva tietokantakieli. Kyseistä kyselykieltä käytetään hakujen lisäksi mm. taulujen luontiin, muokkaamiseen sekä sisältöjen ja taulujen poistamiseen (Date, Kannan & Swamynathan 2003, 69.)

SQL-kielen lisäys-, päivitys- ja poistokäskyt eli insert, update, delete sekä näiden lisäksi myös valintakäsky select tunnetaan käsitteellä DML (Data Manipulation Language). Lisäksi kieleen kuuluu DDL (Data Definition Language) -osuus, jolla voi esimerkiksi luoda tauluja. (Hakkarainen 2011, 76, 124.)

Seuraavassa on yksinkertainen esimerkki SQL-lauseesta, jolla haetaan Opiskelija-taulusta kaikkien tietotekniikan koulutusohjelmalla olevien opiskelijoiden etu- ja sukunimet. Vastausrivit lajitellaan lopuksi nimen perusteella aakkosjärjestykseen.

SELECT sukunimi, etunimi, koulutusohjelma FROM Opiskelija WHERE koulutusohjelma = 'tietotekniikka' ORDER BY sukunimi, etunimi;

2.3.3 Perusavain

Relaatiotietokannan sääntöihin kuuluu, että jokainen taulun rivi on yksilöllinen. Tämä saadaan aikaan luomalla sellainen kenttä, joka voi sisältää vain uniikkeja arvoja. Kyseinen kenttä määritellään taulun perusavaimeksi. Se on erittäin tärkeä suorituskyvyn ja useamman taulun liitoksen kannalta. Perusavaimen pakollisuutta kutsutaan avaineheydeksi. (Nixon 2014, 189; Hovi ym. 2005, 11.)

Perusavain on yhdelle tai joissakin tapauksissa useammalle kentälle määriteltävä ominaisuus, ja se voi olla joko luonnollinen tai keinotekoinen eli surrogaatti. Luonnollisella perusavaimella tarkoitetaan ”luonnossa esiintyviä” yksilöllisiä termejä, näistä hyviä esimerkkejä ovat henkilötunnus ja kirjan ISBN. Keinotekoinen perusavain nimensä mukaisesti luodaan pelkästään sen takia, että taululla on oltava perusavain. Surrogaatista voi helposti tehdä ns. juoksevan numeron, joka kasvaa automaattisesti yhdellä aina uuden rivin lisäyksen yhteydessä. Tämä helpottaa tiedon lisäämistä tauluun, kun perusavaimen arvosta ei tarvitse huolehtia. Surrogaatit ovat yksilöllisiä ja muuttumattomia, mutta eivät anna mitään informaatiota. (Hovi ym. 2005, 9, 62–63.)

2.3.4 Taulujen väliset suhteet

Taulujen väliset suhteet ovat yksi relaatiotietokantojen kulmakivistä. Käsitemallinnuksessa (luku 4) suhteet ilmoitetaan muutamilla erilaisilla yhteyksillä, joiden merkintätapa riippuu käytettävästä notaatiosta. Suhteet kuvaavat kuinka monta ilmentymää käsitteestä on siihen yhteydessä oleviin tauluihin molemmista suunnista katsottuna. Ilman näitä yhteyksiä taulut olisivat vain ”irrallaan”. Pääsääntöisesti suhteet voivat olla joko yksi-yhteen, yksi-moneen tai moni-moneen. (Teorey ym. 2011, 17, 20–21.)

Kaikista tavallisin on yksi-moneen, jossa yhdessä päässä olevaa taulua kutsutaan isäksi (yksi) ja muita lapsiksi, joita voi olla useita. (Hovi ym. 2005, 9). Hyvä esimerkki tästä on opiskelijan ja oppilaitoksen välinen suhde: oppilaitokseen kuuluu monia opiskelijoita, mutta yhden opiskelijan voidaan olettaa kuuluvan vain yhteen oppilaitokseen. Suhteisiin palataan tarkemmin luvussa 4 TIETOKANTOJEN SUUNNITTELU.

2.3.5 Viiteavain ja viite-eheys

Viiteavain on eräänlainen linkkikenttä taulujen välille: se määritetään viittaamaan toisen taulun perusavaimeen. Viiteavaimia tarvitaan tekemään taulujen väliset yhteydet mahdolliseksi. Käytännössä viiteavain on taulujen välisen suhteen vastine relaatiomallissa ja lopullisessa relaatiotietokantatoteutuksessa. (Hovi ym. 2005, 9.)

Viiteavaimiin liittyy myös käsite viite-eheys, jolla tarkoitetaan perusavaimen ja viiteavaimen suhteen pysyvyyttä. Viite-eheys rikkoutuu, jos perusavaimen arvoa yritetään muuttaa tai poistaa. Perusavaimen arvon poistaminen tarkoittaa yleensä kokonaisen rivin poistamista, sillä avaineheyden mukaanhan perusavain ei saisi olla NULL. Taulukossa 2 on esitetty näitä tilanteita varten olevia käyttäytymissääntöjä eli viite-eheyssääntöjä. (Lahtonen 2002, 12–14; Hovi ym. 2005, 11–12.)

TAULUKKO 2. Viite-eheyssäännöt (mukaillen Hovi ym. 2005, 50; Lahtonen 2002, 13)

Eheyssääntö	Perusavaimen muutos	Perusavaimen poistaminen
Cascade	Viiteavain saa uuden perusavaimen arvon	Viiteavain ja vastaava tietue poistetaan
Restrict	Estetty, jos on viittauksia	Estetty, jos on viittauksia
Set Default	Viiteavain saa oletusarvonsa	Viiteavain saa oletusarvonsa
Set Null	Viiteavain saa NULL-arvon	Viiteavain saa NULL-arvon

2.3.6 Sarakekohtaiset eheyssäännöt

Sarakkeille määriteltyä sallittujen arvojen joukkoa kutsutaan arvojoukoksi. Yksinkertaisimmillaan se voi ilmaista sarakkeen arvon pakollisuutta. SQL-kielessä pakollisuus saadaan määreellä NOT NULL. Jos määrettä ei käytetä, ovat ns. tyhjäarvot eli NULL-arvot sallittuja. On huomattava, että NULL on eri asia kuin tyhjä merkkijono, välilyönti tai nolla, ja sitä myös käsitellään SQL-kyselyissä erityisillä lausekkeilla IS NULL ja IS NOT NULL. (Hovi ym. 2005, 113; Hakkarainen 2011, 86–87.)

Monimutkaisempia rajoituksia saa käyttämällä CHECK-rajoituksia. Alla on kaksi esimerkkiä CHECK-lauseen käytöstä:

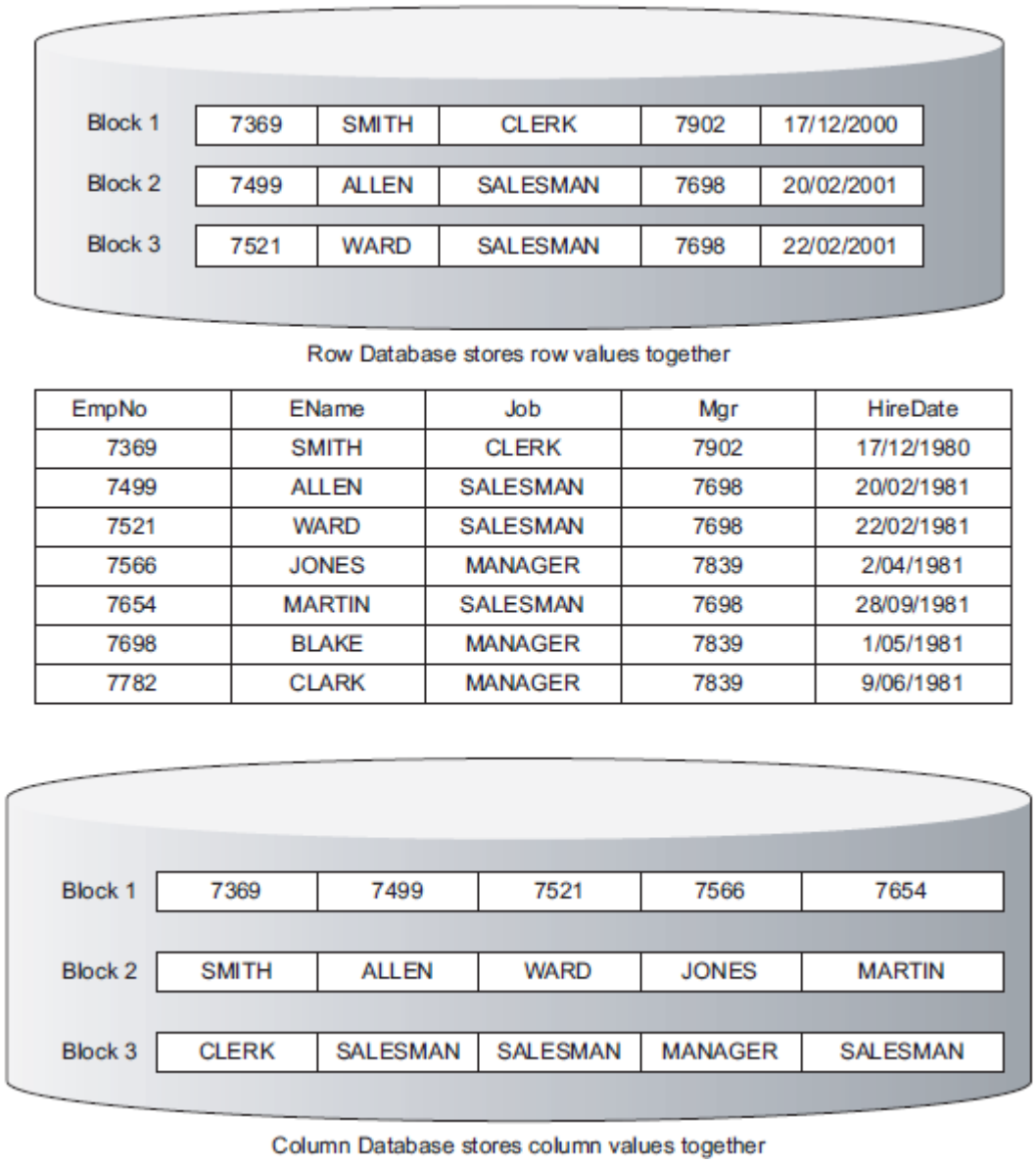
CONSTRAINT SUKUPUOLI_TARK CHECK (sukupuoli IN (’M’, ’N’))
CONSTRAINT VEROPROS_TARK CHECK (veropros BETWEEN 0 AND 100)

(Hovi ym. 2005, 113–114.)

2.4 NoSQL

Relaatiotietokantojen vallankumouksen jälkeen 1970-luvulta alkaen verkko- ja hierarkkiset mallit jäivät syrjään. Vaihtoehtoja on kuitenkin aina ollut, vaikka ne alkoivat saada enemmän suosiota vasta viime vuosikymmenellä. Käyttöön on tullut uusi termi NoSQL, jolla tarkoitetaan relaatiomallista poikkeavien tietokantajärjestelmien joukkoa. NoSQL-tietokannat voidaan jakaa useaan eri tyyppiin sen mukaan, miten tieto on tallennettuna. Seuraavassa on lyhyt esittely keskeisimmistä tyypeistä. (Vaish 2013, 23–24.)

Sarakeorientoituneessa tietokannassa (KUVIO 4) tieto tallennetaan sarakkeittain rivien sijaan, kuten relaatiomallin mukaisesti tehtäisiin. Yksittäisen attribuutin ilmentyvät ovat siis kaikki samalla rivillä. Tällainen rakenne on joustava ratkaisu ainakin attribuuttikohtaiseen tiedonkäsittelyyn ja uusien attribuuttien luomiseen käyttöönoton jälkeen. (Vaish 2013, 43–45.)



KUVIO 4. Vertailussa rivimallinen ja sarakemallinen tietokanta (Naidu 2014)

Dokumenttiorientoitunut tietokanta on ns. osittain jäsennelty tallennusrakenne. Tällä tarkoitetaan sitä, että tiedot voivat noudattaa ennalta päätettyä skeemaa joustavasti tai ei ollenkaan, myös mahdollistaen helpohkon tietorakenteen muuttamisen jälkeenpäin. Tyypillisesti datan määrittelyyn käytetään JSON:ia tai XML-kieltä. Dokumenttiorientoitunut tietokanta on päätynyt melko suureen suosioon viime aikoina web-sovelluksissa. (Vaish 2013, 46–49.)

Seuraavassa esimerkki siitä, miltä JSON-muotoinen henkilöä kuvaava dokumentti voisi näyttää (Vaish 2013, 47).

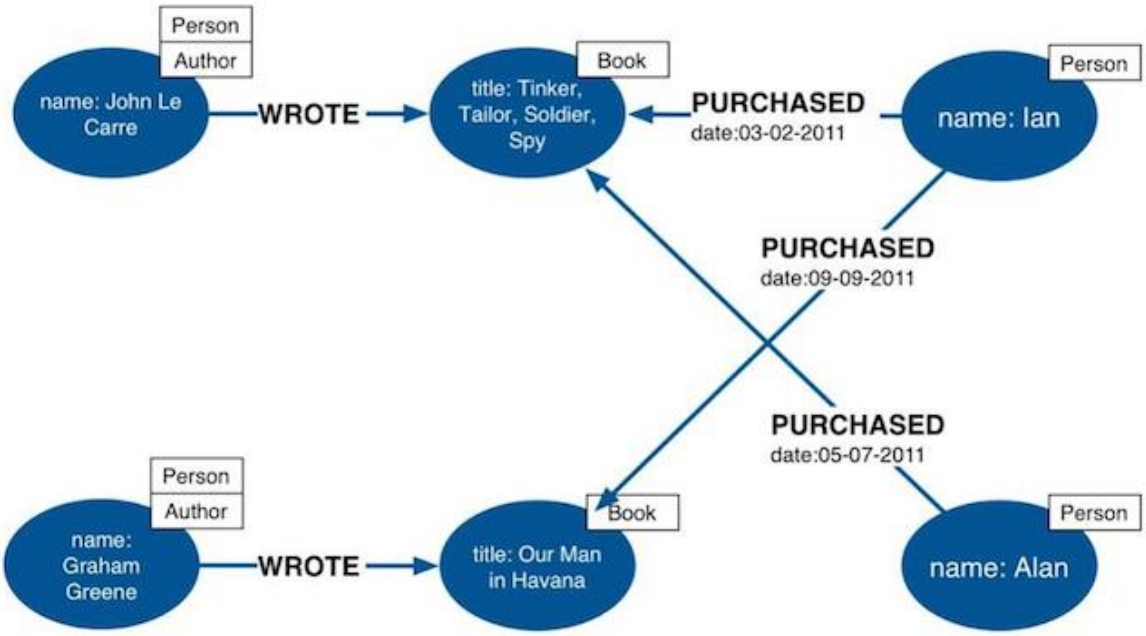
```
{
   "EmployeeID": "SM1",
   "FirstName" : "Anuj",
   "LastName"  : "Sharma",
   "Age"       : 45,
   "Salary"    : 10000000
}
```

Avain-arvo-varasto on samankaltainen ratkaisu kuin dokumenttiorientoitunut tietokanta. Tässä tapauksessa tietokanta koostuu avain-arvo-pareista. Arvoja voidaan hakea avainten perusteella hyvin nopeasti, mutta arvojen mukaan hakuja ei pysty rajaamaan. Tämä on hyvä ratkaisu mm. välimuistitoiminnallisuuteen. (Vaish 2013, 57–58.)

Alla on hyvin yksinkertainen esimerkki Redis-ohjelmistolla tehdystä arvon kirjoituksesta ja lukemisesta (Redis).

```
> set mykey somevalue
OK
> get mykey
"somevalue"
```

Graafi-tietokanta keskittyy solmujen (käytännössä käsitteen ilmentymä) välisiin yhteyksiin. Solmut ja niiden väliset suhteet voivat muodostaa hyvinkin monimutkaisia kokonaisuuksia, ja lisäksi useat yhteydet kahden solmun välillä onnistuvat ilman suorituskyvyn heikkenemistä. Solmut saatetaan toteuttaa esimerkiksi käyttämällä dokumenttilähtöisen tietokannan tai avain-arvo-varaston periaatteita. Graafi sopii hyvin mm. sosiaalisten verkkojen, reittien ja verkkotopologioiden kuvaamiseen. (Neo4j 2017; Vaish 2013, 61–62.)



KUVIO 5. Esimerkki graafi-tietokannasta (Neo4j 2017)

2.5 Relaatiotietokannan hallintajärjestelmät

Seuraavaksi perehdytään muutamiin tunnettuihin ilmaisiin avoimen lähdekoodin tietokannan hallintajärjestelmiin, verraten niitä toisiinsa. Tässä keskitytään relaatiotietokantatuotteisiin, sillä relaatiotietokannat ovat tämän opinnäytetyön pääaihe. NoSQL-tietokantojen ohjelmistoratkaisuja ovat muun muassa Apache Software Foundationin CouchDB, MongoDB Inc.:n MongoDB sekä ylempänä mainitut Redis ja Neo4j.

2.4.1 SQLite

SQLite on hyvin pienikokoinen hallintajärjestelmä, joka liitetään sitä käyttävään sovellukseen. Etuna tässä on äärimmäisen helppo siirrettävyys. SQLite sopii hyvin yksinkertaisempiin tilanteisiin, kuten paikalliseksi yksityiseksi tietokannaksi tai testaus- ja opiskelukäyttöön. SQLite ei ole kovin hyvä vaihtoehto mihinkään monimutkaisempaan. (Tezer 2014.)

2.4.2 MySQL

MySQL on yksi suosituimmista tietokantojen hallintajärjestelmistä. SQLite:stä poiketen MySQL toimii siten, että tietokantaa käyttävä sovellus keskustelee MySQL-taustaohjelman kanssa tietokantaan pääsystä. MySQL on tehokas ja turvallinen ratkaisu monimutkaisempaankin käyttöön. Se on helposti saatavilla ja on ominaisuuksiltaan monipuolinen, vaikka ei aivan kaikkia SQL-standardin mukaisia ominaisuuksia sisälläkään. Rinnakkaisissa luku- ja kirjoitusoperaatioissa MySQL on hieman hidas. (Tezer 2014.)

2.4.3 PostgreSQL

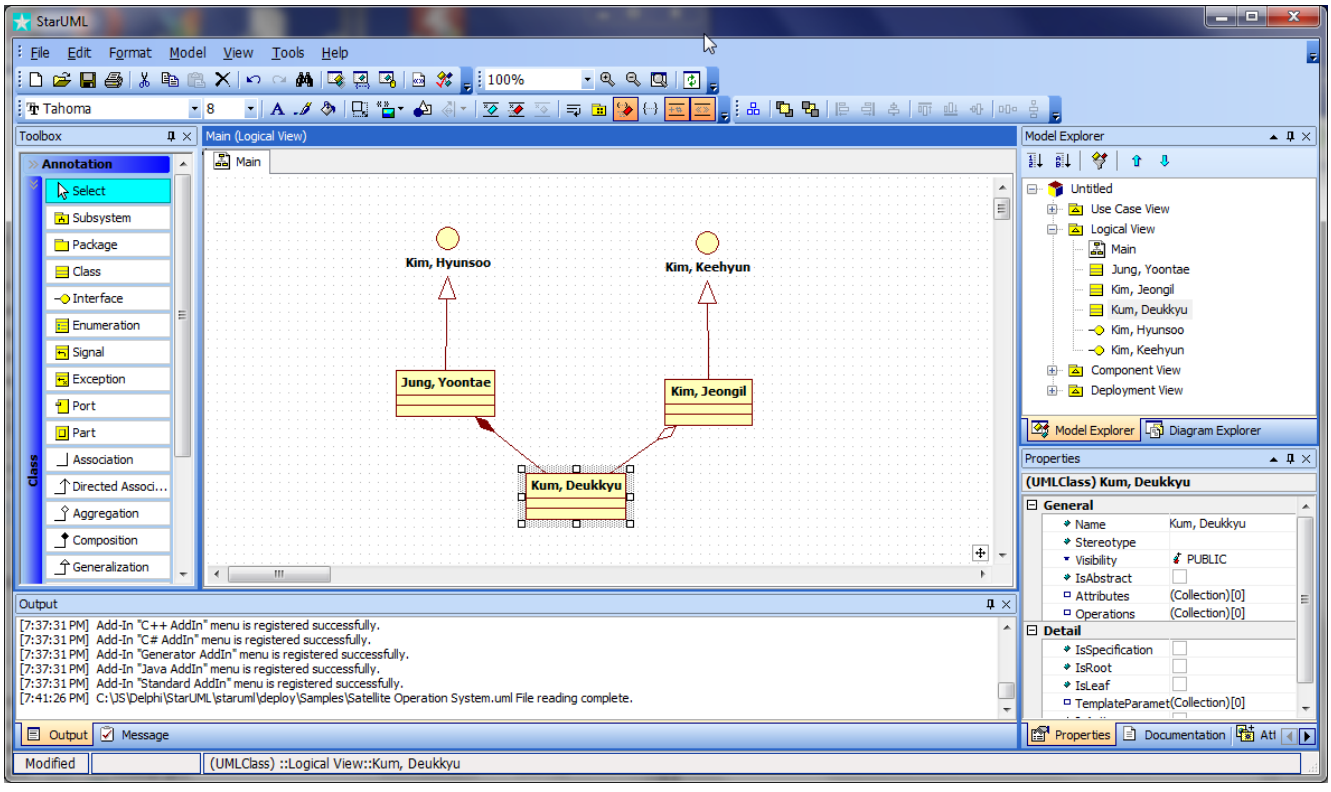
PostgreSQL on vertailun tuotteista monipuolisin ja tehokkain. Se sisältää valtavan määrän tietotyyppejä ja ominaisuuksia. Heikkoja puolia ovat mm. MySQL:ää pienempi suosio, monipuolisuuden tuomat haasteet monimutkaisuudessa sekä hitaus sovelluksissa, joissa luetaan paljon dataa. Se on myös liian raskas yksinkertaisiin sovelluksiin. (Tezer 2014.)

3 KÄYTETTÄVÄT TYÖKALUT

Tässä luvussa tutustutaan käytännön osuudessa käytettäviin työkaluihin, niiden perusominaisuuksiin sekä syihin, miksi juuri ne on valittu. Työkaluja tarvitaan pääasiassa tietokannan suunnitteluun, mutta mukana on myös palvelinympäristö ja yksi tietokannan hallinnan aputyökalu.

3.1 Tietokantojen suunnittelun työkalu

Tietokantojen käsitemalli tehdään StarUML-ohjelmalla, joka on suosittu UML-kaavioiden tekemiseen käytetty työkalu. StarUML on valittu käyttöön sen ilmaisuuden, suosion, helppokäyttöisyyden ja monipuolisuuden takia. Työssä käytetty ohjelmaversio on 5.8.7.0 WhiteStarUML. Sovelluksesta on myös haarukoinnin tuloksena syntynyt musta StarUML 2, joka olisi sopinut aivan yhtä hyvin.



KUVA 1. WhiteStarUML-sovelluksen käyttöliittymä

3.2 Palvelinympäristö

Projektin palvelinympäristönä käytetään WampServer-työkalua. Se on Windowsille tarkoitettu ohjelmistopaketti, joka pitää sisällään Apache-web-palvelimen sekä ympäristön, jossa voi ajaa PHP/Perl/Python-skriptejä ja käyttää MySQL-tietokantaa. WampServer kuuluu ns. WAMP-ohjelmistoihin, joiden nimitys juontaakin edellä mainittujen tuotteiden alkukirjaimista. Vastaavia ohjelmistoja on olemassa muillekin käyttöjärjestelmille. Esimerkiksi Linuxin vastaavasta ohjelmistosta käytetään termiä LAMP ja Mac-käyttöjärjestelmän versiosta termiä MAMP (Nixon 2014, 16).

WampServer on valittu siksi, että se sisältää kaikki tärkeimmät työkalut dynaamisten tietokantoja sisältävien www-sovellusten tekemiseen ja testaukseen sekä on helppokäyttöinen ilmaissovelluskokoelma. Tätä ei loppujen lopuksi juurikaan käytetty työn aikana.

3.3 phpMyAdmin

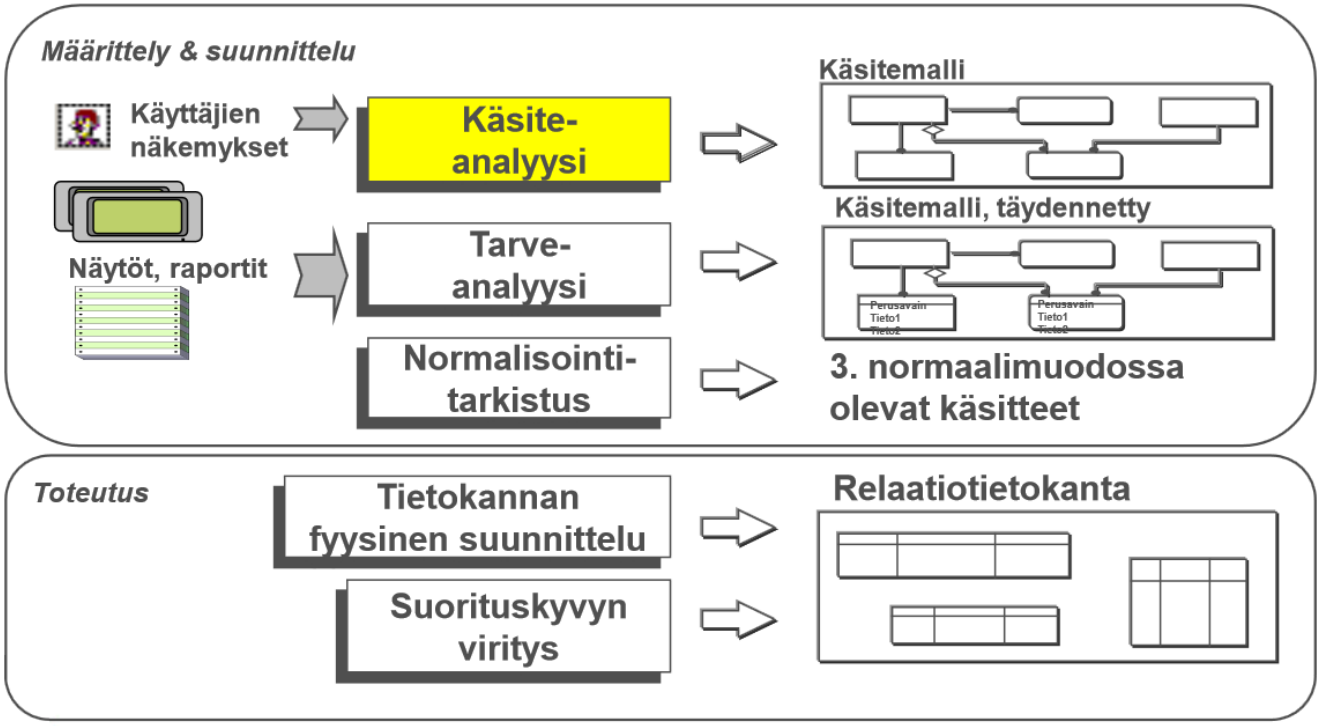
phpMyAdmin on PHP-kielellä toteutettu ohjelma, jota käytetään MySQL-tietokantojen hallintaan. Sovellus itse asiassa kuuluu WAMP-pakettiin, joten ylimääräisiä latauksia ei tästä koitunut. Sovelluksen etuihin kuuluu helppo graafisen käyttöliittymän kautta suoritettava tietokantaoperointi kuten käyttöoikeuksien hallinta, taulujen luonti ja testaus. Myös SQL-komentojen suorittaminen sovelluksesta käsin onnistuu hyvin. (phpMyAdmin 2017.)

4 TIETOKANTOJEN SUUNNITTELU

Tietokantojen suunnittelu on monivaiheinen prosessi. Tässä luvussa käydään läpi nämä vaiheet esimerkkien avulla. Tämä luku myös pohjustaa lukua 5, jossa suunnittelun vaiheita vielä sovelletaan käytännössä harjoittelunohjaushanketietokannan esimerkeillä.

4.1 Suunnitteluputken rakenne

Suunnittelu voidaan jakaa loogiseen ja fyysiseen suunnitteluun. Näistä looginen suunnittelu koostuu tiedon mallinnusvaiheista (käsiteanalyysi, tarveanalyysi ja normalisointi). Fyysinen suunnittelu on vahvasti käyttöympäristöön eli laitteistoon ja tietokannan hallintajärjestelmään sidonnaista suorituskyvyn parantamista. Käsitteet eivät ole aivan yksiselitteisiä, sillä taulujen luonti voidaan ajatella kuuluvan kummalle tahansa puolelle. (Hovi ym. 2005, 24–25; Lightstone, Teorey & Nadeau 2007, 1–2, 5–7.)



KUVIO 6. Tietokantojen suunnitteluputki (Huotari & Hovi 2009, 3).

4.2 Vaatimusmäärittely

Ennen varsinaista loogista suunnittelua on hyvä tehdä vaatimusmäärittely, jossa hieman kartoitetaan tilannetta. Kyseessä on suunnitelma tietokannan ja sitä käyttävän sovelluksen vaatimuksista. Hyvin tehtynä se sisältää kattavan kuvauksen tietojärjestelmän toiminnoista, tiedoista ja tavoitteista. Tässä vaiheessa saattaa syntyä jo jonkinlainen alkeellinen luonnos käsitemallistakin. Määrittely vaatii yhteistyön tekemistä asiakkaan kanssa, joskus sitä ja käsitemallinnusta saatetaan tehdä iteratiivisestikin. (Hovi ym. 2005, 29.)

Vaatimusmäärittelyä voidaan pitää ikään kuin esisuunnitteluvaiheena, sillä se voi olla isonakin apuna, kun lähdetään tekemään käsiteanalyysiä. Vaatimusmäärittely helpottaa myös tarveanalyysin tekemistä.

4.3 Käsiteanalyysi

Käsiteanalyysi on yksi ensimmäisistä suunnitteluvaiheista. Se toteutetaan rajaamalla reaalimaailmasta jokin kohdealue, johon sisältyviä tietoja ja ominaisuuksia tietokantaan tallennetaan. Tämän vaiheen lopputulosta kutsutaan käsitemalliksi, ja se toteutetaan yleensä kaaviona (käsitekaavio, ER-kaavio). Tämän työn kohdalla on päädytty käyttämään UML-notaatiota käsitemallinnuksessa, mutta sitä on käytetty täsmälleen ER-kaavion tapaan. Käsitemallin laatiminen voi olla pitkäkin prosessi, joten sitä saatetaan kehittää asteittain. Pääajatuksena on, että tässä vaiheessa keskitytään pelkästään tietoihin ja unohdetaan suorituskykyasiat. (Hovi ym. 2005, 32–34.)

Käsitemallissa käsite on jokin esine tai asia, josta säilytetään tietoa. Käsitteen tiedot eli attribuutit ovat juuri ne ominaisuudet, jotka kyseisestä käsitteestä tallennetaan. Näiden termien vastineet aiemmin mainitussa relaatiotietokantamallissa ovat taulu ja sarake: käsite voisi siis olla esimerkiksi asiakas, kun taas tietoja voisivat olla asiakkaan nimi ja osoite. (Hovi ym. 2005, 35.)

Taulujen välisten suhteiden suunnittelu

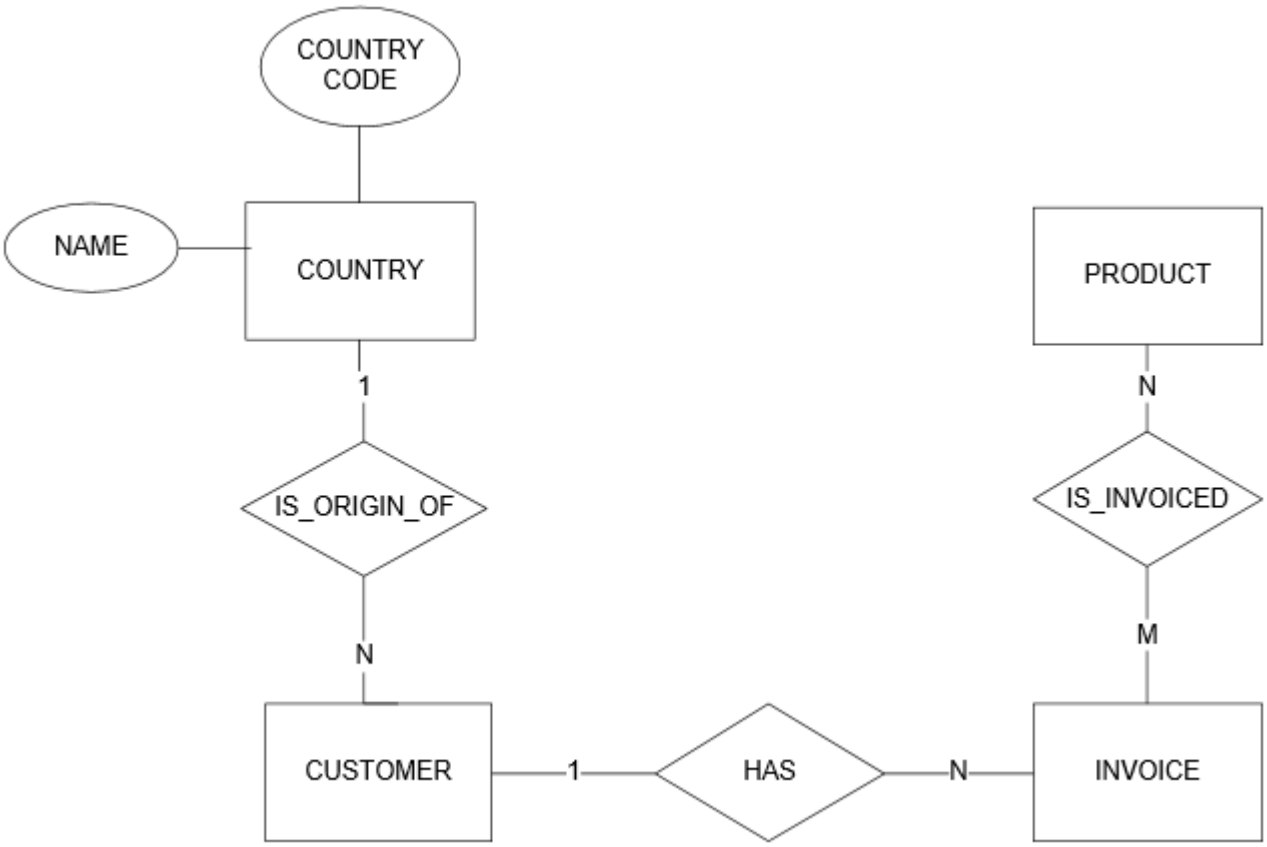
Taulut ovat lähes aina yksi-moneen-suhteessa (Hovi ym. 2005, 37). Niiden taulutoteutuksessa on otettava huomioon, että ensin luodaan isätaulu ja vasta sen jälkeen mahdolliset lapsitaulut, sillä lapsethan

viittaavat isätauluun viiteavaimilla. Tästä mainittiinkin aiemmin jo yksi esimerkki eli käsitteiden oppilaitos ja oppilas välinen yhteys.

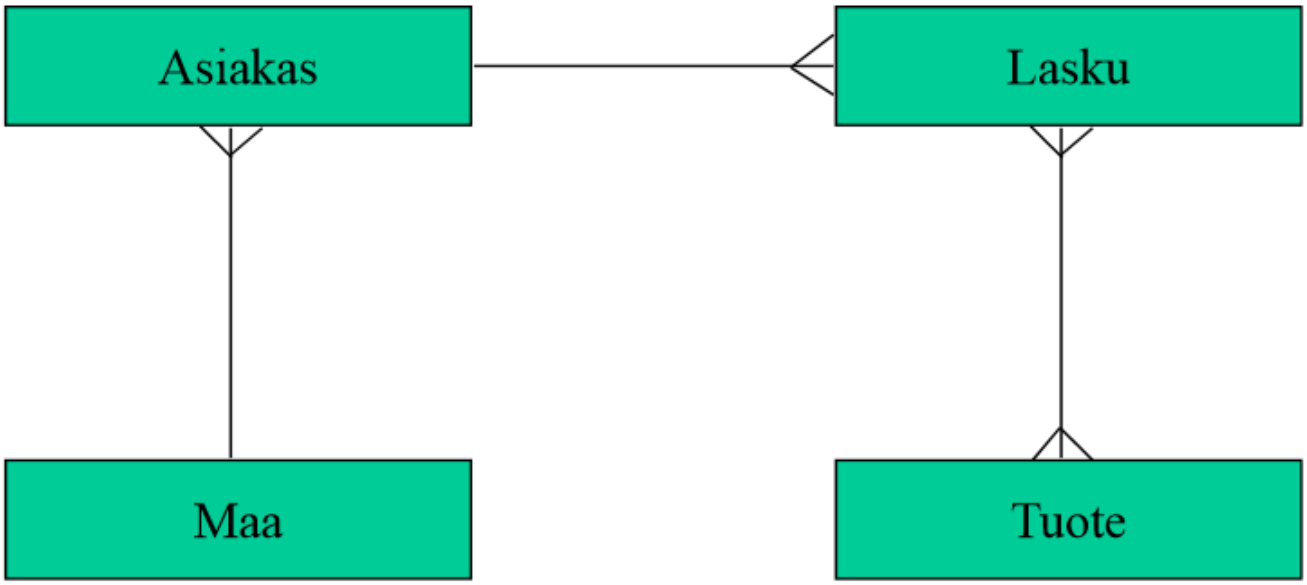
Moni-moneen-yhteyksiä esiintyy myös melko usein. Hyvä esimerkki voisi olla kirjan ja lainaajan välinen suhde: samaa kirjaa voi lainata useampi henkilö ja yksi henkilö voi lainata monia kirjoja samanaikaisesti. Relaatiomalli ei kuitenkaan tue tämmöistä rakennetta, joten se joudutaan purkamaan. Tämä onnistuu, kun luodaan assosiatiivinen käsite kahden moni-moni-käsitteen välille. Kirjan ja lainaajan tapauksessa tämmöinen välikäsite voisi olla lainaus. Tuloksena syntyy kaksi yksi-moneen-yhteyttä. (Hovi ym. 2005, 44, 49.)

Yksi-yhteen-suhteesta hyvä esimerkki on tässäkin työssä esiintyvä opiskelijan ja opinnäytetyön välinen suhde. Yksi-yhteen-yhteys on melko harvinainen, ja sen kohdalla on yleensä hyvä miettiä, voitaisiinko käsitteet sittenkin yhdistää. Joskus voi käydä niinkin, että se paljastuu yksi-moneen-yhteydeksi, kuten tässä esimerkissä lopulta kävi.

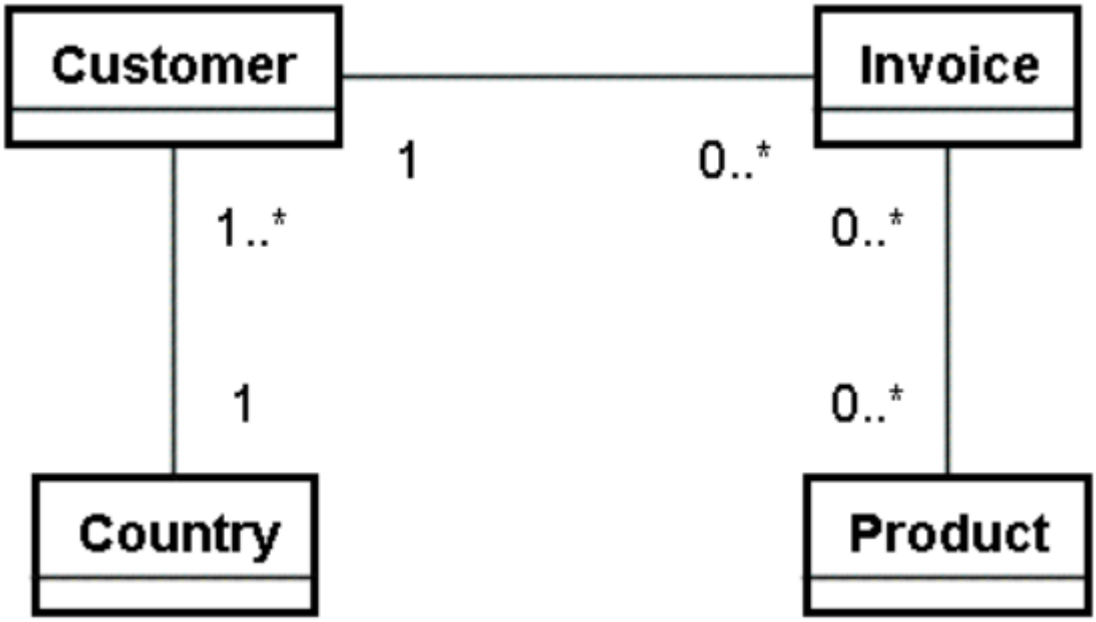
Käytännössä tilanne on hieman monimutkaisempi, sillä todellinen suhteen luonne voi olla rivikohtainen seikka. Käsitemalliin usein merkitään mahdolliset ala- ja ylärajat, jolloin selviää, onko yhteys toiseen suuntaan edes pakollinen. Yhteyksien merkintätapa riippuu käytettävästä notaatiosta. Tunnetuimpiin notaatioihin lukeutuvat Chenin notaatio, harakanvarvasnotaatio ja UML-notaatio eli luokkakaavio. Seuraavissa kuvioissa on esitetty sama käsitekaavio kaikilla näillä kolmella eri notaatiolla.



KUVIO 7. Chenin notaatio (Ranne 2014, 6)



KUVIO 8. Harakanvarvasnotaatio (Ranne 2014, 7)



KUVIO 9. UML-notaatio (Ranne 2014, 8)

4.4 Tarveanalyysi

Tässä vaiheessa täydennetään käsitemallia. Tähän asti tehtyä mallia analysoidaan yksityiskohtaisemmin tutkimalla tietokannan kaikkia käyttökohteita mahdollisimman tarkasti. Näitä kohteita sanotaan tietotarpeiksi, ja niitä ovat muun muassa lomakkeet, käyttöliittymä, listat ja raportit. Tarveanalyysissä voidaan myös miettiä tietojen ja tapahtumien lukumääriä, mutta nykyään sen merkitys on pienentynyt. (Hovi ym. 2005, 80, 83.)

Tässä työssä on käytetty hieman toisenlaista menetelmää, sillä tietokantaa käyttävästä sovelluksesta ei ole valmista toteutusta tai edes tarkkaa suunnitelmaa. Tarveanalyysi on tehty tutkimalla toimeksiantajalta saatuja valmiita tietovaatimuksia ja käymällä keskusteluja toimeksiantajan yhteyshenkilön kanssa. Näiden perusteella on saatu selville tarvittavia toimintoja ja käyttäjiä, joiden perusteella on luotu käyttötapauskaavio (use case diagram). Käyttötapauskaaviossa kuvataan järjestelmän käyttäjiä ja näiden käyttäjien tapoja käyttää järjestelmää, eli siis ”käyttötapauksia”. Sitä tutkimalla on sitten saatu parannettua käsitemallia.

4.5 Normalisointi

Normalisoinnin avulla tietokantataulut saatetaan parempaan muotoon. Se on prosessi, jossa muun muassa minimoidaan tietojen toisto. Käytännössä normalisointi aiheuttaa melkein aina taulujen pilkkomista useampiin tauluihin. Normaalimuotoja on monta, joista seuraavaksi käydään läpi kolme ensimmäistä. Kolmannessa normaalimuodossa olevaa tietokantaa voidaan pitää riittävän hyvänä. (Hovi ym. 2005, 86.)

Normalisoinnin tuloksena oleva tietokanta on selkeä ja hyvin päivitettävissä, toisin kuin normalisoimaton, huonosti suunniteltu tietokanta. Ehkä kaikista tärkein etu on tietojen yhdenmukaisuus: tietoja päivitettäessä laitetaan uusi arvo aina sille kuuluvaan, yksittäiseen tauluun. Normalisoimattomassa tietokannassa tietoja muutettaessa tai poistettaessa saatetaan muutos joutua tekemään useaan tauluun. Jos jokin näistä tauluista unohdetaan, rikkoutuu tietokannan eheys. Huonona puolena normalisoinnissa on kuitenkin se mahdollisuus, että haut ovat hieman hitaampia ja monimutkaisempia, sillä hakukyselyissä joudutaan nyt tekemään enemmän taulujen liitoksia. Juuri tästä syystä tietovarastoja ei saateta yhtä korkeaan normaalimuotoon. (Nixon 2014, 211; Hovi ym. 2005, 86, 97.)

4.5.1 Ensimmäinen normaalimuoto

Tietokantoja suunniteltaessa voi usein tulla vastaan tilanteita, joissa tarvitaan toistuvia samankaltaisia kenttiä. Esimerkiksi kaupan tietokanta ylläpitää tuotteiden hintoja. Entä sitten, kun hinnat muuttuvat? Yksi tapa toteuttaa tämä olisi laittaa kullekin hinnalle ja sen voimassaoloajalle omat sarakkeensa eli hinta1, pvm1, hinta2, pvm2 ja niin edelleen. Tämä ei kuitenkaan olisi hyvä ratkaisu, ja se rikkookin juuri ensimmäistä normaalimuotoa. Nämä niin sanotut toistuvat ryhmät saadaan eliminoitua luomalla uusi taulu, johon tulee sarakkeiksi hinta, hinnan alkamis- tai päättymispäivämäärä sekä tietenkin tuote, jonka hinta on kyseessä. (Hovi ym. 2005, 87–89.)

Toinen ensimmäisen normaalimuodon ohje on poistaa moniarvoiset kentät. Kenttää, jota ei voi jakaa osiin, sanotaan atomiseksi. Moniarvoisesta kentästä hyvin yleinen esimerkki on puhelinnumerot: jos yhdelle henkilölle halutaan voivan tallentaa monta puhelinnumeroa, saattaa silti tulla mieleen luoda vain yksi sarake kaikille numeroille, jotka sitten erotellaan toisistaan pilkulla. Tämä ei välttämättä nopeasti ajateltuna vaikuta huonolta ratkaisulta, mutta pitkän puhelinnumerolistan sisällä olevien numeroiden poistaminen tai päivittäminen olisi hyvin epäkäytännöllistä. Ensimmäisen normaalimuodon mukaisessa

tietokantarakenteessa puhelinnumerot voi viedä uuteen tauluun. On myös mahdollista nimetä eri puhelinnumerot jollain loogisella tavalla, esimerkiksi matkapuhelin ja työpuhelin. (Hovi ym. 2005, 87–89; Lahtonen 2012, 31–32.)

4.5.2 Toinen normaalimuoto

Toisessa normalisointimuodossa tutkitaan taulukohtaisesti sarakkeiden välisiä riippuvuuksia. Kuvitellaan taulu, jossa on sarakkeet opiskelijatunnus, etunimi ja sukunimi. Kutakin opiskelijatunnusta vastaa aina yksi ja sama etunimi. Tästä sanotaan, että etunimi on funktionaalisesti riippuvainen opiskelijatunnuksesta, kuten on myös sukunimikin. (Hovi ym. 2005, 90–91.)

Mikäli taululla on yksiosainen perusavain ja se on ensimmäisessä normaalimuodossa, on se jo valmiiksi toisessakin normaalimuodossa. Jos taululla on moniosainen perusavain, on kaikkien muiden kuin perusavaimeen kuuluvien sarakkeiden oltava funktionaalisesti riippuvia koko perusavaimesta. Osittaisia riippuvuuksia eli riippuvuuksia vain jostain perusavaimen osasta ei saisi olla. Toinen normaalimuoto tuo tietokannan sellaiseen muotoon, jossa tauluihin liittyy vain selkeästi juuri siihen kuuluvia sarakkeita. Esimerkiksi verkkokaupan tietokannan tilausrivi-taulussa ei saisi olla asiakkaan nimitietoja. (Hovi ym. 2005, 90–92; Lahtonen 2002, 33, 36.)

4.5.3 Kolmas normaalimuoto

Tämä normaalimuoto tuo vielä yhden lisäsäännön aikaisempaan: sarakkeet saavat olla funktionaalisesti riippuvaisia ainoastaan perusavaimesta. Jos esimerkiksi taulussa on sarakkeina postinumero ja postitoimipaikka, on näiden välillä funktionaalinen riippuvuus, joka ei enää ole sallittua. Kolmanteen normaalimuotoon tämä saadaan luomalla postinumeroille ja postitoimipaikoille oma taulunsa, ja sitten alkuperäiseen tauluun tulee vain postinumero. (Nixon 2014, 217; Hovi ym. 2005, 93–94).

4.5.4 Denormalisointi

Joskus tauluja ei haluta normalisoida liian pitkälle. Esimerkiksi suuret tietovarastot voivat sisältää jonkin verran toistoa. Jos tietoa enimmäkseen vain haetaan, niin pienempi normaalimuoto voi olla jopa nopeampi ratkaisu. Denormalisoinnilla tarkoitetaankin suuremmasta normaalimuodosta palaamista pienempään. (Hovi ym. 2005, 95–97.)

4.6 Taulujen muodostaminen

Hyvän suunnittelun jälkeen on melko helppoa muodostaa varsinaiset tietokantataulut: kukin käsite vastaa aina yhtä taulua ja attribuutit ovat taulujen sarakkeita. Taulujen välisissä yhteyksissä lisätään viiteavaimet. Varsinaisiksi haasteiksi jäävät tietokantatuotteen valinta ja käytettävät tietotyypit.

4.7 Indeksointi

Relaatiotietokannan hallintajärjestelmä käyttää ns. optimointiohjelmaa eli optimoijaa tehokkaimman ja nopeimman suorituskeinon etsimisessä tietylle kyselylle. Indeksejä perustamalla saadaan kyselyille uusia suoritustapoja eli saantipolkuja, ja optimoija voi niitä hyödyntämällä suoriutua tehtävästä nopeammin. Indeksointi kuuluu tietokantasuunnittelun viimeisimpiin vaiheisiin ja se on oikeastaan ”vain” suorituskyvyn parantamista. Indeksoinnin ollessa hyvin laaja aihealue sitä ei käsitellä kovin yksityiskohtaisesti. (Hovi ym. 2005, 149.)

Puutteellisella tai huonolla indeksoinnilla haut suurista tauluista voisivat nousta monta sekuntia tai jopa minuuttia kestäviksi. Indeksoinnin merkitys siis nousee tärkeäksi, kun taulut ovat hyvin suuria, esim. vähintään kymmenien tai satojen tuhansien rivien luokkaa. Hyvänä puolena on se, että indeksointia pystyy muuttamaan jälkeenpäin, siis vaikka tietokanta olisi jo otettu käyttöön kauan sitten (Hovi ym. 2005, 158). Indeksejä suunniteltaessa on kuitenkin muistettava, että ylimääräiset indeksit hidastavat taulujen päivityksiä, joten niiden lisäämistä on harkittava tarkkaan (Lightstone ym. 2007, 58).

Indeksit toimivat eräänlaisina hakemistoina. Yksinkertaisinta on verrata niitä kirjan sisällysluetteloon tai vaikka puhelinluetteloon. Teknisesti indeksit toteutetaan usein ns. B-puuna (B-tree). Kyseessä on useasta tasosta koostuva puu-tietorakenne. Ylintä osaa kutsutaan juurisivuksi ja alimpana ovat lehtisivut,

joilta on suorat osoittimet tietokantasivuille. Tiedot indeksissä on lajiteltuna, mikä pääasiassa tekeekin siitä niin tehokkaan. Esimerkiksi indeksi sukunimi olisi lajiteltuna sukunimien mukaan aakkosjärjestyksessä. Usein select-lauseille johdetut parhaimmat indeksit koostuvat monesta kentästä, esim. sukunimi, kunta, etunimi, asiakasnumero. (Hovi ym. 2005, 153, 156–157, 186).

Hyvänä lähtökotana on aloittaa kaikista välttämättömimmistä indekseistä. Niitä ovat taulujen perus- ja viiteavaimet, joiden indeksointi tekee liitoksista nopeampia. Tämä on järkevää, vaikka taulujen koot eivät vielä olisikaan kovin isoja. Voi olla, että tietokannan hallintajärjestelmä luo nämä indeksit automaattisesti, mutta asia kannattaa silti varmistaa. Liitteessä 1 on esitetty perusindeksointi phpMyAdmin-sovelluksella taulujen luomisen yhteydessä. (Lightstone ym. 2007, 55–56.)

5 HANKETIETOKANNAN SUUNNITTELU

Nyt on aika toteuttaa hanketietokanta edellisen luvun suunnitelman mukaan. Käydään suunnitteluputki läpi samalla logiikalla, vaatimusmäärittelystä normalisointiin.

5.1 Vaatimusmäärittely

Kehittämistehtävän kuvaus. Tavoitteena on suunnitella ja toteuttaa tietokanta harjoittelunohjaushankkeeseen. Vaatimuksena on, että tietokantaan pystytään tallentamaan eri osapuolten yhteystiedot sekä vastuuhenkilöt. Osapuolet voivat olla joko ammattiopistoja, ammattikorkeakouluja tai yrityksiä. Toimeksiantajana on Centria-ammattikorkeakoulun hallinnoima hanke Ydinosaajat.

Tiedot. Toimeksiantajalta saatiin moniste, joka sisältää tarvittavat perustiedot kohtuullisen tarkalla tasolla. Moniste tarjosi siten hyvän pohjan vaatimusmäärittelylle ja käsitemallin ensimmäiselle versiolle. Seuraavassa on listattuna kyseisen monisteen tiedot.

Tiedot yrityksestä

- Yrityksen nimi
- Toimiala
- Koulutusalat, joilta yritys ottaa harjoittelijoita (luettelo)
- Osoitetiedot
- Puhelin
- Www-sivut
- Kumppanuussopimus tehty (kyllä / ei)
- Harjoittelijarekrytointi verkkosivujen kautta (kyllä / ei)
- Yritys tarjoaa harjoittelupaikkoja, yritykselle parhaiten soveltuvat harjoittelujaksot
    - Jatkuvasti
    - Tammikuu – huhtikuu
    - Toukokuu – elokuu
    - Syyskuu – joulukuu
    - Sopimuksen mukaan ajoittain
- Harjoittelijan tehtävät ovat
    - Opistotasoista harjoittelua
    - AMK-tasoista harjoittelua
- Yritys tarjoaa opinnäytetyön/päättötyön aiheita/toimeksiantoja
    - Opistotason opiskelijoille
    - Ammattikorkeakoulututkintoa suorittaville
    - Ylempää amk-tutkintoa suorittaville
    - Yrityksellä ei toimeksiantoja

- tarjoaa näyttömahdollisuuksia opintoihin (kyllä / ei)

Harjoittelun yhteyshenkilö yrityksessä

- Nimi
- Tehtävänimi
- Sähköposti
- Puhelin

Harjoittelun työpaikkaohjaajat yrityksessä

- Nimi
- Tehtävänimike
- Sähköposti
- Puhelin
- Harjoittelunohjaajakoulutus suoritettu (kyllä / ei)

Tietoa oppilaitoksesta

- Oppilaitoksen nimi
- Osoitetiedot
- Puhelin
- Www-sivut
- Oppilaitoksen harjoittelun yhteyshenkilö(t)
    - Nimi
    - Tehtävänimike
    - Sähköposti
    - Puhelin
- Oppilaitoksen työssä ohjaajat/harjoittelunohjaajat
    - Nimi
    - Tehtävänimike
    - Sähköposti
    - Puhelin

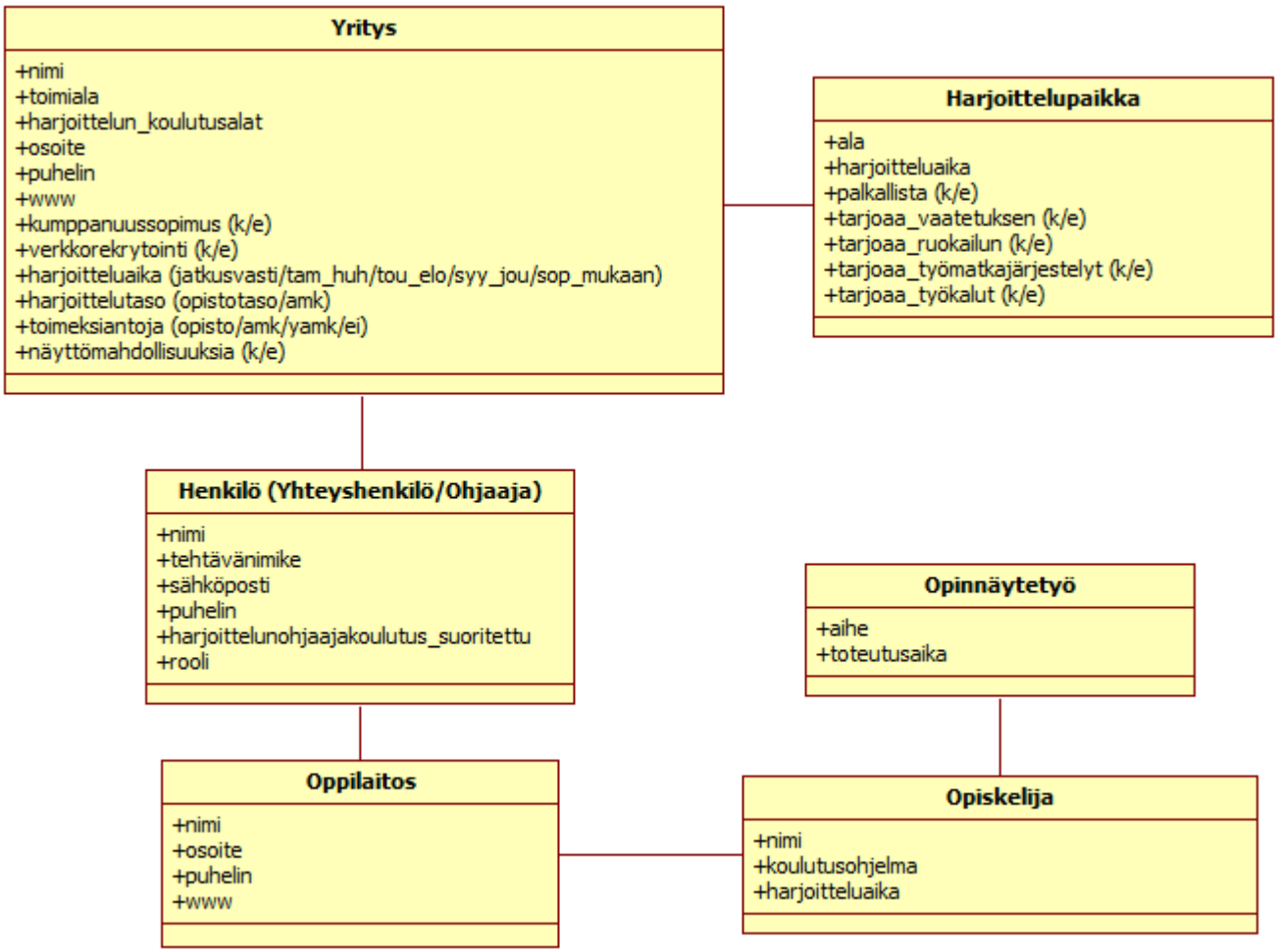
Opiskelija harjoittelussa

- Oppilaitos, jossa opiskelee
- Koulutusohjelma
- Opiskelijan nimi
- Harjoitteluaika
- Harjoittelu on (palkallista / palkatonta)
- Työpaikka tarjoaa
    - (Turva)vaatetuksen
    - Ruokailun
    - Työmatkajärjestelyt
    - Työkalut
- Opinnäytetyö
    - Opinnäytetyön aihe
    - Oppilaitos
    - Koulutusohjelma
    - Opiskelijan nimi
    - Toteutusaika

Toiminnot.

Käsitteiden ja tietojen perusteella saadaan selville, että pakollisia toimintoja ovat vähintäänkin:
P1. Yritystietojen (ainakin nimi, ala ja yhteystiedot) ylläpito.
P2. Oppilaitostietojen (ainakin nimi ja yhteystiedot) ja opiskelijatietojen (ainakin henkilötiedot, oppilaitos ja koulutusohjelma, opinnäytetyön perustiedot) ylläpito.
P3. Yritysten ja oppilaitosten yhteyshenkilö- ja ohjaajatietojen (ainakin nimi, tehtävänimike, koulutus, yhteystiedot) ylläpito.
P4. Harjoittelupaikkatietojen ylläpito.
P5. Harjoittelupaikkoja on voitava hakea monenlaisten kriteerien mukaan, ainakin yrityksittäin, aloittain sekä harjoitteluajan, harjoittelualan tason ja palkan olemassaolon perusteella. Näitä voi tietenkin yhdistellä.

Koska monisteesta selvisi paljon tietoja, tehdään sen perusteella alustava käsitemalli jo nyt (kuvio 10). Kuvion käsitteet ja attribuutit on tässä vaiheessa otettu lähes suoraan listalta. Yksinkertaisuuden vuoksi vielä tässä vaiheessa ei mietitä yhteyksien luonteita.



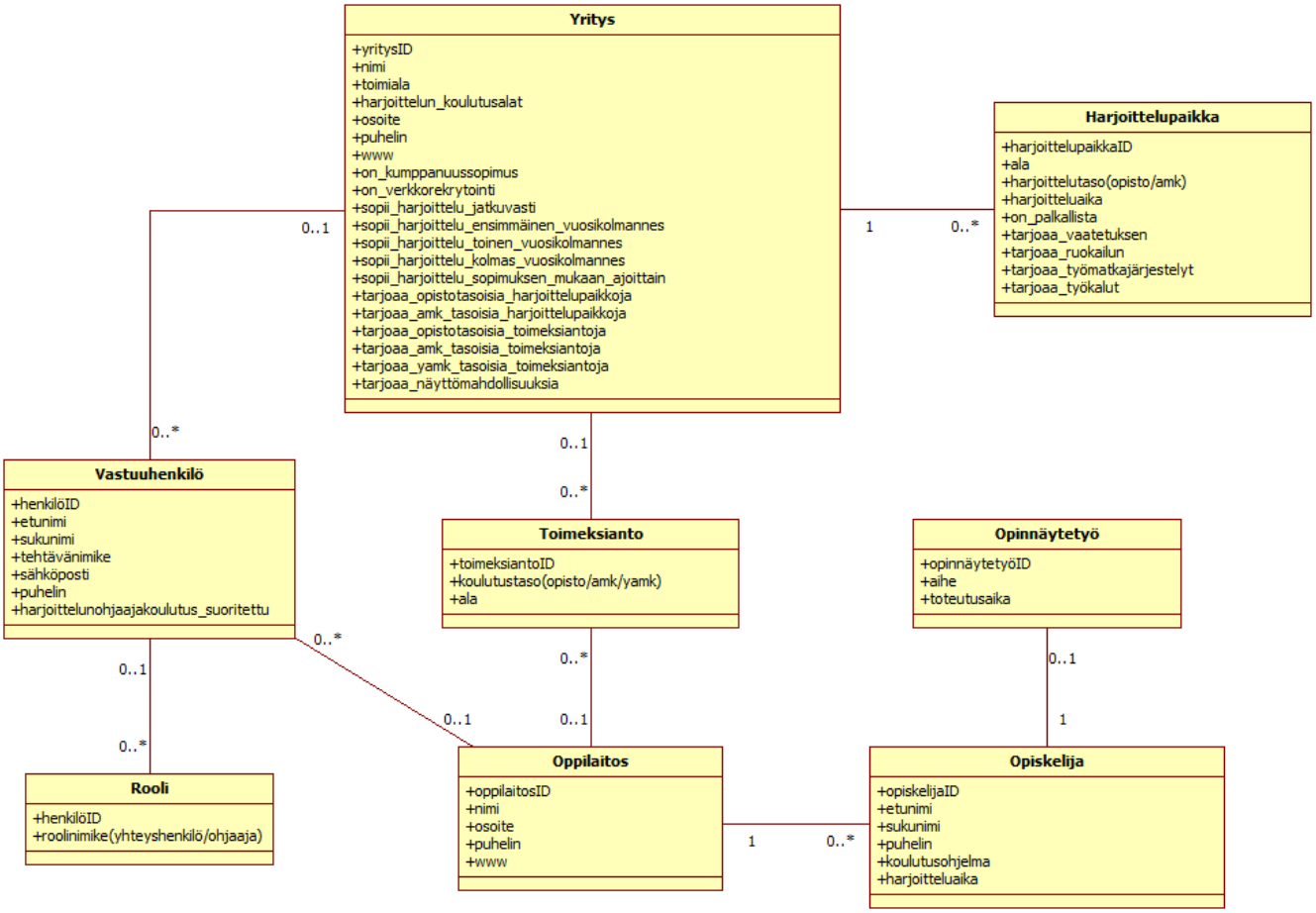
KUVIO 10. Alustava käsitekaavio

5.2 Käsiteanalyysi

Alustava käsitemalli on jo muodostettu, joten parannetaan sitä. Tässä vaiheessa merkitään kaikki käsitteiden väliset yhteydet sekä lisätään perusavainkentät. Lisäksi pohditaan, löytyykö käsitteille uusia attribuutteja tai löytyykö jopa uusia käsitteitä sekä kannattaako joitakin nykyisiä käsitteitä tai attribuutteja muuttaa paremmiksi esimerkiksi nimeämisen osalta. Viiteavaimia ei lisätä, koska ne kuuluvat tauluihin eivätkä käsitemalliin.

Käsiteanalyysivaiheessa ilmenee seuraavia muutoksia alustavaan käsitemalliin:

- Lisätään perusavaimet jokaiseen käsitteeseen.
- Lisätään yhteyksien luonteet.
- Lisätään Oppilaitokselle ja Yritykselle kenttä osoite. Henkilöille ei koettu osoitteen lisäämistä välttämättömäksi.
- Muutetaan kyllä-/ei-kenttien nimiä siten, että ne kaikki alkavat verbillä. Tämä tehdään loogisemman nimeämisen saamiseksi.
- Muutetaan Henkilö-käsitteen nimeksi Vastuuhenkilö.
- Muutetaan henkilöiden nimet erillisiksi etu- ja sukunimitiedoiksi.
- Hajotetaan harjoitteluaika-kenttä erillisiin totuusarvokenttiin. Toinen ratkaisuvaihtoehto olisi luoda harjoitteluajalle oma taulu, mutta se edustaisi yksi-yhteen-suhdetta ja tekisi hakuja monimutkaisemmiksi.
- Hajotetaan harjoittelutaso-kenttä kahdeksi totuusarvokentäksi (yritys voi tarjota paikkoja molemmillekin tasoille) ja lisätään harjoittelupaikalle samanlainen toinen kenttä nimellä harjoittelutaso (voi olla vain joko opistotaso tai amk-taso). Nyt Yritys-taulusta voi hakea yrityksen mahdollisesti tarjoamia harjoittelutasoja ja Harjoittelupaikka-taulun kautta näkee yksittäisen paikan tason.
- Lisätään käsite Toimeksianto, jonka voi myöntää yritys tai oppilaitos. Lisäksi Yritys-käsitteen toimeksianto hajotetaan samalla periaatteella kuin harjoittelutasokin.
- Lisätään käsite HenkilönRooli, jonka yksi ilmentymä kuvaa yhtä roolia yhdelle henkilölle. Henkilö-taulun kautta päästään näkemään, kuuluuko henkilö oppilaitokselle vai yritykselle.



KUVIO 11. Paranneltu käsitekaavio

5.3 Tarveanalyysi

Tarveanalyysi on aloitettu vaatimusmäärittelyn tarkentamisella. Nykyisten tietojen ja toimeksiantajan yhteyshenkilön kanssa käytyjen keskustelujen tuloksena ilmaantui vielä joitakin lisätarpeita:

P6. Toimeksiantoja on voitava hakea monenlaisten kriteerien mukaan, ainakin yrityksen tai oppilaitoksen tarjoamat toimeksiannot sekä alan ja tason mukaiset haut.
P7. Koulutusalatiedot: kytkettävä oppilaitoksiin (mitä koulutuksia tarjoaa: useita), vastuuhenkilöihin (miten koulutettu) ja opiskelijaan (miten kouluttautuu).
P8. Harjoittelualatiedot: kytkettävä yrityksiin (minkä alojen töitä tarjoaa: useita) ja harjoittelupaikkoihin (minkä alan työtä kukin työ on).
P9. On esitettävä erikseen harjoittelusopimukset eli toteutuneet harjoittelujaksot. Yritys voi tarjota useita paikkoja samalle työlle.
P10. Harjoittelu voi olla perusharjoittelua tai ammatillista harjoittelua. Tämä täytyy saada selville jokaisesta harjoittelupaikasta.
P11. Tarvitaan tiedot opiskelijoiden menneistä harjoittelujaksoista. Tämä onkin itse asiassa opiskelijan harjoitteluaika-kentän todellinen merkitys. Tämän voisi yhdistää kohdan P9 kanssa yhdeksi käsitteeksi.

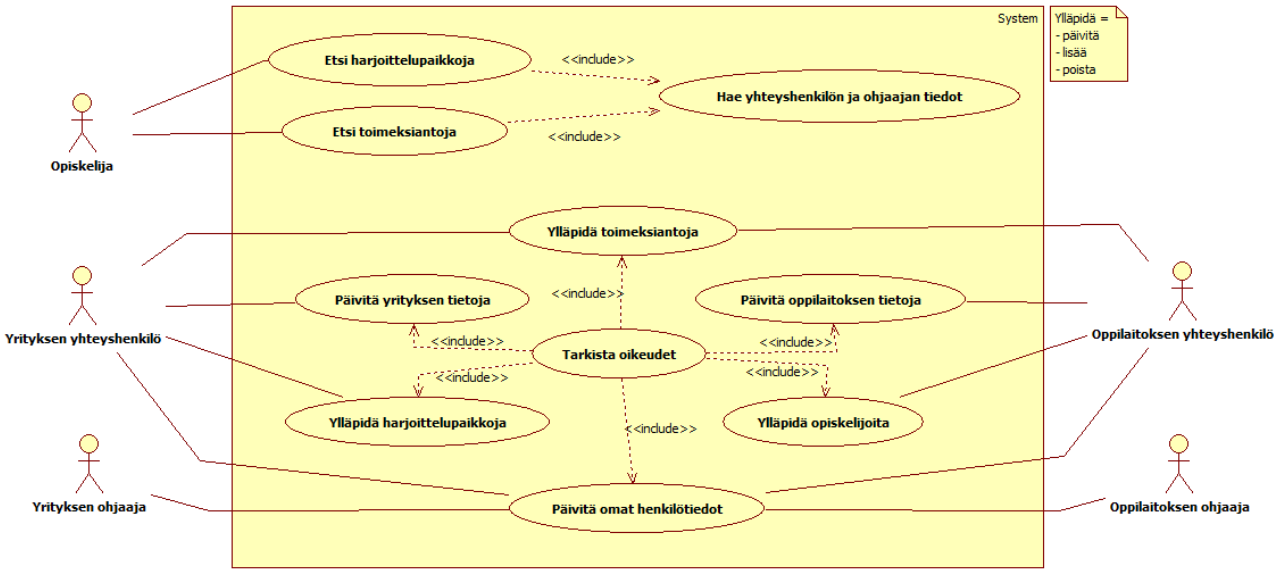
Käyttötapauskaavio (Use case diagram)

Tässä vaiheessa on mietitty tietokannan käyttäjiä ja kyseisten käyttäjien käyttötapauksia. Aluksi on laadittu tekstimuotoinen lista, jonka jälkeen on tehty graafinen UML-käyttötapauskaavio.

Tietokannalla on seuraavia käyttäjiä (actors)

- Opiskelija
  - Harjoittelupaikkojen hakeminen
  - Toimeksiantojen hakeminen
  - Haetun harjoittelupaikan tai toimeksiannon yhteyshenkilön ja ohjaajan hakeminen
- Oppilaitoksen ohjaaja ja yrityksen ohjaaja
  - Tarvittaessa kaikki opiskelijan käyttötapaukset
  - Omien henkilötietojen päivitys
  - Listaus ohjattavista toimeksiannoista
- Oppilaitoksen yhteyshenkilö
  - Tarvittaessa kaikki opiskelijan käyttötapaukset
  - Omien tietojen päivitys
  - Oppilaitoksen (vain oman työpaikan) tietojen päivitys
  - Opiskelijoiden (vain oman työpaikan) tietojen päivitys, lisäys ja poisto
  - Opiskelijoiden (vain oman työpaikan) opinnäytetöiden tietojen päivitys, lisäys ja poisto. Yksinkertaistaen voidaan opiskelijan opinnäytetyön tietoja pitää opiskelijan tietoina.
  - Oppilaitoksen (vain oman työpaikan) tarjoamien toimeksiantojen päivitys, lisäys ja poisto
- Yrityksen yhteyshenkilö
  - Tarvittaessa kaikki opiskelijan käyttötapaukset
  - Omien tietojen päivitys
  - Yrityksen (vain oman työpaikan) tietojen päivitys
  - Yrityksen (vain oman työpaikan) tarjoamien harjoittelupaikkojen päivitys, lisäys ja poisto
  - Yrityksen (vain oman työpaikan) tarjoamien toimeksiantojen päivitys, lisäys ja poisto
- Järjestelmän ylläpitäjä
  - Kaikkien taulujen lukeminen, päivitys, tietojen lisäys ja poisto

Kaaviota on yksinkertaistettu hieman, jotta se pysyisi selkeänä ja auttaisi paremmin hahmottamaan tilannetta. Jokaista mahdollista hakua ei kannata lähteä kaavioon listaamaan, ja esimerkiksi ylläpitäjää ei ole merkitty.



KUVIO 12. Tietokannan käyttötapauskaavio

Yrityksellä voi olla suuri määrä ohjaajia ja yhteyshenkilöitä erilaisille projekteille. Käyttötapauskaaviosta selviääkin, että vastuuhenkilöllä (yhteyshenkilö tai ohjaaja) on yhteys myös harjoittelupaikkaan ja toimeksiantoon. Tästä voidaan johtaa, että vastuuhenkilö voi seurata paikkoja ja toimeksiantoja, joihin hän itse liittyy. Tämä ominaisuus jätettiin kaavioon merkitsemättä, mutta periaatteessa kyseessä on vain yksi tapa etsiä harjoittelupaikkaa tai toimeksiantoa. Voidaan lisäksi olettaa, että kaikissa saman harjoittelupaikan harjoittelusopimuksissa on keskenään samat vastuuhenkilöt.

Kaaviosta suuren osan vie yritysten ja oppilaitosten vastuuhenkilöt. Näihin on kaikkiin tapauksiin merkitty, että ennen tietojen päivittämistä, lisäämistä ja poistamista on tarkistettava oikeudet. Muiden yritysten, oppilaitosten tai henkilöiden tietoja ei pitäisi päästä muokkaamaan. Oikeustarkistukset ovat tietenkin erilaisia riippuen taulusta, mutta selkeyden vuoksi ne on laitettu samaan käyttötapaukseen, joka suoritetaan ennen mitään näistä operaatioista. Tarkistus edellyttää jonkinlaisen autentikointijärjestelmän tekemistä, mutta sen toteutus jää tämän työn ulkopuolelle.

Käydään vielä läpi pakolliseksi määritellyt tarpeet ja käsitekohtaisesti mietitään, puuttuuko jotain. Nykyisille käsitteille löydettiin hieman uusia tietoja, kuten syntymäaika opiskelijalle ja Yritys-taulun mukaiset erilliset totuusarvokentät harjoittelupaikan harjoitteluajalle. Opiskelija-tauluun muutettiin perusavainratkaisuksi oppilaitoksen ja opiskelijatunnuksen yhdistelmä. Lisäksi opinnäytetyö kytkettiin toimeksiantoon ja opinnäytetyölle mahdollistettiin useampi tekijä.

Uusia käsitteitä syntyi seuraavasti:

Koulutusala ja **OppilaitoksenKoulutusala** – Saadaan selville kunkin oppilaitoksen koulutusohjelmat. Tästä piti tehdä kaksi taulua, jotta vältyttäisiin moni-moneen-yhteydeltä. Näistä Koulutusala on myös yhteydessä opiskelijaan ja vastuuhenkilöön.

Harjoitteluala ja **YrityksenHarjoitteluala** – Nämä on lisätty samalla periaatteella kuin Koulutusala ja OppilaitoksenKoulutusala. Nämä luodaan erikseen, sillä harjoittelupaikkakohtaiset alat poikkeavat koulutusohjelmien nimikkeistä.

Harjoittelusopimus – Tämä on toteutunut opiskelijakohtainen harjoittelupaikkatapaus. Yhteen Harjoittelupaikkaan voi liittyä monta sopimusta. Tähän lisätään vielä harjoittelujakson aloitus- ja lopetuspäivämäärät. Lopetuspäivämäärään voisi tarvittaessa sallia NULL-arvoja.

Tässä vaiheessa huomattiin vielä se, että jos vastuuhenkilöllä on useita rooleja, ei henkilön roolia tietyssä harjoittelupaikassa saa selville. Toinen ilmennyt ongelma oli harjoittelupaikan yhteys vastuuhenkilöön. Koska harjoittelupaikalla on sekä yksi yhteyshenkilö että yksi ohjaaja, syntyi tästä ”kaksi-moneen”-suhde. Näiden syiden perusteella koettiin järkevämmäksi sittenkin hajottaa Vastuuhenkilö-käsite, joten saatiin käsitteet **Yhteyshenkilö** ja **Ohjaaja**. Rooli-käsite voitiin poistaa tarpeettomana.

5.4 Normalisointi

Seuraavaksi normalisoidaan käsitemalli kolmanteen normaalimuotoon. Normalisointivaiheessa ei synny enää paljoa muutoksia, sillä käsitemalli on jo melko hyvällä tasolla. Normalisoinnin kohdalla on kuitenkin muistettava, että jos käsitemallia muutetaan tulevaisuudessa, on normalisointi tarkistettava uudestaan. Seuraavassa on listattuna kunkin normaalimuodon tuomat muutokset.

- Ensimmäinen normaalimuoto: Toistuvia ryhmiä ei löydy yhdestäkään käsitteestä. Osoitteessa on tapana olla mukana postinumero ja postitoimipaikka (eli se on oikeastaan moniarvoinen), joten päivitetään käsitteitä Yritys ja Oppilaitos osoitetietojen osalta. Yhteyshenkilö-, Ohjaaja- ja Opiskelija-käsitteille ei ole koettu osoitetietoja välttämättömäksi, mutta tarvittaessa niillekin voitaisiin ne lisätä. Henkilöiden nimet olisivat olleet myös moniarvoisia, mutta ne korjattiin jo aiemmin. Uusia käsitteitä ei syntynyt.
- Toinen normaalimuoto: Kaaviossa ei esiinny juurikaan sellaisia käsitteitä, joilla on moniosainen perusavain, koska suurimmaksi osaksi on käytetty surrogaatteja. Ainoastaan YrityksenHarjoitteluala, OppilaitoksenHarjoitteluala, Harjoittelusopimus ja Opiskelija poikkeavat tästä, mutta ne

ovat valmiiksi toisessa normaalimuodossa. Tietokanta täyttää siis ilman muutoksia toisenkin normaalimuodon vaatimukset.

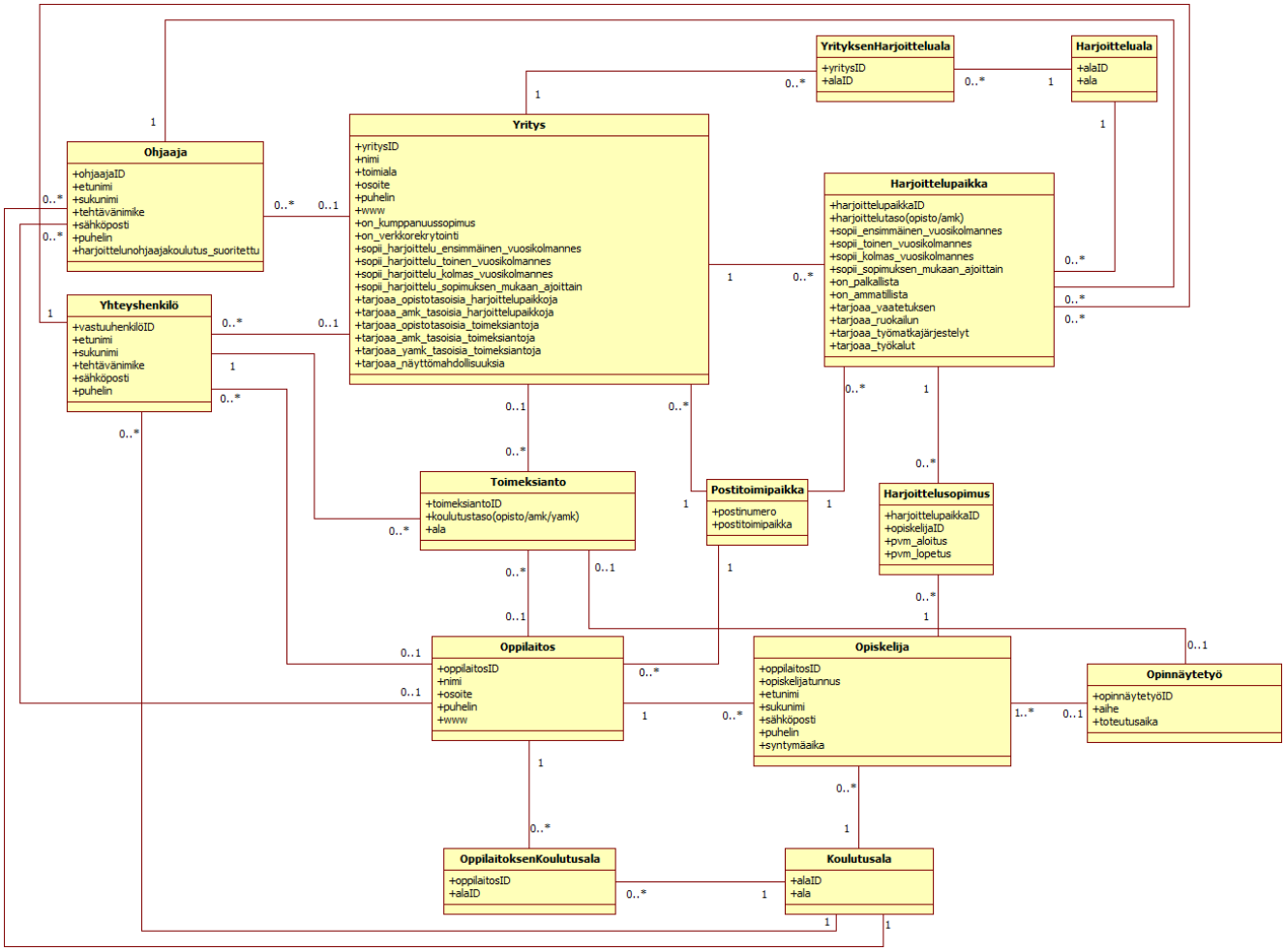
- Kolmas normaalimuoto: Oppilaitos- ja Yritys-käsitteissä on nyt kentät postinumero ja postitoimipaikka. Tehdään uusi käsite Postitoimipaikka, jolla on nämä kaksi kenttää. Postinumero voidaan laittaa perusavaimeksi. Oletetaan yhteyden olevan yksi-moneen.

5.5 Jatkokehitysideoita ja valmis käsitemalli

Tässä luvussa listataan vielä joitakin toivottavia käsitemalliin liittyviä parannuksia. Näitä ei lisätä enää käsitemalliin ensimmäisen kohdan pientä muutosta lukuun ottamatta.

T1. Yritys tai oppilaitos voi toimia usealla paikkakunnalla. Tämä aiheuttaisi taas uuden moni-moneen-yhteyden, joka pitäisi purkaa. Asiaa yksinkertaistetaan tässä siten, että päätoimipaikkaa voidaan kohdella kuin se olisi ainoa toimipaikka. Lisätään kuitenkin olemassa olevaan käsitemalliin harjoittelupaikalle yhteys Postitoimipaikka-tauluun.
T2. Harjoittelupaikkoja voi tarjota oppilaitoskin. Tämän voisi ratkaista joko lisäämällä tarvittavia kenttiä oppilaitokselle tai yhdistämällä taulut Yritys ja Oppilaitos, esimerkiksi nimellä Organisaatio. Tämänhetkisessä ratkaisussa joudutaan olettamaan, että oppilaitos ei tarjoa harjoittelupaikkoja tai jos tarjoaa, niin se tiedetään oppilaitoksen sisällä.
T3. Käsitemalliin jäi vielä kaksi enum-tyyppistä kenttää, Toimeksianto.koulutustaso(opisto/amk/yamk) ja Harjoittelupaikka.harjoittelutaso(opisto/amk). Näitä varten voisi muodostaa yhden tai kaksi uutta taulua, niin ratkaisu olisi eheämpi. Toisaalta se tekisi käsitemallista ja kyselyistä taas monimutkaisempia.

Nyt käsitemalliin on tullut huomattavan paljon muutoksia. Alla on tämän työn lopullinen versio käsitemallista.



KUVIO 13. Käsitemallin lopullinen versio

6 POHDINTA

Työn tavoitteena oli suunnitella relaatiotietokanta Ydinosaajat-hankkeelle, jota hallinnoi Centria-ammattikorkeakoulu. Toimeksiantona ollut tietokantatehtävä oli tarkoitettu harjoittelunohjaushankkeeseen, jossa mukana olevista ammattiopistoista, ammattikorkeakouluista ja organisaatioista saadaan monipuolinen yhteystieto-, toimeksianto- ja työharjoittelupaikkarekisteri. Tämän pohjalta voitaisiin myöhemmin luoda tietokantaa käyttävä sovellus esimerkiksi web-ympäristöön.

Ennen tätä työtä minulla oli perustiedot relaatiotietokannoista ja SQL-kielestä, mutta yhtäkään kunnollista tietokantaa en ollut ikinä suunnitellut. Ensimmäisenä tietokantasuunnitteluprojektina tämä työ tarjosi uutta oppimista, mikä on aina palkitsevaa. Opin myös paremmin ymmärtämään kuinka pitkä ja yksityiskohtainen prosessi relaatiotietokannan suunnittelu on.

Eniten jäi harmittamaan, että laajuuden takia monia asioita piti pudottaa pois. Alun perin oli ajatuksena tehdä kokonainen tietokantademototeutus ja päästä testaamaan sitä, mutta laajuus ja aikaraja tulivat vastaan. Osittain syynä tähän oli teoriaosuus, joka vei yllättävän paljon aikaa työstä. Työhön päätyi mukaan joitakin hieman varsinaista aihetta sivuavia asioita. Tästä seurasi teoriaosuuden hieman oppikirjamainen olemus.

Vaikka käytännön osuus ei edennyt alkuperäisten suunnitelmien mukaan, tuli käsitekaaviosta silti melko yksityiskohtainen. Käsitemallin lisäksi syntyi yksinkertainen käyttötapauskaavio oikeasta ympäristöstä ja pari mietittyä kehitysehdotustakin. Toivon tämän suunnitelman helpottavan varsinaista toteutusta tulevaisuudessa, kun projektia aletaan jatkokehittää.

LÄHTEET

Hakkarainen, A. 2011. Oracle-tietokannan tehokas hallinta. Asenna, ylläpidä, varmista ja suojaa Oracle-tietokanta. Helsinki: A Bonnier Group Company.

Hovi, A., Huotari, J., Lahdenmäki, T. 2005. Tietokantojen suunnittelu & indeksointi. 2. laitos, 1. painos. Jyväskylä: Docendo.

Huotari, J. & Hovi, A. 2009. Tietokantojen suunnittelu. Www-dokumentti. Saatavissa: http://docplayer.fi/9164783-Tietokantojen-suunnittelu.html. Viitattu: 31.8.2017.

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

Lightstone, S., Teorey, T., Nadeau, T. 2007. Physical Database Design. The Database Professional's Guide to Exploiting Indexes, Views, Storage, and More. Saatavissa: https://ebookcentral-proquest-com.ezproxy.centria.fi/lib/cop-ebooks/reader.action?docID=313887. Viitattu: 31.8.2017.

MySQL. 2017. Date and Time Types. Www-dokumentti. Saatavissa: https://dev.mysql.com/doc/refman/5.7/en/date-and-time-types.html. Viitattu: 31.8.2017.

MySQL. 2017. Fixed-Point Types (Exact Values). Saatavissa: https://dev.mysql.com/doc/refman/5.7/en/fixed-point-types.html. Viitattu: 31.8.2017.

MySQL. 2017. Floating-Point Types (Approximate Value). Saatavissa: https://dev.mysql.com/doc/refman/5.7/en/floating-point-types.html. Viitattu: 31.8.2017.

MySQL. 2017. The ENUM Type. Www-dokumentti. Saatavissa: https://dev.mysql.com/doc/refman/5.7/en/enum.html. Viitattu: 31.8.2017.

MySQL. 2017. Numeric Types. Www-dokumentti. Saatavissa: https://dev.mysql.com/doc/refman/5.7/en/numeric-types.html. Viitattu: 31.8.2017.

MySQL. 2017. String Types. Www-dokumentti. Saatavissa: https://dev.mysql.com/doc/refman/5.7/en/string-types.html. Viitattu: 31.8.2017.

Neo4j. 2017. What is Graph Database? Www-dokumentti. Saatavissa: https://neo4j.com/developer/graph-database/. Viitattu: 31.8.2017.

Nixon, R. 2014. Learning PHP, MySQL & JavaScript. With jQuery, CSS & HTML5. 4. painos. Farnham: O'Reilly.

Ollikainen, V. Osa 1: Tietokanta. Saatavissa: http://slideplayer.fi/slide/2847087/. Viitattu: 31.8.2017.

PhpMyAdmin. 2017. Www-dokumentti. Saatavissa: https://www.phpmyadmin.net/. Viitattu: 31.8.2017.

Ranne, P. 2014. Käsiteanalyysi ja relaatiomalli. Www-dokumentti. Saatavissa: http://docplayer.fi/16065658-2-kasiteanalyysi-ja-relaatiomalli.html. Viitattu: 31.8.2017.

Redis. An introduction to Redis data types and abstractions. Saatavissa: https://redis.io/topics/data-types-intro. Viitattu: 31.8.2017.

Teorey, T., Lightstone, S., Nadeau, T. & Jagadish, H.V. 2011. Database Modeling and Design. Logical Design. 5th edition. Burlington: Elsevier Science. Saatavissa: https://ebookcentral.proquest.com/lib/cop-ebooks/reader.action?docID=667713&ppg=1. Viitattu: 31.8.2017.

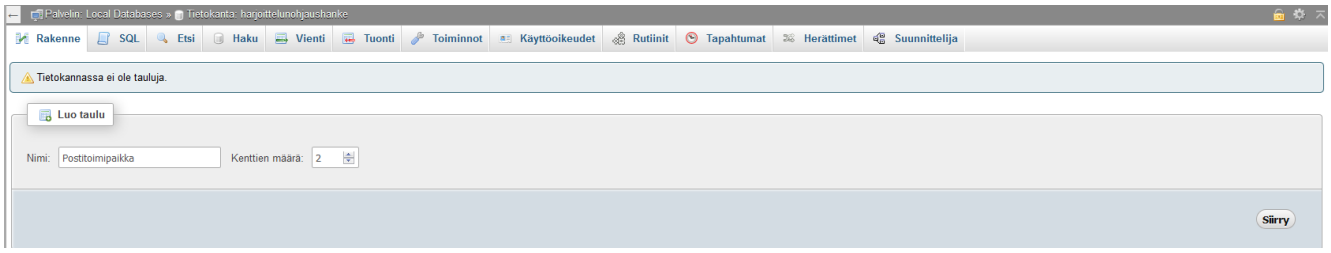
Tezer, O.S. 2014. SQLite vs MySQL vs PostgreSQL: A Comparison of Relational Database Management Systems. Saatavissa: https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems. Viitattu: 31.8.2017.

Vaish, G. 2013. NoSQL Starter. Birmingham: Packt Publishing. Saatavissa: https://ebookcentral.proquest.com/lib/cop-ebooks/reader.action?docID=1142875. Viitattu: 31.8.2017.

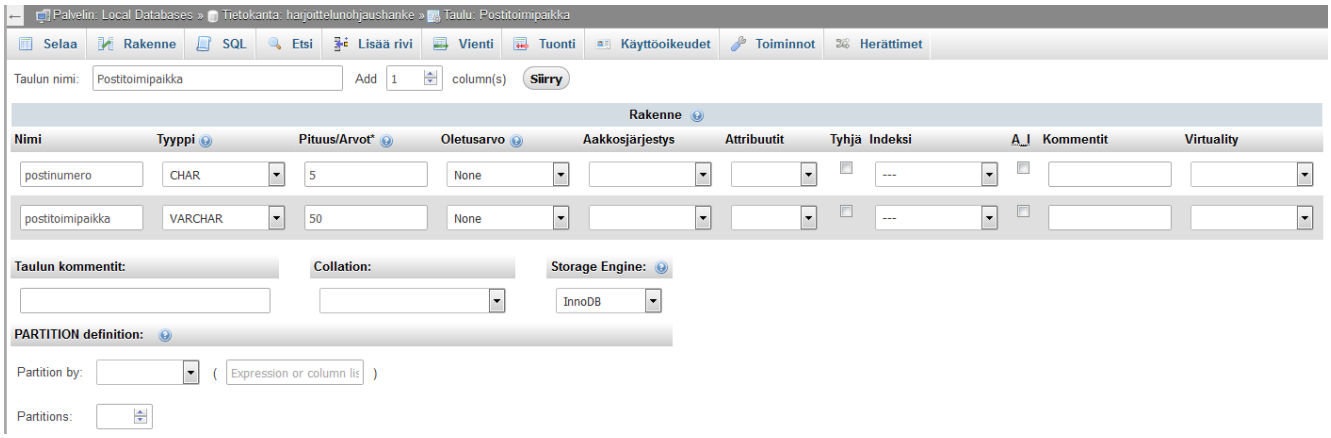
Taulujen, indeksien ja viiteavaimien luominen phpMyAdmin-ohjelmalla

Tutustutaan phpMyAdmin-ohjelman käyttöön. Tähän on valittu näytettäväksi taulut Postitoimipaikka ja Yritys käsitemallin viimeisestä versiosta. Käydään näiden taulujen luonti läpi kuvien kanssa.

Ennen taulujen lisäystä on tietenkin luotava itse tietokanta, jonne taulut tulevat. Annetaan nimeksi Harjoittelunohjaushanke. Annetaan myös aakkosjärjestys, jonka perusteella merkkijonosarakkeissa tiedot lajitellaan.

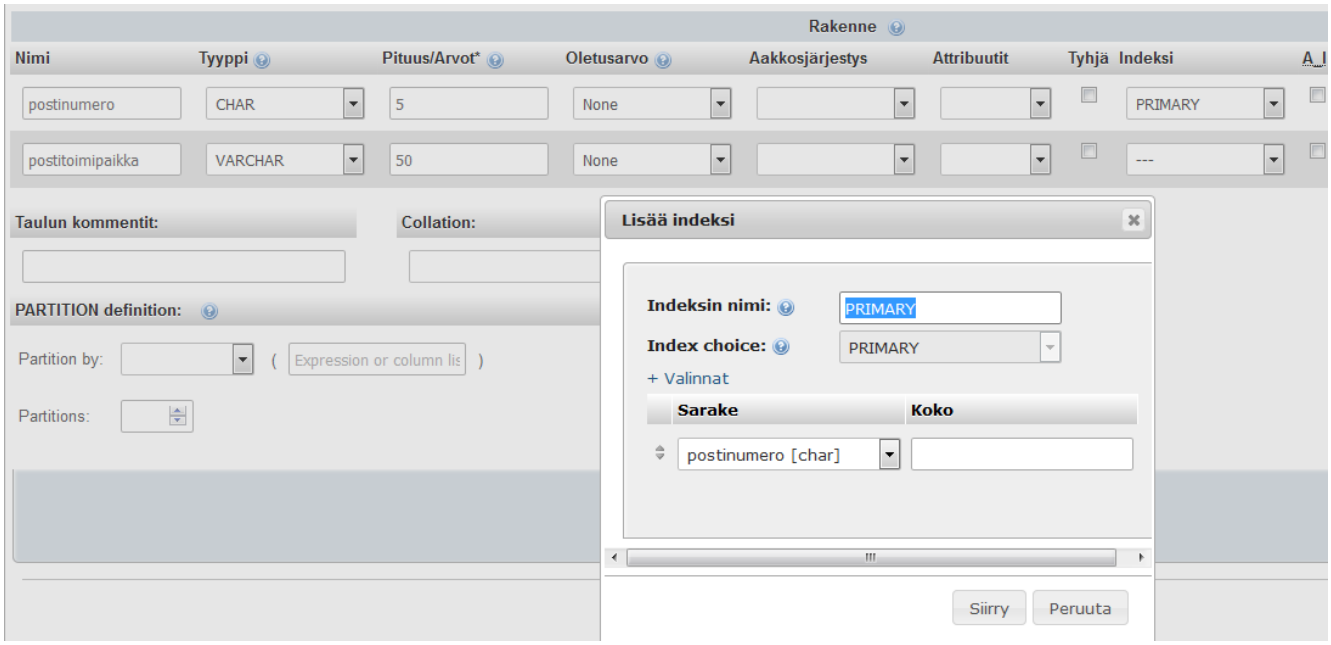


Uuden taulun luonti phpMyAdmin-sovelluksella.

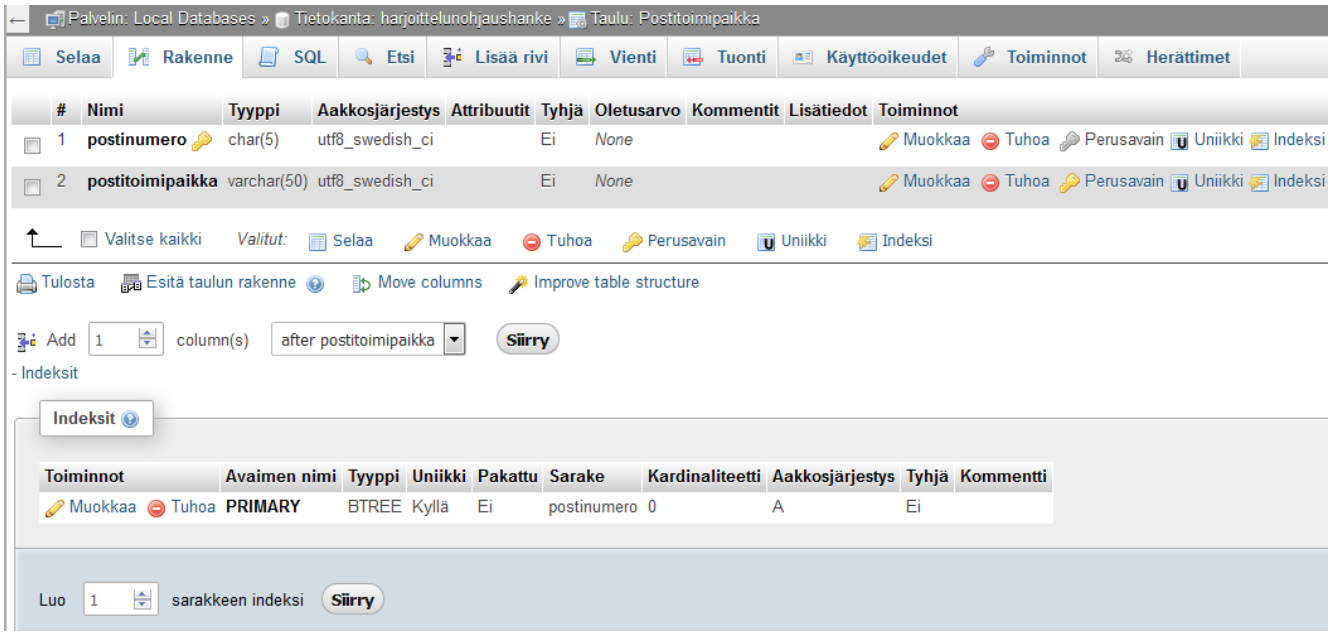


Uuden taulun luonti: sarakemääritykset

Uuden taulun luonti aloitetaan antamalla taulun nimi ja sarakkeiden määrä, tosin niitä pystyy muuttamaan myöhemminkin. Sitten päästään määrittämään taulun rakennetta yksityiskohtaisesti. Määritetään käsitemallin mukaiset kentät postinumero ja postitoimipaikka. Postinumerolle annetaan tietotyypiksi viisimerkkinen CHAR ja postitoimipaikalle korkeintaan 50 merkkiä pitkä VARCHAR. Lisätään vielä perusavaimelle postinumero perusavainindeksi, jolloin siitä tulee samalla perusavain.

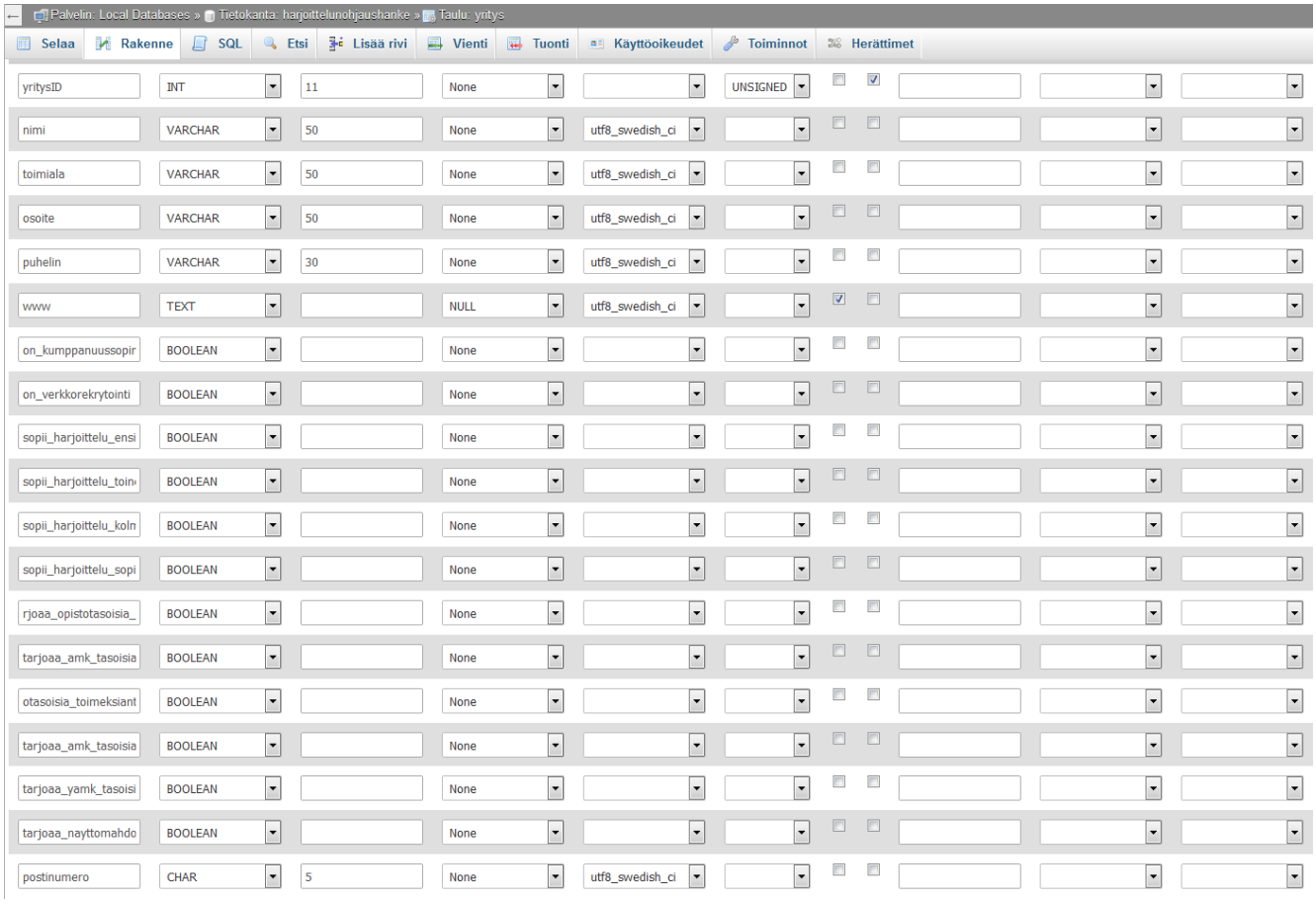


Indeksin ja perusavaimen luominen



Taulun rakenne ja indeksi käyttöliittymässä

Luodaan seuraavaksi hieman isompi taulu: Yritys. Lisätään sille taulun luonnin jälkeen viiteavain, joka viittaa Postitoimipaikka-taulun perusavaimeen postinumero.

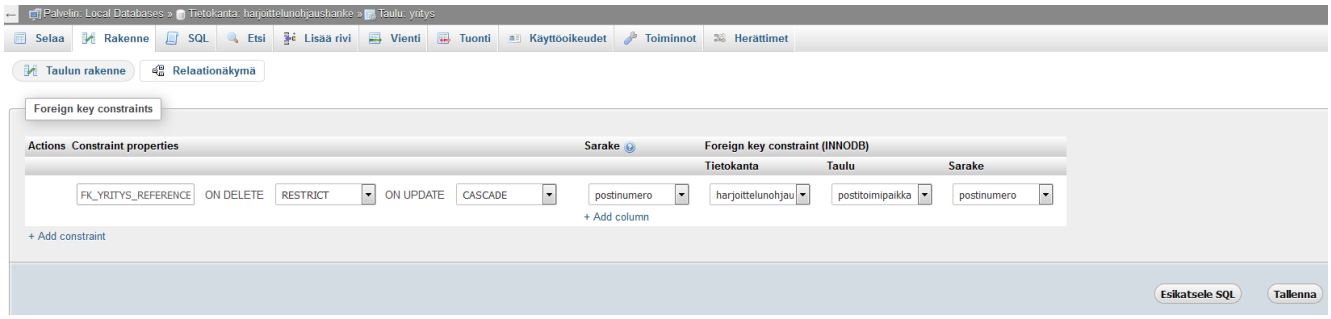


Yritys-taulun kentät ja niiden asetukset

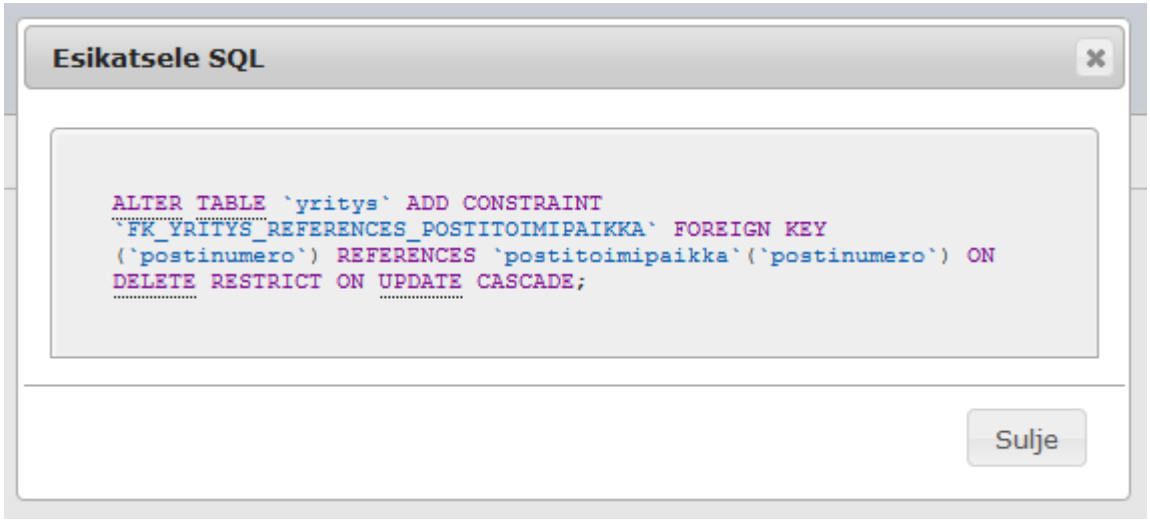
Viiteavain voidaan graafisen käyttöliittymän avulla luoda seuraavasti:

- Lisätään viittaavaan tauluun viiteavainkenttä, joka on samanlainen kuin viitattavan taulun perusavain.
- Lisätään tälle kentälle tavallinen indeksi.
- Määritetään käyttöliittymän relaationäkymän kautta viite-eheyssäännöt sekä viitattava perusavain.

Vaihtoehtoinen tapa eli SQL:n käyttö esitellään kuvassa 8. Kuvan koodi on saatu phpMyAdminin ”Esikatsele SQL” -toimintoa käyttämällä.



Viite-eheyden lisääminen graafisen käyttöliittymän avulla



Viiteavain ja viite-eheys SQL-koodin avulla.

Tästä huomataan, ettei phpMyAdmin-sovellus välttämättä ole nopeampi tapa hallita tietokantaa. Graafinen käyttöliittymä tekee asiat selkeämmäksi, mutta phpMyAdmin-sovelluksen isoimpana etuna on se, ettei komentoja tarvitse muistaa ulkoa.