HAAGA-HELIA
ammattikorkeakoulu

Tietokannan suunnittelu ja toteutus

Joni Winberg

Opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

2012

HAAGA-HELIA
ammattikorkeakoulu

Tiivistelmä

3. joulukuuta 2012

Tietojenkäsittelyn koulutusohjelma

Tekijä tai tekijät Joni Winberg	**Ryhmätunnus tai aloitusvuosi** Tv8Ti
Raportin nimi Tietokannan suunnittelu ja toteutus	**Sivu- ja liitesivumäärä** 48+ 3
Opettajat tai ohjaajat Niina Kinnunen	

Opinnäytetyön tavoitteena oli tutkia kuinka tietokannan toteutusprosessi etenee ideasta toteutukseen. Tutkimuksessa kerrotaan relaatiotiekannan historiasta, jaottelusta, suunnittelusta, toteutuksesta ja hallinnasta. Opinnäytetyö pyrittiin kirjoittamaan niin, että lukijalla ei tarvitse olla ennestään mitään ymmärrystä tietokannoista. Tutkimus on rajattu Windows ympäristössä toimiviin relaatiotietokantoihin. Opinnäytetyö ei myöskään ota kantaa eri relaatiotietokantojen lisensointitapoihin. Opinnäytetyön empirian tavoitteena oli luoda BaliBeachBungalows-verkkopalvelulle toimiva tietokanta. Työn rajauksien ulkopuolelle jäävät kaikki menetelmät, joiden avulla verkkopalvelu on luotu. Verkkopalvelun taustalla oli ystäväni tekemä huomio siitä, että Balilla olevia majoituskohteita ei löytynyt mistään Internetistä.

Produktin raportoinnissa kerrotaan kaikki vaiheet tietokannan suunnittelusta sen lopulliseen toteuttamiseen. Raportoinnissa käy ilmi myös, että relaatiotietokannaksi on valittu MySQL-tietokanta ja sen hallintaan on käytetty ohjelmaa nimeltä phpMyAdmin.

Työn tulokseksi saatiin toimiva relaatiotietokanta verkkopalvelumme tarpeisiin. Pohdinnassa esitellään myös muutoksia, joita tietokanta on opinnäytetyön kirjoituksen jälkeen kokenut, sekä esitellään ideoita joiden avulla työtä voisi jatkaa verkkopalvelumme näkökulmasta.

Asiasanat
Tietokannat, relaatiotietokannat, suunnittelu, toteutus, SQL

HAAGA-HELIA
University of Applied Sciences

Abstract

3 December 2012

Degree Programme in Information Technology

Authors Joni Winberg	**Group or year of entry** Tv8Ti
The title of thesis Database Design And Implementation	**Number of pages and appendices** 48+3
Supervisor(s) Niina Kinnunen	

The goal of this thesis was to study database processes from ideas to implementation. The study explains the history, division, planning, implementation and management of the relational database model. The thesis was intended to be written so that the reader does not need any previous knowledge of databases. The study is limited to relational database models in Windows OS environment. The study does not take a stance on the licensing methods of different relational database models.

The aim of the empirical part of the thesis was to create a working database for BaliBeachBungalows web service. The study does not focus on how the web service was created. The idea behind the web service was my friend's observation that accommodation services in Bali cannot be found on the Internet.

The report of the product explains all the phases from the planning to the implementation of the database. In the report it also comes out that the MySQL database was chosen as the relational database and it is managed by the program called phpMyAdmin. The final product is a working relational database model that meets the needs of our web service. The discussion part presents the changes that the database has gone through after the writing of the thesis and explains ideas for how the work could be continued from the point of view of our web service.

Key words
Databases, relational databases, design, implementation, SQL

Sisällys

Keskeiset käsitteet

Algoritmi

Matematiikassa ja tietotekniikassa käytetty. Tarkkaan määritelty joukko käskyjä, ongelman ratkaisuun. (SuomiSanakirja.fi. 2012)

Data ja informaatio

Datan ja informaation ero on selitettävissä seuraavanlaisesti, data on esitystyyppi, jolla ei ole merkitystä. Vasta kun dataa on käsitelty ja sille on annettu merkitys, syntyy siitä informaatiota. (Wikipedia. 2012)

Eheys

Tiedonhallinnassa, tieto on oikeellista ja yhteensopivaa ennalta määriteltyihin ehtoihin nähden. (Wikipedia. 2012)

Foreign key, viiteavain

Viiteavaimet ovat kentän ominaisuuksia, jotka mahdollistavat viittaamisen johonkin toisen taulun perusavaimeen. Eli mahdollistaa eri tauluissa olevien tietueiden liittämisen toisiinsa. (Koulutuskeskus Salpaus. 1999)

Liitos

Liitos mahdollistaa kahden taulun välisen yhteyden. Liitos tehdään pääavaimia ja viiteavaimia käyttäen. Liitos voidaan myös tehdä linkitystaulun avulla. (Oulun seudun ammattiopisto. 2012)

QBE

Graafinen kysely kieli, jossa käyttäjä visuaalisia taulukoita käyttäen syöttää komentoja, elementtejä tai ehtoja. (Wikipedia. 2012)

Relaatio

Osa joukkoteoriaa, tarkemmin erilaisuussuhde kahden asian välillä. (Hernandez 1997, 11-17) (Wikipedia. 2012)

Tietorakenne

Tapa tallentaa ja järjestää dataa tietokoneella, jotta dataa voitaisiin käyttää mahdollisimman tehokkaasti. (Wikipedia. 2012)

1 Johdanto

Lähtökohtana opinnäytetyön tekemiseen oli ajatus, kuinka opinnäytetyön tekemisestä voisi hyötyä rahallisesti. Keskustelin asiasta ystäväni kanssa, joka työskentelee digitaalisen mainonnan parissa. Hänen työhön kuuluu Internet -palvelujen luominen. Syntyi idea luoda yhdessä jokin web-palvelu, josta saisin tehtyä myös opinnäytetyön. Tulen myöhemmin tässä suunnitelmassa ja itse opinnäytetyössä käyttämään ystävästäni nimitystä Repa. Repa toimii projektin päällikkönä, sekä vastaa palvelimesta ja tekniikkaan liittyvistä asioista. Hän tulee myös kirjoittamaan suurimman osan verkkopalvelun lähdekoodista, lisäksi hän on ainoa jolla on kokemusta vastaavista projekteista. Projektiin liittyy myös kolmas osapuoli, joka tulee vastaamaan kuvaamisesta ja kuvien editoinnista ja graafisista linjauksista. Tässä työssä häntä kutsutaan nimellä Heebo. Verkkopalvelun aihe syntyi tarpeesta; Repa oli edellisenä talvella lomaillut Indonesian Balilla. Hän havaitsi, että Balilta ja sen lähisaaristoista löytyy bungaloweja ja muita majoituskohteita aivan toistensa vierestä. Kohteita oli kuitenkin vaikeaa, ellei jopa mahdotonta löytää netistä. Tavoitteena on siis vastata tähän tarpeeseen. Tarkemmin ajatuksena oli luoda verkkopalvelu, joka listaa mahdollisimman kattavasti majoituskohteet Balilta ja sen lähialueilta.

Tästä päästään opinnäytetyöni aiheeseen, joka on tietokannan suunnittelu ja sen toteutus. Luon siis produktin, jonka lopputuotteena saan verkkopalveluun vaadittavan tietokannan. Lähtökohtana projektiin oli pelkkä idea. Opinnäytetyön alkuosa koostuu tietoperustasta, jossa käyn läpi tietokantoihin liittyviä perusasioita yleisellä tasolla. Tietoperustan tarkoitus on toimia hyvänä perustana opinnäytetyön empiiristä osaa ajatellen. Viitekehyksessä käytän lähteenä Haaga-Helian kirjastosta löytyviä kirjoja, sekä Internetistä löytyviä artikkeleita.

Empiirisessä osassa käydään läpi omaan palveluumme tarvitsemamme tietokannan luomisen vaiheet, tietokannan määrittelystä sen toteuttamiseen. Lisäksi kerrotaan siitä, minkälaisia teknillisiä valmisteluja tietokannan toteuttaminen vaatii. Tietokantaa koskeva projektisuunnitelma, on syntynyt palavereissa, joita olemme pitäneet ennen matkalle lähtöä. Valitsemistamme tekniikoista ja menetelmistä lisää empiirisen osuuden alussa.

Opinnäytetyön tavoitteet voidaan karkeasti jakaa kahteen osaan; tietoperustan tavoitteet, sekä empiirisessä osassa syntyvän produktin tavoitteet. Tietoperustan tutkimuksessa päätavoite on kertoa, mitkä ovat tietokannan tekemisen vaiheet ja mitä tulisi huomioida hyvää tietokantaa tehdessä. Lisäksi tavoitteiksi voidaan lukea myös oman tietämyksen syventäminen erityyppisistä tietokannoista ja niiden hallinnoinnista. Lisäksi tutkitaan mielenkiinnolla, mitä eroja eri relaatiotietokannat pitävät sisällään. Empiirisen osuuden tavoite on toteuttaa tietokanta, tietoperustassa tutkituilla oikeaoppisilla menetelmillä. Empiirisen työn tavoitteet liittyvät myös puhtaasti palveluumme, jota olemme luomassa. Tavoitteena on tietysti saada palvelu toimintaan. Tämän vaatimuksena on toki se, että palvelun taustalogiikassa oleva tietokanta on valmis ja toimiva. Tietokantaa tulee olla myös helppo käyttää web-palvelun ylläpitoa ajatellen. Tarkemmin, ylläpitäjän tulee pystyä käyttämään tietokannan toimintoja ilman ongelmia niin, että se on loogi-nen palvelun ominaisuuksiin nähden.

Työn on rajattu niin, että opinnäytetyön tietoperustassa sekä empiirisessä osassa käsitellään ainoastaan Windows-ympäristössä toimivia tietokantasovelluksia. Opinnäytetyössäni keskityn ainoastaan relaatiotietokantoihin, enkä ota kantaa erilaisiin tietokantatuotteita koskeviin lisensointitapoihin. En myöskään puutu siihen, kuinka tietokannan tietoja saadaan tulostettua web-sivulla. En siis mene PHP-, Asp- tai minkään muunkaan ohjelmointikielen saloihin. Vaikka verkkopalvelumme on työn rajauksien ulkopuolella, tullaan palvelua esittelemään pohdinnassa, jotta lukija saa selkeän kokonaiskuvan luomani tietokannan tarpeesta.

2 Tietokanta

ATK-sanakirja (1990) määrittelee tietokannan siten, että se on *"kokoelma tiettyä kohdetta kuvaavia tietoja, joita yksi tai useampi tietojärjestelmä käyttää"* Tietokanta koostuu yhdestä tai useammasta taulusta (table). Tauluun määritellään sarakkeita (column), tai toiselta nimeltä kenttiä. Sarakkeet kuvaavat taulun sisältämien tietueiden (record) ominaisuuksia. Tietueet ovat määrämuotoista dataa, jotka kuvaavat kohdetta tai useampien kohteiden välisiä suhteita. Kuviossa 1 on esitelty tietokannan taulun ja sen sisältämien tietueiden koostumusta MS Officen sisältämästä Access nimisestä tietokantaohjelmasta. Tietokantaan voidaan tallentaa erityyppistä dataa kuten tekstiä, ääntä, kuvia tai videota. (Haasio 2005, 11-12.)

Kuvio 1. Kuvitteellinen oppilas tietokanta, taulu nimeltä Oppilaat

Nykyisin pääsääntöisesti käytössä oleva tietokantatyyppi on nimeltään relaatiotietokanta. Ennen relaatiotietokantamallin yleistymistä käytettiin kuitenkin hierarkista tietokantamallia ja verkkotietokantamallia.Seuraavissa kappaleissa tulen esittelemään pintapuolisesti miten nykyiseen relaatiotietokantamalliin on päädytty (Hernandez 1997, 4.)

2.1 Historiaa

Edgar F. Coddia pidetään nykyisen relaatiotietokantamallin isänä. Hän kehitti teorian nykyiseen tietokantamalliin vuonna 1970. Uuden mallin oli tarkoitus parantaa 1960-luvulla yleisessä käytössä olleiden ensimmäisten hierarkisten ja verkkotietokantamallien puutteita. (Hernandez 1997, 11-12)

Hierarkista tietokantamallia kuvaa hyvin ylösalaisin käännetty sukupuu. Ylimpänä siinä on (isä) taulu, joka voi jakaantua moneen alihaaraan (lapsi) tauluun. Lapsia voi olla monta, mutta jokaisella voi olla yksi isä. Tämä aiheuttaa ongelmia vähänkin monimut-

kaisemmissa taulukkorakenteissa, joissa syntyy ns. monesta-moneen yhteys. Tällaisesta voisi olla esimerkkinä lääkäri- ja potilastaulut. Lääkärillä on luonnollisesti monta potilasta eli yhden suhde moneen, Kun potilas käy useammalla lääkärillä samaan aikaan, syntyy yhteys monesta-moneen. Tällainen ei ole mahdollista hierarkisessa mallissa ja tämän takia tilanne synnyttää ylimääräistä dataa koko tietorakennetta ajatellen. Mallissa tietokannan rakenne tuli osata hyvin ja sen sisältämän datan haku alkoi aina juuresta. Malli soveltui hyvin suurimpaan osaan 1970 luvulla käytössä olevista nauhatallennusjärjestelmistä. (Hernandez 1997, 4-8)

Hierarkisen tietokantamallin korvasi verkkotietokantamalli. Rakenne pysyi muuten samana, mutta tietorakenteessa oli useita käännettyjä puita, joilla saattoi olla yhteisiä alahaaroja. Tämä voidaan nähdä niin, että ns. isä-lapsitaulurakenne muuttui omistaja/jäsenrakenteeksi, jossa yksi jäsen eli taulu, saattoi kuulua enemmän kuin yhden omistajataulun alle. Verkkotietokantamallissa etuna on, että tietoa haetaan ns. sopivien joukkorakenteiden läpi, jolloin tiedon haun voi aloittaa mistä tahansa taulusta ja kulkea eteen- tai taaksepäin. Tämä mahdollistaa nopeamman tiedon haun. Verkkotietokantamalli mahdollistaa myös monimutkaisempien kyselyiden muodostamisen kuin hierarkisessa tietokantamallissa. Tämän mallin haittapuolena oli myös, että rakenne piti tuntea hyvin osatakseen suunnistaa taulujen välillä. Lisäksi rakenteen ongelmaksi muodostui se, että muutosten teko vaati usein myös sen kanssa toimivien asiaksohjelmien muuttamista. Vaikka parannus hierarkisesta tietokantamallista oli ilmeinen, tiedostettiin, että isoja tietovarastoja säilyttäville on olemassa parempi tapa. Tietorakenteiden kasvaessa tietoa hakevat kysymykset monimutkaistuivat , mikä asetti tietokantamallille lisää vaatimuksia. Tähän relaatiotietokantamalli toi vastauksen. (Hernandez 1997, 8-11)

1960-luvun loppupuolella matematiikan tohtori Edgar F. Codd pohti miten käsitellä suuria määriä tietoa. Ennen kaikkea hän mietti miten ratkaista sen hetkisten tietokantamallien olemassa olevia ongelmia. Suurimpia näistä olivat jo aiemmin mainitut asiat, kuten ylimääräinen data, suuri riippuvuus tietokannan fyysisestä toteutuksesta ja tietojen heikko eheystaso. Codd valitsi luonnollisesti matemaattisen lähestymistavan. Malli perustui kahdesta matematiikan haarasta, joukkoteoriasta ja ensimmäisen kertaluvun predikatiikasta. Tämä tekee mallista kestävän ja luotettavan. Relaatiotietokanta nimi johdetaan termistä relaatio, joka on osa joukkoteoriaa. Yleisen harhakäsityksen mukaan

se tulee tietokannan taulujen mahdollisista suhteista. Relaatiotietokantamallissa taulujen fyysinen järjestys on yhdentekevä. Jokainen tietue on tunnistettavissa kentällä, jonka arvo sisältää muista tietueista poikkeavan arvon. Käyttäjän ei siis tarvitse tietää, missä tieto fyysisesti sijaitsee löytääkseen sen. Relaatiotietokantoja hallitaan rakenteellisella kyselykielellä SQL. Kielen avulla voidaan luoda, muokata, ylläpitää tai hakea tietoja tietokannasta. SQL-kielestä kerrotaan tarkemmin myöhemmin. Relaatiotietokantamallin ehdottomia etuja ovat ainakin taulu ja tietuetasolla monitasoinen eheys. Tämä tarkoittaa, että tietokantarakenteeseen voidaan tehdä muutoksia ilman, että pitäisi muuttaa tietokantasovelluksia. Lisäksi etuna on tiedonhaun helppous, sillä tieto voidaan hakea mistä vaan tietämättä koko tietokannan rakennetta. (Hernandez 1997, 11-17)

2.2 Tietokannan jaottelu

Nykyisin tietokannat voidaan jakaa karkeasti kahteen tyyppiin, jotka ovat käyttötietokannat ja analyyttiset tietokannat. Käyttötietokantoihin kerätään dynaamista dataa, mikä tarkoittaa, että data saattaa muuttua jatkuvasti. Hyviä esimerkkejä käyttötietokannoista ovat muun muassa inventaariotietokannat, potilastietokannat ja tilaustenhallintatietokannat. Analyyttiset tietokannat sen sijaan säilyttävät niin sanottua staattista dataa, mikä tarkoittaa, että tietoja ei juurikaan muokata niiden syöttämisen jälkeen. Analyyttiset tietokannat koskevat usein tiettyä ajankohtaa. Yritykset ja organisaatiot tekevät usein päätöksiään tällaisten tietokantojen tuottaman informaation pohjalta. Hyviä esimerkkejä analyyttisistä tietokannoista ovat mm. yrityksen tulosta seuraavat tietokannat, kyselytutkimustietokannat ja historiallisia tapahtumia tallentavat tietokannat. (Hernandez 1997, 3-4)

Tietokannat voidaan edelleen jakaa sisällön mukaan kahteen luokkaan, joita ovat lähdetietokannat ja korviketietokannat. Lähdetietokannat sisältävät tiedon joko tekstien tai tilastoiden muodossa. Tieto voi olla myös muussa muodossa, kuten ääninä, kuvina tai videoina. Lähdetietokannoille on yhteistä, että ne sisältävät suoraan käyttäjän tarvitseman tiedon. Korviketietokanta taas sisältää ainoastaan viitteen missä kyseinen tieto sijaitsee. Hyvä esimerkki korviketietokannasta on kirjastojen kokoelmaluettelot. Haaga-Helian kirjastossa käytössä olevaan tietokantaan on tallennettu tiedot sen kokoelmiin sisältyvistä tallenteista. Tallennettu tieto sisältää tiedot siitä, mitä tietokantoja koskevia

teoksia löytyy, kuka ne on kirjoittanut ja missä toimipisteessä ne fyysisesti sijaitsevat. Tieto ei kuitenkaan sisällä informaatiota itse kirjojen sisällöstä. (Haasio 2005, 13-14.)

Tietokannat voidaan jaotella myös jakelutavan mukaan. Tämä tarkoittaa jakoa sen mukaan missä tietokanta fyysisesti sijaitsee. Tietokannat voidaan tallentaa esimerkiksi cd-levylle, muistitikulle tai käyttäjän kovalevylle. Yhä yleisempää on kuitenkin Online-tietokantojen käyttö. Ne sijaitsevat palvelimella, johon otetaan yhteys käyttäjän koneelta verkkoa käyttäen. Online-tietokannat vaativat yleensä käyttäjätunnuksen ja salasanan. Palvelimella olevien tietokantojen suurin hyöty on, että niihin pääsee käsiksi miltä tahansa tietokoneelta, jolla on Internet-yhteys käytössä. Tämä helpottaa tietojen pitämistä ajan tasalla. Käyttötarkoituksesta riippuen Online-tietokannat voivat olla joko maksullisia tai maksuttomia. Internetistä löytyy myös lukemattomia vapaasti saatavilla olevia tietokantoja, kuten esimerkiksi Tilastokeskuksen tietokannat. (Wikispaces 2012)

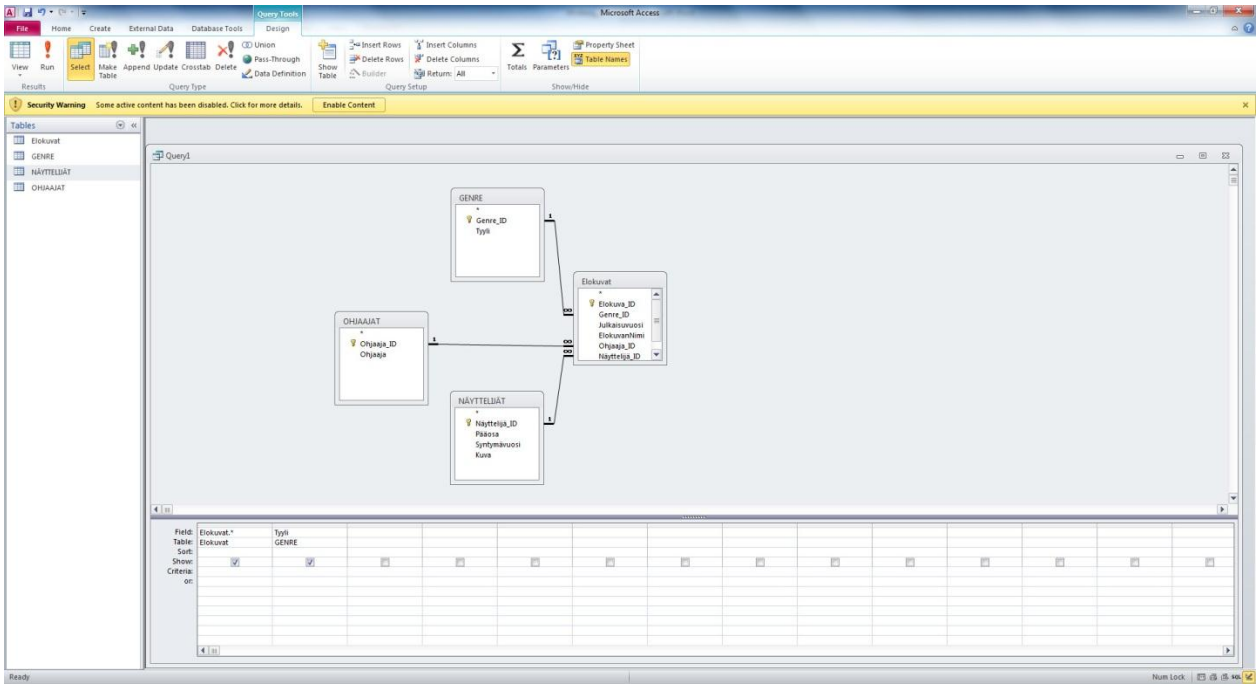
2.3 Relaatiotietokantojen esittelyä

Tässä luvussa vertaillaan pintapuolisesti yleisesti markkinoilla käytössä olevia relaatiotietokantoja Microsoftin Access ja Oraclen MySQL. Tietokantatuotteita on hyvin eri hintaisia, mikä tulee ottaa huomioon sopivaa tietokantatoimittajaa mietittäessä. En kuitenkaan esittelyssäni ota kantaa ohjelmien hintoihin.

MySQL on tunnetuin ja suosituin avoimen lähdekoodin tietokanta. Tietokanta on paljon käytetty WWW-sovelluksissa ja sitä käyttävät esimerkiksi Facebook, Google, Adobe, ja Twitter. Tietokanta on saatavilla yli 20:lle käyttöjärjestelmälle. Se on käytännössä todettu vauhdikkaaksi, mutta se ei kuitenkaan sisällä kaikkia ominaisuuksia, joita jotkut vaativat järjestelmät tarvitsevat. MySQL-tietokanta mahdollistaa automaattisen varmuuskopioinnin datasta, jota ei ole vielä varmuuskopioitu. Varmuuskopioinnin ansiosta on mahdollista palauttaa tietokannan data mihin tahansa aikaisempaan hetkeen. (Oracle Inc. 2012) (PHP.net. 2012)

Microsoft Accessissa on valmiudet hyvin eritasoisien tietokantojen luontiin yksinkertaisista kortistoista kokonaisiin yritysten käyttämiin tietojärjestelmiin. Access tarjoaa erittäin laajan työkalupakin. Se sisältää paljon ohjattuja toimintoja joita kutsutaan vel-

hoiksi (wizard). Ne helpottavat esimerkiksi taulukoiden, kyselyiden, raporttien ja lomakkeiden luomista. Access sisältää myös hyvät näkymävalikoimat (QBE, Query by example), jotka helpottavat huomattavasti taulukkorakenteiden ja taulujen välisten yhteyksien ymmärtämistä ja kyselyiden luomista (Kuvio 2). Mielestäni Accessin vahvuuksiin lukeutuu sen selkeys ja helppokäyttöisyys.(Groh ym. 2007, 9.)



Kuvio 2. Kuvakaappaus erään tietokannan QBE-näkymästä Microsoft Accesissa

3 Tietokannan tekemisen vaiheet

3.1 Suunnittelu

Tietokannan suunnitteluun on käytettävä runsaasti aikaa, eikä sitä tule sivuuttaa tai jättää toissijaiseksi. Hyvä suunnitelma on yhdenmukaisuuden, paikkaansapitävyyden ja eheyden kannalta hyvinkin ratkaiseva tekijä. Pahimmassa tapauksessa huonosti suunniteltu tietokanta antaa käyttäjän hauille virheellistä informaatiota, mikä saattaa olla haitallista koko yrityksen liiketoimintaa ajatellen. Useat tietokantasovellukset tarjoavat velhotyökaluja tietokannan luomisprosessiin. Suunnittelijan tulee ymmärtää, että velhot auttavat vasta tietokannan fyysisessä luomisessa, eikä huolellista suunnitteluvaihetta voi jättää pois velho-työkalujakaan käytettäessä. Hyvin suunniteltu tietokantarakenne säästää aikaa pitkällä aikavälillä. Huonosti suunniteltua rakennetta sen sijaan joutuu korjaileman jatkuvasti. Tietokannan suunnitteluprosessi tulee suorittaa aina alusta loppuun saakka laiminlyömättä määrittelyyn tai toteutukseen liittyviä suunnitteluvaiheita. (Hernandez 1997, 21-22.)

Suunnitteluprosessin tulee täyttää tietynlaisia tavoitteita, jotta tietokanta saa kestävän rakenteen. Taulurakenteiden tulee olla tehokkaasti muodostettuja ja jokaisen taulun kuvata vain yhtä asiaa. Taulujen on hyvä sisältää selkeitä kenttiä ja mahdollisimman vähän ylimääräistä dataa. Eheystasojen pitää olla kunnossa. Kun eheystasot ovat kerralla luotu kuntoon, ovat datan arvot aina paikkansapitäviä. Datan on tuotettava liiketoiminnalle merkityksellistä ja voimassa olevaa informaatiota. Lisäksi tietokannan tulee sopeutua yrityksen vaatimuksiin, ja tarpeiden muuttuessa sen on oltava helposti muokattavissa. (Hernandez 1997, 26-27.)

Hyvällä suunnittelulla on monia etuja. Etukäteen mietittyä rakennetta on helppo muokata niin, että kenttiin tai tauluihin tehdyt muutokset eivät vaikuta vahingollisesti tietokannan muuhun rakenteeseen. Päivittäessä kenttien arvoja, muutokset eivät vaikuta taulun muihin kenttiin. Hyvässä rakenteessa muutokset tarvitsee tehdä vain yhteen

paikkaan. Lisäksi hyvin suunnitellun tietokantarakenteen päälle on helppo rakentaa sovelluksia, jotka käyttävät tietokantaa. (Hernandez 1997, 27-28.)

3.2 Määrittely

Uuden tietokannan luominen on hyvä aloittaa määrittelemällä sille päämäärä, mikä tehdään laatimalla tietokannalle *tehtäväselostus* ja *tehtävän tavoitteet*. Tehtäväselostus aloitetaan haastattelemalla yrityksen johtoa. Haastattelun tavoitteena on saada selville tietokannan tehtävä ytimekkäästi, ja yleensä yrityksen johto on paras vastaamaan tähän. Mikäli projektiin liittyy ainoastaan yksi henkilö, eli sinä itse, haastattelet itseäsi. Haastattelua tehtäessä on hyvä pyrkiä avoimiin kysymyksiin, jotta saataisiin mahdollisimman paljon tietoa. Tehtäväselostuksen tarkoitus on antaa tietokannan suunnittelijalle selkeä keskipiste ja kertoa, mihin tarkoitukseen tietokanta on tulossa. Kun tietokannalle on määritelty tehtäväselostuksessa tarkoitus, voidaan varmistaa, että sille aletaan luoda rakennetta, joka tukee sitä. Määrittelyvaiheessa varmistetaan myös, että tietokantaan on mahdollista kerätä oikeanlaisia tietoja. (Hernandez 1997, 61-62.) (Hernandez 1997, 81-87.)

Seuraavaksi on mietittävä tehtävän tavoitteita, jolla tarkoitetaan yleisiä toimintoja, selvityksiä tai tehtäviä, joista tulee pystyä suoriutumaan tietokannan sisältämän datan avulla. Tehtävätavoitteiden määrittelyn tarkoitus on ohjata tietokannan kehityksen yleistä suuntaa, sen on myös tarkoitus auttaa eri rakenteiden hahmottamisessa. Tehtävän tavoitteiden määrittely on helppo prosessi, joka saadaan vastaamalla kysymykseen ”Minkälaisia yleisiä tehtäviä tietokannan tietojen avulla tulee pystyä suorittamaan?” Kysymykset olisi hyvä esittää ainakin tietokannan tuleville käyttäjille. Tässä kohtaa tulee huomioida, että eri työtehtävissä toimivilla henkilöillä voi olla hyvinkin erilaisia käyttötarkoituksia tietokannalle, johtuen eriävistä työtehtävistä. On siis hyvä heti alusta alkaen ottaa huomioon eri käyttäjien tai käyttäjäryhmien erilaiset näkemykset tietokannan käyttötarkoituksesta. Näin on siis saatu tietokannalle luotua päämääriä. (Hernandez 1997, 61-62.) (Hernandez 1997, 89-95.) (Connolly & Begg 2002, 111-112.)

Päämäärän määrittely pitää sisällään myös mahdollisen jo olemassa olevan tietokannan analysoinnin. Yrityksellä voi olla esimerkiksi papereihin perustuva tietokanta, jonka he haluavat sähköistää. Analyysiin liittyy datan nykytilan arviointi, jossa arvioidaan datan

käsittelyä ja käyttöä. Lisäksi huomioidaan nykyistä tietokantaa käyttävät ohjelmistot, jos sellaisia on. Analysoinnin toisessa vaiheessa otetaan mukaan tietokannan käyttäjät ja jälleen yrityksen johto. Heidän kanssaan selvitetään kuinka käyttäjät ovat vuorovaikutuksessa nykyisen tietokannan kanssa, sekä millaista informaatiota tietokannan tulee tuottaa johdolle. (Hernandez 1997, 62-63.)

3.3 Toteutus

Tietokannan toteutuksessa otetaan huomioon suunnitteluprosessissa ilmoitetut suunnittelutavoitteet, sekä määritellyt tietovaatimukset. Näiden perusteella luodaan tarvittavat taulut, jonka jälkeen tauluihin liitetään sopivat kentät. Kun taulut on luotu, tarkistetaan, että jokainen taulu kuvaa vain yhtä asiaa, eivätkä ne sisällä moninkertaisia kenttiä. Kenttiä läpikäydessä tulee myös tarkastaa, että ne eivät sisällä moniosaisia arvoja. Jokaisen kentän tulee siis sisältää vain yksi arvo. Lisäksi tarkastetaan, että jokainen kenttä edustaa taululle sopivaa ominaisuutta. Kun taulut ja kentät ovat paikoillaan, aloitetaan kenttämääritelmät. Ensiksi luodaan pääavain, jonka avulla jokainen taulun yksittäinen tietue on tunnistettavissa. Tämän jälkeen määritellään jokaisen kentän tyyppi siihen tulevan datan perusteella. Esimerkiksi tarkasteltaessa kenttää johon kerätään työntekijöiden puhelinnumeroita, kentästä tehdään numeerinen. Sille määritetään myös maksimipituus, joka on suuntanumero mukaan lukien 13. Jälleen on hyvä haastatella käyttäjiä ja johtoa, jotta ymmärretään mitkä kentistä ovat erityisen tärkeitä. Tärkeiltä kentiltä voidaan kieltää esimerkiksi arvo ”NULL” eli tyhjäarvo. (Hernandez 1997, 63-64.)

Kun taulut ja kentät ovat kunnossa, voidaan aloittaa taulujen yhteyksien luominen. Jälleen kerran tulee haastatella johtoa sekä käyttäjiä. Useimmat yrityksen työntekijät tuntevat käyttämänsä datan väliset yhteydet varsin hyvin. Ulkopuolinen ei voi tietää miten yritys käyttää omaa dataansa. Saadaan erittäin hyödyllistä tietoa yrityksen käyttämästä datasta, sekä datan mahdollisista yhteyksistä. Kun yhteydet ovat tiedossa, luodaan taulujen välille loogisia liitoksia. Liitosten luominen tapahtuu pääavaimia sekä viiteavaimia käyttäen. Yhteyksille voidaan myös määritellä tyyppi, sekä osallistumisaste. Joskus nämä ovat ilmeisiä datan luonteen takia, mutta joskus ne perustuvat yrityksen liikesääntöihin. Yhteyksien luominen on erityisen tärkeää tietokannan eheyden ja sen sisältämän ylimääräisen datan kannalta. (Hernandez 1997, 64-65.)

Käytännössä yhteydet muodostetaan pohtimalla mikä taulun ominaisuuksista on toistuva arvo. Esimerkiksi kuvitellaan koulun ylläpitämää oppilastietokantaa. Tietokanta sisältää taulun *oppilaat*, oppilaalla on ominaisuus koulutusohjelma. Koulutusohjelmavaihtoehdot ovat kaikki ne, jotka kuuluvat koulun tarjontaan. Voidaan olettaa, että toistoja tulee monen oppilaan osallistuessa samaan koulutusohjelmaan. Ilman taulujen välisiä yhteyksiä tämä tarkoittaisi sitä, että jokaisen oppilaan kohdalla tulisi erikseen kirjoittaa oppilaan koulutusohjelman arvoksi se ohjelma, johon oppilas osallistuu. Ensimmäinen ongelma tällaisessa mallissa olisi ylimääräinen data, koska tietokantaan olisi merkitty satoja tai tuhansia kertoja koulutusohjelmaksi sama arvo. Toinen ongelma olisi tietokannan eheys, sillä mitä enemmän samaa arvoa toistetaan, sitä suuremmaksi todennäköisyys kirjoitusvirheille kasvaa. Aina ei ole edes kirjoitusvirheestä kysymys, vaan esimerkiksi jonkin kaupungin nimi voidaan kirjoittaa useammassa kuin yhdessä kirjoitusasussa. Nämä kaksi ongelmaa voidaan välttää käyttämällä taulujen välisiä yhteyksiä. Yhteys mahdollistaa, että tietty ominaisuus vaatii ainoastaan yhden syöttökerran tietokantaan. Käytännössä oppilaan koulutusohjelman arvo haetaan toisesta taulusta nimeltä *koulutusohjelmat* . Tätä menetelmää kutsutaan nimellä tietokannan normalisointi. Seuraavissa kuvioissa havainnollistetaan nämä kaksi edellä mainittua tapausta kuvitteellista oppilastietokantaa käyttäen (Kuvio 4). (Groh ym. 2007, 90-96.)

OPPILAAT

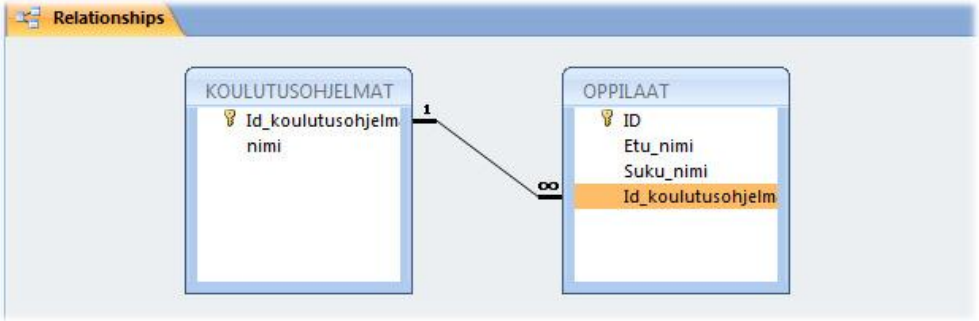
ID	Etu_nimi	Suku_nimi	Koulutusohj	Add New Field
1	Joni	Winberg	It-tradenomi	
2	Katja	Tanskanen	Tradenomi	
3	Pekka	Wallenius	It-tradenom	
(New)				

Kuvio 4. Kuvakaappaus kuvitteellisesta oppilas tietokannasta

Kuviossa 4 tietokanta on tehty ilman yhteyksiä, eli ilman erillistä KOULUTUSOHJELMAT-taulua. Kuviosta voidaan havaita, että OPPILAAT-taulussa on syntynyt toistoa pelkästään kolmen tietueen lisäyksen jälkeen (samat koulutusohjelmat kaksi kertaa). Kuviossa 4 Pekan koulutusohjelmaan on tahallaan jätetty kirjoitusvirhe havainnollis-

taakseen eheyden merkitystä. Tietokanta tulee normalisoida oikeinluotuja yhteyksiä käyttäen, jotta edellä mainituilta ongelmilta vältyttäisiin. (Groh ym. 2007, 98-100.)

Seuraavaksi havainnollistetaan tilanne normalisoitua tietokantaa käyttäen, eli luodaan kaksi taulua ja niiden välille yhteys (Kuvio 5). Seuraavassa esimerkissä on luotu yhteys taulujen KOULUTUSOHJELMAT ja OPPILAAT välillä. Yhteys on tyyppiä yhden suhde - moneen, koska yksi oppilas voi kuulua ainoastaan yhteen koulutusohjelmaan, mutta yhteen koulutusohjelmaan kuuluu monta opiskelijaa. Nyt kun oikeanlainen yhteys taulujen välille on saatu muodostettua, tietueet tauluissa näyttävät Kuvion 6 ja 7 mukaisilta. (Groh ym. 2007, 115-120.) (Groh ym. 2007, 98-100.)



Kuvio 5. Kuvakaappaus kuvitteellisen Oppilas tietokannan yhteydestä

OPPILAAT

ID	Etu_nimi	Suku_nimi	Id_koulutusohjelma	Add New Field
1	Joni	Winberg	1	
2	Katja	Tanskanen	2	
3	Pekka	Wallenius	1	
(New)				

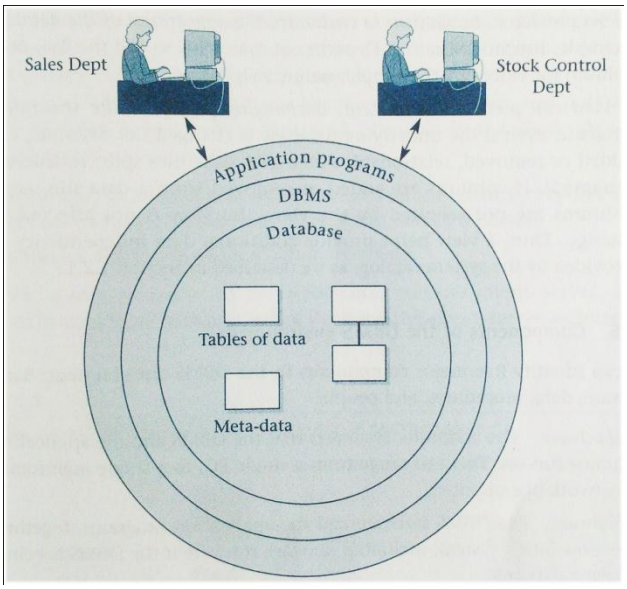
Kuvio 6. Kuvakaappaus kuvitteellisesta oppilas tietokannasta. Taulusta OPPILAAT

Nyt kenttä *Id_koulutusohjelma* toimii viiteavaimena taulussa OPPILAAT, joten koulutusohjelman nimi saadaan KOULUTUSOHJELMA-taulun pääavainta vastaavasta nimi-kentän arvosta. Tulee huomata, että pääavaimen on oltava samaa muotoa kuin OPPILAAT taulussa sijaitsevan viiteavaimen. (Groh ym. 2007, 98-100.)

Kuvio 7. Kuvakaappaus kuvitteellisesta oppilas tietokannasta. Taulusta KOULUTUSOHJELMAT

4 Tietokannan hallinta

Aloitan kappaleen kertomalla Database Management System:istä. DBMS-standardi tarkoittaa tietokannan hallintajärjestelmää, joka mahdollistaa tietokantaa käyttävän rajatut oikeudet tietokannan hallintaan. DBMS on ohjelmisto, jonka ansiosta interaktiivisuus käyttäjän, tietokoneohjelman (sovellus) ja tietokannan välillä on mahdollista. Toisin ilmaistuna se on yleinen rajapinta, joka mahdollistaa relaatiotietokantojen käsittelyn sovellusten ja tietokantojen välillä. Kuviossa 8 nähdään esimerkki yrityksen myynnin ja varaston valvonnasta tietokannan näkökulmasta. Käyttäjät saavat ohjelmillaan yhteyden tietokantaan DBMS rajapinnan kautta. Tästä pääsemme jouhevasti SQL-kieleen, jonka avulla käskyjä lähetetään tietokannan hallintajärjestelmään. (Connolly & Begg 2004, 8-9.)



Kuvio 8. DBMS rajapinta (Connolly & Begg 2004, s.8-9)

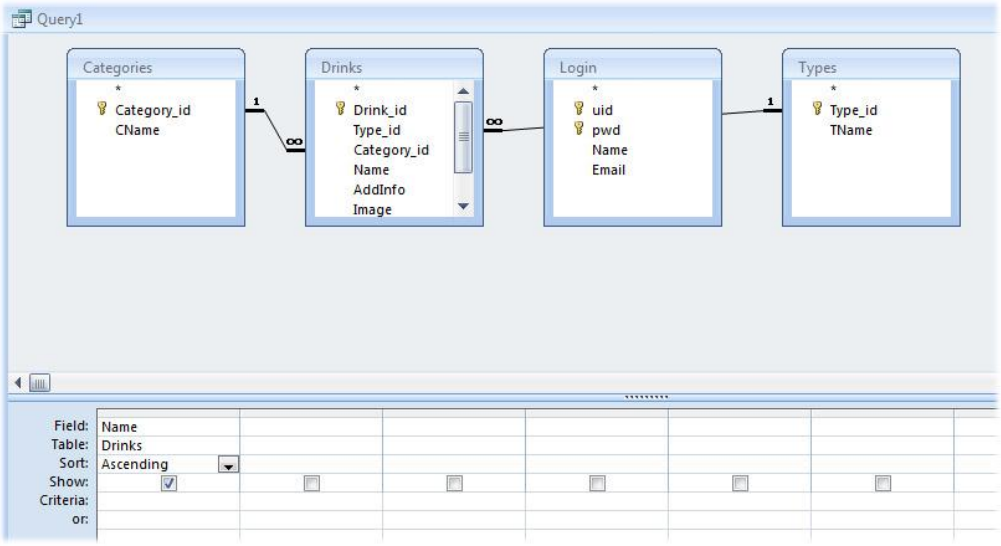
4.1 SQL -kieli

SQL- Structured Query Language, tarkoittaa rakenteellista kyselykieltä. SQL on standardikieli kaikkia relaatiotietokantaa koskevissa käskyissä. Sen avulla voidaan hakea, muokata, lisätä tai poistaa tietoja tietokannasta. SQL-komennot voidaan kohdistaa koko tietokantaan, tietokannan sisältämiin tauluihin tai taulujen sisältämiin tietueisiin. SQL-kielen kehitys alkoi IBM:n laboratoriossa San Josessa vuosina 1974-1977. Alkupe-

räisissä projekteissa se tunnettiin nimellä SEQUEL ja System-R. (Connolly & Begg 2002, 112-113.)

SQL-kielelle on ominaista, että sitä on helppo oppia. Käyttäjän tarvitsee tietää ainoastaan mitä tietoa hän haluaa käsitellä. Hänen ei siis tarvitse tietää metodia siihen, miten tieto on löydettävissä. Kieli on vapaa, jolla tarkoitetaan sitä, että SQL-lausetta ei tarvitse kirjoittaa ennalta määritettyyn paikkaan ruudulla. Käsky-lauseet perustuvat tavallisista englannin kielen sanoista, kuten SELECT, FROM, WHERE. Lisäksi kieltä voi käyttää kaikki tietokannan kanssa vuorovaikutuksessa olevat käyttäjät. (Connolly & Begg 2002, 111-112.)

Moni relaatiotietokantaohjelma sisältää jonkinlaisen SQL-sovelluksen. Usein käyttäjän annetaan kirjoittaa SQL-kyselyjä suoraan komentokehotteella, joka tarkoittaa yksinkertaisesti merkkijonon (käskyjen) syöttämistä tekstikenttään. Jotkut ohjelmat sisältävät myös graafisia käyttöliittymiä komentokehoitteen lisäksi. Tällaisia työkaluja kutsutaan nimellä QBE (Query By Example). Käytännössä se tarkoittaa, että ohjelma antaa käyttäjälle visuaalisen näkymän tauluista ja niiden sisältämistä kentistä (Kuvio 9). Kuviossa 9 käyttäjä on valinnut eräästä tietokannasta Accessin QBE-näkymään taulut ”Categories”, ”Drinks”, ”Login”, ”Types”. Seuraavaksi hän haluaa tehdä kyselyn, joka sisältää kaikki nimet (”Names”) taulusta ”Drinks”. Hän on määritellyt myös missä järjestyksessä ne tulee tulostaa (=Ascending). Käyttäjä voi valita tarvitsemiaan tietoja tauluista ilman, että hänen tarvitsee osata koota vastaavaa SQL-lausetta, jolloin ohjelma muodostaa käyttäjän valitsemista tiedoista kyselyyn tarvittavan SQL-lauseen (Kuvio 10). (Hernandez 1997, 15.) (Connolly & Begg 2004, 37-39.)



Kuvio 9. Accessin QBE näkymä

Kuvio 10. Accessin QBE-työkalun luoma SQL-lause, kuvion 9 tilanteesta

4.2 SQL-lauseen muodostus

Alkuun tulee huomauttaa, että SQL lauseen rakenteet ovat ns. case insensitive, eli lauseen toimivuuden kannalta ei ole väliä kirjoittaako käskyt isoilla vai pienillä kirjaimilla. On kuitenkin hyvä käytäntö kirjoittaa lauseen sisältämät käskyt kokonaan isoilla kirjaimilla, mikä selkeyttää lauseiden hahmottamista. Kuviosta 9 voidaan huomata, että myös Accessin luomassa lauseessa käskyt on kirjoitettu isoilla kirjaimilla. Tässä kohtaa pitää kuitenkin huomata, että SQL-lauserakenteen käskyt ovat täysin eri juttu kuin tietokannan sisältämä data. Esimerkiksi oletetaan, että käyttäjällä on taulu nimeltä Asiakkaat, johon on tallennettu asiakas nimeltä ”Meikäläinen”. Kun käyttäjä SQL-lauseen avulla etsii asiakasta ”Meikäläinen”, tulee hänen kirjoittaa nimi täsmälleen samassa kirjoitusasussa, kuten se on tietokantaan tallennettu tai muutoin hän saa tulokseksi nolla tietuetta. Sama koskee tietenkin myös tietokannan sisältämien taulujen nimiä. SQL-lause lopetetaan puolipisteellä. (Connolly & Begg 2004, 40-41.)

SQL-lause voi siis pitää sisällään monenlaisia toimintoja, kuten datan hakeminen, muokkaaminen, lisäys tai poisto. Sillä voidaan luoda tietokantaan myös kokonaisia tauluja. SQL-lauseen avulla voidaan olla vuorovaikutuksessa monella tapaa tietokannan kanssa. Seuraavaksi aion esittää yksinkertaisia esimerkkejä kuinka SQL-lauseilla manipuloidaan tietokannan sisältämää dataa eri tilanteissa. Esimerkeissä tulen käyttämään jo aiemmin esiintynyttä Alko.mdp tietokantaa ja SQL-lauseiden tulokset näytän kuvakaappauksina Accessin näkymästä. Lauseissa aion käydä läpi ainoastaan päätoiminnot:, jotka ovat mielestäni: tiedon hakeminen, lisäys muokkaus ja poisto, sekä uuden taulun luominen. (Groh ym. 2007, 471-472.)

Tietokannasta halutaan **hakea** kaikkien juomien nimet joiden alkoholipitoisuus on yli 4,7 %. -> SELECT Name FROM Drinks WHERE "Vol" > '4,7%' ;

SELECT Name kertoo, että Name kentän tieto näytetään tuloksessa. *FROM Drinks* kertoo, että tieto haetaan Drinks nimisestä taulusta. *WHERE "Vol" > '4,7%'* kertoo, että haluan ainoastaan ne tietueet, jotka täyttävät ehdon, eli joiden alkoholipitoisuus on yli 4,7 %. Kuviosta 11 nähdään tulos, jonka SQL-lause tulostaa.

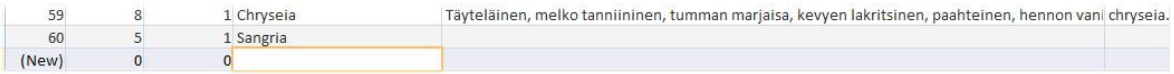

Query1

Name
Yellow Cat
El Tiempo Rosado
Melnik 139
Furia Shiraz
Frizzantino Amabile
Chryseia
Fizz Dry Cider
Hartwall Upcider Perry
Kopparbergs Strawberry Cide
Carlsberg Elephant Beer IV B
Koff Portteri IV B
Olvi Strong IV B
Saku Sorts IV B
Karahvivalkoviini
El Tiempo Tinto
Café
Magic Ice Red
Baila Valkoinen Salmiakkilikö
Peñascal Rosado
Saint-Paulin Rosé
Carlsberg Beer IV A
Sandels III
Zhujiang Beer IV A

Kuvio 11. Kaikki alkoholijuomat Drinks-taulusta, jotka ovat yli 4,7 %

Tietokantaan halutaan **Lisätä** drinkki nimeltä Sangria, juoman tyypiksi arvioidaan numero viisi, sekä kategoriaksi yksi. -> INSERT INTO Drinks("Name", "Type", "Category") VALUES ("Sangria", "5", "1") *;*

INSERT INTO Drinks("Name", "Type", "Category") kertoo, että halutaan lisätä Name, Type ja Category tauluun Drinks. *VALUES ("Sangria", "5", "1")* kertoo, mitä arvoja halutaan tallentaa tietueeseen. Kuviosta 12 nähdään, että SQL-lause lisäsi Sangria nimisen juoman Drinks-tauluun.



59	8	1	Chryseia	Täyteläinen, melko tanniininen, tumman marjaisa, kevyen lakritsinen, paahteinen, hennon vani	chryseia.
60	5	1	Sangria		
(New)	0	0			

Kuvio 12. Sangrian lisäys

Myöhemmin huomataan, että juoman alkoholipitoisuus unohtui, joten **muokataan** sille arvo 12 % -> UPDATE Drinks SET Vol = ”12,00%” ;

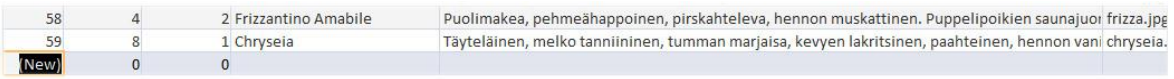
UPDATE Drinks kertoo, että halutaan muokata taulun Drinks tietuetta. *SET Vol = ”12,00%”* kertoo, että halutaan muuttaa kentän Vol arvoksi 12 %. Kuviosta 13 nähdään, että SQL-lause päivitti Sangrian alkoholipitoisuudeksi 12 %.

1	Chryseia	Täyteläinen, melko tanniininen, tumman marjaisa, kevyen lakritsinen, paahteinen, hennon vani	chryseia.jpg	13,00%
1	Sangria			12,00%
0				0,00%

Kuvio 13. Sangrian alkoholipitoisuus päivitetty

Lopuksi halutaan **poistaa** koko juoma tietokannasta. -> DELET FROM Drinks WHERE Name = ”Sangria” ;

DELET FROM Drinks kertoo, että halutaan poistaa jotakin Drinks-taulusta. *WHERE Name = ”Sangria”* kertoo, että halutaan poistaa tietue, jonka nimessä esiintyy Sangria. Kuviosta 14 nähdään, että SQL-lause poisti tietueen, jonka nimi oli Sangria.

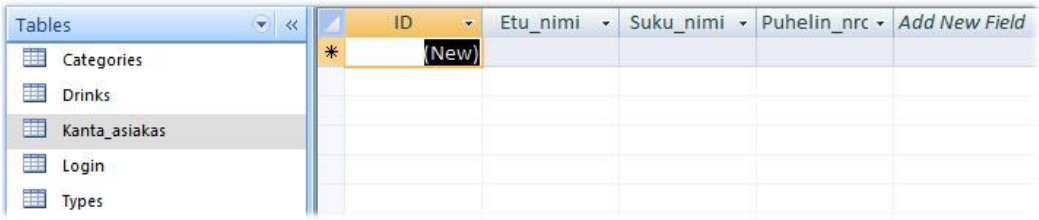


58	4	2	Frizzantino Amabile	Puolimakea, pehmeähappoinen, pirskahteleva, hennon muskattinen. Puppelipoikien saunajuor	frizza.jpg
59	8	1	Chryseia	Täyteläinen, melko tanniininen, tumman marjaisa, kevyen lakritsinen, paahteinen, hennon vani	chryseia.
(New)	0	0			

Kuvio 14. Sangrian poisto

Halutaan **lisätä** tietokantaan taulu, joka sisältää kanta-asiakas tietoja. Taulun tulee sisältää kentät Etu_nimi, Suku_nimi ja Puhelin_nro. -> CREATE TABLE Kanta_asiakas (Etu_nimi varchar(255), Suku_nimi varchar(255), Puhelin_nro varchar(13));

CREATE TABLE Kanta_asiakas (Etu_nimi varchar(255), Suku_nimi varchar(255), Puhelin_nro varchar(13) kertoo, että halutaan luoda taulu nimeltä Kanta_asiakas, joka sisältää kentät Etu_nimi, Suku_nimi, Puhelin_nro. Kenttien tyypiksi halutaan varchar ja pituuksiksi 255, 255, ja 13 merkkiä. Kuviosta 15 nähdään, että SQL-lause loi uuden taulun sisältäen halutut kentät. (Connolly & Begg 2004, 40-41.)



Kuvio 15. Uuden taulun lisäys

5 Case BaliBeachBungalows

Haluan jakaa produktin raportoinnin selkeästi kahteen osaan, suunnitteluun ja itse toteutukseen. Jako tarkoittaa, että suunnittelu osuus pitää sisällään vaiheet, joita on tehty kynä-paperi tekniikoilla. Toteutus sen sijaan alkaa teknisemmillä asioilla, kuten sopivan tietokantasovelluksen valinnalla, sekä tietokannan hallintaan liittyvien menetelmien suunnittelulla. Toteutuksen päämääränä kertoa kuinka tietokanta toteutetaan alusta loppuun, hyvän suunnitelman pohjalta.

5.1 Projektisuunnitelma

Projektisuunnitelmasta käy ilmi, että tarkoituksena on luoda tietokanta majoituskohteille. Projektisuunnitelma pitää myös sisällään menetelmiä, joita minun tulee huomioida tietokantaa suunnitellessa, sekä toteuttaessa. Menetelmiin luetaan ainakin seuraavat asiat, miettiä kaikki mahdolliset majoituskohteen ominaisuudet, valita edellä mainituista sopivat verkkopalveluamme ajatellen ja määritellä näiden ominaisuuksien tietotyypit.

Projektisuunnitelmasta käy ilmi, että eniten aikaa projektissa on varattu tiedon keräämiselle tietokantaan (Liite 1). Tiedon kerääminen tietokantaan menee kuitenkin opinnäytetyön aiheen ulkopuolelle, sillä tiedon kerääminen viittaa enemmän valmiin tietokannan käyttöön, kuin sen suunnitteluun ja toteutukseen.

5.2 Tietokannan suunnittelu

Suunnitteluosio tulee keskittymään tietoperustan läpikäymiin suunnitteluvaiheisiin. Vaiheet ovat siis tehtäväselostuksen ja tehtävätavoitteiden määrittely. Vasta näiden vaiheiden jälkeen olen valmis taulujen ja kenttien suunnitteluun. Kerron siis vaiheista, joita käyn läpi toteuttaessani tietokannan suunnittelua. Mielestäni vanha sanonta ”hyvin suunniteltu on puoliksi tehty” kuvaa erittäin hyvin tietokannan luomisprosessia. Hyvin käytetty aika suunnittelussa lyhentää ratkaisevasti toteutusvaihetta.

5.2.1 Tehtäväselostuksen määrittely

Minun tapauksessani tehtäväselostuksen määrittely tarkoitti, että yhdessä Repan ja Heebon kanssa pohdimme mihin tarkoitukseen tietokantaa ollaan luomassa. Pidimme palaverin, miettiäksemme vastausta kysymykseen mikä on BaliBeachBungalows ja mikä sen tarkoitus on? Vastaukseksi kysymykseen päädyimme seuraavanlaiseen ajatelmaan BaliBeachBungalows on verkkopalvelu, joka listaa mahdollisimman kattavasti majoituskohteita ja niiden ominaisuuksia Indonesiassa, tarkemmin Balilta ja sen lähiympäristöstä. Olimme siis saaneet käsityksen siitä, mitä olemme luomassa ja mikä on sen päämäärä. Nyt voitiin johtaa tehtäväselostus tietokannan näkökulmasta: BaliBeachBungalowin tietokannan tarkoitus on pitää sisällään kaikki olennainen tieto majoituskohteista. Näin olimme määritelleet tietokannan tehtäväselostuksen.

5.2.2 Tehtävän tavoitteiden määrittely

BaliBeachBungalows palvelun tietokannan tehtäviä pohtiessa tulimme seuraavanlaisiin päätelmiin. Meidän tulee ylläpitää mahdollisimman täydellisiä tietoja majoituskohteista, joita palvelumme pitää sisällään. Mieleen tuli myös muita mahdollisia tehtäviä, joita tietokannalta saattaisimme vaatia myöhemmin, kuten: Meidän täytyy pitää kirjaa asiakkaidemme kanssa sovituista sopimuksista. Asiakkailla tarkoitan majoituskohteiden omistajia tai niiden yhteyshenkilöitä. Esiin nousi myös palveluun rekisteröityneet käyttäjät. Meidän pitää kerätä rekisteröitymistietoja, palveluumme rekisteröityneistä käyttäjistä. Teimme myös rajauksia tietokannan tehtäviä koskien. On mahdollista, että palvelumme tulee pitämään sisällään varausjärjestelmän. Varausjärjestelmä vaatisi ehdottomasti tietokannalta lisätauluja, kuten *Reservations*. Jätämme kuitenkin aiheen myöhemmälle, sillä tavoitteena on ensin saada palvelumme listaamaan mahdollisimman täydellisesti sen sisältämät kohteet.

5.2.3 Taulujen ja kenttien valitseminen

Seuraavaksi määritellään taulut, joista tietokanta koostuu. Taulut muodostavat tietokannan tietorakenteen perustan. Jokaisen taulun tulee kuvata vain yhtä asiaa. Ensimmäinen ja tärkein taulu on nimeltään ACCOMODATIONS, tauluun tullaan keräämään majoituskohteiden nimet ja kaikki niille olennaiset ominaisuudet. Toinen taulu on ni-

meltään CONTRACTS, joka sisälää managerien kanssa tehdyt sopimukset. Kolmanneksi ajattelin taulua nimeltä USERS, joka tulee sisältämään palveluamme käyttävien käyttäjien rekisteröitymistietoja.

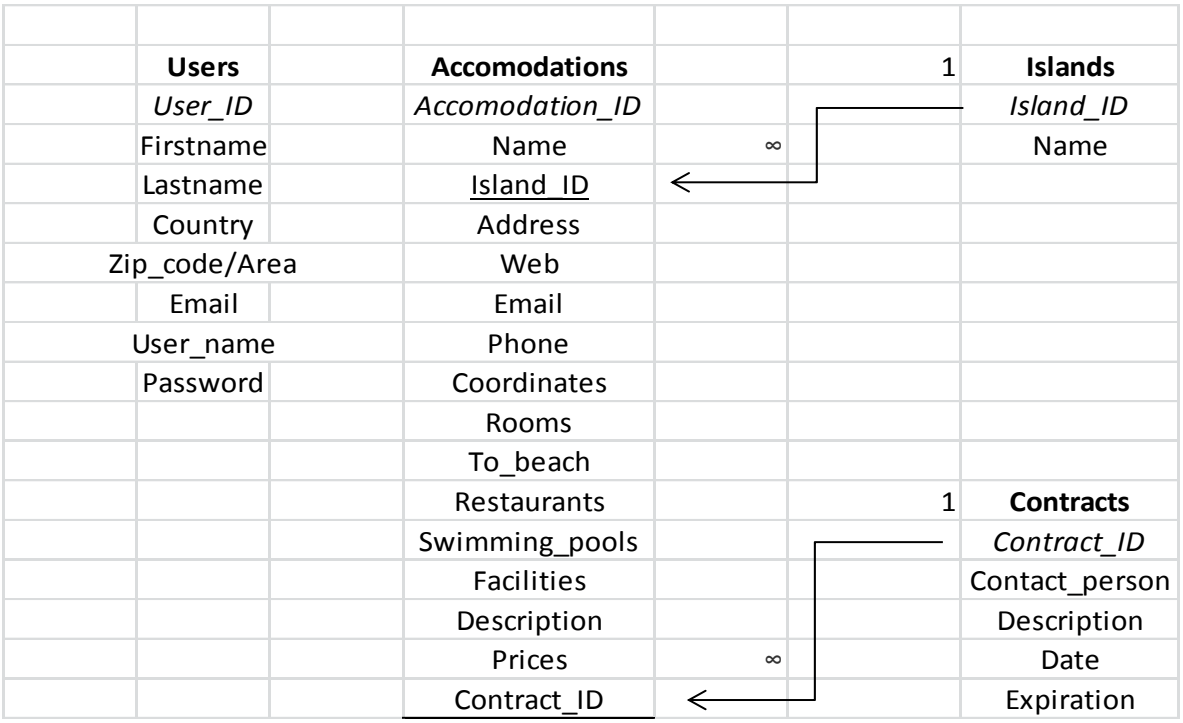
Nyt alkuperäisen suunnitelman mukaiseen tietokantaan on valittu kolme taulua, jotka ovat nimiltään ACCOMODATIONS, CONTRACTS ja USERS. Alleviivaan sanan alkuperäiseen, sillä minun tulee olla valmis muuttamaan suunnitelmaa, mikäli koen sen tarpeelliseksi. Olemme aiemmin miettineet, mitä tehtäviä tietokannan tulee pystyä suorittamaan ja sen mukaan valinneet jokaista tehtävää vastaamaan yhden taulun. Seuraavaksi määritellään kentät, joista taulut koostuvat. Mielestäni jokaisen kentän tulee kuvata ainoastaan yhtä asiaa ja saada vain yksi arvo. Minun tulee olla varma, että jokainen kenttä kuvaa juuri sen taulun ominaisuutta, johon se tullaan sijoittamaan. Seuraavaksi suunnittelin paperille jokaista taulua koskevien kenttien nimet.Kenttien tyypit ja muut määrittelyt tullaan tekemään vasta toteutus vaiheessa. Poikkeuksena pää (Primary key)- ja viiteavaimien (Foreign key) valitseminen. Pääavain tarkoittaa kenttää jolla jokainen yksittäinen tietue on tunnistettavissa. Pääavaimen tulee olla myös yksiselitteinen ja sisältää ainoastaan yksinkertaisia arvoja. Mielestäni hyvä käytäntö tähän tarkoitukseen on käyttää ns. ”id” kenttää. Kenttä on numerotyyppinen ja se asetetaan kasvamaan yhdellä edelliseen tietueeseen verrattuna, joka kerta kun uusi tietue lisätään tauluun. Tämän jälkeen valitaan viiteavaimet tauluihin, viiteavaimet mahdollistavat yhteyksiä taulujen välillä. Liitteinä paperille tehdyt hahmotelmat alkuperäisistä tauluista ja niihin liittyvistä kentistä. (Liite 2).

5.2.4 Tietokannan rakenne ja taulujen väliset yhteydet

Tähän vaiheeseen tultaessa minulla on valmiit suunnitelmat tauluista ja kentistä, joita tietokanta tulee pitämään sisällään. Rakennetta suunnitellessa, otan huomioon myös tietueet, joita tietokantaan tulee. Tämä on tärkeää, sillä jos kentän arvo näyttää toistuvan tietueissa, on fiksumpaa tehdä ominaisuudelle oma taulu välttääkseen ylimääräistä dataa sekä pitääkseen tietokannan eheyden mahdollisimman hyvänä, eli normalisoida tietokantaa. Haluan jälleen korostaa, että taulujen alkuperäiset rakenteet eivät välttämättä jää lopulliseen tietokantaan sellaisinaan. Käytännössä tarkoitan, että lopullista raken-

netta toteuttaessani olen valmis viemään taulun ominaisuuksia (kenttiä) omiin tauluihinsa, mikäli koen sen tarpeelliseksi.

Seuraavaksi aloitan lopullisen tietokannan rakenteen ja sen sisältämien yhteyksien hahmottamisen. Toimenpide on jälleen parempi tehdä paperille, tai esimerkiksi Excel-taulukoita hyväksi käyttäen. Kuviosta 16 nähdään, että alkuperäisen suunnitelman mukainen *Island* -kenttä on nyt tuotu omaksi tauluksi. Tämä juuri siksi, että jokainen majoituskohde sijoittuu jollekin tietylle saarelle. Koska saarivaihtoehtoja on vain noin viisi, toistuva arvo on erittäin ilmeinen. Kuviosta selviää myös, että *Contracts* -taulu on yhteydessä tauluun *Accomodations*. Tämä yhteys selittyy yksinkertaisesti sillä, että *Contracts* ei kuvaa majoituskohteen ominaisuutta (siksi omaan tauluun), mutta silti jokainen sopimus asiakkaan kanssa liittyy yksittäiseen majoituskohteeseen. Kuvio kertoo myös pääja viiteavainten määrittelyn. Pääavaimet kursivoituna ja viiteavaimet alleviivattuna. Taulujen välisten yhteyksien takia pää- ja viiteavainten valitseminen on mielestäni tärkeä tehdä mahdollisimman varhaisessa vaiheessa. Kuviosta selviää myös yhteystyypit; merkki 1 tarkoittaa yhtä ja merkki ∞ tarkoittaa monta. Ensimmäinen suunnittelemistani yhteyksistä on muotoa yhden suhde moneen. Tämän tyyppinen yhteys on määriteltävissä seuraavasti: *"Yhdestä moneen –yhteys määritellään yhteydeksi, jossa ensimmäisen taulun yksittäinen tietue voi liittyä yhteen tai useampaan toisen taulun tietueeseen, mutta toisen taulun yksittäinen tietue voi liittyä vain yhteen ensimmäisen taulun tietueeseen."* (Michael J Hernandez, 1997, 277) Esimerkiksi saarella A, voi sijaita monta majoituskohdetta, mutta majoituskohde X voi sijaita ainoastaan yhdellä saarella (Kuvio 16). Nopeasti ajateltuna toinen yhteys saattaa näyttää yhteydeltä, yhden suhde – yhteen. Vaihtoehto kuitenkin pois sulkisi mahdollisuuden, jossa sama omistaja tekisi kansamme useamman sopimuksen. Edellä mainittu tilanne synnytetään niinkin helposti, että omistaja omistaa useamman majoituskohteen. Myös toisen yhteyden tyypiksi valitaan siis yhden suhde – moneen. (Hernandez 1997, 277-278.)

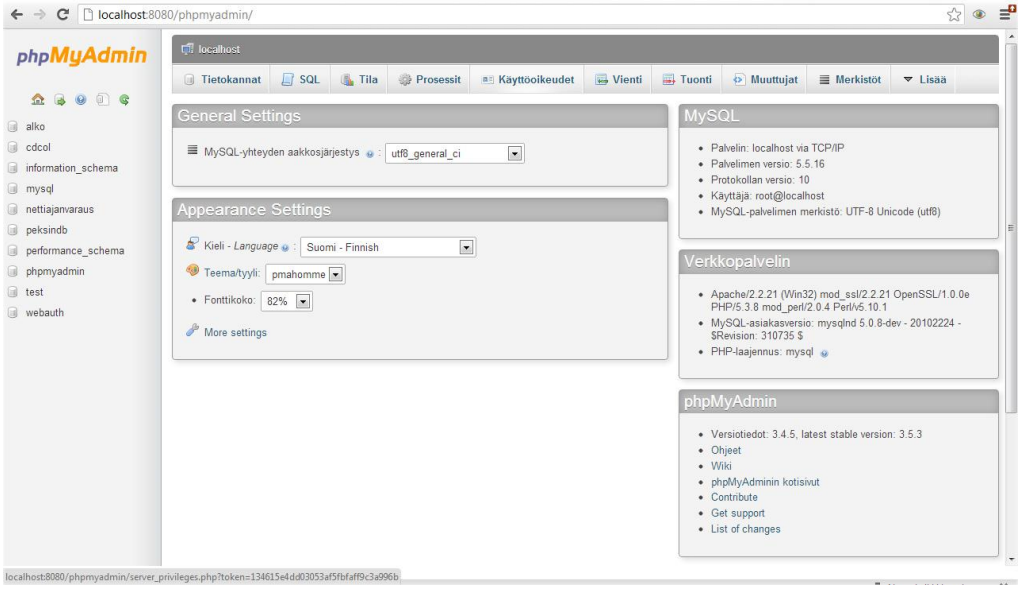


Kuvio 16. Excel-taulukko tietokannan rakenteen hahmottelemisesta

5.3 Toteutus

Relaatiotietokannan tyypiksi valitsemme MySQL -tietokannan. Se toimii hyvin PHP-ohjelmointikielen kanssa ja PHP tulee olemaan kieli jolla web-palvelumme tullaan kirjoittamaan. MySQL -ympäristö oli työryhmällemme myös ennestään tuttu. Emme ole vielä valinneet serveriä, jolla web-palvelumme tulee pyörimään. Ajatuksena olisi kuitenkin valita serveri, joka tarjoaa ilmaista PhpMyAdmin -nimistä hallintatyökalua ohjauspaneelissaan. Tämä on erittäin kätevä työkalu MySQL -tietokannan hallintaa varten. Se tarjoaa käyttäjäystävällisen ja visuaalisen käyttöliittymän kaikkiin tavallisimpiin tietokantaa koskeviin tehtäviin, kuten taulujen, kenttien tai käyttäjien hallintaan ja tietueiden lisäykseen. Lisäksi se tarjoaa myös mahdollisuudet luoda ja ajaa SQL –lauseita komentokehotteen kautta. (Ao media. 2012)

Työn ollessa tässä vaiheessa, serveripalvelua ei ollut vielä vuokrattuna. Minun tuli luoda niin sanottu localhost -serveri, eli niin sanottu paikallinen palvelin. Tähän on olemassa hyvä ilmaisen lähdekoodin web-palvelinpaketti nimeltä XAMPP. Ohjelma sisältää MySQL tietokannan ja PhpMyAdmin hallintatyökalun. Kun olin saanut palvelinympäristön valmiiksi, pääsin aloittamaan tietokannan luomisen. (Kuvio 17).



Kuvio 17. Yleisnäkymä PHPMyAdmin hallintapaneelista

5.3.1 SQL vs. PHPMyAdmin

Minulla on kaksi erilaista vaihtoehtoa tietokannan fyysiseen luomiseen. Ensimmäinen vaihtoehto on käyttää SQL-lauseita. MySQL -tietokannan kanssa SQL-lauseet olisivat muotoa: CREATE DATABASE BaliBeachBungalows; ja CREATE TABLE Accomodiations (Accomodiation_ID Int, Name Varchar, Island_ID Int); Syötettäessä nämä lauseet SQL-komentokehotteeseen syntyy tietokanta nimeltä *BaliBeachBungalows*. Tietokanta sisältää taulun nimeltä *Accomodations*. Taulu koostuu kentistä *Accomodiation_ID, Name* ja *Island_ID*. Kentät olisivat tyyppiä Integer, Varchar ja Integer. Varchar -tyyppi hyväksyy kaikki kirjaintyypit, ja Integer -tyyppi hyväksyy pelkkiä numeroita. Palaan opinnäytteeni myöhemmässä vaiheessa kenttätyyppien määrittelyyn syvemmin.

Toinen vaihtoehto on käyttää serverin sisältämää PHPMyAdmin:iä, visuaalista hallintapaneelia. XAMPPin avulla loin aikaisemmassa vaiheessa palvelin ympäristön koneelleni. Ohjelma mahdollistaa PHPMyAminin käytön, jota suosin SQL-komentokehotteen sijasta. Hallintapaneeli mahdollistaa tietokannan hallinnan ilman, että SQL-komentoja tarvitsisi muistaa ulkoa. Lisäksi kentille on huomattavasti helpompi määritellä tyypit hallintapaneelin tarjoamasta alasvetovalikosta. Visuaalisessa näkymässä taulujen välisten yhteyksien luominen on huomattavasti helpompaa taulujen ollessa fyysisesti näkyvillä.

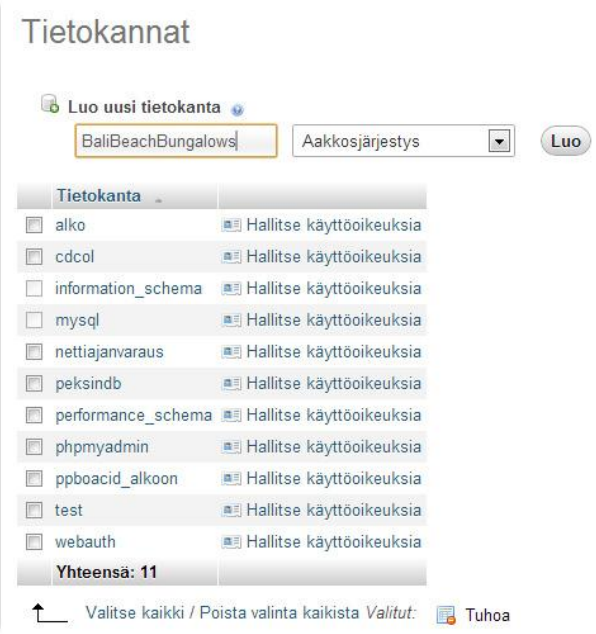
PHPMyAdminin:iä käytettäessä työn ja virheiden määrä vähenee mielestäni huomattavasti SQL -vaihtoehtoon verrattuna. Valitsen työskentelyalustakseni PHPMyAdminin.

5.3.2 Tietokannan sisältämien taulujen ja kenttien luominen

Seuraavissa kappaleissa tulen mahdollisimman yksityiskohtaisesti käymään läpi, kuinka PHPMyAdminiä käyttäen luodaan suunnittelemani tietokanta nimeltä BaliBeachBungalows. Seuraavat esimerkit ja kuvakaappaukset ovat PHPMyAdminin sisältä, vaikka sitä ei erikseen enää mainittaisi.

Luomisessa läpikäyn vaiheet tietokannan nimeäminen, testikäyttäjän luominen, taulujen luominen, kenttien luominen ja määrittely. Seuraavassa on kukin vaihe kuvattu tarkemmin.

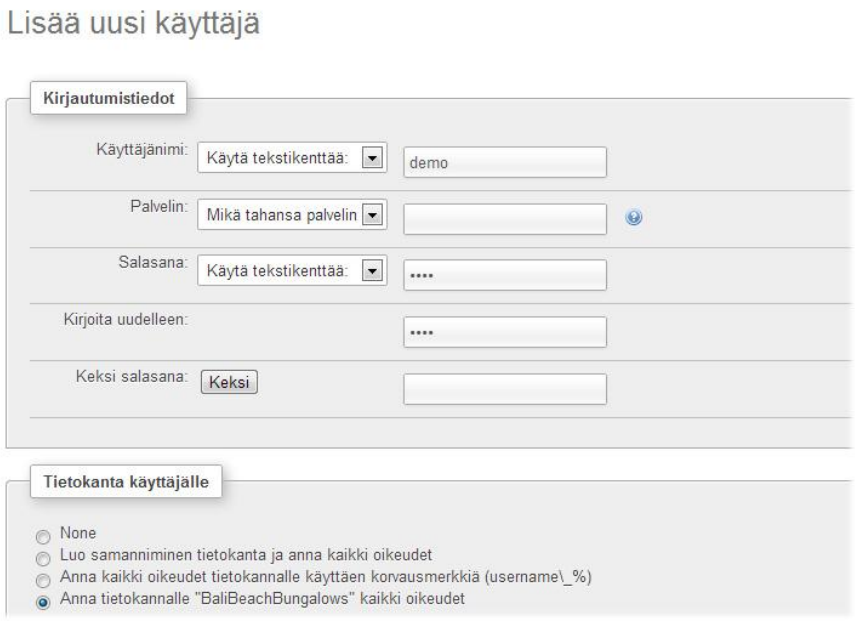
Ensimmäiseksi annan uudelle tietokannalle nimen BaliBeachBungalows (Kuvio 18).



Kuvio 18. Uuden tietokannan luominen

Seuraavaksi luon tietokannalle niin sanotun testikäyttäjän, jolla on kaikki oikeudet tietokannan hallintaan. Annan käyttäjänimeksi *demo* ja salasanaksi *omed* (Kuvio 19). Huomautan, että kun tietokanta on valmis ja viimeistelty, se siirretään ulkoiselle serverille, jossa sen on tarkoitus palvella web-sivustomme pohjana. Silloin luon uudet käyttä-

jät tietokannalle ja rajaan heidän käyttöoikeutensa tietokantaa koskien. Käyttäjämäärityksiä voidaan hallita ja tehdä missä vaiheessa tahansa.



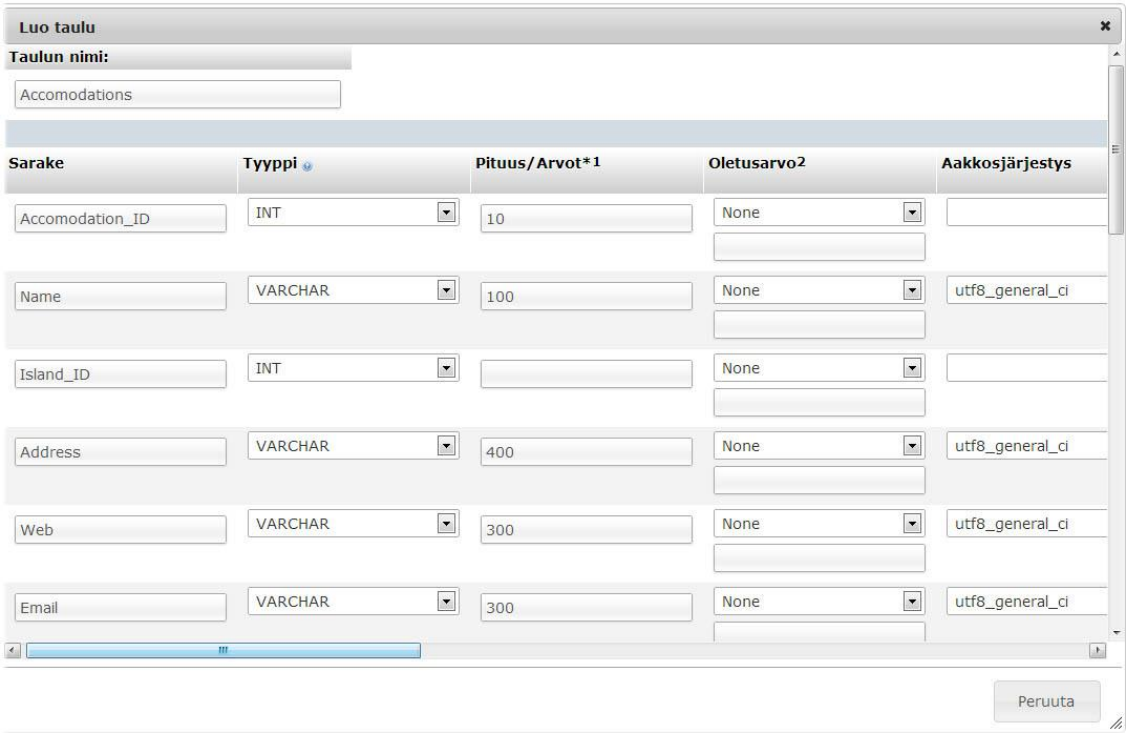
Kuvio 19. Testikäyttäjän luominen

Kolmanneksi tietokantaan luodaan taulut. Ensimmäisen taulun nimeksi annan *Accomodations*. Seuraavaksi pääsen valitsemaan, kuinka monta kenttää haluan tauluun. Kenttiä voi lisätä missä tahansa vaiheessa, mutta hyvän suunnittelun ansiosta tiedän, että kenttien lukumäärä tässä taulussa tulee olemaan 17. Annan suoraan taulun sisältämien kenttien lukumääräksi 17 (Kuvio 20).

Kuvio 20. Uuden taulun lisääminen tietokantaan

Neljännessä vaiheessa kentille annetaan nimet ja tehdään niitä koskevat määritykset (Kuvio 21). Huomautan, että tärkein yksittäinen osa kenttiä koskeviin määrityksiin on jo tehty, sillä tulevat pää- ja viiteavaimet on valittu jo tietokantaa suunniteltaessa. Seu-

raava esimerkki koskee *Accomodiations* -taulun ensimmäistä kenttää, joka on pääavain. Käyn läpi kohta kohdalta, pääavain-kentän määrittelyt.



Kuvio 21. Uuden taulun luominen ja sen sisältämien kenttien määrittely

Ensin pääavaimelle annetaan nimi, ”Accomodation_ID”. Toinen kohta on kentän tyypin määrittäminen. Kentästä halutaan tehdä laskuri, joten sen tyypiksi valitaan numeraalinen vaihtoehto. Tähän sopiva tyyppi on Integer, eli kokonaisluku. Valitsen siis alasvetovalikosta vaihtoehdon int.

Kolmanneksi kentälle määritetään pituus. Kentän pituudella tarkoitetaan sitä, kuinka monta merkkiä kenttään mahtuu. On syytä huomata, että Integer hyväksyy arvoja väliltä 2147483648 ja 2147483647, joten tyypin maksimiarvo voi olla enintään kymmenen merkin pituinen. Sen sijaan jos kentän pituudeksi annettaisiin kaksi (2), olisi sen suurin mahdollinen arvo 99. Annan kentän pituudeksi kymmenen, jotta kenttään mahtuu riittävästi tietueita (majoituskohteita). Seuraavaksi kentälle on mahdollista antaa oletusarvo. Tämä kohta jätetään tyhjäksi pääavainta määriteltäessä. Samoin myös seuraava kohta, jossa on mahdollista määritellä aakkoset joita kenttä käyttää. Kentän ollessa numeraalinen, ei tähän tarvitse valita mitään. (Oracle. 2012)

Seuraavassa vaiheessa on mahdollista antaa kentälle attribuutti, joka tarkoittaa kentän arvon muotoilua. Esimerkiksi voitaisiin valita kohta unsigned zerofill, joka muuttaisi ID -kentän siten, että id-numeron eteen tulisi nollia. Eli ensimmäinen tallennettava tietue saisi id:n arvoksi 0000000001. Kentälle voitaisiin määritellä update current_timestamp, eli tietuetta muutettaessa myös päiväys muuttuisi automaattisesti sen hetkiseen. Haluan kuitenkin, että id on tyyppiä 1- 2147483647, joten en valitse erityisiä attribuutteja kentälle. Seuraavaksi päätetään hyväksyykö kenttä tyhjiä arvoja (NULL). Koska kyseessä on pääavain, sitä tietenkään sallita, joten jätän valintalaatikon (checkbox) tyhjäksi.

Sitten pääavaimen kannalta tärkein kenttä, eli Indeksi, joka tarkoittaa eräänlaista kirjanmerkkiä kentälle. Indeksi nopeuttaa tiedon etsimistä tietokannasta. Tähän valitaan alasvetovalikosta arvo primary (PRIMARY KEY). Tämä tarkoittaa, että kentästä tehdään pääavain. Seuraava checkbox A_I (Auto Increment) on myös tärkeä pääavaimen kannalta, sillä tämä valinta tekee kentälle toivomani laskuriominaisuuden. Auto Increment kasvattaa automaattisesti uuden tietueen id-numeroa yhdellä edelliseen tietueeseen verrattuna. Seuraavassa kohdassa kentälle voidaan antaa kommentii. Annan tähän selityksen ”Automaattinen laskuri id-kentän arvolle”. Kommentteja on hyvä lisätä silloin tällöin, etenkin jos kentän tarkoitus on epäselvä. Näin ollen loppukäyttäjän on helpompi ymmärtää tietokannan toteuttajan visiota.

Seuraavat ominaisuudet ovat tarkoitettu kokeneemman käyttäjän työkaluiksi. Ominaisuudet koskevat muunnosmäärittelyjä. MIME-tyyppi, Selaimen muunnos ja Muunnosvaihtoehdot 3 määrittävät tavan, jolla selain näyttää tekstimuotoisesta datasta poikkeavan datan. Esimerkiksi tietokannan sisältäessä kuvia, ei selain osaa näyttää kuvia ilman oikeanlaisia muunnosmäärittelyjä, ja kuva näkyisi vain sekalaisena teksti- ja merkkiryhmänä.

Seuraavaksi määrittelen loput 16 kenttää edellä esitetyllä tavalla. Nostan kuitenkin esille joitain kenttiä koskevia määrityksiä, jotka eivät välttämättä ole itsestään selviä. Majoituskohteiden nimet tulevat olemaan enimmäkseen indonesialaisia. Indonesiassa käytetään latinalaisia aakkosia, joten tämä on otettava huomioon määriteltäessä kenttää Name. Tähän sopisi Latin-1 merkistö, valitsen kuitenkin mielummin Utf-8 generalin,

koska se on laajempi, jonka ansiosta erikoismerkkien käyttö on helpompaa. (Wikipedia. 2012)

Island_ID-kentästä tehdään indeksi, joka mahdollistaa kentän käytön viiteavaimena. Island_ID-kentän tullessa viiteavaimeksi Islands-taulun pääavaimelle, tekee se Accomodiations-taulusta niin sanotun lapsitaulun Islands-taulun näkökulmasta. Island_ID-kentän tyyppi ja pituus tulee olla sama kuin Islands-taulussa, jossa se on pääavain.Eli kentän tyyppi määräytyy Islands-taulun mukaan. Address-kenttään hyväksyn tyhjän arvon, sillä meillä tulee olemaan paikan koordinaatit, joiden avulla majoituspaikka on helppo paikantaa. Koordinaateista kerron lisää edellä.

Puhelinnumeron kohdalla tulee pohtia, millaista muotoa ovat Indonesialaiset puhelinnumerot. Lyhyimmillään numero koostuu maakoodista (+62), aluekoodista (1 merkki) ja itse numerosta (6 merkkiä). Eli kaikki yhteen laskettuna minimipituus on 10 merkkiä. Minulle oleellisempi tieto on kuitenkin se, kuinka pitkä se maksimissaan voi olla. Numeron maksimipituudeksi saadaan arvo 18. Se muodostuu maakoodista (+62), välilyönti, aluekoodista (3 merkkiä), välilyönti ja numerosta (10 merkkiä). Eli Phone-kentän maksimipituudeksi annan arvon 18. Puhelinnumeron ollessa 18 merkkiä pitkä, se menee yli Integerin maksimiarvon, johon mahtuu ainoastaan kymmenen merkin pituinen numerosarja. Vaikka kyseessä on numerokenttä, tulee sen hyväksyä myös plus-merkki, joten tyypiksi valitsen Varchar:in. (howtocallabroad.com. 2012)

Coordinates-kenttään tulee majoituskohteen tarkka sijainti kartalla. Tarkoitus on iPhonen kompassisovellusta käyttäen saada majoituskohteen leveys- ja pituusasteet (Kuvio 22). Kun leveys- ja pituusasteet ovat selvillä, voidaan ne muuttaa latitude- ja longitude-arvoiksi, jotka säilötään Coordinates-kenttään. Käytettävyyden kannalta haluan jakaa Coordinates-kentän omiin osiinsa Latitude ja Longitude. Kun Coordinates-kentän arvot ovat omissa kentissään (Latitude & Longitude) toimii niiden hakeminen tietokannasta huomattavasti nopeammin. Kenttien tyypiksi määritän FLOAT(10,6), mikä tarkoittaa, että desimaalipilkun jälkeen hyväksytään kuusi numeroa koko merkkijonon saadessa maksimiarvon kymmenen merkkiä.

Kuvio 22. iPhonen kompassi

Facilitilities-, Description- ja Prices-kentät saavat ainakin aluksi jäädä hyvin vapaamuotoisiksi, siksi määritän kenttien tyypiksi Text. Tekstimuotoiselle tyypille ei tarvitse ennalta määrittää maksimipituutta. Verrattuna Varchariin, se myös pienentää tietokannan kokoa pitkiä tekstimääriä käsitellessä. Contract_ID:stä tulee foreign key (viiteavain), joten indeksoin kentän. Hyväksyn myös tyhjän arvon tähän kohtaan, sillä sopimuksia tullaan tekemään vasta palvelun ollessa kokonaan valmis. Viimeisenä kenttänä on Secret_email. Kenttään on tarkoitus tallettaa vaihtoehtoisia sähköpostiosoitteita, joita ei välttämättä ole tarkoitettu yleisölle. Esimerkiksi majoituskohteen markkinoinnista vastaavan henkilön sähköpostiosoite.

Ennen taulun tallennusta, on mahdollista liittää kommentti taulua koskien. Myös taulun tallennusmoottori valitaan tässä vaiheessa. Tallennusmoottoriksi on valittava InnoDB, koska se on ainoa tallennusmoottoreista, joka tarkistaa viite-eheydet. Tässä kohtaa voidaan myös määrittää koko taulua koskeva aakkosjärjestys, jolloin sitä ei tarvitse erikseen valita jokaisen yksittäisen kentän kohdalla.

Edellä mainittuja menetelmiä käyttäen luon muut tietokantaan suunnitellut taulut Users, Islands ja Contracts. Toiston välttämiseksi en tule kommentoimaan kuin muutamalla lauseella jäljellä olevien taulujen luomisprosessia.

Islands-taulussa on huomioitava, että se tulee toimimaan Accomodiations taulun isätauluna, kentän Island_ID suhteen. Tästä syystä jokaista Island_ID:tä täytyy vastata yksi saaren nimi eli Name. Tyhjää arvoa (NULL) ei siis sallita.

Contracts-taulun kohdalla tilanne on hyvin samanlainen. Erona kuitenkin se, että ainoa tieto, jonka tulen varmuudella vaatimaan sopimukselta on kenttä Description. Tämä johtaa tilanteeseen, jossa jokaiseen Accomodiations-taulussa viitattuun sopimukseen kuuluu ainakin sopimuksen kuvaus (Description). Contracts-taulun viimeiset kentät tulevat sisältämään päivämääriä. Näihin kenttiin täytetään sopimuksentekopäivä Date ja sopimuksen sulkeutumispäivä Expiration. Annan kentille tyypin DATE. Lisäksi haluan helpottaa tietueiden syöttämistä niin, että tietueen lisäyksen yhteydessä oletuksena tarjotaan Date:n arvoksi sen hetkistä päivämäärää. Huomautan, että MySQL käyttää amerikkalaista tapaa merkitä päivämääriä, eli päivämäärät ovat muotoa yyyy-mm-dd.

User-tauluun vaadin ainakin tiedot User_ID, User_name, Password sekä Email. Sähköpostiosoite tulee pakolliseksi kentäksi, koska nimimerkin rekisteröiminen tulee todennäköisesti vaatimaan sähköpostiosoitteen, jotta rekisteröityminen saadaan vahvistettua. Muilta kentiltä hyväksyn arvon NULL. Password-kentän määrityksessä käytän md-5 hash-tyyppistä salausta. ”MD5 generates a 128-bit hash value. You can use CHAR(32) or BINARY(16)” (http://stackoverflow.com). Valitsen kentän tyypiksi Binary(16). Tällöin palveluun rekisteröityessä käyttäjän antama salasana suojataan md-5 hash:illä. Tämä tapahtuu seuraavanlaisella SQL-lauseella: INSERT INTO contracts VALUES ('member1',MD5('password')…); Käyttäjän kirjautuessa palveluun salasanan oikeellisuus tarkistetaan vastaavanlaisella SQL-lauseella. (MySQL Consulting and NoSQL Consulting: MySQL DBA. 2006)

Kaikkien neljän taulun luontiprosessin ollessa ohi. Kuvio 23 näyttää tietokannan yleisnäkymässä kaikki sen sisältämät taulut, taulujen sisältämien tietueiden lukumäärän, tyy-

pin, aakkosjärjestyksen sekä koon. Lisäksi kuvakaappaukset kaikista tauluista erikseen (Kuviot 24, 25, 26, 27). Kuvioista nähdään kaikki tauluille tekemäni määritykset.

Taulu	Toiminnot	Kpl rivejä	Tyyppi	Aakkosjärjestys	Koko	Ylijäämä
accomodations	Selaa Rakenne Etsi Lisää rivi Tyhjennä Tuhoa	0	InnoDB	utf8_general_ci	48,0 kt	-
contracts	Selaa Rakenne Etsi Lisää rivi Tyhjennä Tuhoa	0	InnoDB	utf8_general_ci	16,0 kt	-
islands	Selaa Rakenne Etsi Lisää rivi Tyhjennä Tuhoa	0	InnoDB	utf8_general_ci	16,0 kt	-
users	Selaa Rakenne Etsi Lisää rivi Tyhjennä Tuhoa	0	InnoDB	utf8_general_ci	16,0 kt	-
4 taulua	Summa	0	InnoDB	latin1_swedish_ci	96,0 kt	0 tavua

Kuvio 23. Yleisnäkymä tietokannasta BaliBeachBungalows

#	Sarake	Tyyppi	Aakkosjärjestys	Attribuutit	Tyhjä	Oletusarvo	Lisätiedot	Toiminnot
1	Accomodation_ID	int(10)			Ei	None	AUTO_INCREMENT	Muokkaa Tuhoa Lisää
2	Name	varchar(100)	utf8_general_ci		Ei	None		Muokkaa Tuhoa Lisää
3	Island_ID	int(11)			Ei	None		Muokkaa Tuhoa Lisää
4	Address	varchar(400)	utf8_general_ci		Kyllä	NULL		Muokkaa Tuhoa Lisää
5	Web	varchar(300)	utf8_general_ci		Kyllä	NULL		Muokkaa Tuhoa Lisää
6	Email	varchar(300)	utf8_general_ci		Kyllä	NULL		Muokkaa Tuhoa Lisää
7	Phone	int(18)			Kyllä	NULL		Muokkaa Tuhoa Lisää
8	Coordinates(LAT)	float(10,6)			Kyllä	NULL		Muokkaa Tuhoa Lisää
9	Coordinates(LNG)	float(10,6)			Kyllä	NULL		Muokkaa Tuhoa Lisää
10	Rooms	int(4)			Kyllä	NULL		Muokkaa Tuhoa Lisää
11	To_beach	int(5)			Kyllä	NULL		Muokkaa Tuhoa Lisää
12	Restaurants	int(2)			Kyllä	NULL		Muokkaa Tuhoa Lisää
13	Swimming_pools	int(2)			Kyllä	NULL		Muokkaa Tuhoa Lisää
14	Facilities	text	utf8_general_ci		Kyllä	NULL		Muokkaa Tuhoa Lisää
15	Description	text	utf8_general_ci		Kyllä	NULL		Muokkaa Tuhoa Lisää
16	Prices	text	utf8_general_ci		Kyllä	NULL		Muokkaa Tuhoa Lisää
17	Contract_ID	int(11)			Kyllä	NULL		Muokkaa Tuhoa Lisää
18	Secret_email	varchar(100)	utf8_general_ci		Kyllä	NULL		Muokkaa Tuhoa Lisää

Valitse kaikki / Poista valinta kaikista *Valitut:* Selaa Muokkaa Tuhoa Perusavain Uniikki Indeksi

Kuvio 24. Taulu -accomodations

#	Sarake	Tyyppi	Aakkosjärjestys	Attribuutit	Tyhjä	Oletusarvo	Lisätiedot	Toiminnot
1	Island_ID	int(11)			Ei	None	AUTO_INCREMENT	Muokkaa Tuhoa Lisää
2	Name	varchar(30)	utf8_general_ci		Ei	None		Muokkaa Tuhoa Lisää

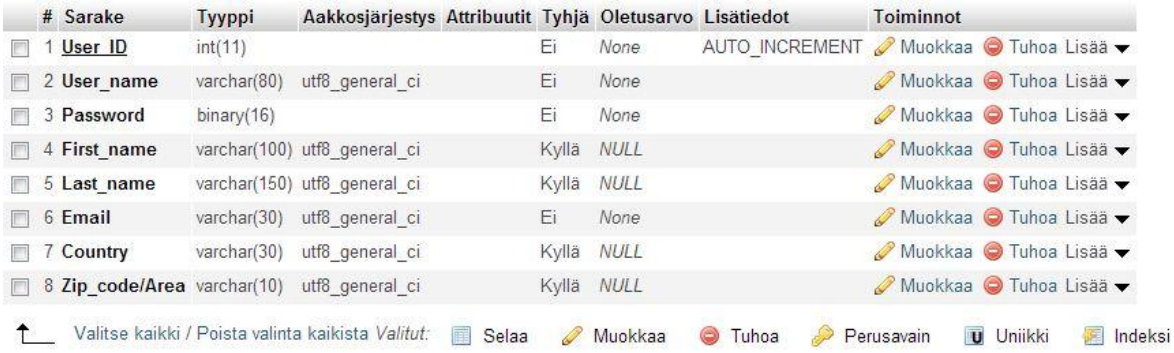
Valitse kaikki / Poista valinta kaikista *Valitut:* Selaa Muokkaa Tuhoa Perusavain Uniikki Indeksi

Kuvio 25. Taulu-islands

#	Sarake	Tyyppi	Aakkosjärjestys	Attribuutit	Tyhjä	Oletusarvo	Lisätiedot	Toiminnot
1	Contract_ID	int(11)			Ei	None	AUTO_INCREMENT	Muokkaa Tuhoa Lisää
2	Contact_person	varchar(100)	utf8_general_ci		Kyllä	NULL		Muokkaa Tuhoa Lisää
3	Description	text	utf8_general_ci		Ei	None		Muokkaa Tuhoa Lisää
4	Date	date			Kyllä	NULL		Muokkaa Tuhoa Lisää
5	Expiration	date			Kyllä	NULL		Muokkaa Tuhoa Lisää

Valitse kaikki / Poista valinta kaikista *Valitut:* Selaa Muokkaa Tuhoa Perusavain Uniikki Indeksi

Kuvio 26. Taulu-contracts



#	Sarake	Tyyppi	Aakkosjärjestys	Attribuutit	Tyhjä	Oletusarvo	Lisätiedot	Toiminnot
1	User_ID	int(11)			Ei	None	AUTO_INCREMENT	Muokkaa Tuhoa Lisää
2	User_name	varchar(80)	utf8_general_ci		Ei	None		Muokkaa Tuhoa Lisää
3	Password	binary(16)			Ei	None		Muokkaa Tuhoa Lisää
4	First_name	varchar(100)	utf8_general_ci		Kyllä	NULL		Muokkaa Tuhoa Lisää
5	Last_name	varchar(150)	utf8_general_ci		Kyllä	NULL		Muokkaa Tuhoa Lisää
6	Email	varchar(30)	utf8_general_ci		Ei	None		Muokkaa Tuhoa Lisää
7	Country	varchar(30)	utf8_general_ci		Kyllä	NULL		Muokkaa Tuhoa Lisää
8	Zip_code/Area	varchar(10)	utf8_general_ci		Kyllä	NULL		Muokkaa Tuhoa Lisää

Valitse kaikki / Poista valinta kaikista *Valitut:* Selaa Muokkaa Tuhoa Perusavain Uniikki Indeksi

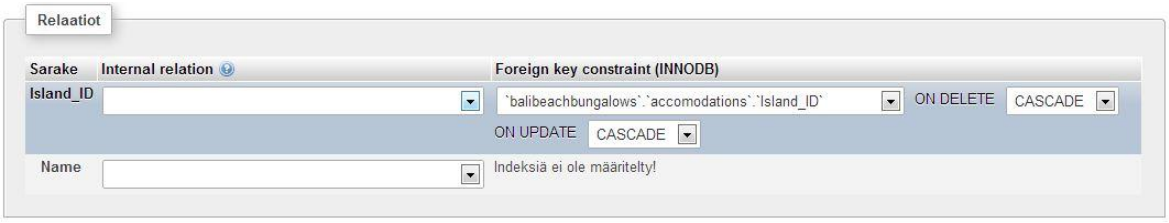
Kuvio 27. Taulu-users

5.3.3 Taulujen välisten yhteyksien luominen

Tietokannan luomisen viimeisessä vaiheessa luodaan yhteydet taulujen välille. Ensimmäiseksi luon yhteyden taulun accomodiations ja taulun islands välille. Tarkemmin kentästä Island_ID kenttään Island_ID. Aloitan valitsemalla taulun, jossa pääavain sijaitsee eli niin sanotun isätaulun. Tässä tapauksessa valitsen taulun islands. Island-taulun näkymässä klikkaan vaihtoehtoa Relaationäkymä. Seuraavaksi pääsen valitsemaan relaatioita islands-taulusta. Ensimmäinen sarake on Infernal relation, ja tässä määritellään taulun mahdollisia sisäisiä yhteyksiä. Huomioni kiinnittyy kuitenkin seuraavaan sarakkeeseen Foreign key constrait. Tässä sarakkeessa määrittelen Island_ID:lle viiteavaimen. Tämän taulun kohdalla viiteavaimia pystyy luomaan ainoastaan Island_ID-kenttää käyttäen, sillä se on ainoa kenttä, joka on määritelty avaimeksi. Mikäli kenttä Name olisi indeksoitu olisi se myös mahdollinen kenttä relaatioiden luomiseen. Valitsen kuvion 28 tavoin Island_ID pääavaimelle viiteavaimeksi Island_ID:n taulusta accomodations.

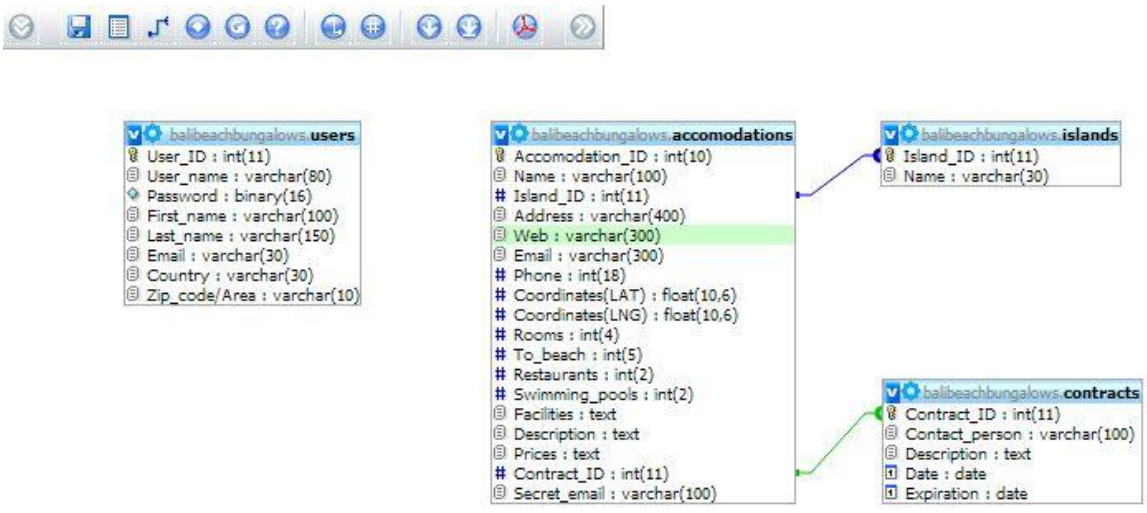
Seuraavaksi määrittelen miten yhteys tulee toimimaan lapsitaulussa tilanteissa, joissa isätaulua päivitetään tai sieltä poistetaan tietueita. Valitsen CASCADE-vaihtoehdot molemmissa tapauksissa. CASCADE-vaihtoehdon ansiosta muutokset tulevat voimaan myös lapsitaulussa. Esimerkiksi oletetaan tilanne, jossa island-taulun ensimmäinen tie-

tue (id=1), on nimeltään bali. Nyt accomodiations-tauluun on lisätty tietueita, jotka sijaitsevat saarella bali. Myöhemmin huomataan, että bali on väärin kirjoitettu, sillä se olisi pitänyt kirjoittaa isolla alkukirjaimella. Sen sijaan, että minun tulisi nyt käydä läpi kaikki tietueet taulusta accomodations, tarvitsee minun päivittää iso alkukirjain ainoastaan kerran tietueeseen numero yksi taulussa islands. Nyt päivittyvät myös kaikki accomodation-taulun tietueet, jotka sijaitsevat saarella Bali.



Kuvio 28. Relaation luonti

Vertauksen vuoksi esittelen SQL-lauseen, jolla olisi saavutettu täsmälleen samanlainen yhteys tietokantaan. `islands` ADD FOREIGN KEY (`Island_ID`) REFERENCES `balibeachbungalows`. `accomodations` (`Island_ID`) ON DELETE CASCADE ON UPDATE CASCADE ;
Tietokannan sisältämä toinen yhteys on täsmälleen samantyyppinen, joten en selitä yhteyden muodostamista uudestaan. Lisään kuitenkin kuvakaappauksen PHPMyAdminin tarjoamasta relaationäkymästä. Näkymässä on esillä kaikki tietokannan sisältämät taulut ja niiden väliset yhteydet (Kuvio 29).

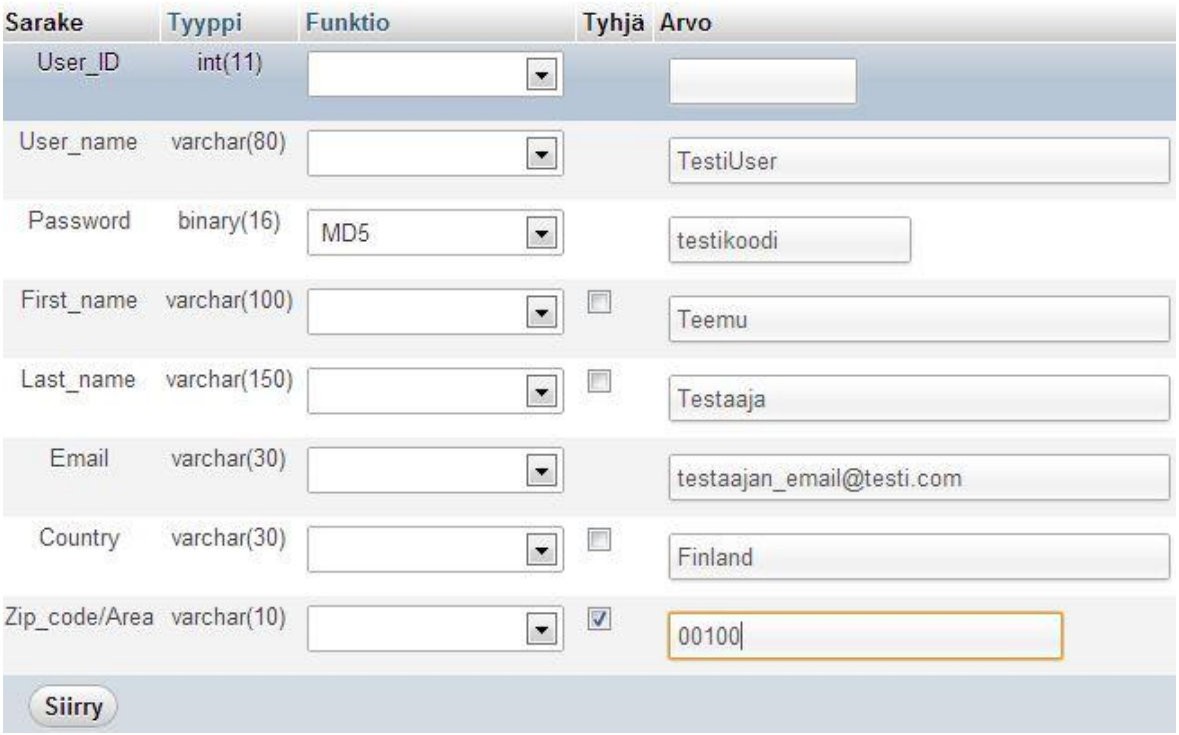


Kuvio 29. Valmiit taulut ja niiden väliset yhteydet

5.4 Tietokannan testaaminen

Tietokannan ollessa valmis on aika syöttää tietokantaan joitain kuvitteellisia tietueita. Testauksen tärkein tavoite on varmistaa, että taulujen väliset yhteydet toimivat halutulla tavalla. Toinen asia johon tulee kiinnittää huomiota on, että taulut on määritelty oikein ja että ne pystyvät vastaanottamaan niille tarkoitetun datan. Testauksen pääpaino on siis käytettävyyden testaamisessa. En tule kiinnittämään huomiota suorituskykyyn. Mielestäni suorituskyvyn kehittäminen tulee kysymykseen vasta kun tietokanta alkaa kasvamaan erittäin isoksi. Tietokannan testaamisessa tulen käyttämään PHPMyAdminiä.. Yhtä hyvin voisin syöttää datan käyttämällä pelkkää SQL-kieltä, komentokehotteen avulla.

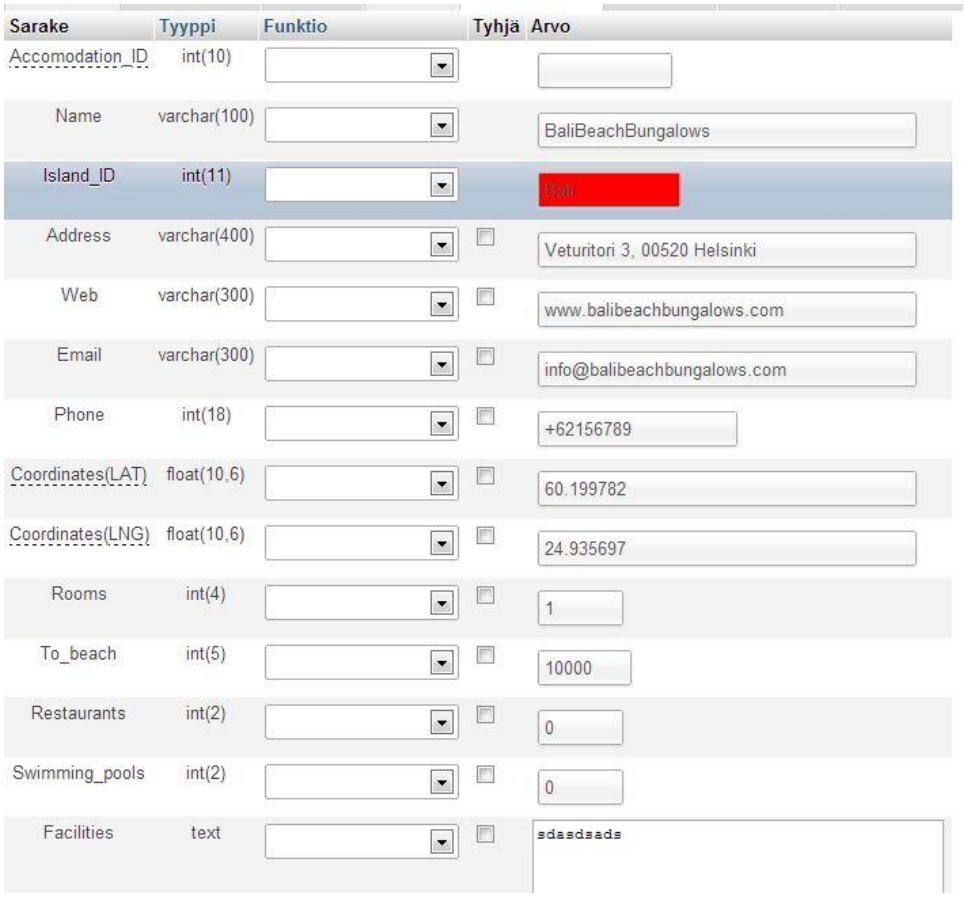
Aloitan luomalla testikäyttäjän tauluun users. Tämä onnistuu klikkaamalla lisää rivi painiketta taulun users yleisnäkymässä. PHPMyAdmin avaa kuvion 30 mukaisen näkymän. Ensimmäinen kenttä User_ID voidaan suosiolla jättää tyhjäksi, sillä siihen on asennettu autolaskuri, joten se kasvattaa id:n arvoa aina yhdellä verrattuna edelliseen tietueeseen. Kuten kuvasta nähdään, olen nyt valinnut Password kentälle funktion MD5. Tämä on algoritmi jonka tarkoitus on kryptata (salata) antamani salasanan 16 merkkiä pitkään heksa koodattuun muotoon. Täytän vielä loputkin tiedot ja painan siirry nappulaa. Avatessani users-taulun yleisnäkymän, huomaan uuden tietueen ilmestyneen tauluun (Kuvio 31). Kuvasta nähdään myös, että salasanan salaus on onnistunut ja antamani salasana on kryptatussa muodossa. (Wikipedia. 2012)



Kuvio 30. Testikäyttäjän syöttö

Kuvio 31. Taulu users, sisältäen yhden käyttäjän

Seuraavaksi siirryn tauluun accomodations. Kentän Island_ID kohdalla meillä on odotettavissa ongelma, yhteyden ollessa kunnossa. Island_ID:n isätaulusta islands olisi löydettävä id:tä vastaava saari (Esimerkiksi id=1 Bali). Yritän kirjoittaa Island-ID kentän arvoksi bali, koska kentän tyyppi on numeraalinen PHPMyAdmin värjää kentän punaiseksi (Kuvio 32). Vaihdan kentälle arvoksi numeron yksi. Tämän jälkeen tallennan. Tallennuksen ei tulisi onnistua, sillä saari taulusta ei löydy id numeroa yksi vastaavaa tietuetta (islands taulun ollessa tyhjä).



Kuvio 32. Testi majoituskohteen lisäys

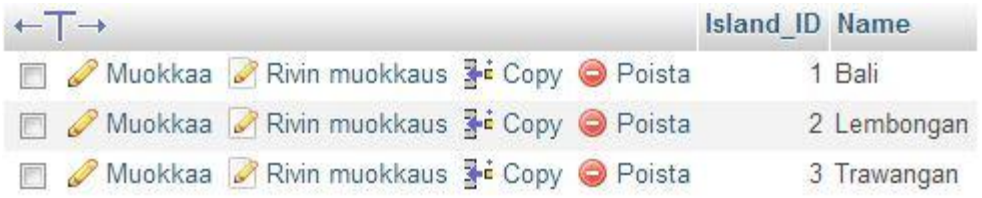
Vastoin odotuksiani tietueen tallennus onnistuu. Tämä kertoo, että yhteys ei toimi haluamallani tavalla. Seuraavaksi yritän syöttää islands tauluun saaren. Saaren lisäys ei onnistu. Tämä vahvistaa minun epäilykseni, yhteys on luotu väärinpäin. Eli lapsitauluna toimii islands taulu ja isätauluna accomodations. Islands tauluun ei voida lisätä saarta, jota ei löydy accomodations taulusta. Tarkoitus oli juuri päinvastainen. Tietokannan molemmat yhteydet luotiin identtisin menetelmin, joten molemmat ovat väärin.

Seuraavaksi avaan tietokannan relaationäkymän ja yksitellen valitsen yhteyden ja delete näppäintä käyttäen poistan sen. Tämän jälkeen haluan tyhjentää taulun accomodiations. Käytän siihen SQL-lausetta TRUNCATE accomodations; Sen lisäksi, että lause tyhjentää taulun se myös nollaa id-kentän autolaskurin, joka on toivomani lopputulos.

Korvatakseni vanhat väärät yhteydet, avaan nyt accomodations taulun relaationäkymän. Näkymässä määrittelen Island_ID:lle foreign key constraintiksi vaihtoehdon 'balibeachbungalows'.'islands'.'Island_ID' samoin Contract_ID:lle 'balibeachbunga-

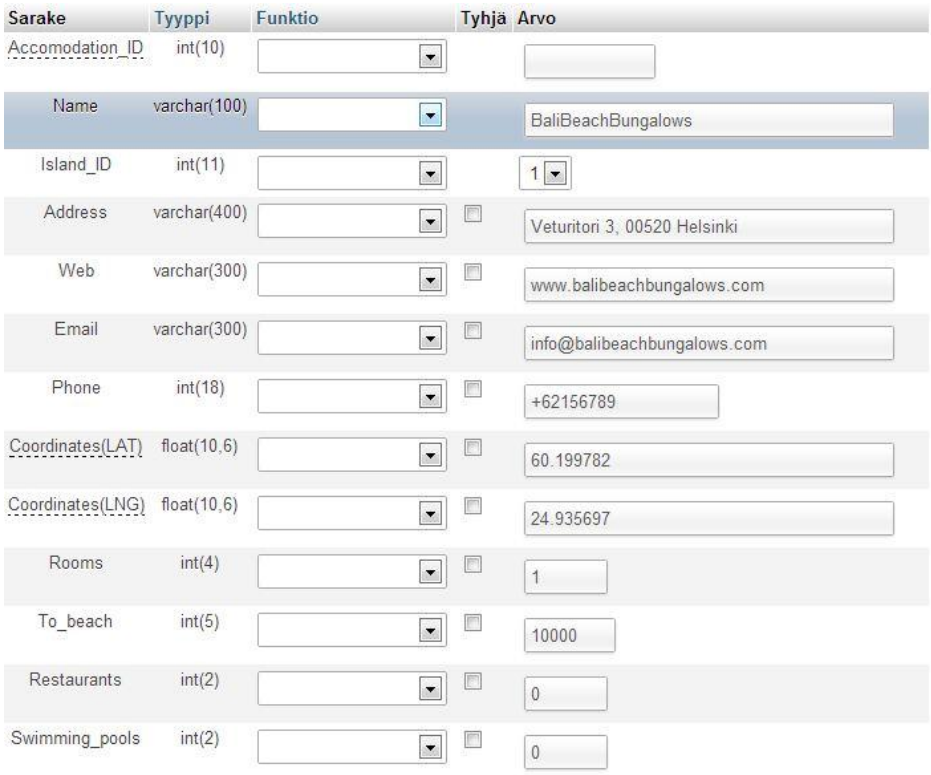
lows'.'contracts'.'Contract_ID'. Yhteyksien tyypiksi valitsen jälleen CASCADE on update and on delete.

Nyt kun uudet yhteydet on luotu, tehdään uusi testi. Tällä kerralla aloitan syöttämällä, saaria islands tauluun (Kuvio 33). Teen myös yhden testisopimuksen tauluun contracts (Kuvio 34). Kun saaret ja sopimus on onnistuneesti lisätty, on aika lisätä testimajoituskohde, jossa on tarkoitus viitata edellä syöttämiini tietueisiin (Kuvio 35). Kuten kuvasta huomataan, tarjoaa nyt PHPMyAdmin Island_ID:n kohdalla alasvetovalikon. Alasvetovalikko tarjoaa numerot 1-3, eli islands-tauluun lisäämäni tietueet. Samoin toimii myös kentän Contract_ID kohdalla. Syötettyäni kaikki testidatan ominaisuudet lisäys lomakkeelle, painan jälleen siirry painiketta, joka tallentaa onnistuneesti tietueen tietokantaan. Koska tietueen lisäys onnistui, voin todeta yhteyksien olevan kunnossa (Kuvio 36). Huomaan kuitenkin vielä yhden virheen sillä Phone kentän tietotyypiksi on epähuomiossa jäänyt INT, joten se ei hyväksy merkkiä "+". Korjaan asian käyttämällä yksinkertaista SQL-lausetta ALTER TABLE accomodations CHANGE Phone Phone varchar(16); Tämän jälkeen muokatessani valmista tietuetta, pystyn antamaan puhelin numerolle merkin "+". Löytämäni virheet tietokannan rakenteesta ja sen puutteista, todistaa erinomaisesti kuinka tärkeä osa testidatan syöttö on tietokannan luomisprosessin loppuun saattamiselle.



Kuvio 33. Testisaaret lisätty islands-tauluun

Kuvio 34. Testisopimus lisätty tauluun contracts



Kuvio 35. Majoituskohteen lisäys

Accomodation_ID Automaattinen laskuri Id kentän numerolle	Name	Island_ID	Address	Web	Email	Phone	Coordinates(LAT) Tähän latitude arvo	Coordinates(LN Tähän longitude arvo
1	BaliBeachBungalows	1	Veturitori 3, 00520	www.balibeachbungalows.com	info@balibeachbungalows.com	62156789	60.199783	24.9356

Kuvio 36. Testimajoituskohde lisätty tietokantaan.

5.5 Tulokset

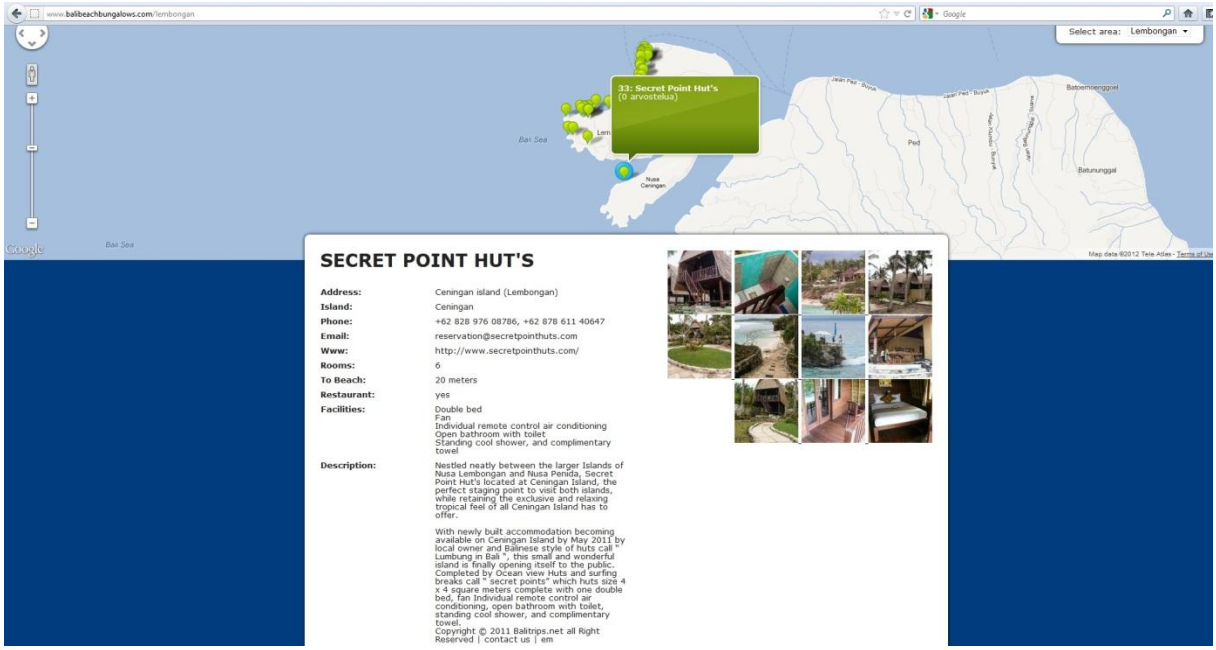
Tietokanta syntyi melkein alkuperäisen suunnitelman mukaiseksi. Suurin ero lopputuotoksen ja suunnitelman välillä on island nimisen kentän tuominen omaan tauluunsa. Joidenkin kenttien nimet poikkeavat hieman alkuperäisestä suunnitelmasta syystä, että nimet kuvaavat nyt paremmin kenttien ominaisuuksia. Kenttiä nimetessä käytin samanlaista kirjoitusasua jokaisen kentän kohdalla. Kenttien nimiä olisi voinut miettiä vieläkin tarkemmin, sillä nyt tietokanta pitää sisällään saman nimisiä kenttiä, kuten Name ja Description. En uskon, että tämä tulee tuottamaan ongelmaa, sillä kenttiä on vaikea sekoittaa keskenään. Kenttien nimeämisestä päästään taulujen nimeämiseen. Taulujen nimet oli tarkoitus kirjoittaa isolla alkukirjaimella, mikä oli kuitenkin MySQL:än yleisten käytäntöjen vastaista. PHPMyAdmin muutti automaattisesti kaikkien taulujen alkukirjaimet pieniksi. Tämä selittää taulujen nimien erilaiset kirjoitusasut läpi opinnäytetyön. Suunnitelmassa taulujen nimet on kirjoitettu isolla alkukirjaimella ja toteutuksessa jäl-

keen pienellä alkukirjaimella. Olisi voinut olla järkevää nimetä myös kaikki kentät MySQL:än käytäntöjen mukaan, eli kaikki merkit pienellä. PHPMyAdminin saa käyttöön suomenkielisenä, mikä ei kuitenkaan tarkoita, että kaikki MySQL-tietokannan ominaisuudet kääntyisi suomeksi. Kenttien tyyppejä määritellessäni minulle tuotti hieman ongelmia joidenkin ominaisuuksien ymmärtäminen. Tästä johtuen loin aluksi taulujen väliset yhteydet väärinpäin. Lopulta yhteydet saatiin korjattua ja tietokanta normalisoitua. Projektin onnistumisen kannalta tietokannan testaaminen oli merkittävässä roolissa. Sillä testidatan syöttäminen osoitti minulle perustavaa laatua olevan virheen.

6 Pohdinta

Aloitan pohtimalla tietoperustan tavoitteita. Mielestäni tietoperustan päätavoite täyttyi erinomaisesti, sillä sen oli määrä kertoa tietokannan tekemisen vaiheista. Uskon, että tietoperusta on hyvin luotettava, sillä se on kirjoitettu enimmäkseen alan kirjallisuuden pohjalta. Lähteinä käytin, suomen sekä englannin kielistä kirjallisuutta. Oma tietämykseni syveni laajasti, erityisesti tietokannan suunnitteluun liittyvien menetelmien kohdalla. Tämä auttoi huomattavasti oman tietokantaprojektin suunnittelua. Tietoperustassa vertailin kahden relaatiotietokannan ominaisuuksia ja löysin vertailun avulla projektiimme sopivan tietokantamallin. Empiirisen osuuden tavoitteissa pääpaino oli tietysti tehdä tietokanta, josta palvelumme hyötyy. Mielestäni onnituin tavoitteessa hyvin, sillä näitä johtopäätöksiä kirjoittaessa tietokanta on jo käytössä. Tietokantaan on myös onnistuttu keräämään kaikki olennainen tieto koskien noin 150 majoituskohdetta Balilla ja sen lähiympäristössä. Tietokannan rakenne on looginen ja palvelee erinomaisesti palvelumme tarpeita. Tietokannan työstämiselle oli projektisuunnitelman mukaan varattu aikaa kokonaisuudessaan yksi kuukausi. Suunnitelmassa on huomioitu käytettävä aika sekä tietokannan suunnittelulle ja toteuttamiselle, että tiedon keräämiselle tietokantaan. Suunnitelmassa aikaa on varattu yksi viikko suunnitteluun ja toteutukseen, joka oli riittävästi. Tietokannan täyttöön kulutettuun aikaan en ota kantaa, sillä se liittyy tietokannan käyttöön ja käyttö on rajattu opinnäytetyöaiheeni ulkopuolelle. Matka Balille sujui kuitenkin hienosti ja tietokantaan saatiin kerättyä kohteita, mikä tulee ilmi myös seuraavassa kappaleessa, jossa esitellään tämän hetkistä palveluamme.

Domain-nimeksi on valittu balibeachbungalows.com. Verkkopalvelu perustuu tällä hetkellä Google mapilla oleviin pisteisiin. Pisteet ovat majoituskohteita, joita palveluumme on tällä hetkellä listattu. Majoituskohteen tiedot saadaan esille klikkaamalla pistettä kartalla (Kuvio 38). Käyttäjän klikatessa majoituskohdetta, lähetetään serverille SQL-lause, joka käskee hakea kaikki tiedot kyseisen majoituskohteen kohdalta. Kun tieto on haettu serverillä olevasta tietokannasta, näytetään se käyttäjän selaimella, kuten kuviossa 38.



Kuvio 38. Verkkopalvelu balibeachbungalows.com

Kaiken kaikkiaan olen tyytyväinen lopputulokseen. Tietoperusta on luotettavalta pohjalta kirjoitettu ja projekti on vahvalla ammattitaidolla toteutettu. Läpi opinnäytetyön olen kehittynyt erityisesti työn tulosten raportoinnissa. Vaikka tietoperustan aiheet olivat entuudestaan melko tuttuja, opin silti paljon uuttakin. Erityisesti tietokannan hallinnasta kertova kappale muistutti hyvin mieleeni tärkeitä perusasioita SQL-kieleen liittyen. Täysin uutena asiana minulle tuli erilaiset tietokannan sisältämän datan salaus menetelmät. Lisäksi aivan uutta minulle oli kenttien välisten yhteyksien luominen PHPmyAdmin ohjelmassa.

Tietokantaan on tähän päivään mennessä tehty seuraavanlainen parannus. island-taulu on muutettu area-tauluksi. Muutos on tehty syystä, että haluamme verkkopalvelumme pystyvän kohdentamaan karttaa alueittain. Näin asiakas pystyy tarkemmin näkemään kohteet alueittain, saarien sijaan. Myös tietokannassa olevat tietueet on muutettu vastaamaan tarpeeseen. Area-taulun tietueet haetaan oikeassa yläkulmassa olevaan alasvetovalikkoon (Kuvio 39). Alasvetovalikkoa käyttämällä asiakas voi valita minkä alueen kohteet näytetään.

Kuvio 39. area-taulun sisältö alasvetovalikossa

Tietokantaa voisi jatkaa niin, että sen olisi mahdollista toimia myös varausjärjestelmän pohjana. Tämä tarkoittaisi, että käyttäjä pystyisi tekemään varauksia haluamaansa majoituskohteeseen, haluamalleen ajankohdalle. Tietokannan pitäisi pystyä pitämään myös lukua jokaisen majoituskohteen tämän hetkisestä varaustilanteesta.

Lähteet

Haasio, A. 2005. Internetin Tietokannat. BTJ Kirjastopalvelu. Helsinki

Hernandez, M-J. 1997. Tietokannat suunnittelu käytännössä. Kääntänyt Tomi Kujala 2000. IT-Press. Jyväskylä.

Wikispaces 2012. Tietokantakoulutus. Luettavissa: http://tietokantakoulutus.wikispaces.com/Tietokantojen+tyyppej%C3%A4. Luettu 8.11.2012.

Connolly, T., Begg, C. 2002. Database Systems. Pearson Education Limited. Harlow.

Connolly, T., Begg, C. 2004. Database Solutions. Bell & Bain Ltd. Glascow.

Groh, M-R., Stockman, J-C., Powell, P., Prague, C-N., Irwin, M-R., Reardon, J. 2007. Access 2007 Bibble. Wiley Publishing, Inc. Canada .

Oracle Inc. 2012. Why MySQL. Luettavissa: http://www.mysql.com/why-mysql/. Luettu. 8.11.2012.

Oracle Inc. 2012. MySQL Enterprise Backup. Luettavissa: http://www.mysql.com/why-mysql/. Luettu. 8.11.2012.

Oracle Inc. 2012. Numeric Types. Luettavissa: http://dev.mysql.com/doc/refman/5.5/en/numeric-types.html#integer-types. Luettu. 8.11.2012.

PHP.net. 2012. Johdatus PHP-kieleen. Luettavissa: http://users.jyu.fi/~kolli/ITK215/PHP/tietokanta.html. Luettu. 8.11.2012.

Ao media. 2012. phpMyadmin, About. Luettavissa:
http://www.phpmyadmin.net/home_page/. Luettu 8.11.2012.

Wikipedia Foundation. 2012. Indonesian kieli. Luettavissa:
http://fi.wikipedia.org/wiki/Indonesian_kieli. Luettu 8.11.2012.

Wikipedia Foundation. 2012. MD5 Luettavissa: http://fi.wikipedia.org/wiki/MD5.
Luettu 8.11.2012.

HowToCallAbroad.com. 2012. Internation calling guide. Luettavissa:
http://www.howtocallabroad.com/indonesia/. Luettu 8.11.2012.

MySQL Consulting and NoSQL Consulting: MySQL DBA . 2006. Storing Passwords in MySQL. Luettavissa:
http://mysqldatabaseadministration.blogspot.fi/2006/08/storing-passwords-in-mysql.html. Luettu 8.11.2012.

Tilastokeskus. 2012. Luettavissa:
http://www.tilastokeskus.fi/. Luettu 14.11.2012

SuomiSanakirja.fi. 2012. Algoritmi. Luettavissa:
http://suomisanakirja.fi/algoritmi. Luettu 15.11.2012

Wikipedia Foundation. 2012. Data. Luettavissa:
http://fi.wikipedia.org/wiki/Data. Luettu 15.11.2012.

Wikipedia Foundation. 2012. Eheys. Luettavissa:
http://fi.wikipedia.org/wiki/Eheys. Luettu 15.11.2012.

Koulutuskeskus Salpaus. 2004. SQL-kieli. Luettavissa:
http://edu.phkk.fi/opiskelu/sqlkieli/sql_osa30.htm. Luettu 15.11.2012.

Oulun seudun ammattiopisto. 2012. Tietokantojen peruskäsitteet. Luettavissa: http://www.okol.org/verkkokurssit/datanomi/tietojarjestelmien_kehittaminen/tiedon hallintajarjestelmat/peruskasitteet.htm. Luettu 15.11.2012.

Wikipedia. 2012. Query By Example. Luettavissa: http://en.wikipedia.org/wiki/Query_by_Example. Luettu 15.11.2012.

Wikipedia Foundation. 2012. Relaatio. Luettavissa: http://fi.wikipedia.org/wiki/Relaatio. Luettu 15.11.2012.

Wikipedia Foundation. 2012. Data structure. Luettavissa: http://en.wikipedia.org/wiki/Data_structure. Luettu 15.11.2012.

Liitteet

Liite 1. Projektisuunnitelma

Tavoitteena luoda tietokanta, joka sisältää majoituspalveluiden tietoja. Tietokannan rakenne tulee suunnitella siten, että tietojen käyttö ja tulostus on mahdollista sekä tehokasta palvelimen näkökulmasta ja, että tieto on normalisoitu sekä muutoinkin rakenteeltaan optimaalinen sitä käyttävälle palvelulle (web-sivustolle). Suunnittelun perustana toimii tosielämässä esiintyvien objektien, tässä tapauksessa majoituskohteiden, ominaisuudet. Ensimmäisenä siis listaamme kaikki mahdolliset ominaisuudet joita kohteella voi olla. Tämän jälkeen valitsemme niistä palvelun tarvitsemat ominaisuudet ja niiden variaatiot, eli mahdolliset vaihtoehdot, jos niitä ominaisuudella on. Ominaisuus voi olla yksinkertaisuudessaan vaikkapa majoituskohteen tyyppi: bungalow, villa, hotelli tai guest house. Seuraavaksi määrittelemme ominaisuuden tietotyypin tietokannan näkökulmasta. Tietotyyppiin liittyy lisäksi tiedon koko, minimi- sekä maksimiarvo ja itse tietotyypin muoto. Esimerkkinä vaikkapa numeraalinen arvo, joka voi olla vaikkapa uima-altaiden määrä, minimissään nolla, maksimissaan ääretön, tai tässä tilanteessa valitsemamme tietotyypin maksimiarvo (tinyint).

Aikataulut

Viikko 1 :

Mieti tiedon rakenne, tietokantaan tarvittavat taulut ja niiden sisältämät ominaisuudet.

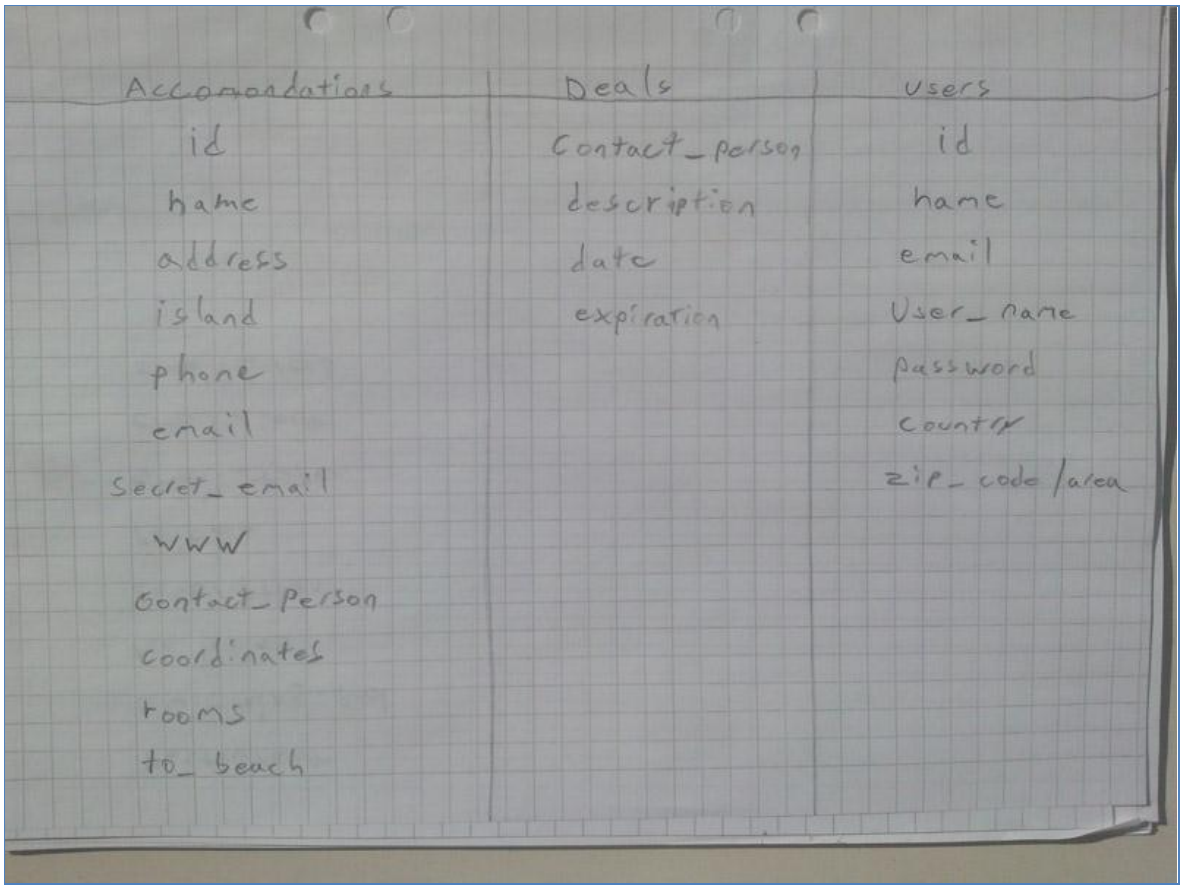
Tee ominaisuuksille vaadittavat tietotyyppimääritelmät ja mieti mahdollisia relaatioita taulujen välillä.

viikot 2-4:

Mieti mitä aiot kysyä paikanpäällä lomakohteissa. Mieti myös etukäteen miten otat tiedot talteen.

Täytetään taulut haastattelemalla majoituskohteiden työntekijöitä paikanpäällä. Siirrä tiedot tietokantaan saatujen muistiinpanojen mukaisesti.

Liite 2. Lyijykynä hahmotelma tietokannan tauluista



Liite 2. Jatkoa