TIETOJÄRJESTELMÄN TIETOKANNAN TOIMIVUUS JA JATKOKEHITYS

LAHDEN AMMATTIKORKEAKOULU
Tietojenkäsittelyn koulutusohjelma
Yritysviestintäjärjestelmät
Opinnäytetyö
Kevät 2007
Jari Korpela

Lahden ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma

Korpela, Jari: Tietojärjestelmän tietokannan toimivuus ja jatkokehitys

Yritysviestintäjärjestelmien opinnäytetyö, 46 sivua, 4 liitesivua

Kevät 2007

TIIVISTELMÄ

Tämän opinnäytetyön tarkoituksena oli parantaa lahtelaisen muotoilutoimiston Neodesign Oy:n käyttämän tietojärjestelmän tietokannan toimivuutta. Työ pohjautuu työharjoittelun aikana toteuttamaani järjestelmään yritykselle. Toteutetulla järjestelmällä yrityksen työntekijät pystyivät hallinnoimaan asiakas-, tuote- ja tilaustietoja. Tarvetta jatkokehitykseen ilmeni, koska tietokannan rakenne jäi puutteelliseksi tiukan aikataulun vuoksi. Työssä tarkastellaan puutteellisen tietokantarakenteen aiheuttamia ongelmia ja rajoituksia.

Teoriaosuudessa selvitetään relaatiotietokantoihin liittyvää teoriaa ja erilaisia tietokantojen suunnittelumenetelmiä. Erityisesti paneudutaan tietokannan toimivuuteen vaikuttaviin asioihin. Lisäksi käydään läpi SQL-kielen toimintoja sekä tietokannan hallintajärjestelmä MySQL.

Opinnäytetyön empiirinen osuus käsittää kvalitatiivisen tutkimuksen, jossa esitellään nykyisen tietokannan rakenne, jonka jälkeen sen toimivuus analysoidaan. Mahdollisten parannusten pohjana käytetään relaatiotietokantoja koskevaa teoriaa. Tutkimuksen tiedonkeruuaineisto koostuu tietokantoihin liittyvästä kirjallisuudesta, Internet-sivuista ja omista havainnoista. Lisäksi joitain tietokantaan liittyviä tulevaisuuden tietovaatimuksia on arvioitu yrityksen kanssa.

Tutkimuksen tulosten perusteella voidaan todeta, ettei tietokannan toteuttaja ole ymmärtänyt teorian merkitystä tietokannan suunnitteluvaiheessa. Tietokannan taulujen sisältö on sekavaa eikä niitä ole liitetty loogisesti toisiinsa. Puutteellinen tietokantarakenne voi myöhemmin aiheuttaa rajoituksia talletettaessa tietoa tietokantaan. Tietokannan toimivuutta voidaan parantaa hajoittamalla taulut osiin normalisoinnin avulla. Tietokannan valvontakielen hyödyntäminen auttaa tietokantaa toipumaan erilaisista virhetilanteista.

Avainsanat: Tietojärjestelmät, relaatiotietokannat, tietokantaohjelmat, SQL

Lahti University of Applied Sciences
Faculty of Business Studies

Korpela, Jari: Database functionality and development of the information system

Bachelor's Thesis in Business Information Systems, 46 pages, 4 appendices

Spring 2007

ABSTRACT

The purpose of this thesis was to improve the database functionality of the information system used by a Lahti-based design office Neodesign Oy. This thesis is based on a system which I had created for the company during my job training. With the created system the company's employees were able to manage information about customers, products and orders. Hovewer, there was need for further development because the database structure was left incomplete due to lack of time. In this thesis the problems and limits caused by the inadequate database structures are discussed.

The theory part discusses relational database theory and different database planning methods. Specifically the aspects that influence the functionality of the database are closely examined. In addition, the theory part introduces functions of SQL language and MySQL database management system.

The empirical part of the study consists of a qualitative study where the structure of the current database is introduced and its functionality analyzed. All the possible improvements are based on the theory of the relational database model -theory. As background information sources for this study, literature for relational database theory, my own conclusions and the internet were used. In addition, some database- related future requirements were evaluated with the the company.

The results showed that the creator of the database did not care about the importance of theory during database planning. The content of the database tables was often chaotic and the tables did not have a logical connection between them. The inadequate database structure might possibly later cause certain limits while saving new information into the database. The functionality of the database can be improved by disintegrating tables into smaller units with a method called normalization. Utilizing database control language helps the database recover from the critical errors.

Keywords: Information systems, relational databases, database systems, SQL

SISÄLLYS

1 JOHDANTO

Suunnittelin ja toteutin lahtelaiselle muotoilutoimistolle Neodesignille tietokantapohjaisen tietojärjestelmän keväällä 2005 työharjoittelun aikana. Yritys tarvitsi uuden järjestelmän nopeuttamaan tietojen hallinnointia sekä poistamaan siinä syntyviä virheitä. Valmiin sovelluksen avulla yrityksen työntekijät pystyivät hallinnoimaan mm. asiakas-, tuote-, ja tilaustietoja.

Järjestelmän suunnittelu aloitettiin yhdessä yrityksen työntekijöiden kanssa, mutta itselläni oli suurin vastuu tietokannan ja ohjelmoinnin toteutuksessa. Suunnittelu ja kehitystyö alkoi helmikuussa 2005 ja jatkui aina kesäkuuhun 2005 asti erilaisten päivitysten muodossa. Projektia varten muodostettiin ryhmä, joka vastasi järjestelmän tekniikasta sekä ideoinnista. Ryhmä koostui lähinnä mediatekniikan insinööreistä sekä web-suunnittelijoista.

Koska aikataulu oli varsin tiukka, kaikkea suunniteltua ei luonnollisesti saatu mukaan toteutettuun järjestelmään. Lisäksi järjestelmän tietokannan rakenne jäi osittain puutteelliseksi, mikä voisi myöhemmin aiheuttaa ongelmia yritykselle tietokannan eheyden ja laajennettavuuden kannalta. Koska tarkoituksenani oli oppia paremmin MySQL-tietokannan käyttöä, oli tietokannan toimivuuden kehittäminen täydellinen valinta opinnäytetyön aiheeksi.

1.1 Opinnäytetyön tutkimuskysymykset

Opinnäytetyö tutkii kuinka toteutetun tietokannan toimivuutta olisi mahdollista saada luotettavammaksi pidemmällä aikavälillä. Opinnäytetyö vastaa seuraaviin tutkimuskysymyksiin:

- Mitä puutteita toteutetussa tietokannassa ilmenee?
- Minkälaisia ongelmia puutteet voivat aiheuttaa yritykselle tulevaisuudessa?
- Millaisia menetelmiä tulisi käyttää tietokannan rakenteen parantamisessa?
- Mitä uusia kehitysideoita voidaan soveltaa nykyiseen tietokantaan ja sitä kautta tietojärjestelmään?

1.2 Opinnäytetyön tarkoitus ja tavoite

Opinnäytetyön tarkoituksena on selvittää kuinka Neodesignin nykyisen tietojärjestelmän tietokannan toimivuutta voidaan parantaa tulevaisuutta silmälläpitäen. Yrityksen nykyinen tietokanta on tarkoitus analysoida ja tehdä siihen korjaukset havaittujen puutteiden perusteella. Eräänä tavoitteena on tuoda esille myös konkreettisia esimerkkejä kestävänlaatuisesta tietokantasuunnittelusta, joita voidaan mahdollisesti hyödyntää muidenkin yritysten tarpeisiin. Tutkimus tehdään ohjelmistosuunnittelijan näkökulmasta.

1.3 Opinnäytetyön menetelmät ja aineisto

Tutkimus on luonteeltaan kvalitatiivinen. Tietokannan analysoinnissa selvitetään tietokannan toimintaa haittavaavia virheitä sekä paikannetaan ne. Mahdollisten parannusten pohjana käytetään relaatiotietokantoihin liittyvää teoriaa. Tutkimuksen lähdeaineisto koostuu relaatiotietokantoihin liittyvästä kirjallisuudesta, Internet-sivuista ja omista havainnoista. Lisäksi joitain tietokantarakenteeseen liittyviä vaatimuksia on arvioitu yrityksen kanssa.

1.4 Opinnäytetyön rajaus ja sisältö

Tutkimus rajautuu käsittelemään relaatiotietokantoihin liittyvää teoriaa. Tutkimuksessa ei käydä läpi tietojärjestelmän ohjelmointiin käytettyä PHP-kieltä ja käyttöliittymäsuunnittelua, koska niiden käsitteleminen ei olisi järkevää aiheiden laajuuden vuoksi. Lisäksi järjestelmään liittyvä tietoturva on jätetty pois.

Tutkimuksen sisältö on jaettu neljään lukuun (luvut 2-5), joiden jälkeen käydään lyhyt loppukeskustelu jossa vastataan tutkimuskysymyksiin. Luvussa 2 kerrotaan lyhyesti Neodesignin liiketoiminnasta, työympäristöstä sekä sen tuotteista ja palveluista. Luvussa 3 syvennytään järjestelmän toteuttamiseen käytettyyn tekniikkaan sekä järjestelmän eri osa-alueisiin.

Luvussa 4 käydään läpi relaatiotietokantoihin liittyvää teoriaa ja erilaisia tietokantojen suunnittelutapoja. Luvussa korostetaan erityisesti tietokantojen eheyteen ja toimivuuteen vaikuttavia asioita. Lisäksi esitellään tietokantojen käyttöön liittyvä SQL- kieli sekä tietokannan hallintajärjestelmä MySQL. Luvussa 5 käydään lyhyesti läpi toteutetun tietokannan suunnittelemiseen ja rakentamiseen liittyviä vaiheita jonka jälkeen nykyisen tietokannan rakenne analysoidaan. Analysoinnin aikana pyritään tuomaan esille tietokannan toimivuutta edistäviä parannusehdotuksia sekä mahdollisesti uusia kehitysideoita. Luvussa 6 vastataan tutkimuskysymyksiin. Opinnäytetyön keskeisistä käsitteistä on listaus liitteessä 1.

2 ASIAKASYRITYS

2.1 Sijainti ja tilat

Neodesign Oy on Lahdessa ja Helsingissä toimiva muotoilutoimisto, joka palvelee asiakkaitaan visuaalisen markkinointiviestinnän ja imago-tuotemuotoilun kentällä. Lahdessa vaikuttava toimisto sijaitsee Yrityshautomo ITU:n tiloissa, muiden pienten pk-yritysten läheisyydessä. Yrityksen henkilöstöllä on mahdollisuus pitää päivittäisiä palavereja toimiston vieressä sijaitsevassa neuvotteluhuoneessa. Lisäksi asiakkaiden on helppo ja nopea tavoittaa Neodesignin tuotteet ja palvelut, koska yritys sijaitsee lähellä keskustaa.

2.2 Henkilöstö

Yrityksen henkilöstö koostuu lähinnä muotoilijoista, web-suunnittelijoista ja mediatekniikan insinööreistä. Päivittäisiin tehtäviin kuuluvat mm. logojen suunnittelu, tuotteiden muotoilu, 3D-mallintaminen ja www-sivujen toteuttaminen. Neodesignin käyttämiin toimistosovelluksiin kuuluvat mm. Adobe Photoshop, Macromedia Flash sekä 3D Studio Max.

2.3 Palvelut ja tuotteet

Palveluista liikelahjatuotanto suunnittelee ja toteuttaa yhdessä asiakkaan kanssa sopivan liikelahjakokonaisuuden, imago- tai kampanjatuotteen. Tuotteen muotoilussa lähtökohtana ovat yrityksen toiminta, identiteetti sekä visuaalinen ilme. Muita palveluita ovat mm. graafinen suunnittelu, multimediasuunnittelu ja www-sivujen toteuttaminen.

Tuotteet suunnitellaan erityisesti yritysten henkilöstö- sekä liikelahjakäyttöön. Hyvällä lahjalla voidaan vahvistaa asiakassuhteita, parantaa yritysimagoa tai avata mahdollisesti keskusteluyhteyksiä uusien asiakkaiden välille. Tuotteet valmistetaan itse käsityönä korkealaatuisista materiaaleista. Neodesignin tuotemallistoon kuuluvat mm. erilaiset korut, vapaajan design-tuotteet sekä keittiö- ja kattaussarjat.

2.4 Asiakkaat

Asiakaskunta koostuu monilla eri toimialoilla vaikuttavista yrityksistä pk-sektorilta aina suuryrityksiin asti. Palveluita ovat käyttäneet mm. Helsingin Energia, Lahden ammattikorkeakoulu, Sibeliustalo, Uponor Suomi Oyj ja Nokia Oyj. Neodesignin toimintaperiaatteisiin kuuluu kokonaisvaltainen yhteistyö asiakkaan kanssa.

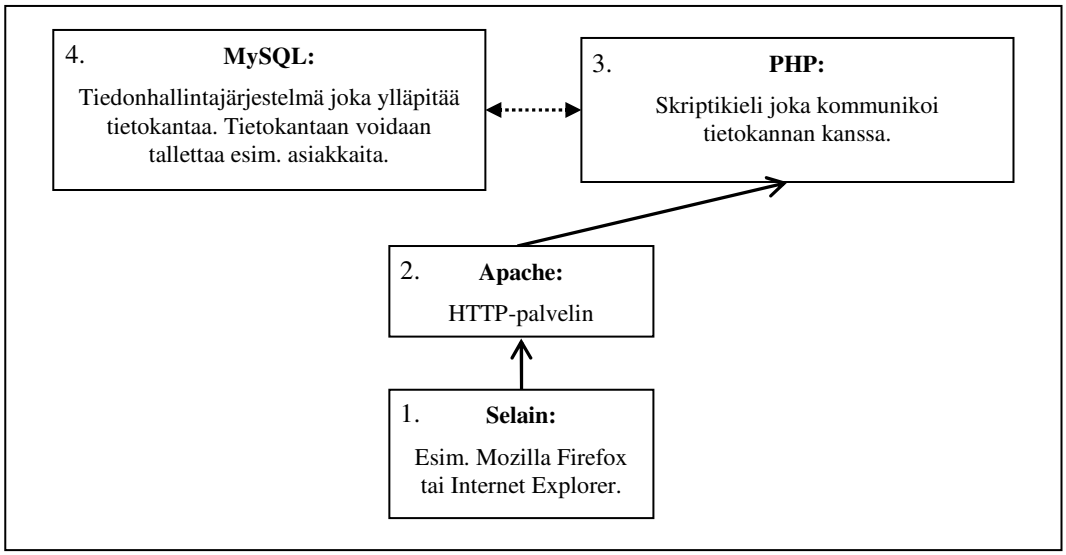
3 TOTEUTETTU JÄRJESTELMÄ LYHYESTI

3.1 Käytetty tekniikka

Tietojärjestelmän kehitys aloitettiin Apache HTTP-palvelimella, johon oli asennettu Linux-käyttöjärjestelmä (kuvio 1). Linuxia päätettiin hyödyntää Windowsin sijasta koska se on vapaalevitteinen ja luotettava etenkin palvelinkäytössä. Palvelimen asetuksiin ei ollut mahdollista tehdä muutoksia koska yrityksen palvelin oli vuokrattu Nebula Oy:ltä.

Tietokannan hallintajärjestelmäksi valittiin MySQL koska se oli myös ilmainen ohjelmisto. Vaikka MySQL ei vastannut kyselyiden suorituskyvyltä järempiä kaupallisia sovelluksia, oli se riittävän tehokas toteutetulle tietojärjestelmälle. Palvelinpuolen ohjelmointikieleksi valittiin PHP (Hypertext Preprocessor), koska se oli helppokäyttöinen ja tarjosi hyvän tuen valitulle tietokannan hallintajärjestelmälle.

Tietokantojen luontiin ja muokkaukseen käytettiin Putty-nimistä ohjelmaa, jolla oli mahdollista ottaa suojattu pääteyhteys Nebulan palvelimeen. Neodesign huolehti päävaltaisesti kaikista palvelinkustannuksista ja muista kuluista. Toteutettu järjestelmä koostuu useasta toisiinsa liitetystä www-sivusta (liite 2).



KUVIO 1. *Rakennekaavio yrityksen palvelimesta.*

3.2 Kalenteri ja ilmoitustaulu

Järjestelmän etusivulta löytyvät kalenteri ja ilmoitustaulutoiminnot, joiden tehtävänä on helpottaa erilaisten muistiinpanojen tallettamista. Yrityksen tapahtumat ja uutiset listautuvat ilmoitustaululle, joita niiden kirjoittaja tai järjestelmän ylläpitäjä voi myöhemmin muokata. Kalenteri ei ole sidottu ilmoitustauluun, ja sitä käytetään pääasiassa neuvotteluhuoneen ja videotykin varaamiseen. Edellä mainitut toiminnot parantavat yrityksen sisäistä informaation kulkua työntekijöiden välillä.

3.3 Asiakasrekisteri

Asiakasrekisteriin siirrytään etusivun linkkien kautta. Uuden asiakkaan lisäys aloitetaan ensin määrittelemällä alasvetovalikosta onko kyseessä henkilö vai yritys. Seuraavaksi valitaan yhteistyöryhmä kuten alihankkija johon asiakas voi mahdollisesti kuulua. Lopuksi yrityksen tai henkilön nimi kirjoitetaan vapaavalintaiseen kenttään ja suoritetaan tallennus tietokantaan. Tallennuksen jälkeen uusi asiakas päivittyy asiakasrekisteri-sivulla olevaan listaukseen. Näkymää on mahdollista suodattaa erilaisista alasvetovalikoista valitsemalla onko asiakas henkilö vai yritys tai kuuluuko se tiettyyn yhteistyöryhmään. Valitsemalla asiakkaan nimi listauksesta siirrytään joko henkilö- tai yrityskorttiin, jossa voidaan muokata mm. yhteys- ja lisätietoja.

3.4 Tuote- ja palvelurekisteri

Tuote- ja palvelurekisterin kautta voidaan lisätä uusia tuotteita ja palveluita tietokantaan. Tuotteelle tai palvelulle on mahdollista määrittää nimi, ominaisuudet ja kategoria johon se kuuluu. Tuotekategoria koostuu pääasiassa erilaisista koruista, keittiösarjoista ja vapaa-ajan tuotteista. Palvelukategoria pitää sisällään mm. liikelahjasuunnittelun ja www-sivujen toteuttamisen. Tietoja tuotteista tai palveluista on mahdollista muokata tai poistaa missä tahansa vaiheessa.

3.5 Tarjoushallinta

Tarjouksenteko aloitetaan luomalla uusi tarjous, tai vaihtoehtoisesti valitsemalla asiakas. Kun asiakas on valittu, siirrytään palvelu- ja tuotevalintavaiheeseen. Listasta poimitaan halutut tuotteet tarjoukseen ja niitä on mahdollista hakea myös tuotekoodien avulla. Jos tuotteet liittyvät johonkin palvelukokonaisuuteen kirjoitetaan palvelun nimi vapaamuotoiseen kenttään.

Kun tuotteet ja siihen mahdollisesti kuuluva palvelu on määritetty, päästään tarjoushallintasivulle. Sivulta määritellään tarjouksen otsikko, tarjouksen tehnyt myyjä sekä muut mahdolliset lisätiedot. Tarjous laskee tuotteista ja palveluista koostuvat yksikköhinnat automaattisesti. Tarjouksen tietoja voi toki palata muuttamaan missä tahansa vaiheessa. Lisäksi tarjouksia on mahdollista poistaa ja myyjä voi muuttaa manuaalisesti vielä kokonaishintaa tai vaihtoehtoisesti lisätä sellaisen tuotteen tarjoukseen jota ei löydy tuotelistoilta. Tarjous voidaan tarvittaessa myös tulostaa.

3.6 Tilaushallinta

Jos asiakas päättää ostaa tarjouksen sisältämät tuotteet tai siihen kuuluvat palvelut, tehdään siitä silloin tilaus tarjoushallintasivun kautta. Tilaus voidaan tarvittaessa peruuttaa eli muuttaa takaisin tarjoukseksi. Tilaushallintasivulta on mahdollista myös tulostaa tilausta koskevat sopimukset, laskut ja kuitit. Lisäksi on mahdollista syöttää tilausta koskevaa erilaista lisätietoa kuten toimituspäivä ja maksutiedot. Kerätyt tarjoukseen liittyvät tuotteet voidaan halutessa nähdä myös listauksena.

3.7 Ylläpitosivusto

Ylläpitosivustolta voidaan kontrolloida järjestelmän yleisiä ominaisuuksia. Uuden hallinnoijan lisääminen järjestelmään tapahtuu syöttämällä uuden hallinnoijan nimi ja sähköpostiosoite. Tunnukset ja salasasana lähetetään uudelle hallinnoijalle määritettyyn sähköpostiosoitteeseen.

Muita hallintasivujen ominaisuuksia ovat mm. asiakkaiden lopullinen poistaminen tietokannasta, kalenterin ja ilmoitustaulun tyhjentäminen sekä tietokannan varmuuskopionti.

3.8 Arkistointi

Kun kaikki tuotteet ja palvelut on kerätty ja tilaus on kokonaisuudessaan maksettu yrityksen tilille, on tilaus arkistokelpoinen. Valmiit tilaukset siirretään identtisiin tietokannan tauluihin myöhempää säilytystä varten. Passiivisia tilauksia ja tarjouksia voidaan myös siirtää arkistointiin väliaikaisesti, jolloin aktiivisten tilausten käsittely nopeutuu. Passiivisia tietoja on mahdollista myös myöhemmin palauttaa arkistosta takaisin aktiiviseksi.

4 TIETOKANTAMALLIN VALINTA

4.1 Tietokanta käsitteenä

Tietokanta on kokoelma tietoja, joilla on yhteys toisiinsa. Tietokanta voi esiintyä useassa eri muodossa kuten paperilla tai tietokoneen kiintolevyllä. Kuitenkin koneellisesti järjestetyn tietokannan edut ovat manuaaliseen nähden ilmeisiä: tiedonhaku on helppoa ja nopeaa. Tiedot on silloin tallennettu siten, että niitä voidaan hyödyntää useaan eri tarpeeseen nopeasti. Liikeyrityksissä paperitietokannoista pyritään pääsemään eroon virheiden vähentämiseksi ja liiketoimintaan kohdistuvien prosessien nopeuttamiseksi. Tietokoneella sijaitsevaa tietokantaa pitää yllä tietokannan hallintajärjestelmä (TKHJ). Tunnettuja esimerkkejä ovat mm. Oracle, DB2, MySQL ja Access. (Hovi, Huotari & Lahdenmäki 2003, 4-5.)

4.2 Tietokantojen tyypit

Tietokantojen tyypit voidaan jakaa käyttötietokantoihin ja analyyttisiin tietokantoihin. Käyttötietokantoja voidaan hyödyntää erilaisissa hallinnollisissa sovelluksissa, joissa tietoa talletetaan, päivitetään ja poistetaan. Tieto on tällöin dynaamista, koska se muuttuu jatkuvasti. Käyttötietokantoja voivat olla erilaiset asiakas- ja tuoterekisterit. Analyyttisen tietokannan data on aina staattista, mikä tarkoittaa että sitä ei ole mahdollista muokata uudelleen. Esimerkkejä analyyttisistä tietokannoista ovat tilasto- ja kyselykannat, joita liikeyritykset voivat käyttää analysoimaan yrityksen kehityksen suuntaa. (Hernandez 2000, 3-4.)

4.3 Varhaiset tietokantamallit

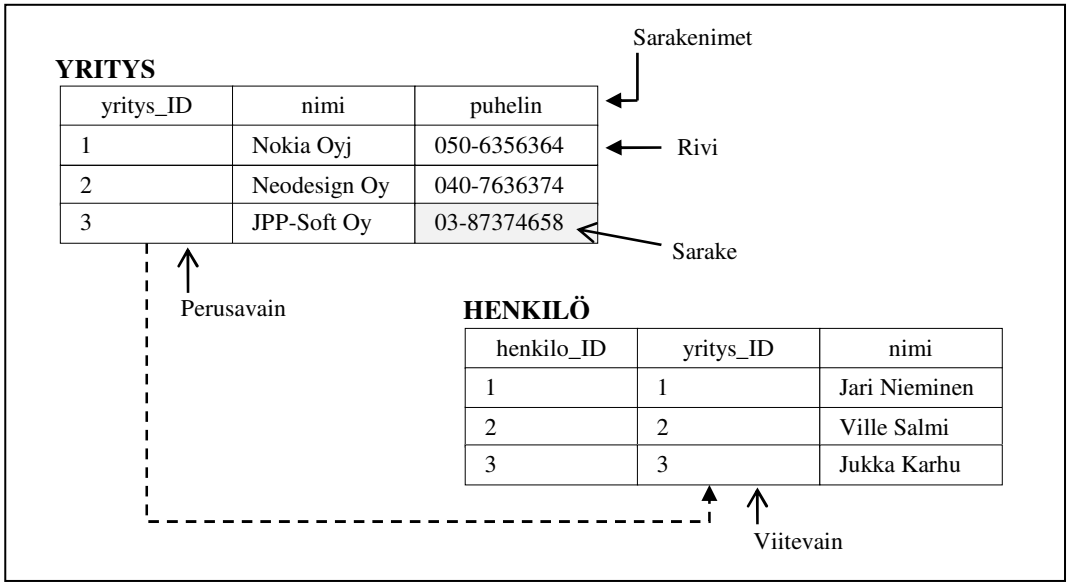
Hierarkkinen tietokantamalli muistuttaa ylösalaisin käännettyä puuta. Tässä tietokantamallissa tiedonhakeminen aloitetaan ylhäältä isätaulusta kohti alaspäin lapsitauluja. Hierarkkinen tietokantamalli kykenee erittäin nopeisiin tiedonhakuihin, mutta sen eheydessä ilmenee yleensä erilaisia ongelmia. Mallilla ei voida esittää kovin monimutkaisia rakenteita ja tietokannan rakenne sisältää usein tietojen turhaa toistoa. (Hernandez 2000, 4-8.)

Toinen varhaisista tietokantamalleista on verkkotietokantamalli, joka oli yritys korjata hierarkkisen mallin ongelmia. Verkkotietokantamalli muistuttaa hierarkkista tietokantamallia, ja se voidaan nähdä myös ylösalaisin käännettynä puuna. Tässä mallissa tietokannan välisiä yhteyksiä kuvataan erityisillä joukkorakenteilla jotka ovat läpinäkyviä rakenteita. Joukkorakenteen avulla kaksi taulua voidaan liittää yhteen siten, että toisesta tulee omistaja ja toisesta jäsen. Tässä mallissa käyttäjä voi aloittaa mistä tahansa taulusta ja kulkea joukkorakenteiden läpi jokaiseen suuntaan. (Hernandez 2000, 8-11.)

4.4 Relaatiotietokantamalli

Toteutetussa tietojärjestelmässä päädyttiin hyödyntämään IBM:n tutkijan Tri E. F. Codding 1960-luvun lopulla kehittämää relaatiotietokantamallia, koska varhaiset tietokantamallit eivät tarjonneet riittävää luotettavuutta. Relaatiotietokannat ovat helpompia käyttää kuin perinteisemmät hierarkkiset ja verkkomalliset tietokannat, koska käyttäjän ei tarvitse tietää tiedon fyysistä sijaintia voidakseen hakea sen tiedot. (Hernandez 2000, 11-12.)

Relaatiotietokantamallissa tieto talletetaan tauluihin (Table), jotka koostuvat sarakkeista tai kentistä (Column). Monesta rinnakkaisesta sarakkeesta muodostuu rivi tai tietue. Rivien ja sarakkeiden fyysisellä järjestyksellä ei ole tietoa haettaessa mitään merkitystä, mutta jokaisessa taulussa on oltava yksi uniikki perusavaimen arvo. Perusavain huolehtii siitä että tauluun ei tule tupla-arvoja. Viiteavaimella voidaan luoda useamman taulun välinen yhteys asettamalla niiden kesken jaetun kentän arvot täsmääväksi (kuvio 2). Alla olevassa kuviossa yritys-taulun perusavain yritys_ID on yhdistetty henkilö-taulun viiteavaimeen yritys_ID. Yksi henkilö voi kuulua yhteen yritykseen ja yhdessä yrityksessä voi olla useita henkilöitä. (Hovi ym. 2003, 7-9.)

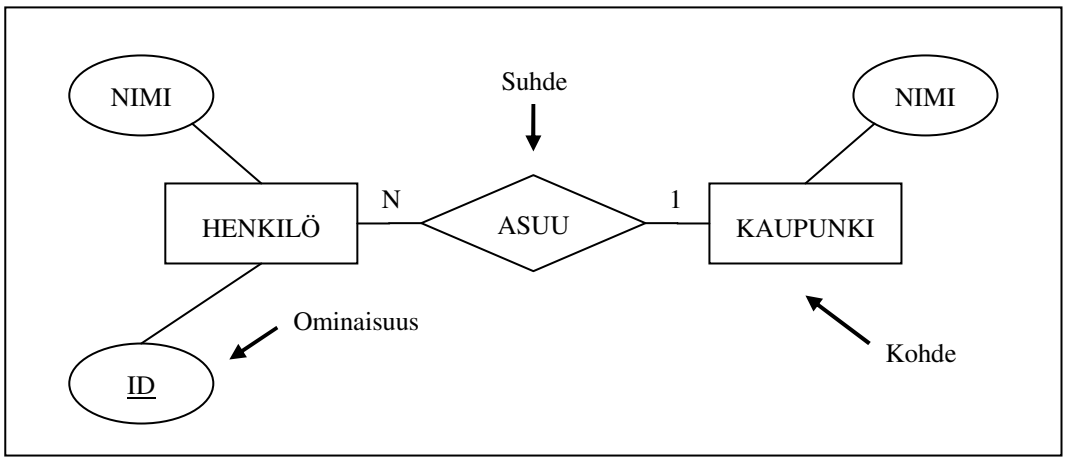


KUVIO 2. *Esimerkki relaatiotaulujen välisestä yhteydestä.*

4.5 Käsiteanalyysi

Käsiteanalyysi on ensimmäisiä vaiheita tietokannan suunnittelussa. Sen tavoitteena on kuvailla kokonaisvaltaisesti tietokantaan talletettavia tietoja ja taulujen välisiä suhteita. Lopputuloksena syntyy käsitemalli, joka kuvaa tietokannan fyysistä rakennetta. Käsiteanalyysiä käytetään yleensä IT-ammattilaisten ja liiketoiminnan edustajien kommunikaatiovälineenä. (Hovi ym. 2003, 32-33.)

ER-käsitemallimenetelmän (Entity Relationship modelling) esitteli Peter Chen v. 1976. Siinä kohteita kuvataan suorakulmioilla. Kohteita voivat olla henkilöt, tilat tai tuotteet. Kohteilla voi olla ominaisuuksia kuten nimi tai syntymäaika. Ominaisuutta kuvataan ellipsillä. Jos ominaisuus kuvaa taulun perusavainta se alleviivataan. Suhteella kuvataan useamman kohteen välistä toiminnallisuutta. Suhteet ovat timanttikuvioita ja ne ovat yleensä verbejä kuten koostuu tai kuuluu. Suhteeseen liittyvät kohteet yhdistetään viivalla ja jokaisen kohdalle merkitään joko 1 tai N, riippuen siitä onko kyseessä yhden suhde yhteen, yhden suhde moneen tai monen suhde moneen yhteys. Esimerkiksi kohteiden henkilö ja kaupunki välillä vallitsee suhde asuu (kuvio 3). Yhdessä kaupungissa voi siis asua monta henkilöä (1:N). Jos yksi henkilö omistaisi vain yhden auton olisi kyseessä yhden suhde yhteen riippuvuus (1:1). Jos henkilö omistaisi useita autoja jotka kuuluisivat usealle henkilölle, olisi kyseessä monen suhden moneen riippuvuus (N:M). (Hovi ym. 2003, 74-77.)



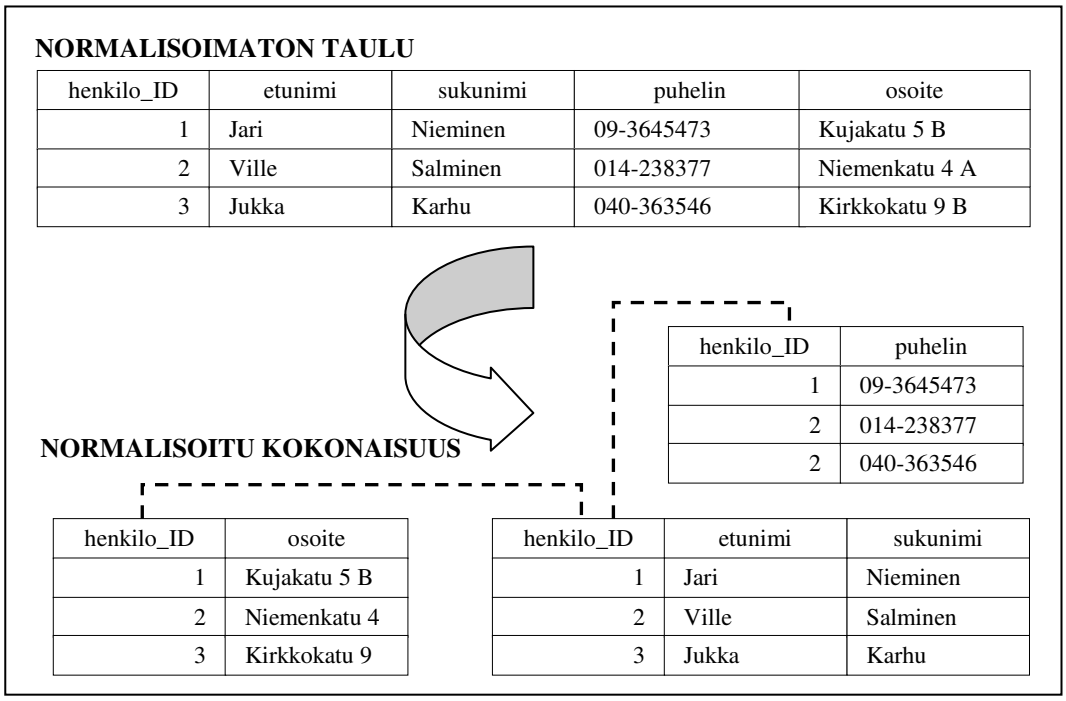
KUVIO 3. *Esimerkki yksinkertaisesta käsitemallista.*

4.6 Tietokannan eheys

Tietokannan eheys (integrity) tarkoittaa yhden tai usemman tietovaraston sisäistä ristiriidattomuutta. Tietokannan eheys vaarantuu jos esimerkiksi sama henkilö talletetaan kahteen kertaan. Eheyden saavuttamiseksi on laadittu eheyssääntöjä, joiden toiminnallisuus vaihtelee eri tietokannan hallintajärjestelmissä. Arvoalue-eheyden mukaan vain tietyn muotoinen arvo voidaan tallettaa tauluun. Tällä estetään esimerkiksi numeraalisen tiedon tallentaminen sellaiseen kenttään, joka voi sisältää ainoastaan merkkijonoja. Viite-eheys huolehtii taulujen välisten viittausten säilymisenä oikeana poistettaessa ja lisättäessä tietueita. Avaineheyden mukaan perusavaimen arvo ei saa olla tyhjä, eli toisin sanoen perusavain on pakollinen jokaisessa taulussa. (Hovi ym. 2003, 11-12; Hovi 2004, 9-10.)

4.7 Normalisointi

Normalisointi on vaiheittainen menetelmä, jota seuraamalla relaatiotietokannan rakenne saadaan tukemaan ehjää tallennusta. Normalisointi tehdään hajoittamalla taulut osiin taulujoukkioksi. Normalisoimattomasta taulusta erotellaan keskeiset käsitteet kuten henkilö, puhelin ja osoite. Jokaisesta käsitteestä luodaan oma itsenäinen taulunsa (kuvio 4). Normalisoinnin tasoja ovat 1-5 normaalimuodot sekä Boyce/Coddin normaalimuoto. Käytännön tietokantasovelluksissa voidaan pitää riittävänä, jos taulu täyttää ehdot kolmanteen normaalimuotoon (3NF) asti. Normalisointi vähentää saman tiedon uudelleen tallentamista ja parantaa tiedon laajennettavuutta tietokannassa tulevaisuudessa. (Hernandez 2000, 30-31.)



KUVIO 4. *Normalisointi kolmannen normaalimuodon mukaisesti.*

Ensimmäisessä normaalimuodossa jokainen kenttä määritellään ydintiedoksi. Samaan kenttään ei esimerkiksi kannata laittaa etu- ja sukunimeä, koska tiedon lajittelu vaikeutuu. Toisessa normaalimuodossa perusavain on pakollinen jokaisessa taulussa, jolloin jokainen kenttä taulussa riippuu täysin avaimesta. Kolmannen normaalimuodon mukaan taulun kaikkien kenttien tulee olla keskenään riippumattomia, jolloin minkään taulun sisältö ei saa perustua jonkin toisen taulun sisältöön. (Meloni 2003, 32-35.)

Denormalisoinnissa taulujoukkiot voidaan palauttaa yksinkertaisempaan muotoon käyttämällä pienempää normaalimuotoa. Tarvetta denormalisointiin voi esiintyä jos kyselyt tietokantaan ovat liian hitaita. Samalla vähennetään taulujen välisiä liitoksia jolloin tiedonhaku nopeutuu. Hintana tietokannan eheys voi vaarantua. (Hovi ym. 2003, 95-97.)

4.8 SQL

Relaatiotietokantojen hallinnan standardikieli on SQL (Structured Query Language) joka perustuu IBM:n tutkijan Tri E. F. Coddin kehittämään relaatiotietokantamalliin. SQL-kielellä voidaan tehdä relaatiotietokantaan erilaisia hakuja, muutoksia ja lisäyksiä. Kaikki keskeiset relaatiotietokantoja hyväkseen käyttävät ratkaisut hyödyntävät SQL:ää jollain tapaa. Sillä kerrotaan mitä tietoja tietokannasta haetaan, ei sitä kuinka ne haetaan. (Hovi ym. 2003, 10.)

4.8.1 Tietokannan käsittelykieli (DML)

Tietokannassa sijaitsevia tietoja käsitellään SELECT-, INSERT-, UPDATE- ja DELETE- käskyillä. SELECT-käskyllä voidaan valita mitä haetaan tauluista ja näitä hakutoimenpiteitä kutsutaan usein tietokantakyselyiksi. INSERT-käskyllä voidaan tallentaa uutta dataa tauluihin ja UPDATE-käskyllä päivittää jo olemassaolevaa tietoa. DELETE-käskyllä voidaan pyyhkiä yhden rivin tai koko taulun tiedot kerralla. (Hovi 2004, 14-16.)

Alla olevassa esimerkissä (listaus 1) haetaan kentät etunimi ja sukunimi asiakas-nimisestä taulusta siltä riviltä, jossa asiakkaan perusavain ID saa arvon 5. Lisäksi haetut kentät järjestetään nousevaan järjestykseen sukunimen mukaan.

```
SELECT etunimi, sukunimi FROM asiakas WHERE  ID = 5 ORDER BY sukunimi ASC;
```

LISTAUS 1. *Syntaksi rivien hakemiseen taulusta.*

4.8.2 Tietokannan määrityskieli (DDL)

Tietokannan määrityskieli pitää sisällään CREATE-, ALTER- ja DROP- käskyt. CREATE-käskyn avulla voidaan luoda uusi taulu tietokantaan. ALTER-käskyllä voidaan tehdä rakenteellisia muutoksia valmiiseen tauluun rivitasolla. DROP-käskyä käytetään taulun poistamiseen pysyvästi tietokannasta.

Taulun luonnin yhteydessä määritellään myös taulun sisältämät kentät ja niitä vastaavat tietotyypit. Esimerkiksi etunimi on kenttä ja sen tietotyyppi on VARCHAR eli merkkijono (listaus 2). Suluissa olevat arvot kertovat kuinka monta merkkiä kukin tietotyyppi voi varata. Lopuksi määritellään taulun perusavain PRIMARY KEY sekä taulumoottori TYPE. (Hovi 2004, 102-121.)

```
CREATE TABLE asiakas (
asiakas_ID INT AUTO_INCREMENT, etunimi VARCHAR (25), sukunimi VARCHAR(25),
PRIMARY KEY (asiakas_ID)
) TYPE= MyISAM);
```

LISTAUS 2. *Syntaksi taulun luontiin tietokantaan.*

4.8.3 Tietokannan valvontakieli (DCL)

Tietokannan valvontakieli suojelee tietokantaa yllättäviltä tilanteilta kuten systeemivialta. Käyttöoikeuksien luomiseen taulutasolla käytetään GRANT- ja REVOKE-käskyjä. Transaktiokielen hallintaan käytetään BEGIN-, COMMIT- ja ROLLBACK-käskyjä. Transaktio on joukko toisiaan seuraavia kyselyitä jotka suoritetaan yhtenäisenä joukkona. Transaktio ei toteudu, jos kaikki sen yksittäiset operaatiot eivät ole toteutuneet. Hyvä esimerkki transaktiosta on pankkitapahtuma, jossa halutaan siirtää rahaa 100 euroa toiselle tilille (listaus 3). Jotta voidaan tallettaa rahat yhdelle tilille, on ensin otettava toiselta tililtä rahaa. Jos ensimmäiseltä tililtä ei löydy 100:aa euroa, peruutetaan koko tapahtuma. Transaktio alkaa aina BEGIN-käskyllä ja loppuu COMMIT-käskyyn. Virheen sattuessa tapahtuma peruuntuu automaattisesti ROLLBACK-käskyllä. (Meloni 2003, 227-229; Hovi 2004, 146-149.)

```
BEGIN;
        UPDATE tili SET rahanmaara = rahanmaara – 100 WHERE ID = 1;
        UPDATE tili SET rahanmaara = rahanmaara  + 100 WHERE ID = 2;
COMMIT;
```

LISTAUS 3. *Esimerkki transaktion toiminnasta.*

Transaktioihin kuuluvat myös taulujen ja rivien lukitustasot. Lukituksen tarkoituksena on estää tietyn rivin tai taulun muokkaus kesken transaktioiden. Esimerkiksi jos varastossa on vähemmän tuotteita kuin käyttäjä haluaa tilata, transaktio peruutetaan ja lukko avataan. Jos saatavuus vastaa käyttäjän tilaamaa määrää, transaktiota jatketaan ja lisätään käyttäjän tekemät tilaukset tauluun. (Hovi 2004, 131-132.)

Transaktiot tarjoavat lisäksi viite-eheyden valvontaan tarkoitettuja toimintoja, joilla voidaan määrittää sääntöjä ja rajoitteita siitä, kuinka tiedon muokkauksen yhteydessä tulisi toimia. Säännöt luodaan CREATE-käskyn yhteydessä. RESTRICT-määreellä voidaan estää esimerkiksi sellaisen oppilaan poisto jolla on arviointeja jo muissa tauluissa. CASCADE-määreellä voidaan poistaa kaikki opiskelijalle kuuluvat arvioinnit jos itse opiskelija poistetaan. Oppilas on siis isätaulu (listaus 4) ja arviointi lapsitaulu (listaus 5). Säännöt kuinka muokkauksen yhteydessä toimitaan kirjoitetaan siis aina lapsitauluihin. (MySQL Reference Manual 2007.)

```
CREATE TABLE opiskelija  (
opiskelija_ID INT NOT NULL,  nimi VARCHAR (25),  PRIMARY KEY (opiskelija_ID)
) ENGINE = InnoDB;
```

LISTAUS 4. *Isätauluun talletetaan kaikki opiskelijat.*

```
CREATE TABLE arviointi  (
arviointi_ID INT NOT NULL, arviointi VARCHAR (25),
INDEX opiskelija_index (opiskelija_ID),
REFERENCES opiskelija (opiskelija_ID),
FOREIGN KEY (opiskelija_ID),
ON DELETE CASCADE
)  ENGINE = InnoDB;
```

LISTAUS 5. *Jos opiskelija poistetaan isätaulusta, poistuu lapsitaulusta automaattisesti opiskelijalle kuuluvat arvioinnit. Määreillä INDEX, REFERENCES, ja FOREIGN KEY viitataan isätauluun. Määre ON DELETE CASCADE on sääntö, joka tarkoittaa että arvioinnit poistetaan jos oppilas pyyhitään.*

4.9 MySQL

MySQL:ää päätettiin hyödyntää tietojärjestelmässä koska se täytti erinomaisesti valintaperusteet. Se on ilmainen ja suorituskykyinen SQL-tietokannan hallintajärjestelmä, joka on kirjoitettu käyttäen C++ kieltä. Sitä kehittää ruotsalais-suomalainen yritys MySQL AB. MySQL on helppo asentaa ja ylläpitää. Se ei vaadi täyspäiväistä huoltoa kuten kalliit kaupalliset tietokantaohjelmistot. Sillä on tällä hetkellä maailmanlaajuisesti yli 4 miljoonaa käyttäjää. (Hovi 2004, 4.)

4.9.1 Indeksointi

Indeksiä voidaan verrata kirjassa sijaitsevaan hakemistoon, joka luonnollisesti nopeuttaa tietojen löytymistä. Indeksointi kirjoitetaan taulun luonnin yhteydessä tai jälkeenpäin. Taulun ensijainen avain (Primary key) indeksoidaan aina automaattisesti päähakemistoon. Muut kentät voidaan tarvittaessa indeksoida oheishakemistoon antamalla ensin indeksille nimi ja laittamalla perään sulkuihin indeksoitavan kentän nimi. Myös useamman kentän voi indeksoida samaan indeksiin. Indeksit voivat olla sekä hyödyllisiä että haitallisia. Jos tietoa talletetaan vain arkistointiin on indeksointi tarpeetonta. Sellaiset kentät kannattaa indeksoida joita tullaan käyttämään hakukriteereinä usein tietokannoissa. Alla tarvittava syntaksi (listaus 6) indeksin kirjoittamiseen taulun luonnin yhteydessä.
(Hovi 2004, 143-145.)

```
INDEX indeksin nimi (indeksoitavan kentän nimi);
```

LISTAUS 6. *Indeksin luontiin tarvittava syntaksi.*

4.9.2 Taulutyypit

Taulutyyppejä (Table types) kutsutaan varastomoottoreiksi (Storage engines). Ne antavat pohjan tiedolle jota pidetään tauluissa. Jokaisella taulutyypillä on luonnollisesti etunsa mutta myös heikkoutensa. MySQL tukee erilaisia taulutyyppejä joita ovat mm. BerkeleyDB, InnoDB ja MyISAM. Mikäli taulutyyppi tukee transaktioita, niin se tarjoaa yleensä enemmän turvallisuutta. Alla esimerkkejä erilaisista taulutyypeistä. (MySQL Reference Manual 2007.)

Memory

Memory-taulut eroavat muista tauluista siinä, että niiden sisältämä tieto talletetaan ainoastaan tietokoneen muistiin. Mikäli MySQL-järjestelmä käynnistetään uudelleen, tyhjentyvät taulujen tiedot, mutta taulujen rakenteelliset tiedot säilyvät ehjinä kiintolevyllä. Memory-taulut tukevat silppuindeksejä (Hash index). Memory-taulussa voi olla 32 indeksiä ja yhteisindeksi voi sisältää enintään 16 kenttää. (MySQL Reference Manual 2007.)

BerkeleyDB

Berkeley DB:n (BDB) on tehnyt Sleepycat Software, joka on transaktionaalinen taulutyyppi ja tarjoaa sivulukituksen. Se ei ole kuitenkaan optimoiduin tarjolla olevista tyypeistä. BDB-taululle täytyy luoda aina perusavain, muuten MySQL tekee sen itse käyttäen piilotettua kenttää. BDB-taulussa voi olla 31 indeksiä ja yhteisindeksi voi sisältää enintään 16 kenttää. (Meloni 2003, 231; MySQL Reference Manual 2007.)

MyISAM

MySQL:n oletustaulutyyppi on MyISAM, joka tulee sanoista Index Sequential Access Method. MyISAM ei sisällä tukea transaktioille, mutta siitä löytyy tuki taulutason lukitukselle. MyISAM sisältää muutamia suorituskykyä parantavia ominaisuuksia.

Käytetyimmät avaimet on mahdollista tallentaa avainvälimuistiin. Lisäksi se pystyy pakkaamaan indeksin kokoa jos indeksoitava kenttä on tietotyypiltään merkkijono. MyISAM-taulussa voi olla enintään 64 indeksiä ja yhteisindeksi voi sisältää 16 kenttää. (MySQL Reference Manual 2007.)

InnoDB

InnoDB on Innobase Oy:n kehittämä suositumpi ja vakaampi transaktioihin soveltuva taulutyyppi. Jos taululle ei luoda perusavainta, valitsee InnoDB vain uniikkeja arvoja sisältävän kentän perusavaimeksi. InnoDB tukee ryväindeksejä (Clustered Index), jotka voidaan tehdä perusavaimille. Lisäksi se kykenee huomaamaan transaktioiden mahdolliset lukkeutumat. Mikäli hakuja halutaan nopeuttaa, niin taulun kentälle on mahdollista tehdä toisioindeksi (Secondary Index). (Meloni 2003, 232; MySQL Reference Manual 2007.)

4.9.3 Kenttien tietotyypit

MySQL tukee mm. seuraavanlaisia numeerisia tietotyyppejä. Jos tietotyypissä käytetään määrettä UNSIGNED, niin lukujen viemä negatiivinen tila poistetaan ja vain positiivisia lukuja lasketaan eteenpäin. (Meloni 2003, 78-79; MySQL Reference Manual 2007.)

- TINYINT on hyvin pieni kokonaisluku. Mikäli kentässä käytetään määrettä UNSIGNED, niin kenttä voi sisältää arvoja väliltä 0 – 255. Muuten väliltä (-128) – 127. Kenttä vie tilaa noin 1 tavua.

- SMALLINT on pieni kokonaisluku. Mikäli kentässä käytetään määrettä UNSIGNED, niin kenttä voi sisältää arvoja väliltä 0 – 65535. Muuten väliltä (-32768) – 32767. Kenttä vie tilaa noin 2 tavua.

- MEDIUMINT on keskikokoinen kokonaisluku. Mikäli kentässä käytetään määrettä UNSIGNED, niin kenttä voi sisältää arvoja väliltä 0 – 16777215. Muuten väliltä (-8388608) – 8388607. Kenttä vie tilaa noin 3 tavua.

- INT on normaalikokoinen kokonaisluku. Mikäli kentässä käytetään määrettä UNSIGNED, niin kenttä voi sisältää arvoja väliltä 0 – 4294967295. Muuten väliltä (-2147483648) – 2147483647. Kenttä vie tilaa noin 4 tavua.

- BIGINT on suuri kokonaisluku. Mikäli kentässä käytetään määrettä UNSIGNED, niin kenttä voi sisältää arvoja väliltä 0 – 18446744073709551615. Muuten väliltä (-9223372036854775808) – 9223372036854775807. Kenttä vie tilaa noin 8 tavua.

MySQL tarjoaa lisäksi tuen mm. seuraavanlaisille aikaa ja päiväyksiä ilmaiseville tietotyypeille. (Meloni 2003, 79-80; MySQL Reference Manual 2007.)

- DATE on päiväys väliltä ’1000-01-01’ – ’9999-12-31’. Kenttä vie tilaa noin 3 tavua.

- DATETIME on päiväys ja aika väliltä ’1000-01-01 00:00:00’ – ’9999-12-31 23:59:59’. Kenttä vie tilaa tilaa noin 8 tavua.

- TIME on aika väliltä '-838:59:59' - '838:59:59'. Kenttä vie tilaa noin 3 tavua.

- TIMESTAMP(M) on aikaleima, jossa arvo M viittaa siihen kuinka tarkasti aika esitetään. Kenttä vie tilaa noin 4 tavua.

- YEAR on vuosiluku. Jos kentän pituudeksi on määritelty 2, niin aika on väliltä 1970 - 2069. Jos pituudeksi on määritelty 4, niin aika on väliltä 1901 – 2155. Kenttä vie tilaa noin 1 tavun.

Numeeristen ja aikaa ilmaisevien tietotyyppien lisäksi MySQL tarjoaa tuen mm. seuraaville merkkijonoille. (Meloni 2003, 80-81; MySQL Reference Manual 2007.)

- CHAR(M) on kiinteämittainen merkkijono, jossa arvo M voi olla arvojen 0 – 255 välillä. Maksimipituudeltaan 255 merkin kenttä vie tilaa noin 255 tavua.

- VARCHAR(M) on muuttuvamittainen merkkijono. Arvo M eli merkkijonon maksimipituus voi olla väliltä 0 – 255. Kenttä varaa aina yhden ylimääräisen tavun, joten 10 merkkiä sisältävä kenttä vie 11 tavua.

- TINYBLOB tai TINYTEXT ovat muuttuvanmittaisia tekstikenttiä maksimissaan 255 merkkiin asti. Kenttä varaa aina yhden ylimääräisen tavun, joten 10 merkkiä sisältävä kenttä vie 11 tavua. Maksimissaan kenttä vie noin 256 tavua.

- BLOB ja TEXT ovat muuttuvanmittaisia tekstikenttiä 16777215 merkkiin asti. Kenttä varaa aina 3 ylimääräistä tavua, joten 10 merkkiä sisältävä kenttä vie 13 tavua. Maksimissaan kenttä vie noin 16 megatavua.

- MEDIUM BLOB ja MEDIUMTEXT ovat muuttuvanmittaisia tekstikenttiä 16777215 merkkiin asti. Kenttä varaa aina 3 ylimääräistä tavua, joten 10 merkkiä sisältävä kenttä vie 13 tavua. Maksimissaan 16 megatavua.

- SET (’arvo1’,’arvo2’,..) on merkkijono-objekti, jonka sisällöksi voidaan määrittää yksi, useampi tai ei yhtään määritetyistä arvoista. Näitä arvoja voi olla enintään 64. Kenttä vie tilaa arvojen maksimimäärällä noin 8 tavua.

- ENUM (’arvo1’,’arvo2’,..) on merkkijono-objekti, jonka sisällöksi voidaan valita vain yksi määritellyistä arvoista. Näitä arvoja voi olla maksimissaan 65535. Kenttä vie arvojen maksimimäärällä noin 1 - 2 tavua.

5 TIETOKANNAN TOTEUTUS JA TOIMIVUUDEN ANALYSOINTI

5.1 Yrityksen vanha systeemi

Yrityksen vanha systeemi oli kirjoittaa tilaukset ja tarjoukset tietokoneen kansioihin Adobe Acrobat-dokumenteiksi nimeämällä ne asiakkaan mukaan. Tarjouksia ja tilauksia koskevat myyntihinnat jouduttiin käsin etsimään erilaisista katalogeista ja ne laskettiin lopulta laskimella yhteen. Asiakkaille annettujen tilaus- ja tarjoustulosteiden kopioita säilytettiin yleensä niille varatuissa mapeissa aakkosjärjestyksessä.

Alkuperäisen systeemin toimivuutta vaikeuttivat lukuisat seikat. Yrityksen asiakkaat saattoivat olla samanimisiä henkilöitä. Lisäksi samalla asiakkaalla saattoi olla lukuisia tilauksia ja tarjouksia. Oikean dokumentin löytäminen oli luonnollisesti tämän takia työlästä ja aikaa vievää. Myyntihintojen laskeminen käsin aiheutti usein inhimillisiä virheitä ja siihen kului paljon aikaa.

5.1.1 Uuden systeemin vaatimusten määrittely

Uudelle systeemille asetettiin vaatimukseksi parantaa useita yrityksen asiakaspalvelulle ja toiminnalle tärkeitä asioita. Uuden järjestelmän tuli tarjota toimiva asiakas- ja tuoterekisteri. Lisäksi sen tuli vähentää tai poistaa kokonaan tilausten ja tarjousten yhteydessä syntyvät virheet, sekä nopeuttaa tehtyjen tarjous- ja tilaustietojen löytymistä. Työntekijöiden välistä informaationkulkua tuli parantaa toteuttamalla kalenteri ja ilmoitustaulu-toiminto, joilla olisi mahdollista tehdä mm. videotykin ja neuvotteluhuoneen varauksia sekä kirjoittaa ilmoitusluontoisia asioita.

5.1.2 Suunnittelun lähtökohdat

Ensimmäisissä tietokannan rakenteen suunnittelupalavereissa järjestelmästä muodostui 5 vahvaa käsitettä jotka olivat asiakkaat, tuotteet, palvelut, tilaukset ja tarjoukset. Muodostuneet käsitteet pysyivät hyvin samanlaisina työn alusta loppuun saakka saaden työn edetessä hieman erilaisia merkityksiä. Tietokannan rakenteesta yritettiin rakentaa mahdollisimman yleiskäyttöinen. Koska yrityksellä oli myös yhteystyökumppaneita kuten alihankkijoita ja muita organisaatioita, päätettiin niille luoda yhteinen käsite yhteistyöryhmät.

Neodesignin työntekijät ja heidän henkilötietonsa päätettiin pitää omassa kokonaisuudessaan. Jokaisella työntekijällä oli mahdollisuus tehdä asiakkaalle tarjous. Tarjous saattoi koostua tuotteista tai palveluista. Jos asiakas hyväksyi tarjouksen siitä syntyi tilausvahvistus, jonka jälkeen laskettiin lasku. Lopuksi tuote tai palvelu toimitettiin asiakkaalle.

5.1.3 Tietokannan taulut

Toteutettu tietokanta rakentuu erilaisista tietokokonaisuuksista, jotka koostuvat vaihtelevasta määrästä tauluja. Tietokokonaisuuksia ovat asiakasrekisteri, palvelurekisteri, tuoterekisteri, tarjousrekisteri ja tilausrekisteri. Lisäksi kalenterille ja ilmoitustaululle löytyvät omat taulunsa. Tietokokonaisuudet ovat yhdistetty toisiinsa kuitenkin eritavoilla. Tietokannan taulut ovat kaikki toteutettu MyISAM-tyyppisinä, koska transaktioiden käyttöön ei ollut tarvetta. MyISAM-taulujen etuina verrattuna InnoDB-tauluihin on niiden nopeus, vaikka ne eivät tarjoakaan samanlaista tietojen eheyttä.

Aktiivisten taulujen lisäksi tietokannasta löytyy passiivisia tauluja, joita käytetään tilaus- ja tarjoustietojen arkistointiin. Passiiviset tilaus- ja tarjoustaulut ovat identtisiä aktiivisten kanssa. Kun aktiivisiin tauluihin kertyy paljon vanhentunutta tietoa, on sitä mahdollista siirtää passiivisiin tauluihin tilan lisäämiseksi jolloin tietojen prosessointi nopeutuu ja selkeytyy.

Tosin passiivista tietoa on mahdollista palauttaa myöhemmin aktiiviseksi jos siihen on tarvetta. Ennen taulujen fyysistä toteuttamista tietokannasta piirrettiin käsitemalli kokonaisuuden hahmottamiseksi (liite 3).

Liitteestä voidaan todeta, että yksi asiakas voi liittyä yhteen tai useampaan tarjoukseen, mutta yksi tarjous voi sisältää vain yhden asiakkaan tiedot, jolloin kyseessä on yhden suhde moneen yhteys (1:N). Samoin yksi myyjä voi tehdä useamman tarjouksen, mutta yksi tarjous voi kuulua vain yhdelle myyjälle. Yksi tuote tai palvelu voi liittyä useampaan tarjoukseen, mutta yhteen tarjoukseen voi kuulua vain yksi tuote tai palvelu. Tarjouksesta tehdään aina yksi tilaus, mutta yhteen tilaukseen voi liittyä vain yksi tarjous, jolloin kyseessä on yhden suhde yhteen yhteys (1:1).

5.1.4 Asiakasrekisteri

Asiakasrekisteri rakennettiin yhdellä asiakas-nimisellä taululla (kuvio 5), johon kuuluivat seuraavat kentät ja tietotyypit:

ASIAKAS	
nimi	**tietotyyppi**
asiakas_ID	INT PK auto_increment
toimiala	VARCHAR (50)
etunimi	VARCHAR (25)
sukunimi	VARCHAR (25)
yritys	VARCHAR (50)
y-tunnus	VARCHAR (25)
yhteistyoryhma	VARCHAR (25)
puhelin1	VARCHAR (25)
puhelin2	VARCHAR (25)
osoite	VARCHAR (25)
email	VARCHAR (80)
pankkinumero	VARCHAR (25)
asiakastyyppi	SET (’henkilo’,’yritys’)

KUVIO 5. *Asiakas-taulu.*

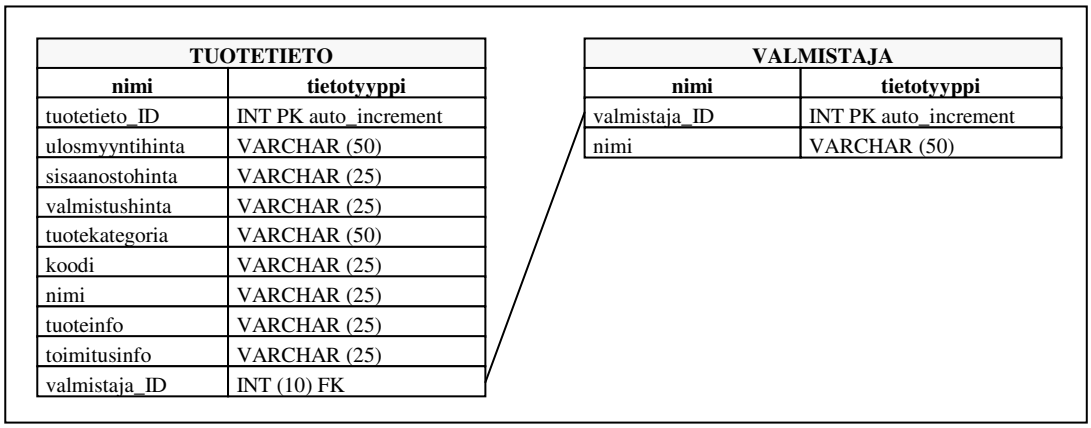
Kuvan taulusta voidaan todeta, että perusavaimen sisältävä kenttä on nimeltään asiakas_ID, jota vastaa kirjainyhdistelmä PK. Määre nimeltään auto_increment pitää huolen siitä, että jokaisella uudella lisätyllä asiakkaalla on aina uniikki perusavaimen arvo. Taulun pakolliset kentät ovat etunimi, sukunimi ja yhteistyöryhmä. Yhteistyöryhmä voi olla esimerkiksi alihankkija tai muu sidos. Vaihtoehtoisesti jos asiakas ei ole henkilö täytetään kentät yritys, toimiala ja y-tunnus. Jokaisen uuden asiakkaan luonnissa asiakastyyppi-kentälle voidaan asettaa arvot henkilö tai yritys, joka toimii siis tunnisteena sille minkä tyyppinen asiakas on kyseessä.

Muut yhteystietoja kuvaavat kentät kuten osoitteet ja puhelinnumerot voidaan vapaavalintaisesti jättää tyhjäksi, mikäli asiakas ei halua jättää itsestään enempää tietoa. Kaikki kentät käyttävät VARCHAR-tietotyyppiä joka tarkoittaa merkkijonoa. Tietotyyppien varaamat merkkien määrät kuitenkin vaihtelevat kenttäkohtaisesti.

Asiakasrekisteri on tärkeä osa tietojärjestelmää, sillä se on yhteydessä moniin muihin rekistereihin kuten tarjouksiin, tilauksiin ja laskuihin. Asiakasrekisteri liittyy muihin rekistereihin viiteavaimen asiakas_ID avulla. Jos asiakkaan perusavaimen arvo täsmää muissa rekistereissä esiintyviin asiakkaan viiteavaimiin, kuuluu kyseisten rekisterien ominaisuudet aina asiakkaalle.

5.1.5 Tuoterekisteri

Tuoterekisteri toteutettiin kahdella toisiinsa liitetyllä taululla (kuvio 6). Taulujen nimet olivat tuotetieto ja valmistaja. Tuoterekisteri on jatkuvasti yhteydessä tarjousrekisteriin. Uuden tarjouksen luonnissa asiakkaalle voidaan tarjousrekisteriin mahdollisesti liittää erilaisia tuotteita tuoterekisteristä.



TUOTETIETO	
nimi	**tietotyyppi**
tuotetieto_ID	INT PK auto_increment
ulosmyyntihinta	VARCHAR (50)
sisaanostohinta	VARCHAR (25)
valmistushinta	VARCHAR (25)
tuotekategoria	VARCHAR (50)
koodi	VARCHAR (25)
nimi	VARCHAR (25)
tuoteinfo	VARCHAR (25)
toimitusinfo	VARCHAR (25)
valmistaja_ID	INT (10) FK

VALMISTAJA	
nimi	**tietotyyppi**
valmistaja_ID	INT PK auto_increment
nimi	VARCHAR (50)

KUVIO 6. *Tuote-taulu, johon on liitetty valmistaja-taulu.*

Kuviosta voidaan todeta, että tuotetieto-taulun perusavain on tuotetieto_ID. Valmistaja-taulu liittyy tuotetieto-tauluun viiteavaimen valmistaja_ID avulla, jota vastaa kirjainyhdistelmä FK. Tuotetieto-taulu koostuu siis erilaisista tuotteista, joilla usealla voi olla sama valmistaja.

Taulujen kenttien tietotyypit ovat asiakasrekisterin tapaan VARCHAR-tietotyyppiä. Koodi-kenttä vastaa yleensä tuotteiden tehdaskoodeja. Erilaiset hinta-kentät kertovat tuotteeseen kohdistuvasta hinnoittelusta sen valmistuksen, sisäänoston tai ulosmyynnin yhteydessä. Tuoteinfo-kenttään voidaan merkitä informatiivisia lisätietoja tuotteesta kuten väri, mitat tai paino. Tuotekategoria-kenttään voidaan vapaavalintaisesti määrittää mihin tuoteryhmään kyseinen tuote mahdollisesti kuuluu. Yleisimpiä tuoteryhmiä ovat mm korut, design-tuotteet ja keittiökalusteet. Toimitusinfo-kenttään voidaan merkitä vapaavalintaisesti lisäinformaatiota toimitukseen liittyvästä aikataulusta, maksutavasta ja toimitustavasta. Lopuksi nimi-kenttään kirjoitetaan tuotteen nimi.

5.1.6 Palvelurekisteri

Palvelurekisteri koostuu yhdestä palvelutieto-taulusta (kuvio 7). Palvelutieto-taulun perusavain on palvelutieto_ID. Nimi-kenttä kertoo palvelun nimen ja palveluinfo-kenttä sisältää yleensä palveluun liittyvää lisäinformaatiota.

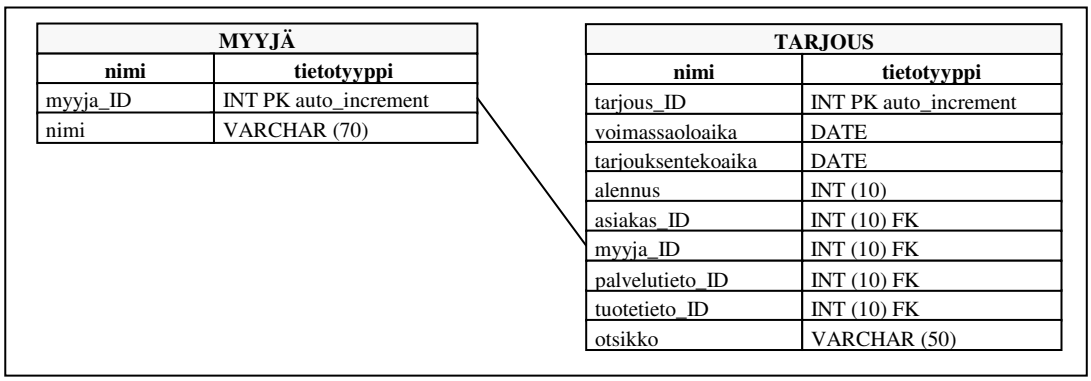
Palvelutyyppejä ovat mm. websuunnittelu, liikelahjatuotanto ja graafinen suunnittelu. Palveluihin kuten liikelahjatuotantoon liittyy yleensä myös tuotteita jotka suunnitellaan loppuun yhdessä asiakkaan kanssa. Seuraavaksi esiteteltävässä tarjousrekisterissä kerrotaan kuinka palvelut ja tuotteet liitetään tarjous-tauluun.

PALVELUTIETO	
nimi	**tietotyyppi**
palvelutieto_ID	INT PK auto_increment
nimi	VARCHAR (25)
palveluinfo	TEXT
hinta	INT (10)
palvelutyyppi	VARCHAR (25)

KUVIO 7. *Palvelutieto-taulu.*

5.1.7 Tarjousrekisteri

Tarjousrekisteri toteutettiin yhdellä tarjous-taululla ja siihen liittyvällä myyjä-taululla (kuvio 8). Kun myyjä tekee tarjouksen asiakkaalle, tallentuu tarjous-tauluun tiedot tarjoukseen liittyvistä palveluista ja tuotteista. Palvelut tulevat palvelutieto-taulusta tarjous-tauluun viiteavaimen palvelutieto_ID välityksellä. Tuotteet tulevat niinikään tuotetieto-taulusta ja liittyvät tarjous-tauluun viitevaimen tuotetieto_ID välityksellä. Asiakasrekisteri on jatkuvasti yhteydessä tarjoustauluun viiteavaimen asiakas_ID avulla, ja sillä voidaan tunnistaa kenelle asiakkaalle tarjous kuuluu. Myyjä-tauluun voidaan lisätä uusia järjestelmän hallinnoijia ja se on liitetty tarjous-tauluun viiteavaimen myyja_ID avulla.



KUVIO 8. *Tarjous-taulu, johon liittyvät tuotteet, palvelut ja myyjä.*

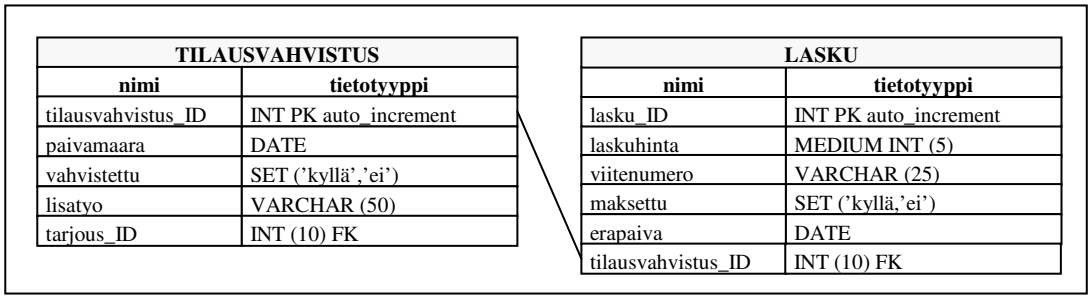
Myyjä-taulu sisältää kentän myyjän nimelle. Tarjouksen voimassaoloaika merkitään tarjous-taulun voimassaoloaika-kenttään, joka on tietotyypiltään aikaa ilmaiseva DATE. Lisäksi tarjouksentekoaika merkitään tietotyypiltään samanlaiseen kenttään. Alennus-kenttään voidaan määrittää tarjouskohtaisesti viimeisin alennuksen määrä. Otsikko-kenttä on varattu tarjouksen viimeistely-vaiheessa tarvittavaan tarjouksen kuvaukseen.

5.1.8 Tilausrekisteri

Tilausrekisteri rakentuu kahdesta taulusta jotka ovat nimeltään tilausvahvistus ja lasku (kuvio 9). Kun asiakas hyväksyy esitetyn tarjouksen tehdään siitä tilausvahvistus ja lasku. Tilausrekisteri on jatkuvasti yhteydessä tarjousrekisteriin viiteavaimen tarjous_ID kautta.

Tilausvahvistus-taulun tilausvahvistus_ID-kenttä sisältää taulun perusavaimen. Päivämäärä-kenttä kuvaa sita aikaa jolloin tilaus tehtiin. Vahvistettu-kenttään voidaan määrittää tilauksen tilanne arvoilla kyllä tai ei. Lisätyö-kenttään voidaan merkitä vapaavalintaisesti tilauksen yhteyteen liittyvä muu palvelu kuten kotiintoimitus tai asennus.

Toinen tilausrekisterin tauluista on lasku-taulu, joka on yhteydessä tilausvahvistus-tauluun viiteavaimen tilausvahvistus_ID kautta. Lasku-taulu sisältää laskulle tyypillisiä kenttiä kuten hinta, viitenumero ja eräpäivä. Maksettu-kenttään voidaan määrittää onko lasku maksettu arvoilla kyllä tai ei, mikäli asiakas on siirtänyt maksun yrityksen tilille.



TILAUSVAHVISTUS	
nimi	**tietotyyppi**
tilausvahvistus_ID	INT PK auto_increment
paivamaara	DATE
vahvistettu	SET (’kyllä’,’ei’)
lisatyo	VARCHAR (50)
tarjous_ID	INT (10) FK

LASKU	
nimi	**tietotyyppi**
lasku_ID	INT PK auto_increment
laskuhinta	MEDIUM INT (5)
viitenumero	VARCHAR (25)
maksettu	SET (’kyllä,’ei’)
erapaiva	DATE
tilausvahvistus_ID	INT (10) FK

KUVIO 9. *Tilausvahvistus-taulu, johon on liitetty lasku-taulu.*

5.2 Tietokannan toimivuuden analysointi

Toteutetusta tietokannasta voidaan todeta, ettei sen suunnittelija ole ymmärtänyt huomioida yrityksen tulevaisuuden tietotarpeita. Järjestelmän tietokanta kärsii tietojen laajenettavuuteen liittyvistä ongelmista. Lisäksi tauluista löytyy runsaasti tiedon turhaa toistoa, joka hidastaa tietokannan toimintaa. Taulujen tietotyypit on valittu osittain virheellisesti, ja ne varaavat enemmän tilaa kuin on tarpeen.

Tietokannan parantamiseen vaikuttaisi ratkaisevasti taulujen normalisoiminen kolmannen normaalimuodon mukaisesti. Käyttämättömien sarakkeiden tyhjäarvot poistuisivat kun sarakkeiden tiedot siirrettäisiin niille tarkoitetuilla uusille tauluille. Samalla tietokantakyselyistä muodostuisi entistä tarkempia. Tietokannan eheyttä olisi mahdollista parantaa ottamalla käyttöön transaktiot suojelemaan järjestelmän kriittisiä osa-alueita tahattomilta vahingoilta. Tämä edellyttäisi MyISAM-taulutyypin vaihtamista innoDB-taulutyyppiin.

5.2.1 Asiakasrekisterin parannus

Asiakasrekisterin yleisin ongelma on sen normalisoimattomuudessa. Asiakas-taulu sisältää runsaasti vapaaehtoisia kenttiä joita asiakas ei välttämättä tarvitse. Esimerkiksi asiakas voi olla henkilö, jolloin ylläpitäjä jättää yritystietojen kentät aina tyhjiksi. Tietojen tallennusprosessin jälkeen tietokantaan jää runsaasti tyhjäarvoja (NULL), jotka syövät turhaan tietokannan resursseja. Lisäksi asiakas-taulu on sekava koska se sisältää monen eri käsitteen tiedot.

Toinen ongelmista liittyy tiedon laajennettavuuteen tulevaisuudessa. Asiakkaalle on varattu rajoitettu määrä kenttiä eri yhteystiedoille. Esimerkiksi pankille on varattu vain yksi kenttä, toisin sanoen asiakas voi omistaa vain yhden pankkitilin. Tämä tietokantarakenne voi aiheuttaa myöhemmin ongelmia, jos asiakas avaa tilin uudessa pankissa. Ongelmasta päästään helposti eroon normalisoimalla asiakas-taulu kolmannen normaalimuodon mukaisesti.

Normalisoinnin avulla taulu hajoitetaan taulujoukkiohin joiden sisältö ei perustu muiden taulujen sisältöön. Normalisointia varten eritellään taulun keskeiset käsitteet joita ovat asiakas, henkilö, yritys, osoite, puhelin, sähköpostiosoite, pankki sekä yhteistyöryhmä. Jokaisesta käsitteestä luodaan omat taulunsa, jotka ovat kuitenkin yhdistetty asiakas-tauluun erilailla. (kuvio 10).

KUVIO 10. *Asiakasrekisterin parannusehdotus.*

Yllä olevassa kuvassa asiakas-taulu on miltein tyhjä normalisoinnin jälkeen. Asiakas-taulua voidaan pitää isätauluna, joka kerää muiden lapsitaulujen tiedot yhteen. Asiakas-taulun perusavain asiakas_ID liittyy useaan lapsitauluun viiteavaimen asiakas_ID avulla. PK-kirjainyhdistelmä viittaa perusavaimeen, jota vastaa aina jokin viiteavain FK.

Käsitteet henkilö ja yritys muodostavat nyt itsenäiset taulunsa, jotka liittyvät asiakas-tauluun viiteavaimen asiakas_ID avulla. Uuden asiakkaan luonnissa asiakas-tauluun lisätään uusi tietue, joka sisältää uuden perusavaimen arvon. Luonnin yhteydessä päätetään onko asiakas yritys vai henkilö, jolloin asiakas-taulun perusavain asiakas_ID kopioidaan joko henkilö- tai yritys-tauluun.

Yhteistyöryhmä sijaitsee myös omassa taulussaan edellä mainittujen syiden vuoksi. Koska Neodesign on kasvava yritys ja sillä paljon erilaisia yhteistyökumppaneita, on järkevää että on olemassa dynaaminen päivitettevä taulu erilaisille ryhmille kuten alihankkijoille ja muille sidoksille. Kun ylläpitäjä on täyttänyt taulun erilaisilla ryhmillä, voidaan jokin ryhmistä liittää luonnin yhteydessä asiakkaaseen kopioimalla perusavain yhteistyoryhma_ID asiakas-tauluun.

Käsite pankki muodostaa nyt oman taulunsa, jota on normalisoitu edelleen sisältämään lapsitaulun pankkityyppi. Pankkityyppi-taulu liittyy pankki-tauluun viiteavaimen pankkityyppi_ID avulla. Itse pankki-taulu liittyy asiakas-tauluun viiteavaimen asiakas_ID avulla. Ylläpitäjä voi milloin tahansa päivittää pankkityyppi-taulua sisältämään uusia pankkeja. Uuden pankin luomisessa asiakkaalle ylläpitäjä täyttää pankki-taulun perustiedot ja liittää haluamansa pankkityypin asiakkaalle. Tämä tietokantarakenne mahdollistaa sen, että yhdellä asiakkaalla voi olla rajaton määrä erilaisia pankkitilejä.

Asiakkaiden sähköpostiosoitteet tallentuvat myös omaan tauluunsa, jolloin asiakkaalle ei välttämättä tarvitse määritellä sähköpostiosoitetta. Tietokantarakenne mahdollistaa rajattoman määrän erilaisia sähköpostiosoitteita yhdelle asiakkaalle. Jos määrä halutaan kuitenkin rajata vain yhteen, niin silloin kenttä on määritettävä uniikiksi.

Asiakkaalla ei tarvitse myöskään olla osoitetta, jolloin tyhjäarvoilta säästytään. Osoitetiedot on siirretty luonnollisesti omiin tauluihinsa. Osoite-taulusta voidaan vielä normalisoida kaksi erillistä taulua, jotka ovat kaupunki ja postinumero, jolloin niidenkin arvojen turhalta toistolta vältytään.

Ylläpitäjällä on mahdollisuus myöhemmin varastoida kaupunki-tauluun uusia kaupungin nimiä, jotka voidaan liittää nopeasti uuteen asiakkaalle määritettyyn osoitteeseen.

Puhelin-taulu normalisoituu tuttuun tapaan useaan osaan. Puhelin-tauluun talletetaan puhelinnumero, suunta-tauluun suuntanumero ja puhelintyyppi-tauluun yleisiä puhelintyyppejä. Erilaisia puhelintyypejä voivat olla esimerkiksi mobiili, lanka tai fax. Suuntanumerot koostuvat pääasiassa kotimaisista suunnista. Uuden puhelinnumeron luonnissa ylläpitäjä voi kirjoittaa numeron ja liittää siihen mahdollisesti kuuluvan suunnan ja puhelintyypin, jotka tallentuvat isätauluun puhelin. Puhelinnumeroita kuvaavat kentät kannattaa jättää merkkijonotyypeiksi, sillä numerot voivat alkaa luvulla 0, jota numeeriset tietotyypit eivät tue.

Alkuperäinen asiakas-taulu sisälsi runsaasti virheellisesti valittuja tietotyyppejä. Itse asiassa suurin osa oli määritetty käyttämään VARCHAR-tietotyyppiä, joka vei turhaan tilaa tietokannasta. Jokaisen asiakasrekisterin kokonaislukuarvoja sisältävät kentät kannattaa vaihtaa positiiviksi luvuiksi (UNSIGNED), jolloin lukujen käyttämä negatiivinen tila poistetaan ja lukuja lasketaan vain nollasta eteenpäin. Positiivisten lukujen maksimiarvoja saadaan näin kasvatettua suuremmaksi.

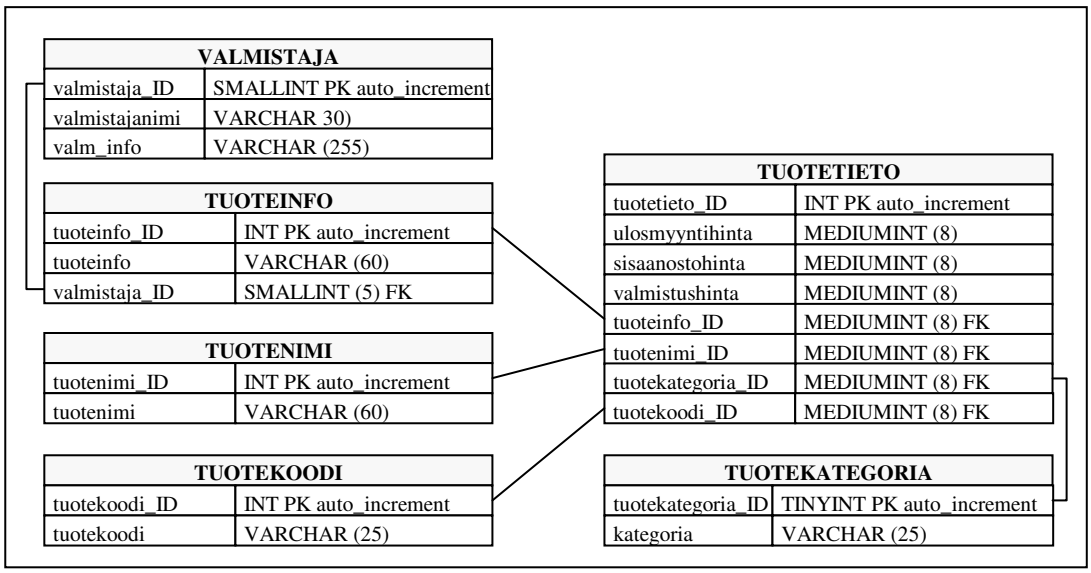
Muita kokonaislukuja on optimoitu arvioimalla niiden tilankäyttöä yrityksen henkilöstön kanssa. Esimerkiksi suunta-taulun sisältämässä suunta_ID kentässä on käytetty TINYINT-tietotyyppiä, koska 255 suuntanumeron uskotaan riittävän yrityksen tarpeisiin. Vertauksena INT-tyyppinen kenttä vie 4 tavua tilaa, kun TINYINT vie yhden tavun. Lisää arvioita tietovaatimuksista on liitteessä 4.

5.2.2 Tuote- ja palvelurekisterin parannus

Tuoterekisterin ongelmat löytyvät asiakasrekisterin tapaan turhasta tiedon toistosta ja käyttämättömistä vapaaehtoisista kentistä. Kentille tuotenimi, tuotekoodi, tuotekategoria ja tuoteinfo on normalisoitu omat itsenäiset taulunsa tyhjäarvojen poistamiseksi. Tuoteinfo-taulua on normalisoitu edelleen sisältämään erillisen taulun valmistajalle (kuvio 11).

Tuotekategorian siirtäminen omaan tauluun sallii ylläpitäjän myöhemmin lisätä uusia mahdollisia tuotekategorioita järjestelmän hallintasivuston kautta. Valmistaja-taulu on eritelty omaan tauluunsa koska tuotteella ei välttämättä ole ulkopuolista valmistajaa, jolloin vältytään tyhjäarvoilta. Lisäksi valmistaja-tauluun voidaan varastoida uusia valmistajia, joita voidaan nopeasti liittää tuotteisiin.

Alkuperäiset kentät tuotteen hinnoille oli määritetty käyttämään merkkijonotyyppejä jotka veivät turhaan resursseja tietokannasta. Kentät on muutettu käyttämään MEDIUMINT-kokonaislukua, joka vie noin 8 tavua vähemmän tilaa. Tuotekategoria-taulun tuotekategoria_ID-kentässä on käytetty TINYINT-tietotyyppiä, joka rajaa tuotekategorioiden määrän 255 kappaleeseen.

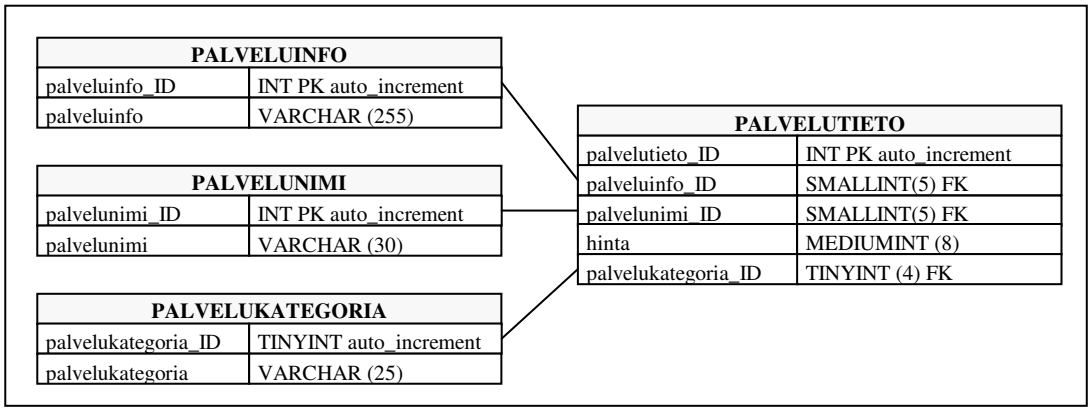


KUVIO 11. *Tuoterekisterin parannusehdotus.*

Palvelurekisteri on hyvin samankaltainen tuoterekisterin kanssa, ja niiden ongelmat muistuttavat toisiaan. Käsitteet palvelunimi, palveluinfo ja palvelukategoria on normalisoitu luonnollisesti erillisiin tauluihinsa, jotka ovat yhteydessä palvelutieto-tauluun. Samalla tyhjäarvot ovat poistuneet (kuvio 12).

Palvelukategoria on sijoitettu omaan itsenäiseen tauluunsa aikaisempien tiedon laajennettavuuteen liittyvien ongelmien takia. Uusi tietokantarakenne mahdollistaa sen, että ylläpitäjällä on myöhemmin mahdollisuus lisätä palvelukategoria-tauluun uusia palvelutyyppejä, joita kasvava yritys saattaa myöhemmin laajentaa palveluvalikoimaansa.

Palvelurekisterin tietotyyppien kokonaislukuja on optimoitu arvioimalla yrityksen kanssa suurinta mahdollista palvelukategorioiden määrää. Palvelukategoria-taulun perusavaimen tietotyyppi on muutettu TINYINT-tyyppiseksi, koska 255 eri palvelutyypin uskotaan riittävän yrityksen tarpeisiin. Tosin tietotyyppejä on mahdollisuus vaihtaa suuremmiksi myös myöhemmin tarvittaessa.

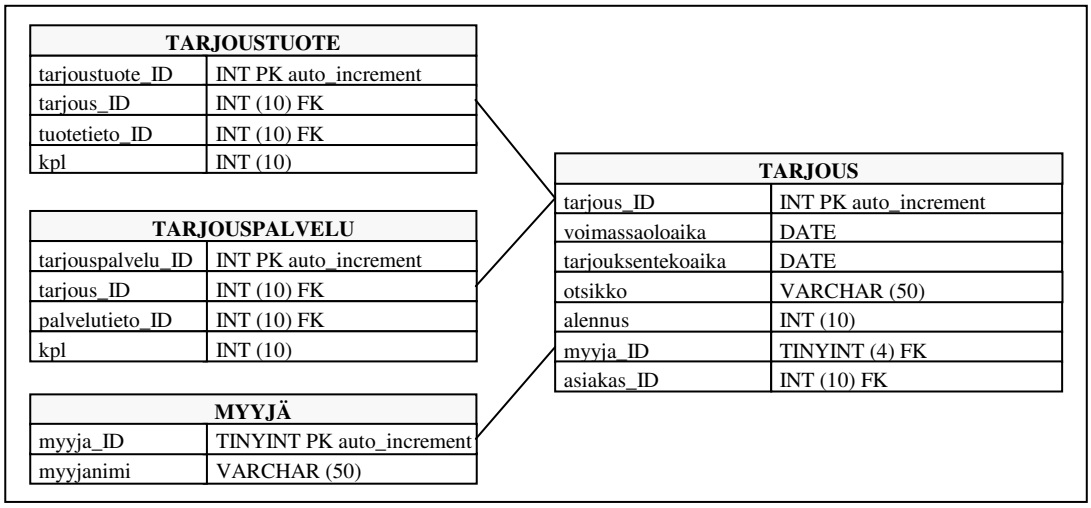


KUVIO 12. *Palvelurekisterin parannusehdotus.*

5.2.3 Tarjousrekisterin parannus

Tarjousrekisterin ongelmat liittyvät myös taulujen normalisoimattomuuteen ja tarjous-tauluun kohdistuviin rajoituksiin. Nykyinen tarjousrekisterin rakenne sallii tarjouksen sisältävän vain yhden palvelun ja siihen kuuluvan tuotteen.

Ongelmasta päästään helposti eroon luomalla kaksi uutta aputaulua tarjouspalvelu ja tarjoustuote (kuvio 13).



KUVIO 13. *Tarjous-taulu, johon on liitetty tarjoustuotteet, tarjouspalvelut sekä myyjä.*

Tarjoustuote-taulu ja tarjouspalvelu-taulu on liitetty tarjous-tauluun viitevaimen tarjous_ID avulla. Tarjouspalvelu-taulun tehtävänä on kerätä tietoa tarjoukseen liittyvistä palveluista palvelutieto-taulusta. Tarjoustuote-taulun tehtävänä on kerätä tietoa tarjoukseen liittyvistä tuotteista tuotetieto-taulusta.

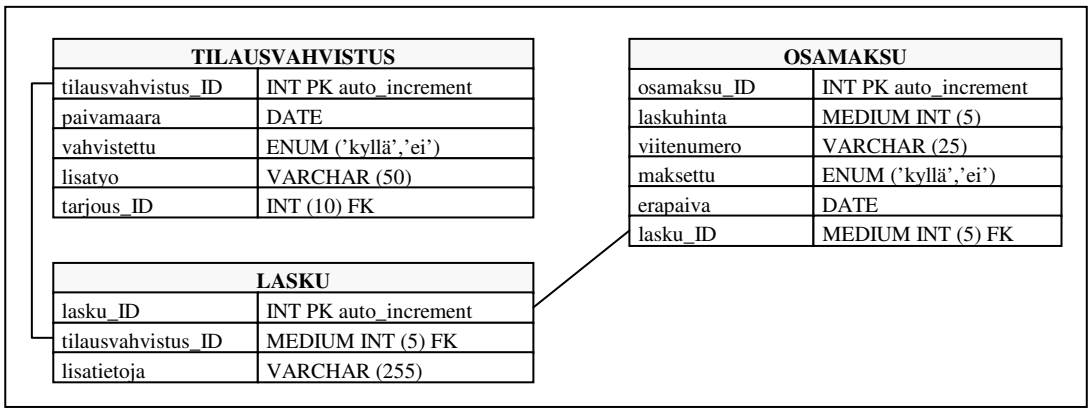
Tarjous-taulua voidaan ajatella eräänlaisena isätauluna, joka kerää muiden lapsitaulujen tiedot. Kun taulujen väliset suhteet on järjestetty näin, voidaan asiakkaalle tehdä tarjous, joka sisältää yhden palvelun, mutta johon voi kuulua useampi tuote. Tällainen palvelu on esimerkiksi liikelahjatuotanto, jossa yritys suunnittelee asiakkaan kanssa yhden tai useamman tuotteen.

Tarjousrekisterin tietotyyppien kokonaislukuja on optimoitu tarjoukseen liittyvien myyjien enimmäismäärän osalta. Myyjä-taulun myyja_ID-kentän tietotyyppi on vaihdettu INTEGER-tietotyypistä TINYINT-tyyppiseksi, sillä yrityksen henkilöstön ei uskota ylittävän 255 henkilöä lähitulevaisuudessa.

5.2.4 Tilausrekisterin parannus

Tilausrekisteri ei välttämättä vaadi tietojen erittelemistä, koska se sisältää aina pelkästään tilauksia. Vahvistus-kentän SET-tietotyyppi tosin kannattaa vaihtaa ENUM-tietotyyppiin, sillä arvo voi silloin olla vain yksi määritellyn arvolistan arvoista, eli kyllä tai ei.

Vaikka tietojärjestelmän laskutus saatiin toteutettua haluamalla tavalla, jäi siitä puuttumaan asiakaskohtainen osamaksun huomiointi tiukan aikataulun vuoksi. Uusi toiminnallisuus on mahdollista toteutettaa luomalla uusi osamaksu-taulu ja liittämällä se lasku-tauluun viiteavaimen lasku_ID avulla (kuvio 14). Pienellä muutoksella lasku-tauluun asiakkaalle on siis mahdollista tehdä lasku joka jakautuu myös useampaan erään.



KUVIO 14. *Parannusehdotus tilausrekisteriin*

5.2.5 Kenttä- ja taulunimien parantelu

Osa tietokannan kenttä- ja taulunimistä olivat harhaanjohtavia tai liian pitkiä. Samankaltaiset nimet kuten hinta, nimi ja ominaisuudet toistuivat useissa tauluissa ja vaikeuttivat tietokantakyselyiden kirjoittamista etenkin useamman taulun liitoksissa. Kenttien nimistä tulisi aina luoda muista nimistä poikkevia, jotka kaikki yrityksessä ymmärtävät. Esimerkiksi valmistaja-taulussa on mahdollista käyttää sellaisia kenttälyhenteitä kuin ValmHinta, ValmNimi tai ValmKoodi.

Toisaalta kenttänimiä ei kannata lyhentää liikaa koska silloin niiden lukijat ymmärtävät niitä väärin. Esimerkiksi Ptieto ei sano lukijalle mitään, ainoastaan sanan lyhentäjä voi ymmärtää sanan alkuperäisen merkityksen. Sen sijaan Palvtieto vihjaa lukijalle, että lyhenne koostuu sanoista palvelu ja tieto. Vaikka nimien tarkentamisella ei olekaan tietokannan suorituskyvyn kannalta mitään merkitystä, selkeyttää se huomattavasti tietokannan luettavuutta myöhemmin.

5.2.6 Transaktioiden hyödyntäminen tietokannassa

Transaktioita tukeva InnoDB-taulutyyppi voidaan ottaa käyttöön muuttamalla taulumoottori ALTER-komennon avulla. Eräs transaktioiden potentiaalinen hyödyntämiskohde on tilausvahvistuksen ja laskutuksen seuranta. Jos tilauksen vahvistus jostain syystä epäonnistuu, peruutetaan koko tapahtuma ja laskua ei synny.

Transaktiot soveltuvat suojelemaan kaikkia sellaisia tapahtumia joissa on useita kyselyitä, tallennuksia tai päivityksiä peräkkäin. Systeemivian sattuessa mitään ylimääräistä kyselyä ei suoriteta, ja tietokanta palautuu automaattisesti alkuperäiseen tilaansa.

Rivilukituksien käyttäminen tauluissa estää useamman ylläpitäjän samanaikaisen kentän muokkauksen. Tällöin esimerkiksi tuotteiden hinnat eivät vahingossa pääse muuttumaan, jos tarjoustuotteiden tallennusprosessi on käynnissä ja toinen myyjistä muokkaa tuotteiden hintaa. Tosin useamman hallinnoijan samanaikainen tietokantaan kohdistuva muokkaus on harvinaista.

InnoDB tuo myös viitee-eheyden automaattisen valvonnan. Tällöin on mahdollista estää sellaisten rivien poistaminen joilla on viitteitä muihin tauluihin. Esimerkiksi sellaisten tuotteiden poistaminen jotka kuuluvat asiakkaille jo tehtyihin tarjouksiin, on mahdollista estää RESTRICT-määreellä. Jos asiakasrekisteristä taas halutaan poistaa kokonaan tietty asiakas, on mahdollista poistaa kaikki asiakkaalle kuuluvat yhteystiedot automaattisesti CASCADE-määreellä.

Toimenpide edellyttää, että jokaisen lapsitaulun viiteavain indeksoidaan ja lisätään viittaus isätaulun perusavaimeen. Tämä säästää merkittävästi eheyden hallintaan vaadittavaa PHP-koodia.

5.2.7 Tietokantakyselyiden suorituskyvyn parantaminen

Erilaisten kyselyiden nopeutta on mahdollista nopeuttaa indeksoimalla kentät joita käytetään usein kyselyiden haku- ja lajitteluehdoissa. Näitä ovat voivat olla esim. tuotteiden ja palveluiden kategoriat tai asiakkaan yhteistyöryhmät. Lisäksi jos haetaan tietyllä kirjaimella alkavien asiakkaiden tietoja on indeksoiminen kannattavaa. Indeksit kuitenkin hidastavat päivitysoperaatioita koska tietokannan tiedonhallintajärjestelmä joutuu ylläpitämään tiedostorakenteita. Indeksoinnissa onkin mahdollista käyttää oheishakemistoja, koska päähakemistot indeksoidaan aina automaattisesti.

Toinen vaihtoehto on denormalisoida taulukokonaisuuksia nopeampien kyselyiden saavuttamiseksi. Jos kyselyt muodostuvat lukuisista toisiin liitetyistä taulukokonaisuuksista, voi erilaisten raporttien käsittely olla hidasta. Denormalisointi kuitenkin rikkoo eheyssääntöjä ja tuo mukanaan eräitä uusia ongelmia kuten kaksinkertaisia kenttiä ja ylimääräistä dataa.

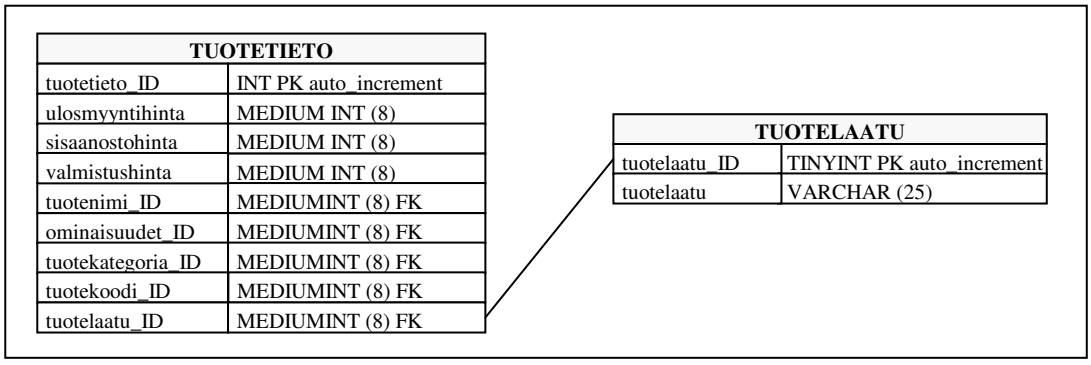
Palvelimissa kyselyiden nopeuteen vaikuttaa olennaisesti kiintolevyn tila ja nopeus. Nebulan vuokraamat palvelit ovat kuitenkin riittävän nopeita joten tietokantarakenteen muutoksiin ei ole tarvetta. Lisäksi tietojärjestelmässä sijaitsevien asiakas-, tilaus-, tuote- ja palvelurekisterin ei odoteta kasvavan lähitulevaisuudessa riittävästi, jotta suorituskyky laskisi olennaisesti.

5.2.8 Uusia kehitysideoita

Tuotteiden jakaminen erilaisiin tuoteluokkiin Boston Consulting Groupin markkinamatriisin mukaisesti auttaa seuraamaan kuinka korkea kasvupotentiaali tuotteilla olisi markkinoilla.

Tuotteet voidaan jakaa karkeasti neljään laatuluokkaan. Rakkikoirat ovat tuotteita, jotka eivät tuota yritykselle enää minkäänlaista voittoa. Lypsylehmät tuottavat yritykselle edelleen rahaa, mutta niiden elinkaari on taantumassa. Tähtituotteet olennaisesti menestyvät ja kysymysmerkeistä voidaan kehittää uusia menestystuotteita mahdollisesti.

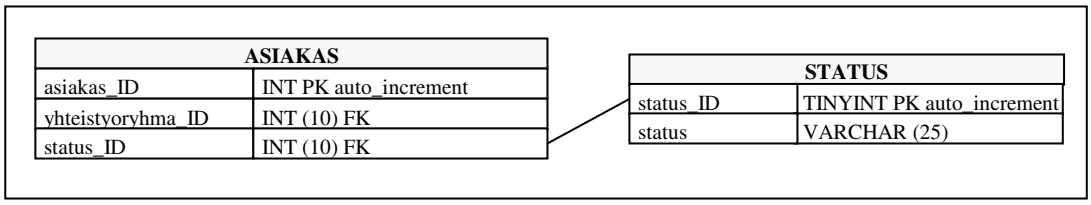
Idea tuotteiden laatuluokasta on mahdollista toteuttaa pienellä lisäyksellä tuotetieto-tauluun sekä luomalla uusi tuotelaatu-taulu (kuvio 15). Tuotetieto-taulun kenttä tuotelaatu_ID tulee luonnollisesti tuotelaatu-taulusta. Jatkossa uusien laatuluokkien lisääminen onnistuu helposti hallinnointi-sivuston kautta ja vanhojen tuotteiden laatuluokkaa on mahdollista päivittää vastaamaan nykymarkkinoiden tilaa.



KUVIO 15. *Kehitysidea laatuluokan lisäämisestä tuotetieto-tauluun.*

Segmentoimalla asiakkaat erilaisten perusteiden avulla auttaa saamaan aikaan kanta-asiakassuhteita. Eräs vaihtoehto on käyttää seuraavia asiakassuhteen vaiheita kuten potentiaalinen asiakas, uusi asiakas, satunnaisasiakas ja kanta-asiakas. Kun tiedetään missä vaiheessa kukin asiakas on, asiakasta voidaan mahdollisesti ohjata etenemään seuraavaan vaiheeseen asiakassuhteen parantamiseksi.

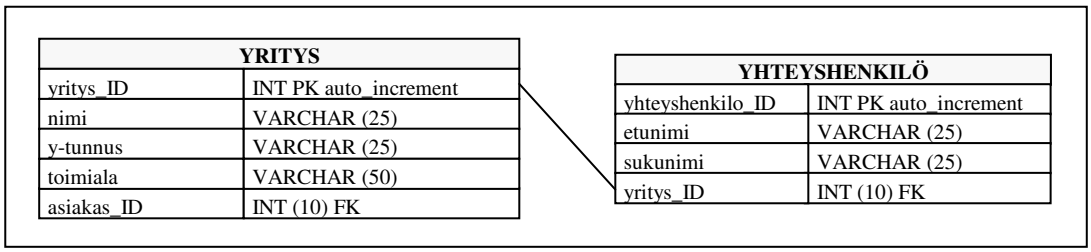
Asiakassuhteen seuranta on mahdollista toteuttaa luomalla uusi status-taulu, jonne hallinnointi-sivuston kautta voidaan myöhemmin päivittää uusia vaiheita (kuvio16). Status-taulu on mahdollista liittää asiakas-tauluun viiteavaimen status_ID avulla.



KUVIO 16. *Kehitysidea asiakasstatuksen lisäämisestä asiakas-tauluun.*

Vaikka asiakasrekisteriin voidaan tallettaa tiedot erikseen yrityksestä ja henkilöstä, ei yritykselle löydy yhteyshenkilöä jonka kautta voidaan edistää asiakassuhteita ja saadaan lisäinformaatiota yrityksestä nopeasti. Ehdotus yhteyshenkilöstä on mahdollista toteuttaa luomalla uusi yhteyshenkilö-taulu ja liittämällä se jo olemassaolevaan yritys-tauluun. Kahden taulun välinen liitos voidaan tehdä viitevaimen yritys_ID avulla, joka siis sijoitetaan yhteyshenkilö-tauluun (kuvio 17). Tämä tietokantarakenne mahdollistaa sen, että yhdellä yrityksellä voi olla yksi tai useampi yhteyshenkilö.

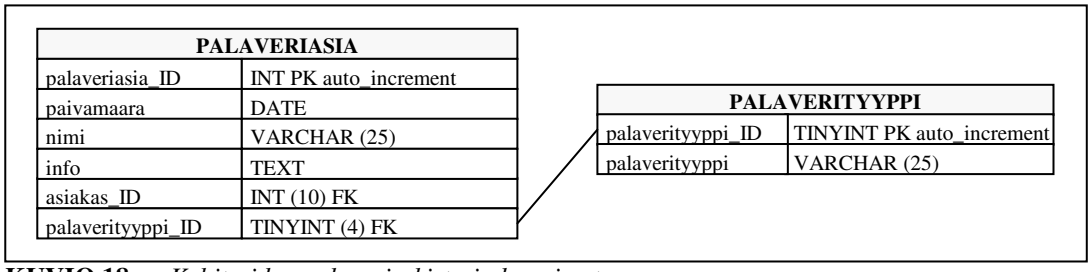
Yhteyshenkilölle on lisäksi mahdollista luoda asiakasrekisterin tapaan lapsitaulut erilaisille yhteystiedoille kuten puhelimelle, sähköpostille ja osoitteille. Tällöin syntyy uusi yhteyshenkilörekisteri.



KUVIO 17. *Kehitysidea yhteyshenkilön lisäämisestä yritystauluun.*

Koska Neodesign pitää säännöllisesti palavereja ja käsittelee niissä erilaisia asiakkaita, on hyvä olla jonkinlainen muistio palavereissa käsitellyille asioille. Idea asiakaskohtaisesta palaverirekisteristä on mahdollista toteuttaa luomalla uusi palaveriasia-taulu (kuvio 18). Lisäksi jos halutaan jakaa palaverit erilaisiin tyyppeihin, voidaan luoda palaverityyppi-taulu ja liittää se palaveriasia-tauluun viitevaimen palaverityyppi_ID avulla. Järjestelmän hallinnoijalla on tällöin mahdollisuus lisätä uudentyyppisiä palavereja tulevaisuudessa.

Palaverirekisteriin voidaan merkitä asiakaskohtaisesti joko yrityksen tai henkilön tiedot, ja se on jatkuvasti yhteydessä asiakasrekisteriin viiteavaimen asiakas_ID avulla. Palaveriasia-tauluun voidaan toki tallettaa muutakin kuin asiakaskohtaista informaatiota. Paivamaara-kenttään merkitään päiväys jolloin palaveri tehtiin. Nimi-kenttään laitetaan luonnollisesti palaverin nimi ja info-kenttään palaverin aikana asiakkaasta tehtyjä muistiinpanoja.



KUVIO 18. *Kehitysidea palaverirekisterin luomisesta.*

Eräs kehitysidea on luoda taulu yrityksen tuotteiden myynnille, jonne voidaan kerätä tietoa siitä, kuinka paljon mitäkin tuotetta myydään ja miten myynti jakautuu eri tuotteille. Myyntitilastoon voidaan tallentaa tuotteen myyntipäivämäärä, yksikköhinta, tuotteen kokonaismyynti prosentteina sekä asiakkaan nimi jolle se on myyty. Myyntitilasto on tällöin dynaaminen, ja tietoa on mahdollista hakea tietyltä aikaväliltä. Lisäksi on mahdollista tarkastella asiakaskohtaista myyntiä myös tuotetasolla.

6 JOHTOPÄÄTÖKSET

Toteutetun tietokannan rakenteesta voidaan todeta, ettei sen tekijä ole huomioinut yrityksen tietotarpeita ja ymmärtänyt teorian merkitystä tietokannan suunnitteluvaiheessa. Työharjoittelun puitteissa rakennettu tietokanta on jäänyt puutteelliseksi ja sekavaksi. Siihen ovat vaikuttaneet tiukka aikataulu, mutta myös oma rajoittunut tietokantoihin liittyvä osaaminen.

Tietokannan yleisimmät ongelmat liittyivät tietokantarakenteen normalisoimattomuuteen. Tietokantarakenteen alhainen normalisoinnin taso voi aiheuttaa myöhemmin yritykselle tietojen päivitys-, lisäys- ja poisto-ongelmia. Tietokantaan ei ollut mahdollista laajentaa uusia yhteystietoja kuten osoitteita ja puhelinnumeroita asiakkaille. Tietokannan taulut saattoivat sisältää runsaasti käyttämättömiä kenttiä, joihin tallentui tyhjäarvoja tietojen tallennusprosessin aikana. Lisäksi taulujen tietotyypit oli valittu osittain väärin ja ne varasivat enemmän tilaa kuin oli tarvetta.

Normalisoinnin avulla tietokannasta saadaan yleiskäyttöinen, joka käyttäytyy hyvin erilaisissa päivitystilanteissa. Lisäksi etuina ovat että tiedot jäsentyvät loogisesti omiin osa-alueihinsa. Menetelmällä saadaan myös käyttämättömien kenttien tyhjäarvot poistettua. Tällöin tietokannan rakenteen muokkaaminen ja ylläpito on tulevaisuudessa helppoa. Toisaalta tauluja ei kannata normalisoida liikaa, koska silloin tiedon hakeminen tietokannasta vaikeutuu. Tietotyyppien valintaan kannattaa aina perehtyä selvittämällä tietotarpeita tarkemmin yrityksen henkilöstön kanssa.

InnoDB-taulutyypin hyödyntäminen tuo mukanaan runsaasti eheyden hallintaan liittyviä toimintoja, jotka suojelevat yritykselle tärkeiden tietojen pilaantumiselta. Lisäksi eheyden hallintaan ei tarvita ylimääräistä PHP-koodia, koska MySQL-tiedonhallintajärjestelmä hoitaa sen automaattisesti. Tämä vähentää luonnollisesti palvelimen tarvitsemaa aikaa koodin suorittamiseen.

Uudet kehitysideat antavat lisäarvoa tietojärjestelmälle ja sitä kautta yritykselle sekä sen asiakkaille. Tuotteiden jakaminen erilaisiin luokkiin auttaa seuraamaan niiden kasvupotentiaalia markkinoilla. Palaverirekisterin avulla voidaan tallentaa yksityiskohtaisia muistiinpanoja palavereissa käsitellyistä asiakkaista. Asiakaskohtaisen myyntitilaston avulla voidaan selvittää mistä asiakassegmentistä löytyvät yrityksen tärkeimmät asiakkaat.

On tärkeää, että tietokannat toimivat ongelmitta suurissa tietojärjestelmissä, koska yrityksen koko liiketoiminta saattaa olla sidoksissa tallennettuihin tietoihin. Hyvin suunniteltu ja toteutettu tietokanta parantaa luonnollisesti myös tietojen hallintaa. Yrityksellä, jonka tiedot ovat suuriakin muutoksia sietävän tiedonhallinnan piirissä, on merkittävä kilpailuetu.

LÄHTEET

Painetut lähteet

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

Hovi, A., Huotari, J. & Lahdenmäki, T. 2003. Tietokantojen suunnittelu & indeksointi. 1. Painos. Jyväskylä: Docendo Finland Oy.

Meloni, Julie C. 2003. MySQL Trainer Kit. 1. Painos. Helsinki: It-Press.

Hernandez, Michael J. 2000. Tietokannat: Suunnittelu ja toteutus. 1. Painos. Helsinki: It-Press.

Sähköiset lähteet

MySQL Reference Manual. 2007. Foreign key constrains. [www-dokumentti]. MySQL AB. [viitattu 11.1.2007].Saatavissa: http://dev.mysql.com/doc/refman/5.0/en/innodb-foreign-key-constraints.html

MySQL Reference Manual. 2007. Storage Engines and Table Types. [www-dokumentti]. MySQL AB. [viitattu 11.1.2007].Saatavissa: http://dev.mysql.com/doc/refman/5.0/en/storage-engines.html

MySQL Reference Manual. 2007. Characteristics of BDB tables. [www-dokumentti]. MySQL AB. [viitattu 11.1.2007]. Saatavissa: http://dev.mysql.com/doc/refman/5.0/en/bdb-characteristics.html

MySQL Reference Manual. 2007. The memory storage engine. [www-dokumentti]. MySQL AB. [viitattu 11.1.2007].Saatavissa: http://dev.mysql.com/doc/refman/5.0/en/memory-storage-engine.html

MySQL Reference Manual. 2007. The MyISAM storage engine. [www-dokumentti]. MySQL AB. [viitattu 11.1.2007]. Saatavissa: http://dev.mysql.com/doc/refman/5.0/en/myisam-storage-engine.html

MySQL Reference Manual. 2007. InnoDB table and Index Structures. [www-dokumentti]. MySQL AB. [viitattu 11.1.2007].Saatavissa: http://dev.mysql.com/doc/refman/5.0/en/innodb-table-and-index.html

MySQL Reference Manual. 2007. Data Type Storage Requirements [www-dokumentti]. MySQL AB. [viitattu 11.1.2007]. Saatavissa: http://dev.mysql.com/doc/refman/5.0/en/storage-requirements.html

Wikipedia. 2006. Tietokannan hallintajärjestelmä. [www-dokumentti]. [viitattu 11.1.2007].Saatavissa: http://fi.wikipedia.org/wiki/Tietokannan_hallintaj%C3%A4rjestelm%C3%A4.

Wikipedia. 2007. SQL. [www-dokumentti]. [viitattu 5.1.2007].Saatavissa: http://fi.wikipedia.org/wiki/SQL.

Wikipedia. 2007. Eheys. [www-dokumentti]. [viitattu 11.1.2007]. Saatavissa: http://fi.wikipedia.org/wiki/Eheys.

Wikipedia. 2007. Tietokannan normalisointi. [www-dokumentti]. [viitattu 11.1.2007].Saatavissa: http://fi.wikipedia.org/wiki/Tietokannan_normalisointi.

Wikipedia. 2007. Linux. [www-dokumentti]. [viitattu 11.1.2007]. Saatavissa: http://fi.wikipedia.org/wiki/Linux.

Wikipedia. 2007. Data Control Language. [www-dokumentti]. [viitattu 11.1.2007].Saatavissa: http://en.wikipedia.org/wiki/Data_Control_Language

Wikipedia. 2007. Data Definition Language. [www-dokumentti]. [viitattu 11.1.2007]. Saatavissa: http://en.wikipedia.org/wiki/Data_Definition_Language

Wikipedia. 2006. Data Manipulation Language. [www-dokumentti]. [viitattu11.1.2007]. Saatavissa:http://en.wikipedia.org/wiki/Data_Manipulation_Language

Wikipedia. 2007. PHP. [www-dokumentti]. [viitattu 11.1.2007]. Saatavissa: http://fi.wikipedia.org/wiki/Php.

Wikipedia. 2007. Apache. [www-dokumentti]. [viitattu 11.1.2007]. Saatavissa: http://fi.wikipedia.org/wiki/Apache.

Wikipedia. 2007. Tietojärjestelmä. [www-dokumentti]. [viitattu 11.1.2007]. Saatavissa: http://fi.wikipedia.org/wiki/Tietoj%C3%A4rjestelm%C3%A4

LIITTEET

1. Keskeiset käsitteet.
2. Rakennekaavio tietojärjestelmästä.
3. Käsitemalli alkuperäisestä tietokantakokonaisuudesta
4. Arvioita järjestelmän tietovaatimuksista.

KESKEISET KÄSITTEET

Tietokanta

Tietokanta on kokoelma tietoja, joilla on yhteys toisiinsa. Tietokanta voi sijaita paperilla tai kiintolevyllä. Relaatiotietokantamallissa tiedot talletetaan tauluihin. (Wikipedia: Vapaa tietosanakirja 2007.)

TKHJ (Tietokannan hallintajärjestelmä)

Tietokannan hallintajärjestelmä on ohjelmisto joka mahdollistaa tietokannan ylläpitämisen tietokoneella. Tunnettuja esimerkkejä ovat Oracle, MySQL ja Access. (Wikipedia: Vapaa tietosanakirja 2007.)

SQL (Structured Query Language)

SQL on IBM:n kehittämä standardoitu kyselykieli, jonka avulla tietokantaan voidaan tehdä erilaisia hakuja, lisäyksiä tai muutoksia. (Wikipedia: Vapaa tietosanakirja 2007.)

Tiedon eheys

Tietokannoissa eheys tarkoittaa tietojen risririidattomuutta ja yhteensopivuutta keskenään. (Wikipedia: Vapaa tietosanakirja 2007.)

Tietokannan normalisointi

Tietokannan normalisointi on menetelmä, jolla tietokannan taulut hajoitetaan taulujoukkioiksi. Etuina ovat turhan tiedon poistuminen tauluista ja parempi tiedon laajennettavuus tulevaisuudessa. (Wikipedia: Vapaa tietosanakirja 2007.)

DML (Data manipulation language)

Tietokannan käsittelykieli. Kieltä käytetään tiedon lisäämiseen, päivittämiseen tai poistamiseen tietokannassa. (Wikipedia: Vapaa tietosanakirja 2007.)

DDL (Data definition language)

Tietokannan rakenteelliseen määrittelyyn käytettävä kieli. Kielellä voidaan luoda tietokantaan tauluja tai tehdä rakenteellisia muutoksia olemassaoleviin tauluihin. (Wikipedia: Vapaa tietosanakirja 2007.)

DCL (Data control language)

Tietokannan valvontakieli. Kielellä voidaan luoda oikeuksia tietokannan lukemiseen sekä määrittää sääntöjä jotka suojelevat tietokantaa virhetilanteilta. (Wikipedia: Vapaa tietosanakirja 2007.)

Linux

Linus Torvalds on kehittänut käyttöjärjestelmäytimen nimeltään Linux. Ytimellä tarkoitetaan käyttöjärjestelmän osaa, joka mahdollistaa käyttöjärjestelmän muiden osien toiminnan. (Wikipedia: Vapaa tietosanakirja 2007.)

PHP (Hypertext preprocessor)

PHP on ohjelmointikieli, jota käytetään web-palvelinympäristössä luomaan dynaamisia sivuja. PHP-ohjelmointikieli suoritetaan aina palvelimella eikä käyttäjän selaimessa. (Wikipedia: Vapaa tietosanakirja 2007.)

Apache HTTP-palvelin

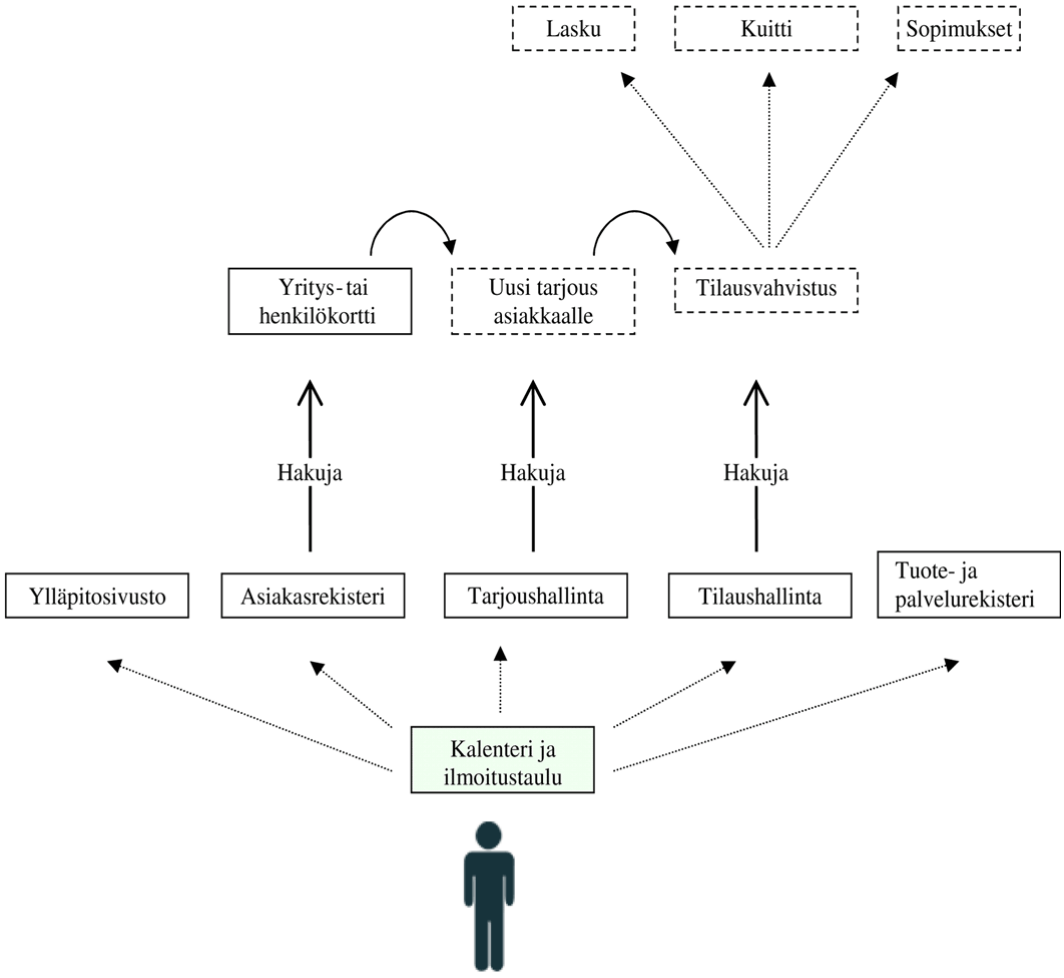
Apache HTTP-palvelin on palvelinohjelma, jonka tarkoitus on tarjota erilaisia palveluita muille ohjelmille verkon yli tai samassa tietokoneessa. Kommunikaatio perustuu asiakkaan ottamaan yhteydenottoon. (Wikipedia: Vapaa tietosanakirja 2007.)

Tietojärjestelmä

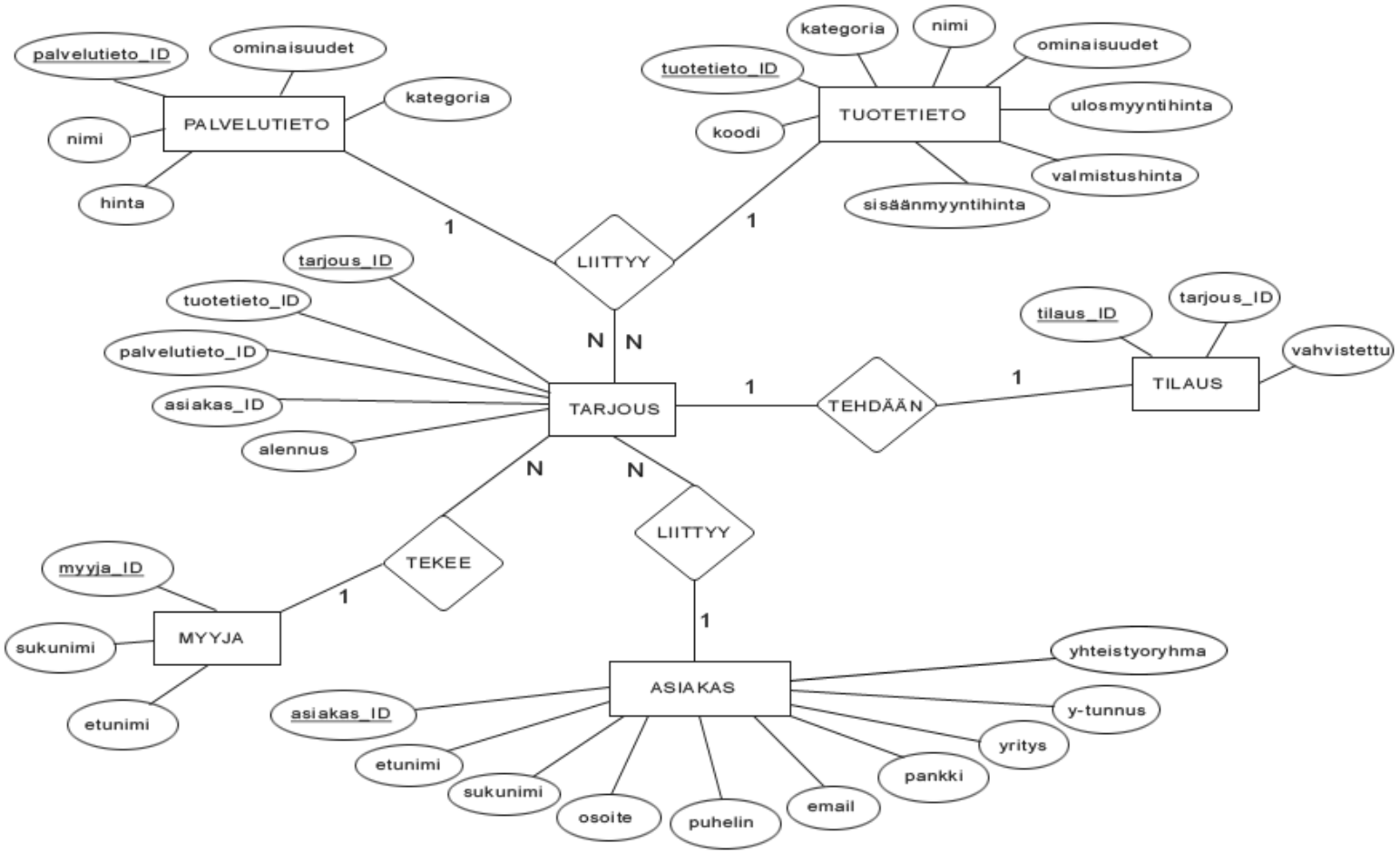
Tietojärjestelmä voi koostua asiakkaista, sovelluksesta, tietokannasta ja lomakkeista ja sen tavoitteena on tehostaa jotain toimintaa kuten tilausten hallintaa.(Wikipedia: Vapaa tietosanakirja 2007.)

LIITE 2

Rakennekaavio tietojärjestelmästä



Käsitemalli alkuperäisestä tietotokantakokonaisuudesta



LIITE 3

ARVIOITA JÄRJESTELMÄN TIETOVAATIMUKSISTA

Asiakkaat: Asiakkaille ei asetettu juuri ylärajaa. Käytetty perusavaimen tietotyyppi oli INTEGER joka voi sisältää arvoja väliltä 0 – 4294967295.

Asiakasstatukset: Käytettyjä statuksia olivat potentiaalinen asiakas, uusi asiakas, satunnaisasiakas ja kanta-asiakas. Käytetty perusavaimen tietotyyppi oli TINYINT joka voi sisältää arvoja väliltä 0 – 255.

Hallinnoijat: Hallinnoijien ei uskottu kasvavan yli 255:n. Käytetty perusavaimen tietotyyppi oli TINYINT joka voi sisältää arvoja väliltä 0 – 255.

Kaupunkityypit: Kaupunkityyppeinä olivat kaikki kotimaiset kaupungit. Käytetty perusavaimen tietotyyppi oli TINYINT joka voi sisältää arvoja väliltä 0 – 255.

Palvelukategoriat: Palvelukategorioita uskottiin olevan enintään 255. Käytetty perusavaimen tietotyyppi oli TINYINT joka voi sisältää arvoja väliltä 0 – 255.

Pankkityypit: Suomessa oli vuoden 2005 lopussa yhteensä 345 pankkia. Käytetty perusavaimen tietotyyppi oli SMALLINT joka voi sisältää arvoja väliltä 0 – 65535.

Puhelinsuunnat: Puhelinsuuntina olivat kaikki kotimaiset suunnat. Käytetty perusavaimen tietotyyppi oli TINYINT joka voi sisältää arvoja väliltä 0 – 255.

Puhelintyypit: Yleisiä käytettyjä puhelintyyppejä olivat yrityspuhelin, kotipuhelin, matkapuhelin, mobiilipuhelin, lankapuhelin ja fax. Käytetty perusavaimen tietotyyppi oli TINYINT joka voi sisältää arvoja väliltä 0 – 255.

Postinumerot: Postinumeroina olivat kaikki kotimaiset postinumerot. Käytetty perusavaimen tietotyyppi oli TINYINT joka voi sisältää arvoja väliltä 0 – 255.

Palaverityypit: Käytetty perusavaimen tietotyyppi oli TINYINT joka voi sisältää arvoja väliltä 0 – 255.

Tuotteet ja palvelut: Käytetty perusavaimen tietotyyppi oli INTEGER joka voi sisältää arvoja väliltä 0 – 4294967295.

Tilaukset ja tarjoukset: Käytetty perusavaimen tietotyyppi oli INTEGER joka voi sisältää arvoja väliltä 0 – 4294967295.

Tuotekategoriat: Tuotekategorioita uskottiin olevan enintään 255. Käytetty perusavaimen tietotyyppi oli TINYINT joka voi sisältää arvoja väliltä 0 – 255.

Tuotelaadut: Käytettyjä tuotelaatuja olivat rakkikoirat, lypsylehmät, tähtituotteet ja kysymysmerkit. Käytetty perusavaimen tietotyyppi oli TINYINT joka voi sisältää arvoja väliltä 0 – 255.

Valmistajat: Valmistajia arvioitiin olevan tuhansittain. Käytetty perusavaimen tietotyyppi oli SMALLINT joka voi sisältää arvoja väliltä 0 – 65535.

Yhteystyöryhmät: Käytetyt yhteisryhmät liikkuivat muutamassa kymmenessä. Käytetty perusavaimen tietotyyppi oli TINYINT joka voi sisältää arvoja väliltä 0 – 255.