Jari Juntunen ja Marko Kotakorva

ASIAKAS- JA HUOLTOTIETOKANTA

Opinnäytetyö
Kajaanin ammattikorkeakoulu
Luonnontieteiden ala
Tietojenkäsittelyn koulutusohjelma
Kevät 2010

Kajaanin ammattikorkeakoulu

OPINNÄYTETYÖ
TIIVISTELMÄ

Koulutusala Luonnontieteiden ala	Koulutusohjelma Tietojenkäsittelyn koulutusohjelma
Tekijä(t) Jari Juntunen & Marko Kotakorva	
Työn nimi Asiakas- ja huoltotietokanta	
Vaihtoehtoiset ammattiopinnot Järjestelmän ylläpito	Ohjaaja(t) Matti Härkönen
	Toimeksiantaja Kainuun Puhelinosuuskunta (KPO)
Aika Kevät 2010	Sivumäärä ja liitteet 56+1

Tämän opinnäytetyön tavoite oli suunnitella ja toteuttaa asiakas- ja huoltotietokanta sekä sen käyttöliittymä. Työn toimeksiantaja oli Kainuun Puhelinosuuskunta (KPO) ja tietokanta tuli esiasennus- ja huolto-osaston käyttöön. Kainuun Puhelinosuuskunta tarvitsi asiakas- ja huoltotietokannan, jonne työntekijät voivat tallentaa ja dokumentoida huoltotapauksia. Näin tietokannasta voidaan seurata avoimien huoltotapauksien tilaa ja tarkastella valmiiden tapauksien ratkaisuja. Valmiit ratkaisut helpottavat ja nopeuttavat vastaavien ongelmien ratkaisemista.

Opinnäytetyön teoriaosuus käsittelee tietokantoja ja etenkin relaatiotietokantoja sekä SQL Server ja Visual Basic teoriaa. Tietokantojen rakenteita tutkittiin suunnittelijan näkökulmasta, eli mitä tietokannan rakentajan tulisi ottaa huomioon erilaisia tietokantoja tehdessä. Teoriaosuudessa on myös tutkittu valmisohjelmien ja räätälöityjen ohjelmien eroja tietokantasuunnittelun kannalta. SQL Server ja Visual Basic ohjelmistoja on käsitelty yleisellä tasolla.

Opinnäytetyö toteutettiin Microsoft SQL Server 2008 Express Editiolla sekä Microsoft Visual Basic 2008 Express Editiolla. SQL Server -ohjelmalla toteutettiin tietokanta ja Visual Basicilla tietokannan käyttöliittymä. Ensin perustettiin SQL–palvelin, johon luotiin tietokanta ja Visual Basic:llä rakennettiin tietokantaan selkeä käyttöliittymä.

Tietokannasta tuli selkeä ja tehokas käyttää, eli esimerkiksi haut toimivat nopeasti. Käyttöliittymän toiminnallisuus on hyvä ja helppo omaksua. Käyttöliittymän avulla tietokannan käyttäminen on helppoa joten tietokannan käyttämistä varten ei käyttäjän tarvitse tietää tietokannoista mitään ja silti tietojen syöttäminen ja hakujen suorittaminen onnistuvat vaivatta.

Kieli	Suomi
Asiasanat	SQL Server 2008, Visual Basic, Tietokanta, Käyttöliittymä
Säilytyspaikka	☒Verkkokirjasto Theseus ☒Kajaanin ammattikorkeakoulun kirjasto

Kajaanin ammattikorkeakoulu
University of Applied Sciences

THESIS
ABSTRACT

School Business	Degree Programme Business Information Technology
Author(s) Jari Juntunen & Marko Kotakorva	
Title Customer and maintenance database	
Optional Professional Studies Systems Maintenance	Instructor(s) Matti Härkönen
	Commissioned by Kainuun Puhelinosuuskunta (KPO)
Date Spring 2010	Total Number of Pages and Appendices 56+1

The purpose of this thesis was to design and implement a customer and maintenance database and a user interface needed for it. The work was made for Kainuun Puhelinosuuskunta (KPO), a local telecommunications company, and the database was for its preinstallation and maintenance department. KPO needed a customer and maintenance database where employees could save and document service cases. In this way, the employees can follow the status of open cases and look for solutions from the solved cases. The solved cases make solving similar cases easier and faster in the future.

The theory part of the thesis presents databases, especially relational databases, SQL Server and Visual Basic. The database structures were studied from the point of view of a designer, keeping in mind what a database builder should take into consideration when making different kinds of databases. The theory part covers the differences between application programs and tailored programs considering the database designs. SQL Server and Visual Basic applications were studied in a general level.

The project was made with Microsoft SQL Server 2008 Express Edition and also with Microsoft Visual Basic 2008 Express Edition. The database was made with the SQL Server application and the user interface was made with Visual Basic application. First, an SQL Server was established where a database was created and a user interface for the database was built with Visual Basic.

The database became easy and efficient to use meaning that for example queries work fast. The user interface has good functionality and it is easy to use. The use of the database with the user interface is easy so the user does not need to know anything about databases and still adding data and making queries is simple.

Language of Thesis	Finnish
Keywords	SQL Server 2008, Visual Basic, Database, User Interface
Deposited at	☒Electronic library Theseus ☒Library of Kajaani University of Applied Sciences

SYMBOLILUETTELO

DBMS	Database Management System, tietokannan hallintajärjestelmä.
Entity integrity	Relaatiomallin avaineheyssääntö.
Form	Lomake.
Gb	GigaByte, gigatavu.
GUI	Graphical User Interface, graafinen käyttöliittymä.
Microsoft .Net Framework 3.5 SP1	.Net Framework palvelun päivityspaketti.
NTFS	New Technology File System, tiedostojärjestelmä.
PC	Personal Computer, tietokone.
Referential integrity	Relaatiomallin viite-eheyssääntö.
SA	System Administrator, järjestelmänvalvoja.
Spatial-data	Karttatieto ja muut objektit.
SQL	Structured Query Language, tietokantakieli, jonka avulla käyttäjä voi määritellä tietokannan, ylläpitää sitä ja kohdistaa siihen kyselyitä.
TKHJ	Tietokannan hallintajärjestelmä.
Windows Installer 4.5	Sovellusten asennus- ja muokkauspalvelu Windowsille.
WPF	Windows Presentation Foundation, toimii käyttöliittymän pohjana ja ohjelmoitavana rajapintana.
XML	eXtensible Markup Language, on merkintäkieli tai standardi, jolla tiedon merkitys on kuvattavissa.
Yhdistetty avain	Composite key, Useammasta sarakkeesta koostuva perusavain.

SISÄLLYS

1 JOHDANTO

Nykyään monet yritykset tarvitsevat tietokantoja kasvavan tietomäärän hallitsemiseksi. Palvelualoilla asiakastietojen tallentamiselle on erityisesti tarvetta yrityksissä, jotka tarjoavat jatkuvia palveluita, esimerkiksi Internet-palveluntarjoajat, puhelinoperaattorit, maksullisten tv-palvelujen tarjoajat (katselukortit). Kaikilla edellä mainituilla on tarve tallettaa asiakkaan nimi ja osoite sekä muita mahdollisesti tarvittavia tietoja.

Internet-palveluntarjoajat, kuten Kainuun Puhelinosuuskunta (KPO), joutuvat myös huoltamaan tarjoamiaan laitteita palvelun jatkumiseksi. Koska asiakkaiden laitteita saapuu huollettavaksi, täytyy tapaukset ja laitteet dokumentoida, jotta laitteiden huoltaminen tehostuu ja laitteet saadaan asiakkaille mahdollisimman nopeasti. Tämän takia täytyy olla myös luotettava laite- tai huoltotietokanta asiakastietokannan lisäksi.

Työssä käydään läpi tietokanta teorian perusteita, jotta lukijan olisi helpompi ymmärtää mistä tietokannoissa ja etenkin relaatiotietokannoissa on kyse. Teoria käsittelee myös Microsoft SQL Server 2008 uusia ominaisuuksia vanhempiin versioihin nähden ja myös jonkin verran Microsoft Visual Basic 2008:n ominaisuuksia ja sen Express version mahdollisuuksia.

Työn tavoitteena on helpottaa ja nopeuttaa Kainuun Puhelinosuuskunnan (KPO) yksityisasiakkaille suunnatun huollon toimintaa. Aihe on ajankohtainen koska KPO:n huollossa on paljon harjoittelijoita, ja työntekijöiden vaihtuvuus on suuri. Tietokantaan tuotetaan uutena tietona asiakkaiden tiedot ja laitteiden vikatiedot, mikä laitteessa on ollut vikana, sekä miten se on korjattu. Näin saadaan aikaan samalla ratkaisutietokanta, josta huollon työntekijät voivat katsoa tarvittaessa ohjeita vian korjaamiseen.

2 TOIMEKSIANTAJA

Opinnäytetyön toimeksiantaja oli Kainuun Puhelinosuuskunta, joka on osa Finnet –ryhmää. Finnet –ryhmän muodostavat 27 aluetoimintaa harjoittavaa, itsenäistä puhelinyhtiötä eri puolilla Suomea. (KPO, 2009 a.)

2.1 KPO yrityksenä

KPO:n toiminta-ajatuksena on toteuttaa asiakkaiden tarvitsemat tietoliikenteen ja –tekniikan kokonaispalvelut. Kyseistä työtä tekee noin 60 työntekijää Kajaanissa ja useita kumppaneita Kainuussa, Koillismaalla ja Pohjois-Karjalassa. Kajaanilainen Ebsolut Oy toimii osana KPO Konsernia. Ebsolut Oy on ohjelmistotuotannon palveluyritys, jonka osaaminen ja laaja yhteistyöverkosto mahdollistavat toimeksiantojen toteuttamisen tehokkaasti ja joustavasti. (KPO, 2009 a.)

2.2 Historia

KPO on perustettu vuonna 1898 nimellä Kajaanin Telefooniyhtiö. 1950 nimi vaihtui Kajaanin Puhelinosuuskunnaksi. Yhtiö ryhtyi rakentamaan kaapeli-TV-verkkoa vuonna 1983. Ensimmäinen valokaapeliyhteys tuli käyttöön 1987. KPO aloitti laajakaistaverkon rakentamisen vuonna 2002. Kainuuseen tuli ensimmäinen langaton Wimax-verkko vuonna 2005 ja Lieksaan verkko tuli 2007. Vuonna 2007 toiminimi vaihtui Kainuun Puhelinosuuskunnaksi. Ebsolut Oy liittyi osaksi KPO Konsernia vuonna 2008. Vuonna 2009 alkoi valokuituverkon rakentaminen laajasti Kainuussa. (KPO, 2009 b.)

3 YLEISTÄ TIETOKANNOISTA

Tässä luvussa käsitellään tietokantoja ja selvitetään mitä ne ovat, kuinka ne toimivat ja miksi niitä käytetään.

3.1 Mikä on tietokanta?

Tietokanta on yhteen liittyvän tiedon kokonaisuus. Tiedoilla tarkoitetaan tosiasioita, jotka voidaan kirjata ja joilla on jokin merkitys, esimerkiksi kirjakokoelma tai videorekisteri voivat olla tietokantoja. Tietokannoilla on yleensä seuraavanlaisia ominaisuuksia:

- Tietokannat esittävät jotain reaalimaailman, minimaailman (miniworld, Universe of Discourse), asioita. Minimaailmaan tehdyt muutokset heijastuvat tietokantoihin.
- Tietokanta on yhtenäinen kokoelma tietoa, jolla on jokin merkitys.
- Tietokannat on suunniteltu, rakennettu ja täytetty tiedolla tiettyä tarkoitusta varten. Tietokannoilla on jokin tarkoitettu käyttäjäryhmä ja joitain ennaltalaadittuja ohjelmia, joita käyttäjät käyttävät.

Toisin sanoen:

- Tietokannoilla on jokin lähde, joista niiden sisältämä tieto on peräisin.
- Tietokannoilla on jotain tekemistä todellisen maailman muutosten kanssa.
- Tietokannoilla on käyttäjiä jotka ovat kiinnostuneita sen sisällöstä.

Tietokanta voi olla erittäin pieni ja yksinkertainen tai vastaavasti erittäin suuri ja monimutkainen, esimerkiksi tavallisen henkilön puhelinnumeromuistio on hyvin pieni ja yksinkertainen tietokanta. Tallennettavaa tietoa ovat yleensä vain nimet, puhelinnumerot ja mahdollisesti osoitteet. Tällainen tietokanta ei tarvitse kovin montaa kymmentä kilotavua tallennustilaa, eikä se myöskään vaadi monimutkaisia haku-, lisäys- ja poistotoimintoja. Altavistan hakukone (http://www.altavista.com), joka ylläpitää hakusanastoa koko WWW:n sisällöstä, on vaatimuksiltaan aivan eri luokkaa. Pelkkää levytilaa kuluu yli 200 gigatavua muista laitteistovaati-

muksista ja tietokantaohjelmiston toiminnoista puhumattakaan. Altavistan tietokanta ei ole maailman suurin, ei edes lähellä suurinta. (Ekonoja, A., Lahtonen, T. & Mäntylä, J. 2004 a.)

Tietokannalle asetetaan seuraavia vaatimuksia:

- Kukin tieto tallentuu kannassa vain yhteen paikkaan, näin ollen tietokannassa ei esiinny turhaa toistoa.
- Tietoa voidaan hakea erilaisin perustein ja myös sellaisin, joita ei tietokantaa suunniteltaessa ole voitu ennakoida.
- Tietokannan rakenteen muuttaminen tulisi olla joustavaa.
- Hyväksikäyttö ja sovellusohjelmat ovat riippumattomia tietojen fyysisestä tallennusrakenteesta, jota kutsutaan tietoriippumattomuudeksi.

3.2 Relaatiotietokanta

Relaatiomallin esitteli E.F. Codd vuonna 1970. Tämä oli siihenastisista tietokantamalleista yksinkertaisin ja joustavin sekä se toteutti parhaiten tietokannalle asetettavat vaatimukset. Relaatiomallin huonona puolena oli suuri koneresurssien tarve. (Ekonoja, A., Lahtonen, T. & Mäntylä, J. 2004 b.)

Relaatiotietokannoissa tiedot esitetään tauluina (table), joita kutsutaan myös relaatioiksi, ja yhtä riviä kutsutaan tietueeksi (record). Taulun jokaisella rivillä on yhtä monta kenttää (field). Jokaisella rivillä täytyy olla yksikäsitteinen perusavain, joka vastaa jotain reaalimaailman kohdetta. Jokaiseen kohteeseen liitetään vain siihen välittömästi liittyvät ominaisuudet. Kukin yksittäinen tieto relaatiokannassa voidaan hakea ilmoittamalla taulun nimi, perusavaimen kentän nimi ja avaimen arvo sekä haettavan tiedon kentän nimi. Lisäksi on olemassa useita muita tapoja hakea tietoa. Relaatiotietokannasta tietoa haetaan pelkästään tiedon nimien ja arvojen perusteella. Ei koskaan tiedon sijainnin ja järjestyksen mukaan. (Ekonoja, A., Lahtonen, T. & Mäntylä, J. 2004 b.)

3.3 Relaatiotietokantojen peruskäsitteitä

Tässä luvussa käsitellään relaatiotietokantojen peruskäsitteitä esimerkki tietokannan avulla ja selvitetään erilaisten käsitteiden ominaisuuksia, sekä niiden käyttämistä tietokannassa.

Esimerkkitietokantana toimii työntekijärekisteri, johon halutaan tallentaa etunimi, sukunimi ja työntekijänumero. Näitä tallennettavia tietoja kutsutaan kentiksi. Kenttä on pienin osa, josta alkaa muodostua varsinainen tietokanta. (Ekonoja, A., Lahtonen, T. & Mäntylä, J. 2004 c.)

Etunimi
Erkki

Taulukko 1. Kenttä.

Kaikki kentät muodostavat yhdessä yhden rivin eli tietueen.

TyontekijaID	**Etunimi**	**Sukunimi**
1	Erkki	Esimerkki

Taulukko 2. Tietue.

Monta tietuetta muodostaa yhden taulun.

TyontekijaID	**Etunimi**	**Sukunimi**
1	Erkki	Esimerkki
2	Maija	Mehiläinen
3	Matti	Näsä

Taulukko 3. Työntekijä-taulu.

3.3.1 Kentät

Kenttiä luotaessa niille määritellään seuraavanlaisia ominaisuuksia:

- Kentän nimi.
- Tietotyyppi, joka voi olla alfanumeerinen (teksti), numeerinen, päivämäärä ja/tai kellonaika, looginen tai bittijono.
- Maksimipituus ja tarkkuus.
- Onko tieto pakollinen.
- Tarkistukset tiedon oikeellisuudelle tai sallittujen arvojoukkojen joukko.
- Oletusarvo, jota käytetään jätettäessä kenttä tyhjäksi.
- Syöttömaski, jolla kuvataan kenttään kelpuutettavat merkit ja merkkien muoto.

Kentän nimenä käytetään selkeää ja kuvaavaa nimeä. Nimessä kannattaa kuitenkin välttää erikoismerkkejä, välilyöntejä sekä skandinaavisia kirjaimia. Kentän maksimipituutta määriteltäessä pitää olla erittäin tarkkana. Hyvin tyypillistä on varata liian vähän tilaa esimerkiksi sukunimien tallentamiseen. Melkein aina kannattaa varata enemmän tilaa kuin tarvitsee, esimerkkinä paperilomakkeet, joiden allekirjoituskenttään ei mahdu kirjoittamaan omaa nimeään. (Ekonoja, A., Lahtonen, T. & Mäntylä, J. 2004 c.)

Tietokantaa suunniteltaessa täytyy päättää, valitaanko kiinteämittainen vai vaihtuvamittainen kenttä. Kiinteämittainen kenttä varaa aina saman verran tilaa riippumatta siitä, kuinka paljon siihen tallennettava tieto oikeasti tarvitsee. Vaihtuvamittaisessa kentässä tilanvaraus muuttuu tarpeen mukaan, kuitenkin ylittämättä määriteltyä maksimipituutta. Kiinteämittainen kenttä on tietokantaohjelmiston kannalta kevyempi ja helpompi, mutta se voi johtaa tilanhukkaan. Kiinteän kentän pohjalta ohjelmiston on helpompi varata tarvitsemansa muisti- ja levytila sekä laskea tietueiden ja kenttien paikkoja muistissa. Tietokannat tosin suunnitellaan käytettäväksi ihmisen ehdoilla. Kenttäkohtaisesti on siis tarkkaan harkittava, käyttääkö kiinteä- vai vaihtuvamittaista kenttää. (Ekonoja, A., Lahtonen, T. & Mäntylä, J. 2004 c.)

Oikeellisuustarkistuksen avulla voidaan määritellä numeerisille kentille arvoväli joka on sallittu (esim. luvut yhdestä kymmeneen) tai jokin laskennallinen tarkistus (esim. <100). Useissa

ohjelmistoissa voidaan määritellä myös syöttömalli. Oikeellisuustarkistuksien määrittely on aina erittäin tuotekohtaista. (Ekonoja, A., Lahtonen, T. & Mäntylä, J. 2004 c.)

Mahdollisimman moni kenttä kannattaisi määritellä pakolliseksi tai käyttää oletusarvoa. Näin vältetään NULL-ongelmia, NULL on merkintätapa puuttuvalle arvolle. NULL ei tarkoita nollaa tai välilyöntiä. Se tarkoittaa tuntematonta arvoa eli se voisi olla mikä tahansa tai ei mikään. (Ekonoja, A., Lahtonen, T. & Mäntylä, J. 2004 c.)

3.3.2 Taulut ja tietueet

Yksi tai useampi kenttä muodostaa tietueen. Taulun yksi rivi on tietue. Taulussa voi olla useampia tietueita tai ei yhtään tietuetta (tyhjä taulu). Tauluja nimettäessä on hyvä noudattaa samaa periaatetta kuin kenttiä nimettäessä, eli vältetään välilyöntejä, erikoismerkkejä ja skandinaavisia kirjaimia. (Ekonoja, A., Lahtonen, T. & Mäntylä, J. 2004 c.)

3.3.3 Avaimet

Jokaiselle taululle täytyy määritellä perusavain, joka yksilöi kyseisen taulun sisältämät tietueet, esimerkiksi jokaisella suomalaisella on oma yksilöllinen henkilötunnus. Vastaavasti täytyy jokaisella taulussa olevalla tietueella olla kenttä tai kenttien yhdistelmä, jollaista ei ole yhdelläkään samassa taulussa olevalla tietueella. Jokaisella tietueella pitää olla yksilöllinen uniikki perusavaimen arvo eli kahta samaa perusavaimen arvoa ei taulussa saa eikä voi olla. (Ekonoja, A., Lahtonen, T. & Mäntylä, J. 2004 c.)

Perusavain muodostetaan yhdestä tai useammasta kentästä. Useammasta kuin yhdestä sarakkeesta koostuvaa perusavainta kutsutaan yhdistetyksi avaimeksi. Perusavain ei saa puuttua eli se ei saa saada arvoa NULL. Tätä sanotaan perusavaimen eheydeksi. (Ekonoja, A., Lahtonen, T. & Mäntylä, J. 2004 c.)

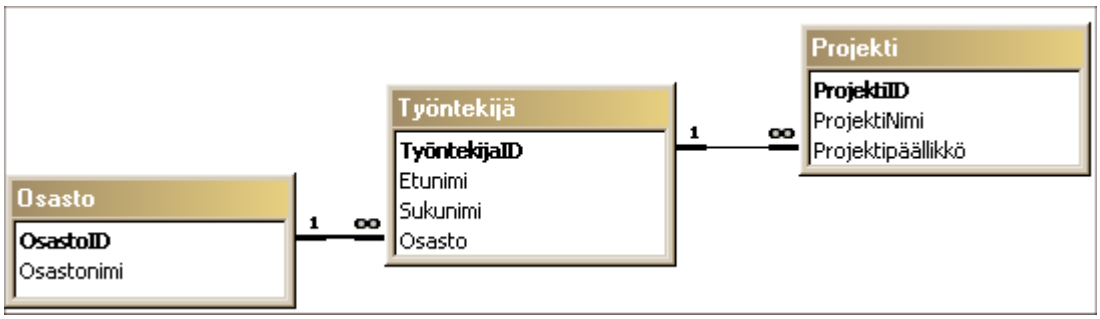
Tietokannanhallintajärjestelmä säilyttää taulun järjestyksessä perusavaimen mukaan. Järjestetyn taulun käsitteleminen on tehokkaampaa kuin järjestämättömän, esimerkkinä tavallisen puhelinluettelon käyttäminen: Puhelinluettelon sisältämät tiedot on järjestetty nimien mukaan eli nimi toimii hieman vastaavasti kuin perusavain toimii tietokannassa. Näin tietyn ihmisen löytäminen puhelinluettelosta onnistuu näppärästi. Jos nimet olisivat puhelinluettelos-

sa sattumanvaraisessa järjestyksessä, olisi hakeminen huomattavasti hankalampaa ja hitaampaa. (Ekonoja, A., Lahtonen, T. & Mäntylä, J. 2004 c.)

Tauluissa esiintyy myös toisenlaisia avaimia kuin perusavaimia, ja joitakin kenttiä määritellään toissijaisiksi avaimiksi. Tietokanta pitää myös automaattisesti yllä järjestysindeksiä avainkentistä. Haettaessa taulun tiedot järjestettynä jonkin avainkentän mukaan tietokantaohjelma pystyy tehokkaammin toteuttamaan haun kuin, jos kyseessä olisi tavallinen kenttä. Toissijaisiksi avaimiksi määritellään juurikin sellaisia kenttiä, joiden mukaan tieto halutaan mahdollisesti järjestellä, esimerkiksi sukunimi-kenttä voisi olla sellainen. Toissijaiset avaimet nopeuttavat taulusta haettavan tiedon järjestämistä. Tosin ne hieman hidastavat päivitys- ja poistooperaatioita, koska näiden tapahtumien yhteydessä täytyy päivittää avainkenttiin liittyvät järjestysindeksitiedot. (Ekonoja, A., Lahtonen, T. & Mäntylä, J. 2004 c.)

3.3.4 Viite-eheys

Taulun kentistä voidaan viitata toisen taulun perusavaimeen tai toissijaiseen avaimeen. Tätä kutsutaan viite-eheydeksi ja kyseinen kenttä määritellään yleensä myös avaimeksi, jota kutsutaan viiteavaimeksi. Viite-eheys määrää, että jokaista viittaavassa taulussa (”lapsi”) esiintyvää viiteavaimen arvoa täytyy vastata sama perusavaimen arvo viittauksen kohteena olevassa taulussa (”isä”). Esimerkiksi yrittäessä syöttää Työntekijä-tauluun (”lapsi”) arvoja, joita vastaavia ei ole Osasto-taulussa, viite-eheys pakottaa syöttämään jotain kelvollista ennen kuin syöttö voidaan hyväksyä (Kuva 1.). Sama tapahtuu, mikäli Projekti-taulun projektipäällikkökenttään yritetään syöttää arvoa, jota ei löydy Työntekijä-taulun tyontekijaID-kentästä. Viite-eheys ei koske NULL-arvoja, joten yleensä on syytä kieltää NULL-arvojen esiintyminen kyseisessä lapsi-taulun kentässä. (Ekonoja, A., Lahtonen, T. & Mäntylä, J. 2004 c.)



Kuva 1. Esimerkki tietokannan viite-eheydet. (Ekonoja, A., Lahtonen, T. & Mäntylä, J. 2004 c.)

Seuraavissa tauluissa on edellisessä kuvassa määriteltyjen viite-eheyksien mukaisen tietokannan esimerkkisisältöä.

OsastoID	**Osastonimi**
0	Palkkahallinto
1	Tuotanto
2	Dokumentointi

Taulukko 4. Osasto-taulu.

TyontekijaID	**Etunimi**	**Sukunimi**	**Osasto**
1	Erkki	Esimerkki	1
2	Maija	Mehiläinen	0
3	Matti	Näsä	2

Taulukko 5. Työntekijä-taulu.

ProjektiID	**ProjektiNimi**	**Projektipaallikko**
0	3D-Sovellus	1
1	Tietokantaprojekti	1

Taulukko 6. Projekti-taulu.

”Isä”-taulussa olevia tietoja voidaan kuitenkin yrittää muuttaa ja poistaa. Jos muuttaminen tai poistaminen on aiheuttamassa viite-eheyden rikkoutumisen, on olemassa muutamia perusperiaatteita joiden mukaan muutoksia tulee tehdä.

- Viiteavaimen nollaus. Kaikki isä-taulusta poistuvaan tai isä-taulussa muuttuvaan perusavaimeen viittaavat lapsi-taulun viiteavaimet muutetaan NULL arvoisiksi. Soveltuu vain kenttiin, joilta ei ole kielletty NOTNULL arvoa.

- Oletusarvon asettaminen. Kaikki isä-taulusta poistuvaan tai isä-taulussa muuttuvaan perusavaimeen viittaavat lapsi-taulun viiteavaimet saavat oletusarvonsa. Tämä edellyttää, että isä-taulusta löytyy oletusarvoa vastaava tietue, ellei oletusarvoksi ole määritelty NULL.

- Johdannaismuutos. Kaikki isä-taulusta poistuvaan perusavaimeen viittaavat lapsi-taulun viiteavaimet ja vastaavat tietueet poistetaan. Isä-taulun muuttuvaan perusavaimeen viittaavan lapsi-taulun viiteavaimen arvo muutetaan uutta isä-taulun perusavainta vastaavaksi.

- Rajoitettu muutos. Vain sellaisia isä-taulun perusavaimia, joihin ei ole viitattu missään lapsi-taulussa, voidaan poistaa tai muuttaa.

Esimerkissä on kaikissa viite-eheyksissä voimassa johdannaismuutos. Muuttamalla tuotanto-osaston TuontantoID:ksi 10, muuttuvat viite-eheys säännön mukaisesti kaikki työntekijä-taulussa olevat kentät, joissa on viitattu kyseiseen Osasto-taulun tietueeseen. (Ekonoja, A., Lahtonen, T. & Mäntylä, J. 2004 c.)

OsastoID	**Osastonimi**
0	Palkkahallinto
10	Tuotanto
2	Dokumentointi

Taulukko 7. Osasto-taulu, jossa tuotanto-osaston OsastoID:ksi on muutettu 10.

TyontekijaID	**Etunimi**	**Sukunimi**	**Osasto**
1	Erkki	Esimerkki	10
2	Maija	Mehiläinen	0
3	Matti	Näsä	2

Taulukko 8. Työntekijä-taulu, jossa Erkin osaston tunnus on päivittynyt.

Samoin jos TyontekijaID-kentän arvo 1 muutetaan arvoksi 15, muuttuvat vastaavat arvot Projekti-taulun Projektipaallikko-kentässä. (Ekonoja, A., Lahtonen, T. & Mäntylä, J. 2004 c.)

3.4 Tietokannanhallintajärjestelmät

Tietokannanhallintajärjestelmät kuten Ingres, Oracle, DB2, Informix, SQL-Server, MySQL, PostgreSQL, Firebird ja Solid sekä henkilökohtaiset tietokantaohjelmat (Paradox, Access) ovat perustaltaan aivan samanlaisia. Tietokannanhallintajärjestelmiin on kuitenkin rakennettu sellaisia ominaisuuksia kuten monipuoliset tietoturvaominaisuudet, elvytystekniikka ja transaktiot, joita ei henkilökohtaisissa tietokantaohjelmissa ole ainakaan samassa laajuudessa. Henkilökohtaisia tietokantaohjelmia ei ole myöskään suunniteltu yhtä suuren tietomäärän tallentamiseen ja käsittelyyn kuin ”oikeita” tietokannanhallintajärjestelmiä. (Ekonoja, A., Lahtonen, T. & Mäntylä, J. 2004 d.)

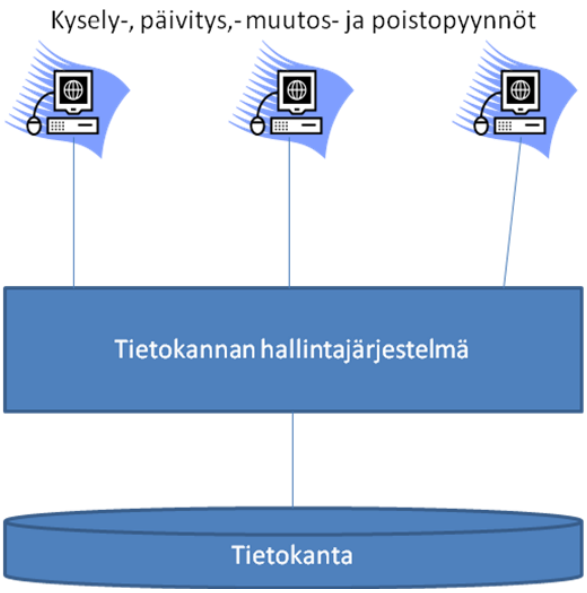
Tietokannan hallintajärjestelmäperusvaatimukset ovat:

- Perusoperaatiot (tallennus, haku, päivitys).
- Tietoriippumattomuus.
- Yhteiskäytön mahdollisuus.
- Ylimäärättömyys.
- Tiedon eheys.
- Turvaaminen.
- Tehokkuus ja hyvä suorituskyky.
- Yhteensopivuus.
- Skaalautuvuus.

3.5 Yhteenveto

Yritysten kannalta tiedot (asiakas, laite, …) ovat erittäin tärkeitä ja niiden varastointi vaatii investointeja. Tietojärjestelmät käyttävät tietokantatekniikkaa tietojen tallentamiseen. Useat yritykset ovatkin täysin riippuvaisia tietokannoistaan. Päätökset syntyvät toimintaa kuvaavista tiedoista. Tietojen tulee olla tallennettu järkevässä muodossa ja niiden tulee olla myös helposti ja nopeasti saatavilla. Tiedonhallinnalla tarkoitetaan kaikkea tallennetun tiedon määrittelemiseen ja käyttämiseen liittyvää toimintaa, esimerkiksi voidaan mainita yrityksen materiaalit ja henkilöstö, jotka ovat tärkeitä resursseja, voidaan puhua resurssien hallinnasta. Tietoresurssin hallinta on tiedonhallintaa. (Hovi, Huotari & Lahdenmäki, 2005, 4.)

Yleisesti tietokanta on loogisesti yhteenkuuluvien, tallennettujen tietojen joukko, jota käsitellään tietokantakielellä (SQL). Tietoja hallinnoidaan erityisillä ohjelmistoilla, eli tietokannan hallintajärjestelmillä (TKHJ, Database Management System, DBMS, Kuva 2.), esimerkiksi: Microsoft SQL Server:llä, DB2:llä, MySQL:llä, Access:lla ja Oracle:lla. Hallintajärjestelmät ovat erittäin monipuolisia ja isoja ohjelmistoja. Tiedot tallennetaan tietokantaan muutosjoustavuuden lisäämiseksi, suorituskyvyn parantamiseksi, tietoeheyden turvaamiseksi sekä sovellusohjelmoinnin helpottamiseksi. Jos TKHJ:ta ei olisi, joutuisimme käyttämään tiedostoja, jolloin monimutkaisten tietokokonaisuuksien ohjelmointi olisi monta kertaa työläämpää, sisällön eheys olisi paljon heikompi ja tietojen hakeminen vaikeampaa. (Hovi, Huotari & Lahdenmäki, 2005, 4.)



Kuva 2. Hallintajärjestelmä hoitaa tietokantaan kohdistuvat operaatiot.

Ennen SQL:ää käytetyt tietokantamallit olivat rakenteeltaan verkkomallisia, esimerkiksi Total, IDMS, DMIV, MDBS ja hierarkkisia hallintajärjestelmiä kuten DL/1. Nykyään tietokannan hallintajärjestelmät ovat valtaosin SQL-pohjaisia relaatiotietokantoja. Relaatiotietokantojen käyttäminen ja muokkaaminen on helpompaa kuin perinteisten hierarkkisten ja verkkomallisten tietokantojen. Suorituskykyongelmat on voitettu, ja nykyisin kaikkein monimutkaisimmat ja vaativimmat järjestelmät tehdään relaatiotietokannoilla. Relaatiotietokannoilla toteutetaan sekä operatiivisia sovelluksia että tietovarastoja. (Hovi, Huotari & Lahdenmäki, 2005, 5.)

Coddin oivallus relaatiomallin yhteydessä oli, että tietoja käsitellään joukko-opillisesti. Joukko-operaatiolla voidaan käsitellä koko yksittäinen taulu tai useita tauluja. Joukko-opillisuus käsittää perinteisistä kyselykielistä poiketen myös päivityksiä. Yhdellä päivityskäskyllä voi esimerkiksi päivittää ison joukon rivejä. (Hovi, Huotari & Lahdenmäki, 2005, 10.)

Joukko-opillisuus toteutetaan relaatiotietokannoissa SQL-kielellä. SQL on tehokas ja monipuolinen tietokantakieli. Sillä kerrotaan mitä haetaan eikä miten haetaan. SQL:n avulla voidaan lukea ja päivittää tietokantaa sekä poistaa ja lisätä tietoa. Sitä voidaan käyttää vuorovaikutteisesti päätteeltä tai työasemalta. SQL voidaan myös upottaa ohjelmointikieleen, esimerkiksi Java, C#, Cobol tai Visual Basic, jota käytetään tässä työssä. (Hovi, Huotari & Lahdenmäki, 2005, 10.)

4 TIETOKANTOJEN SUUNNITTELU

Tässä luvussa käsitellään tietokantojen suunnittelua ja siihen liittyviä ongelmia ja tavoitteita. Tutustumme erilaisten tietokantojen vaatimuksiin ja rakenteiden tavoitteisiin.

4.1 Suunnittelun vaiheet

Tietokantojen suunnitteluun on olemassa yhtä monta erilaista tapaa kuin on suunnittelijoitakin. Suunnittelun eri vaiheita esimerkiksi:

- Vaatimusten määrittelyssä ja analyysissä selvitetään kirjalliseen materiaaliin tutustumalla, haastatteluin ja keskusteluin järjestelmältä vaadittavat ominaisuudet.
- Käsitteellisessä mallintamisessa laaditaan käsitekaavio, jonka avulla kuvataan tietokannan sisältö ja rakenne.
- Tietokannan loogisessa suunnitteluvaiheessa käsitekaavan avulla laaditaan sisällöstä ja rakenteesta relaatiokaavio käytettävissä olevalla DDL:llä (Data Definition Language).
- Toteutusvaiheessa tehdään edellisten osien päälle pieni käyttöliittymä ja testataan toteutuksen toimivuus. (Ekonoja, A., Lahtonen, T. & Mäntylä, J. 2004 e.)

4.2 Käsitteellinen mallintaminen

Käsitteelliseen mallintamiseen on useita menetelmiä. Tunnetuin ja käytetyin on Chenin vuonna 1976 esitelty ER-malli. ER-mallia on myöhemmin laajennettu ja muutettu suuntaan jos toiseen. Seuraavaksi esitellään yksi tapa tehdä ER-malli. (Ekonoja, A., Lahtonen, T. & Mäntylä, J. 2004 f.)

ER-malli koostuu useasta erilaisesta osasta, joita kuvataan tietyillä kuvioilla, jotka yhdistetään toisiinsa tietyillä suhteilla. Näistä muokataan ER-diagrammi, josta nähdään kaikki tarvittavat asiat ja niiden toimintatavat sekä suhteet toisiinsa nähden. (Ekonoja, A., Lahtonen, T. & Mäntylä, J. 2004 f.)

4.2.1 Kohde

Kohteella (entity) tarkoitetaan tunnistettavissa olevaa tapahtumaa tai asiaa. Heikoksi kohteeksi (weak entity) kutsutaan kohdetta, joka on riippuvainen toisesta kohteesta. Se ei siis voi olla olemassa, jos toista kohdetta ei ole olemassa, esimerkiksi tenttitulosta ei voi olla ilman opiskelijaa. (Ekonoja, A., Lahtonen, T. & Mäntylä, J. 2004 f.)

4.2.2 Ominaisuudet

Kaikilla samantyyppisillä kohteilla on tiettyjä yhteisiä ominaisuuksia, esimerkiksi opiskelijoilla on kaikilla nimi, sosiaaliturvatunnus ja osoite. Kaikki ominaisuudet saavat arvonsa tietystä arvojoukosta. Ominaisuudet voivat olla yksittäisiä tai useammasta osasta koottuja, esimerkiksi nimi koostuu etu- ja sukunimestä. Ominaisuudet voivat olla avaimia, jotka yksilöivät kohteen eli ovat 1. uniikkeja ainakin jossakin yhteyksissä. Ominaisuudet voivat olla yksi- tai moniarvoisia ja ne voivat saada tuntemattoman arvon tai tyhjän arvon. Ominaisuuksien arvo voi olla johdettu muista tiedoista, esimerkiksi tilausten kokonaismäärä lasketaan yksittäistilausten kappalemäärästä. (Ekonoja, A., Lahtonen, T. & Mäntylä, J. 2004 f.)

4.2.3 Suhde

Suhteella tarkoitetaan vähintään kahden kohteen välillä olevaa riippuvuutta. Kardinaalisuus kertoo, kuinka moneen suhteeseen kohde voi samalla hetkellä osallistua ja kuinka monta kohdetta voi samalla hetkellä osallistua tiettyyn suhteeseen. (Ekonoja, A., Lahtonen, T. & Mäntylä, J. 2004 f.)

Kardinaalisuus voi olla:

- (1-to-1) yhden suhde yhteen, esimerkiksi avioliitto (henkilö ja henkilö).
- (1-to-M) yhden suhde moneen tai (M-to-1) monen suhde yhteen, esimerkiksi kuntalaiset (kunta ja henkilö).
- (M-to-M) monen suhde moneen, esimerkiksi kirjastokirjan lainaaminen (kirja ja henkilö)

Kardinaalisuudet merkitään ER-kaavioon merkitsemällä jokaisen suhteen ja kohteen väliin 1 tai M. (Ekonoja, A., Lahtonen, T. & Mäntylä, J. 2004 g.)

4.3 Tietokantojen käyttötapoja

Tietokantoja voidaan käyttää monella tavalla. Pankkien tilitietokantoihin kohdistuu raskaita tapahtumakuormia. Konttorit ja pankkiautomaatit lähettävät tapahtumia, pääasiassa tililtäottotapahtumia. Suomalaisissa pankeissa saattaa tapahtua yli 100 tapahtumaa sekunnissa, tämä vaatii laiteresursseja eli laskentatehoja ja suorituskykyä. (Hovi, A., Huotari, J. & Lahdenmäki, T. 2005, 15.)

Tällaiselle tapahtumankäsittelylle on ominaista, että kerralla käsitellään hyvin pientä osaa tietokannasta, eli yleensä yhtä tiliä ja siihen liittyvää tilitapahtumaa. Tällainen tietokannan käyttö kuuluu operatiiviseen eli tuotannolliseen tietojenkäsittelyyn. Operatiivisia tietokantoja päivitetään usein. Esimerkkejä operatiivisista järjestelmistä ovat laskutusjärjestelmä, tilauksenkäsittelyjärjestelmä, paikanvarausjärjestelmä ja kirjanpitosovellukset. Pankeissa tilitiedot yleensä pidetään ajan tasalla, joten Kajaanissa tehty pankkiautomaattinosto näkyy välittömästi porilaisessa konttorissa tehdyssä tilikyselyssä. (Hovi, A., Huotari, J. & Lahdenmäki, T. 2005, 15.)

Tietokantoihin kohdistuu eräajoja. Eräajossa ihminen ei osallistu toimintaan samoin kuin tapahtumankäsittelyssä päätteen kautta, esimerkiksi pankin tiliotteiden tulostus tai tietovarastokannan lataus operatiivisessa järjestelmässä on eräajoa. Voidaan siis todeta, että tietokantoihin kohdistuu monenlaista käyttöä: tapahtumankäsittelyä, kyselyitä ja eräajoja. Kaikkia näitä voi tapahtua jopa yhtä aikaa. Tietokannan hallintajärjestelmä mahdollistaa nämä palvelut ja pitää tietokannan kunnossa. (Hovi, A., Huotari, J. & Lahdenmäki, T. 2005, 15.)

Operatiiviset tietokannat voi jakaa kahteen ryhmään:

- Räätälöityihin, yleensä yhdelle yritykselle tehtyjen sovellusten tietokannat.
- Yleisiin, kaupallisten valmisohjelmistojen tietokannat, esimerkiksi taloushallinto, toiminnanohjaus, henkilöstöhallinta.

Räätälöidyt järjestelmät suunnitellaan juuri tietylle käyttäjäryhmälle ja yritykselle ja tietokannan rakennekin tehdään räätälöidysti. Valmisohjelmistoissa sen sijaan tavoitellaan mahdolli-

simman suurta yleisyyttä, jotta järjestelmä sopisi erilaisille asiakkaille. Tällöin tietokannan rakenteen tulee olla hyvin yleiskäyttöinen. Räätälöidyt tietokannat ovat yleensä selkeitä ja niissä on pyritty ohjelmointimukavuuteen sekä suorituskykyyn. Näitä rakenteita voidaan itse muuttaa tarpeen mukaan. Sarakkeet ja taulut nimetään selkeästi tutuilla termeillä. (Hovi, A., Huotari, J. & Lahdenmäki, T. 2005, 15 - 16.)

Valmisohjelmistojen tietokantojen halutaan sopivan useisiin ympäristöihin ilman, että tietokannan rakennetta tarvitsisi muuttaa. Niinpä tietokantojen rakenteista tulee usein monimutkaisia ja taulujen sekä sarakkeiden nimistä yleisiä ja siksi ei-kuvaavia, jolloin tietokannan tutkiminen ja lukeminen voi olla hankalaa. (Hovi, A., Huotari, J. & Lahdenmäki, T. 2005, 16.)

Operatiiviset tietokannat tukevat mainiosti operatiivisia järjestelmiä, mutta tietojen analysoinnin ja raportoinnin osalta on etenkin vanhemmissa järjestelmissä havaittu seuraavia puutteita:

- On vaikeaa yhdistellä eri järjestelmien tietoja, jotka voivat olla eri koneilla, eri tietokantajärjestelmissä ja sisällöltään yhteen sopimattomia.

- Operatiivisissa järjestelmissä ei ole tarpeeksi historiatietoa tallessa trendi- ja aikasarja analyysejä varten.

- Operatiivisia järjestelmiä ei voi kuormittaa kesken työpäivää raskailla kyselyillä, joita analyysejä varten tarvitaan.

- Tiedot ovat tietokannassa liian vaikeissa rakenteissa ja käyttäjät eivät kykene itse analysoimaan ja tutkimaan niitä, siksi tarvitaan IT-ammattilainen tueksi ohjelmoimaan kyselyt.

Ongelmien ratkaisuksi on kehitelty oma ratkaisunsa, tietovarastointi. Tietovarastoajattelussa tietoja poimitaan, ladataan ja muokataan omaan erilliseen tietovarastokantaansa. Tämä kanta suunnitellaan helppoja kyselyjä ja hyvää vastausaikaa silmälläpitäen. Lisäksi tietovarastoon tallennetaan historiaa, jotta käyttäjä voi tehdä trendianalyysejä. (Hovi, A., Huotari, J. & Lahdenmäki, T. 2005, 16.)

4.4 Erilaisten tietokantojen tavoitteita

Tietokantoja suunniteltaessa on hyvä muistaa yleiset tavoitteet tietokannan rakenteelle:

- Yleiskäyttöisyys: tietokannan rakenteen tulee olla sellainen, että sitä ei tarvitse muuttaa sovellettaessa sitä erilaisiin ympäristöihin ja eri asiakkaille.
- Kattavuus: sisältää kaikki tarvittavat yhteydet ja tiedot.
- Selkeys ja ymmärrettävyys: yksinkertainen rakenne mahdollistaa helpot kyselyt.
- Eheys: tietojen oikeellisuus, toisteisuuden välttäminen ja sisäinen ristiriidattomuus. (oletusarvot, viite-eheyssäännöt, raja-arvot)
- Muutosjoustavuus: laajennettavuus ilman suuria muutoksia ohjelmiin.
- Ohjelmointimukavuus: selkeät tietorakenteet ja sarakkeiden kiinteä merkitys. (sarakkeen merkitys ei saisi olla riippuvainen toisesta sarakkeesta)
- Suorituskyky eli tehokkuus: riittävän tehokkaat eräajot ja riittävä vastausaika tapahtumille. (Huotari, J & Hovi, A. 2008.)

Tietokantoja suunniteltaessa tulisi ottaa myös huomioon tapauskohtaisia tavoitteita esimerkiksi asiakkaalle räätälöidyn tai valmisohjelmiston tietokannan rakenteelle.

Räätälöidyn tietokannan rakenteen on hyvä olla selkeä ja tarkasti tarpeisiin sovitettu. Taulujen ja tietojen nimet ovat tuttuja, omia termejä ja sarakkeet tarkoittavat yleensä yhtä asiaa. Tietokannan muutosjoustavuuden tulee olla hyvä. Tietokannan hoitaja voi lisätä uusia tauluja ja sarakkeita koskematta jo olemassa oleviin ohjelmiin. Jos esimerkiksi henkilötietoihin tarvitaan uusi tutkinto-sarake, se lisätään henkilötauluun ja sitä on helppo kysellä. Pyritään välttämään taulujen pilkkomista ja yhdistämistä, ne johtavat yleensä nykyohjelmien muutoksiin. Vältetään aputauluja ja erikoisvirityksiä, joilla ehkä parannetaan suorituskykyä jossain määrin, mutta jotka monimutkaistavat tietokannan rakennetta ja ohjelmointia. Tietokannan rakenne voidaan pitää melko yksinkertaisena, jolloin siinä on vähemmän liitoksia, mikä taas on eduksi suorituskyvylle. (Hovi, A., Huotari, J. & Lahdenmäki, T. 2005, 21 - 22. ja Huotari, J & Hovi, A. 2008.)

Valmisohjelmistojen tietokantojen, esimerkiksi SAP ja Baan, on oltava erittäin yleiskäyttöisiä, jotta sama tietokantarakenne voitaisiin monistaa jokaiselle asiakkaalle. Valmisohjelmat sovitetaan erilaisilla säädettävillä parametreilla kuhunkin ympäristöön. Esimerkiksi henkilötiedoissa mahdollisesti tarvittava tutkinto-tieto laitetaan käyttöliittymästä erilliseen luokittelutauluun,

josta se voidaan liittää henkilötauluun. Uusi tieto on näin mukana ilman tietokantamuutosta, mutta tiedon hakeminen on mutkikasta. Ilmaisuvoima voi olla hyvä, mutta tietokannan rakenne on selkeä vain järjestelmää ylläpitäville eksperteille. Sarakkeet nimetään yleisnimillä, jolloin ne voivat eri ympäristöissä tarkoittaa eri asioita. Tämän vuoksi ohjaustaulussa kerrotaan usein mitä toisen taulun sarakkeet merkitsevät. Valmisohjelman tietokannan rakenteen tutkiminen ja omien raporttien teko voi olla hankalaa. (Hovi, A., Huotari, J. & Lahdenmäki, T. 2005, 22. ja Huotari, J & Hovi, A. 2008.)

	Valmisohjelma	**Räätälöity ohjelma**
Hyödyt	- Testattu ohjelmisto, vähemmän virheitä - Olemassa olevat referenssit - Toimivat tukipalvelut - Valmiit rajapinnat toisiin ohjelmistoihin; voi mahdollistaa helpomman integroinnin - Hyvä dokumentointi - Jatkuva tuotekehitys - Projektin ja käyttöönoton nopeus	- Joustava ja täsmälleen yrityksen tarpeisiin soveltuva - Yksilöllinen ja vaikea kopioida - Uusien toimintamallien ja -prosessien mahdollistaja
Haitat	- Jäykkä muutoksille - Voi johtaa yrityksen sopeutumiseen järjestelmään eikä päinvastoin - Räätälöinti voi olla osittain mahdotonta - Kustannukset - Tukipalvelujen jatkuvuus	- Onko saatavilla riittävät tukipalvelut? - Hidas rakentaa - Integroiminen muihin järjestelmiin voi olla vaikeaa - Testauksen laadusta ja syvyydestä ei voi olla varma - Dokumentoinnin taso voi olla puutteellinen - Jatkokehitystyö on kohtuullisen kallista

Taulukko 9. Valmisohjelmien ja räätälöityjen ohjelmien hyödyt ja haitat. (Kettunen, S. 2002, 38.)

Tietovarastokantojen tulee olla hyvin selkeitä, ymmärrettäviä ja helppokäyttöisiä. Rakenteen täytyy tukea helppoja kyselyjä ilman varsinaista ohjelmointia, eli toisinsanoin SQL-tyyppisesti. Tiedoilla on selkeät, kiinteät nimet. Koska tietovarastot ovat yleensä räätälöityjä,

ei niiden yleiskäyttöisyys ole tärkeätä. Muutosjoustavuus ja laajennettavuus ovat sen sijaan tavoiteltuja ominaisuuksia. Tiedon toisteisuutta ei vältellä, sitä päinvastoin suositaan kyselyjen nopeuttamiseksi. Toisteisuus on hallinnassa, koska tietojen ylläpito on erittäin keskitettyä. Tietovarastoissa säilytetään useiden vuosien historiaa, jolloin on oltava tarkempana levytilan käytöstä kuin operatiivisten kantojen kanssa. (Hovi, A., Huotari, J. & Lahdenmäki, T. 2005, 22. & Huotari, J & Hovi, A. 2008.)

Yhteisenä tavoitteena on eheys. Viite-eheyssääntöjen avulla ehkäistään orvoksi jäävien tietojen esiintyminen tietokannoissa. Laajemmin määriteltynä eheys sisältää tietokannan tietojen oikeellisuuden ja sisäisen ristiriidattomuuden kuten tiedon hallitsematon toistaminen tauluissa. Esimerkiksi kurssijärjestelmän opintosuoritus-tauluun halutaan lisätä opiskelijan nimi tehokkuussyistä, mutta jos sitä ei päivitetä, niin kannassa on kaksi eri nimeä samalle opiskelijalle. Väärien tietojen syöttäminen uhkaa koko ajan tietokannan eheyttä. Tällaisia tilanteita varten tulisi suojautua mahdollisuuksien mukaan. Virhealttiutta voi pienentää eri ratkaisuilla, kuten käyttämällä raja-arvoja ja oletusarvoja. Tärkeää on sopia yhteiset pelisäännöt siitä, miten ja mitä tietoa tietokantaan tallennetaan, esimerkiksi erilaiset luokittelut. (Hovi, A., Huotari, J. & Lahdenmäki, T. 2005, 22 - 23.)

Muita tavoitteita:

- Yhteensopivuus jo olemassa olevien tietojärjestelmien tai tietokannan hallintajärjestelmien kanssa.
- Skaalautuvuus laiteympäristöstä tai tietokannan hallintajärjestelmästä toiseen, tämä on erityisen tärkeää valmisohjelmistoille.
- Turvallisuus: tietoihin päästään käsiksi vain käyttöoikeuksien mukaan.

Operatiivisten tietokantojen hajauttamista ei suositella, vaan suuntana on nykyään keskitetyt tietokannat. Sen sijaan tietojen monistamisesta on yleisempää, esimerkiksi tietojen kopiointi kannettaviin tietokoneisiin. Hyvä tietokannan hallintajärjestelmä pitää huolta automaattisesti joistakin edellä mainituista hyvän tietokannan ominaisuuksista, kuten viite-eheydestä ja turvallisuudesta. Nämä asiat tulee kuitenkin määritellä ja suunnitella huolellisesti. (Hovi, A., Huotari, J. & Lahdenmäki, T. 2005, 23.)

5 KEHITYSTYÖKALUT

Tämän opinnäytetyön tekemiseen on käytetty Microsoft SQL Server 2008 Express Edition palvelinohjelmistoa ja Microsoft Visual Basic 2008 Express Edition ohjelmaa.

5.1 Microsoft SQL Server 2008

SQL Server 2008 mahdollistaa tiedon saatavuuden missä tahansa, milloin tahansa, ja mihin laitteeseen tahansa. Se mahdollistaa luotettavamman, skaalautuvamman ja paremmin saatavilla olevan alustan, kuin sen aikaisemmat versiot. SQL Server 2008:sta tulee tuottavampi, älykkäämpi ja se toimii myös tietoalustana kokonaisille yrityksille, joka johtuu kehityksessä tärkeimmillä avainalueilla. (Niteshi, R., Subnivis, P. B. & Wadekar, V. 2008, 3.)

SQL Server 2008 ominaisuudet:

- Enterprise Data Platform, mahdollistaa luotettavamman, turvallisemman, skaalautuvamman alustan. Se parantaa saatavuutta käyttäen parannettua tiedon peilaamista, ennakoitavaa kyselyjen ajamista, tiedon ja varmennusten pakkausta ja monia kehitys ominaisuuksia. Toinen innovatiivinen ominaisuus on hallinta policyjen eli ryhmäkäytäntöjen kautta, joka helpottaa palvelimen hallintaa. (Niteshi, R., Subnivis, P. B. & Wadekar, V. 2008, 3. ja Korhonen, P. 2008.)

- Beyond Relational, tämän kategorian ominaisuudet voivat hallita monen tyyppistä tietoa, jopa sellaista joka on epämuodollista ja ei-relaationaalista. Uudet tietotyypit mahdollistavat karttatiedon, dokumenttien ja kuvien hallinnan. Sovelluskehittäjien on nyt mahdollista suunnitella paikkatietoisia sovelluksia ja parantaa heidän kykyä hallita dokumentteja. (Niteshi, R., Subnivis, P. B. & Wadekar, V. 2008, 3. ja Korhonen, P. 2008.)

- Dynamic Development, .NET Framework 3.5:n käyttäminen vähentää kehittämisen monimutkaisuutta ADO.NET Entity Framework:lla ja LINQ:lla SQL:ään. ADO.NET Entity Framework mahdollistaa kehittäjien suoran vuorovaikutuksen business-entiteettien kanssa. Se mahdollistaa myös tietojen synkronoinnin, eli tietojen

tallentamisen palvelimelle ja synkronoinnin takaisin palvelimelle. (Niteshi, R., Subnivis, P. B. & Wadekar, V. 2008, 3. ja Korhonen, P. 2008.)

- Pervasive Insight, mahdollistaa kaikenlaisen tiedon integroinnin tietovarastoon, skaalautuvan tietojen analysoinnin ja raportointiympäristön ja Office integraatiot sekä omien raporttien tekemisen. (Niteshi, R., Subnivis, P. B. & Wadekar, V. 2008, 3. ja Korhonen, P. 2008.)

5.2 Microsoft SQL Server 2008 Express Edition

Microsoft SQL Server 2008 Express Edition on ilmainen versio Microsoftin uusimmasta SQL-palvelinohjelmistosta. Se soveltuu erityisesti ohjelman käytön opiskeluun, web-sovellusten ja pienten palvelinsovellusten kehittämiseen. (Microsoft. 2009 a.)

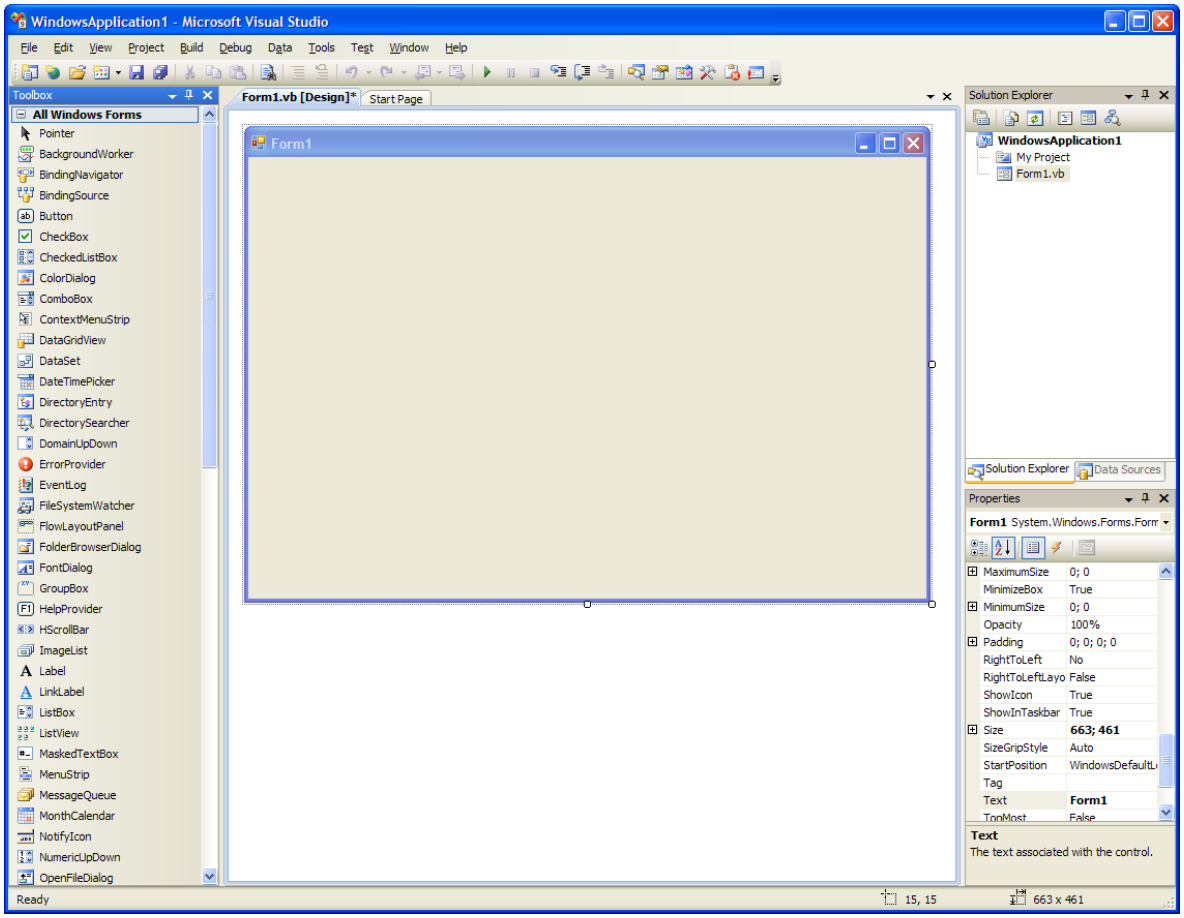
Ominaisuudet:

- Tukee sisäänrakennettuja toimintoja, tapahtumia, funktioita ja näkymiä tietokannassa.
- Tallentaa kaikenlaista dataa ja tukee relaatio-, XML-, FILESTREAM- sekä karttadataa (Spatial-data).
- Suorituskykyisempi kuin edelliset versiot ja helpompi käyttää. Lisäksi se on integroitu Microsoft Office 2007 järjestelmän kanssa SQL Serverin raportointi palveluun (SQL Server Reporting Services).
- Integroitu läheisesti Visual Studioon ja Visual Web Developeriin. (Microsoft. 2009 a.)

5.3 Yleistä Visual Basicista

Visual Basic on Microsoftin vuonna 1991 julkaisema ohjelmointikieli. Se oli merkittävä uudistus vaikeana pidettyyn Windows-ohjelmointiin. Uusien versioiden myötä Visual Basiciin on tullut lukuisia parannuksia, ja sillä pystyy tekemään laajojakin ohjelmia. Visual Basicia ei enää nykyään mielletä aloittelijoille tarkoitetuksi ohjelmointikieleksi, josta täytyisi siirtyä pois ohjelmointitaitojen kehittyessä. (Laaksonen, A. a.)

Tavallisin projektissa oleva tiedosto on lomake (form), joka kuvaa yhtä ohjelmaan kuuluvaa ikkunaa. Se sisältää ikkunassa näkyvät napit, tekstikentät, vierityspalkit ja muut ohjaimet. Lomakkeeseen kuuluu myös melkein aina siihen liittyvää ohjelmakoodia. Kuvassa 3 on muokkaustilassa oleva lomake. (Laaksonen, A. a.)



Kuva 3. Visual Basicin -ohjelmointiympäristö perustilassa.

Lähes jokaiseen Windows-ohjelmaan kuuluu käyttöliittymä: loppukäyttäjälle näkyvä ikkuna ja siinä olevat ohjaimet ja valikot. Käyttöliittymän suunnittelu Visual Basicissa on helppoa, koska suurimman osan ajasta käyttöliittymä on näkyvillä lopullisessa muodossaan jo ohjelmointivaiheessa. Useimpien ohjainten ominaisuudet ja tapahtumat ovat yhteisiä ja niiden tarkoitus on helppo päätellä. (Laaksonen, A. b.)

Visual Basic 2008 on yksi niistä ohjelmointikielistä jotka käyttävät .NET ohjelmistokomponenttikirjastoa. Kuten mikä tahansa puhuttu tai kirjoitettu kieli, Visual Basic:ssä on oma ”lauseoppi” sekä sääntöjä ja varattuja sanoja, joita voi käyttää sovellusten tekemiseen. Visual Basic on suosittu aloittelevien ohjelmoijien keskuudessa, koska jotkut ovat havainneet Visual

Basicin säännöt yksinkertaisemmiksi kuin muiden ohjelmointikielien. Visual Basic 2008:ssa on aiempia versioita käyttäneille tuttu käyttöliittymä ja tuttuja ominaisuuksia. (Pelland, P. 2008, 4.)

Visual Basic 2008 Express Edition, jota on käytetty tässä työssä, on suunniteltu tuotantoa silmälläpitäen. Kuten Visual Studio 2008 täydet versiot, ovat myös Express versiot niin kutsutun nopean kehityksen mallin (RAD, Rapid Application Development) työkaluja. Express versiot ovat helppoja käyttää, helppoja oppia ja virtaviivaisia. Ne sisältävät suurin piirtein samat komponentit kuin Visual Studion täydet versiot, lukuun ottamatta muutamia poikkeuksia. Useimpia ominaisuuksia ja komponentteja on yksinkertaistettu Express-versioissa, jotta oppimiskäyrä ei olisi niin jyrkkä, ja jotta se soveltuisi amatööriohjelmoijien tarpeisiin. (Pelland, P. 2008, 9 - 10.)

5.4 Microsoft Visual Basic 2008 Express Edition

Microsoft Visual Basic 2008 Express Edition on Windows-ohjelmakehitykseen suunnitellun Visual Basic-kielen kehitysympäristö. Ohjelma on karsitumpi kuin kaupallinen versio, mutta esimerkiksi käyttöliittymien teko onnistuu helposti. Windows-ohjelmoinnista kiinnostuneille ohjelma on hyvä vaihtoehto kokeiluun, opetteluun ja ohjelmistokehitykseen. Sen etuna on yhteensopivuus muiden ilmaisten työkalujen kanssa esimerkiksi SQL Server 2008 Express Edition. (Pitkänen 2009.)

Visual Basic Expressillä voi rakentaa seuraavanlaisia sovelluksia:

- Windows sovelluksia. Näissä sovelluksissa on graafinen käyttöliittymä jossa on painikkeita, ikkunoita, valikkoja, työkalurivejä ja niin edelleen, aivan kuten Microsoft Wordissa tai Windows Internet Explorerissa.

- Konsolisovelluksia. Näissä sovelluksissa ei ole graafista käyttöliittymää ja ne käyttävät vain tekstiä vuorovaikutukseen käyttäjän kanssa. Tyypillisesti nämä sovellukset ajetaan komentokehotteessa tai DOS ikkunassa.

- Uudelleen käytettäviä komponentteja tai luokka kirjastoja. Nämä ovat joukko työkaluja jotka on luotu auttamaan muiden sovellusten luomisessa.

Visual Basic Expressillä ei voi rakentaa Web-sivuja tai Web-palveluita. Niiden rakentamiseen tarvitaan Microsoft Visual Web Developer 2008 Express Edition. (Pelland, P. 2008, 10.)

6 SQL SERVER 2008 EXPRESS EDITIONIN KÄYTTÖÖNOTTO

Ohjelmiston saa ladattua sivulta: http://www.microsoft.com/express/Database/.

Käyttäjän tulee varmistaa, että tietokonelaitteisto ja ohjelmisto vastaa seuraavia vaatimuksia:

- Käyttäjän tulee olla järjestelmänvalvoja
- tietokoneeseen on asennettu Windows Installer 4.5
- Tietokoneeseen on asennettu Microsoft .Net Framework 3.5 SP1
- Tiedostojärjestelmä on NTFS.
- Vähintään 2 Gb vapaata kiintolevytilaa.
  - 2 Gb muistia on suositeltavaa, tosin SQL palvelin varaa paljon muistia. Joten on varminta varustaa laitteisto niin suurella muistimäärällä kuin mahdollista.

Tässä työssä on kiintolevy jaettu kahteen asemaan, joista toinen on varattu pelkästään tietokannan data- ja lokitiedostoille. Toiselle eli C asemalle on asennettu tietokoneen käyttöjärjestelmä ja SQL Server asennetaan käyttäjän haluamaan tiedostoon tai oletuksena C:\Program Files\ Microsoft SQL Server polkuun. Asennus tarkistaa, että tietokone soveltuu ohjelman käyttämiseen. Kun tarkistus on valmis, avautuu uusi ikkuna, jossa ilmoitetaan onnistuneesta tarkistuksesta. Tämän jälkeen täytyy valita Install, jotta asennusohjelma asentaa tarvitsemansa komponentit. Seuraavaksi käyttäjä valitsee Perform a new installation of SQL Server 2008. Asennuksen aikana on muutama tärkeä tapahtuma joihin pitää kiinnittää huomiota.

1. Feature Selection

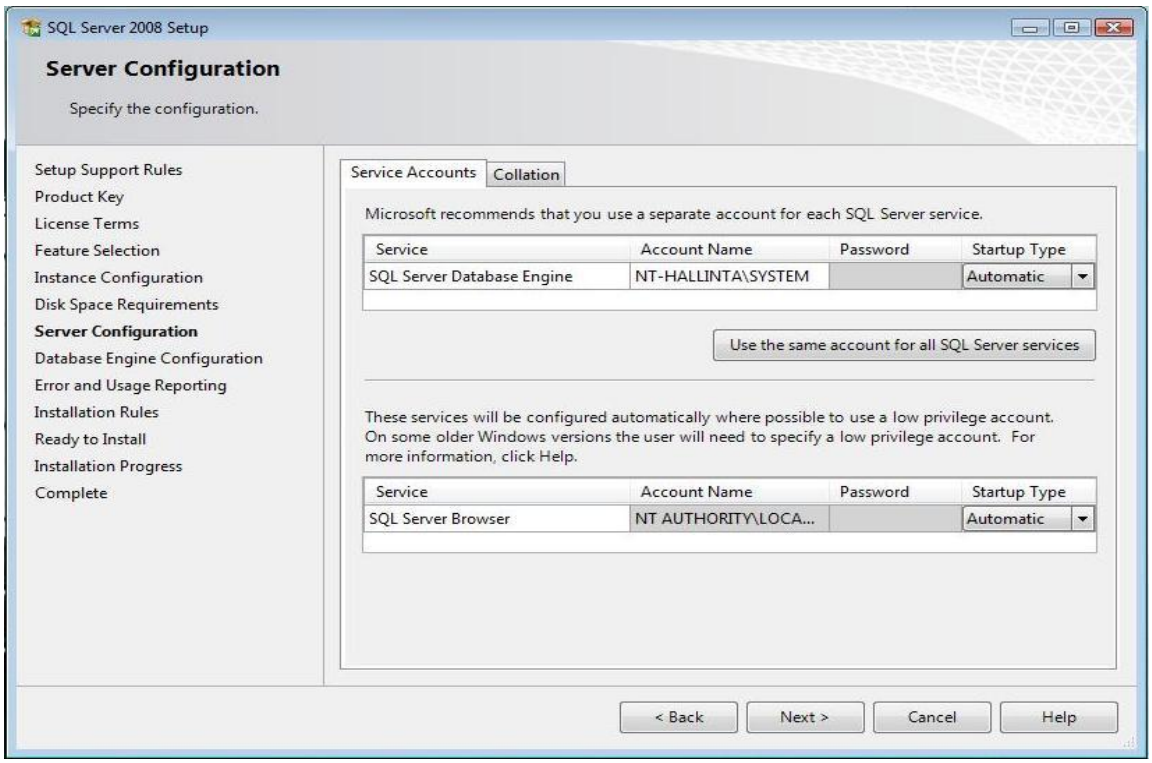
   Tässä vaiheessa täytyy valita mitä ominaisuuksia haluaa SQL-palvelimen sisältävän. Tässä tapauksessa valitaan tietokanta ominaisuudet, eli Database Engine Services.

2. Instance Configuration

Tässä kohdassa määritellään palvelimen nimi. Nimi voi olla joko järjestelmän käyttämä oletusnimi tai käyttäjän määrittelemä. Tässä tapauksessa nimeksi on määritelty HUOLTO, kuin myös instanssin ID:ksi.

3. Server Configuration

Tässä tulee valita service accounts välilehdellä Account Name:n arvoksi NT-HALLINTA\SYSTEM, kuten alla olevassa kuvassa 4 on esitetty.

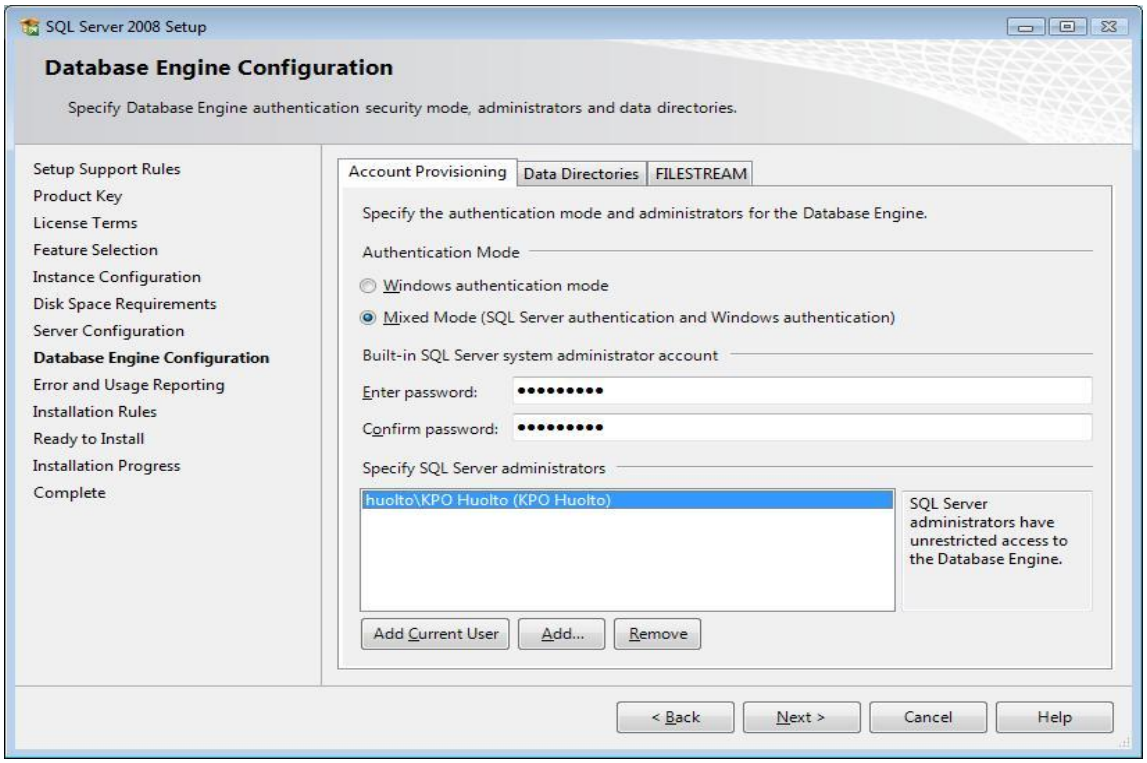


Kuva 4. Server Configuration ikkuna.

4. Database Engine Configuration

Tässä vaiheessa valitaan, että käyttääkö SQL-palvelin Windows-autentikointia vai sekoitettua eli Mixed modea. Windows-autentikointi tarkoittaa, että SQL-palvelinta voivat käyttää windowsissa olevat käyttäjät. Mixed mode tarkoittaa sitä, että palvelinta voivat käyttää windowsissa määritetyt käyttäjät ja palvelimella määritetyt käyttäjät. Tässä tapauksessa on valittu mixed mode ja SQL-palvelimen pääkäyttäjälle, eli SA:lle salasana. Huomioitava on, että SA ei ole sama kuin windowsin järjestelmän valvoja. SA:lla on SQL-palvelimessa kaikki oikeudet. Kuvassa 5 on myös annettu järjestelmä-

valvoja oikeudet KPO Huolto käyttäjälle, koska sillä käyttäjätilillä tullaan käyttämään myös valmista käyttöliittymää ja tietokantaa.



Kuva 5. Käyttäjätilien autentikointi.

Data Directories välilehdellä täytyy määritellä mihin kansioihin ja asemiin tiedot tallennetaan. Tietokannan data ja lokitiedostot on tallennettu D asemalle, kuin myös tietokannan väliaikais- ja varmennustiedostot.

FILESTREAM välilehdellä määritellään voiko tietokantaan ottaa yhteyden esimerkiksi erillisen käyttöliittymän kautta, etäyhteydellä tai molempia keinoja käyttäen. Tässä tapauksessa FILESTREAM on sallittu kaikille paitsi etäyhteyksille ja sen jakonimi on HuoltoTietokanta. Tätä tarvitaan myöhemmin käyttöliittymää luodessa, kun pitää ottaa yhteys tietokantaan.

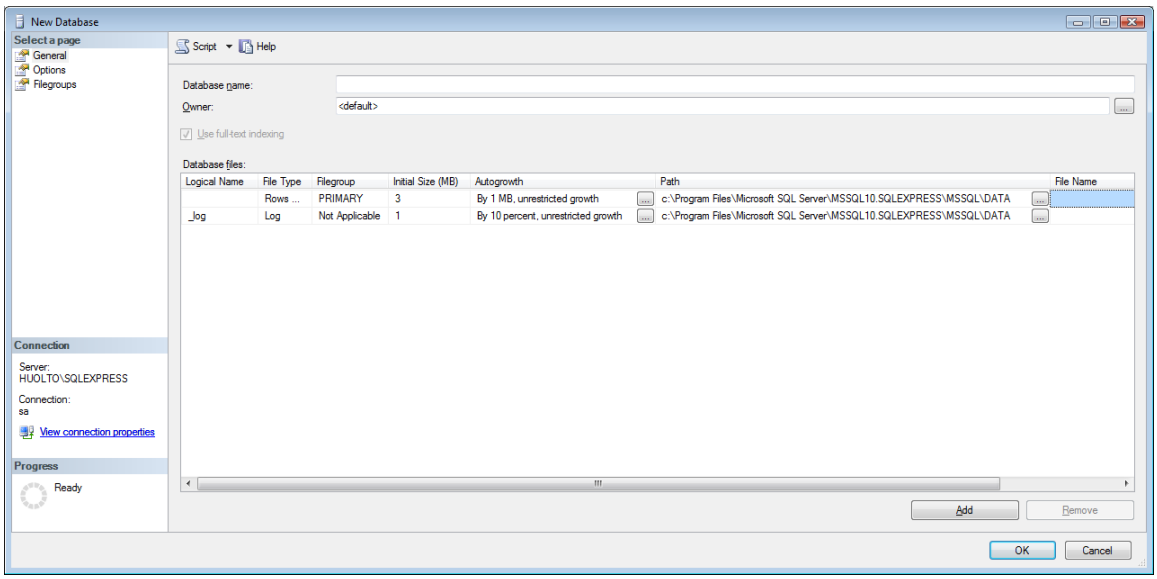
Tämän jälkeen asennus viedään loppuun Next-painiketta klikkailemalla. Asennukseen kannattaa varata aikaa noin yksi tunti.

7 SQL-TIETOKANTA

Tässä kappaleessa käydään läpi tietokannan luomisen vaiheet, taulujen määrittelyt, taulujen sisältämän datan tyypit, niiden arvot ja yhteydet.

7.1 Tietokannan luominen

SQL Server Management Studion päänäkymässä on vasemmalla laidalla Object explorer, jossa näkyy kaikki palvelimella olevat tietokannat. Uuden tietokannan voi luoda klikkaamalla hiiren oikealla painikkeella Databases kansiota ja valitsemalla New Database. Tämä aukaisee new database ikkunan (kuva 6), jonka General-sivulla määritellään tietokannan nimi kohtaan Database name ja pääkäyttäjä eli Owner, joka on tässä tapauksessa KPO Huolto käyttäjä. Alla olevaan tauluun määritellään tietokannan aloituskoko ja autogrowth kohdassa määritellään tietokannan kasvunopeus datamäärän lisääntyessä. Path kohdassa määritellään mihin varsinainen tietokanta tallentuu, tässä tapauksessa D asemalle, Huolto DATA ja Huolto LOG kansioihin. File name kohdassa määritellään data- ja lokitiedoston nimet.



Kuva 6. New database ikkunan general-sivu.

New database ikkunan options kohdassa on kolme pudotusvalikkoa joihin pitää kiinnittää huomiota: Collation, Recovery model ja Compatibility level. Collation kohdassa täytyy valita arvoksi Finnish_Swedish_CI_AS, jotta tietokanta ymmärtää suomalaiset ja ruotsalaiset merkit, joita mahdollisesti joudutaan käyttämään asiakkaiden tietoja tallennettaessa. Recovery

model tarkoittaa palautusmallia, jota käytetään jos tietokanta lakkaa toimimasta. Vaihtoehtoja on kolme: Simple, Full ja Bulk-logged.

Simple on yksinkertainen varmennus, joka ei tarvitse lokivarmennuksia. Se varaa lokin käyttämän tilan, jotta tietokannan tilavaatimukset pysyvät mahdollisimman pieninä. Tämä poistaa käytännössä, myös transaktiolokin käyttötarpeen. Kaikki muutokset, mitä on tehty viimeisen varmennuksen jälkeen, ovat haavoittuvia ja katastrofin sattuessa kaikki nämä muutokset täytyy tehdä uudestaan. (Microsoft, 2010 b.)

Full on täydellinen varmennus, joka tarvitsee lokivarmennuksia. Full malli voi palauttaa tietokannan tiettyyn aikaan esimerkiksi sekuntiin ennen tietokannan kaatumista, olettaen että varmennukset ovat olleet vioittumattomia sillä hetkellä. Tämä malli säilyttää transaktiolokin tiedot niin kauan kunnes transaktioloki on varmennettu. Jos tietokanta on kaatunut kesken lokivarmennuksen ja lokin loppuosa on vioittunut, täytyy viimeisimmät muutokset tehdä uudestaan. Tämä on tosin harvinaista. Tässä työssä on käytetty tätä mallia. (Microsoft, 2010 b.)

Bulk-logged mallia voi käyttää full mallin sijaisena massaoperaatioissa, kuten datan massa tuonnissa toisista tietokannoista ja indeksien luonnissa. Mallin vaihtaminen fullista bulk-loggediksi tällaisten operaatioiden varalle, lisää tietokannan prosessointitehoa ja vähentää lokitilan käyttöä. Lokivarmistukset ovat silti pakollisia. Kuten full-mallissakin, tämä malli säilyttää transaktiolokin niin pitkään, että se on varmistettu. Ainoa ongelma tässä mallissa on suuremmat lokivarmistukset ja lisääntynyt vaara hukata tietoa, koska tämä malli ei tue tiettyyn aikaan palauttamista. (Microsoft, 2010 b.)

7.2 Taulujen luominen ja niiden rakenne

Taulujen luominen Microsoft SQL Server Management Studiolla on hyvin yksinkertaista. Tässä työssä on tarvittu seuraavanlaisia tauluja: asiakas, hinnasto, huoltoliike, laite, raportti ja yhteystietoja. Uuden taulun voi luoda klikkaamalla hiiren oikealla painikkeella tietokannan Tables kohtaa ja valitsemalla New Table kohdan. Ohjelma avaa uuden taulun rakennenäkymään ja käyttäjä voi ruveta syöttämään sarakkeiden tietoja välittömästi. On hyvä suunnitella etukäteen millaisia tietoja missäkin tauluissa tulee olemaan ja mikä niiden tietotyyppi on. Jos taulujen tietotyyppejä joutuu muokkaamaan, täytyy Management Studion asetuksia muuttaa hieman. Tools → Options → Designer polusta löytyy kohta, jossa on oletuksena valittuna

Prevent saving changes that require table re-creation. Tämä asetus täytyy ottaa pois päältä, jos valmiiden taulujen tietotyyppejä haluaa muuttaa. Jos tietotyyppejä haluaa muuttaa, tulee se tehdä nyt luontivaiheessa, koska tietotyyppien muuttaminen yleensä hävittää tauluun tallennetun datan ja rikkoo yhteydet. Alla on kuvaus taulujen rakenteesta ja tietotyypeistä. Alleviivatut sarakkeet ovat perusavaimia ja kursivoidut ovat viiteavaimia.

Asiakas-tauluun tallennetaan asiakkaiden tiedot. Taulussa on seuraavat sarakkeet: Asiakasnro., Etunimi, Sukunimi, Puhelin, Lähiosoite, Postinumero, Postitoimipaikka. Asiakasnro. sarake on tämän taulun pääavain ja sen tietotyyppi on int eli integer. Asiakasnumero on asiakkaan yksilöivä tieto, jonka avulla tietokanta voi eritellä kaikki asiakkaat. Se saa arvonsa automaattisesti aina, kun uusi asiakas luodaan ja kahta samanlaista arvoa ei voi olla olemassa, eikä sitä voi myöskään muuttaa. Arvo ei voi myöskään olla null eli tyhjä. Nimi sarakkeiden tietotyyppi on nvarchar(50), koska nimet koostuvat aina kirjaimista. Puhelin, lähiosoite ja postitoimipaikka sarakkeiden tietotyyppi on text, koska niissä voi olla merkkejä sekä numeroita ja ne eivät ole yksilöiviä tietoja. Postinumero on int, koska se koostuu suomessa poikkeuksetta numeroista.

Laite-taulu on asiakas-taulun lapsitaulu. Laite tauluun tallennetaan asiakkaan huoltoon tuoman laitteen tiedot. Taulussa on seuraavat sarakkeet: *asiakasnro.*, tapausnro., saapumis pvm, laite (pc, tv...), merkki, malli, sarjanumero, tila, huoltoliike, osto pvm. Tässä taulussa tapausnumero yksilöi laitteet samalla tavalla kuin asiakasnumero yksilöi asiakkaat asiakkaat-taulussa. Taulu saa asiakkaan tiedot asiakas-taulusta. Asiakasnumeron tietotyyppi täytyy olla sama kuin asiakas-taulussa yhteyden eheyden varmistamiseksi. Asiakasnumero ja tapausnumero ovat int-arvoisia ja kumpikaan ei voi olla null-arvoinen. Saapumispäivämäärä on date arvoinen, eli siihen ei kelpaa mikään muu arvo kuin päivämäärä. Myös ostopäivämäärä on date-arvoinen, se voi olla null, sillä sitä käytetään vain jos laitteella on takuuta jäljellä ja se lähetetään maahantuojalle huoltoon. Merkki, malli ja sarjanumero sarakkeet ovat nvarchar(50) arvoisia, koska niissä voi olla kaikenlaisia merkkejä. Tila sarake on text-arvoinen ja siinä on vain kaksi arvoa: kesken tai valmis. Laite ja huoltoliike sarakkeet ovat myös text arvoisia. Huoltoliike saraketta käytetään, jos laite lähetetään takuuhuoltoon maahantuojalle tai valmistajan valtuuttamaan huoltoliikkeeseen.

Raportti-taulu on laite-taulun lapsi-taulu ja siinä on seuraavat sarakkeet: vikakuvaus, ratkaisu, työaika, raportti ja *tapausnro.* Tässä taulussa raportti-sarake yksilöi jokaisen raportin samalla tavalla kuin asiakasnumero ja tapausnumero yksilöivät asiakkaan ja laitteen tiedot. Vikakuvaukseen kirjataan asiakkaan ilmoittama vika, ja ratkaisu sarakkeeseen kirjataan, mikä oli varsi-

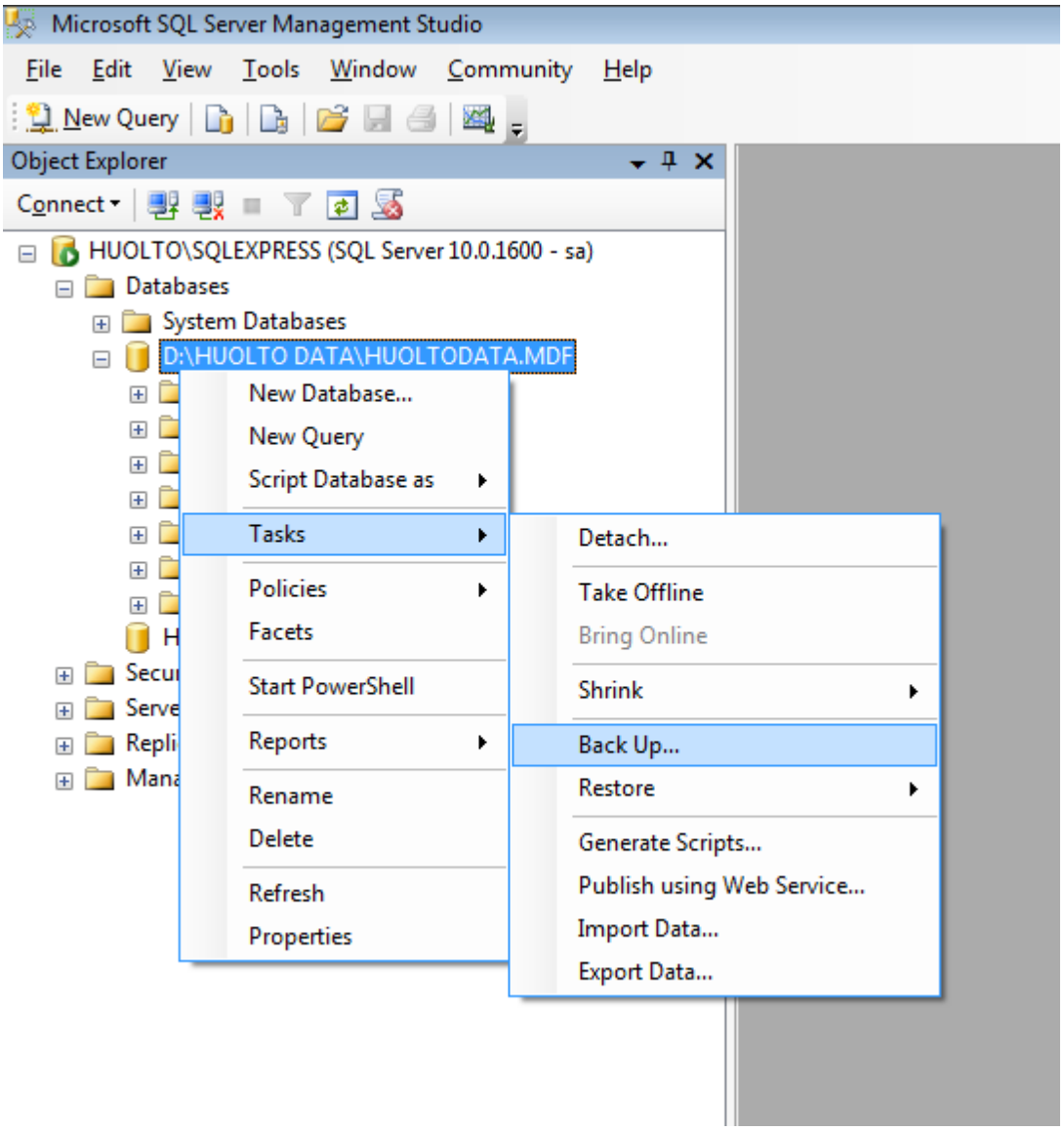
nainen vika ja kuinka se on korjattu tai ratkaistu. Työaika sarakkeeseen merkitään huoltamiseen tai korjaamiseen käytetty aika, jotta myöhemmin voidaan arvioida samanlaiseen työhön käytettävä aika entistä paremmin. Kuten aiemminkin, raportti ja tapausnumero on int arvoisia ja ne eivät voi olla tyhjiä eli null-arvoisia. Työaika-sarake on nvarchar(50) tyyppinen. Ratkaisu ja vikakuvaus sarakkeet ovat text-arvoisia.

Yhteystieto-taulussa on kaksi saraketta: numero ja laite (pc, tv...). Tämä taulu on huoltoliiketaulun isä-taulu, joten laite-sarake ei liity edellä mainittujen taulujen laitetauluun. Yhteystieto-ja-taulu ja huoltoliike-taulu on luotu vain sitä varten, että takuuhuoltoliikkeisiin voidaan ottaa yhteyttä ja kaikkien niiden yhteystiedot olisivat yhdessä paikassa. Numero-sarake yksilöi laitteet samalla tavalla kuin asiakas-taulussa ja sen tietotyyppi on numeric(18, 0). Numeron ei tarvitse näkyä käyttäjälle, jolloin käyttäjä valitsee listasta vain laitteen (pc, tv, yms.) ja valitsee sitten huoltoliiketaulusta haluamansa huoltoliikkeen. Laite-sarake on text tietotyyppiä.

Tähän tauluun kirjataan takuuhuoltoon lähteviä laitteita varten huoltoliikkeet ja niiden yhteystiedot, jotta käyttäjä voi tarvittaessa katsoa liikkeen puhelinnumeron, www-sivun tai sähköpostin sekä mahdollisesti tarvittavan palautusnumeron. Taulussa on seuraavat sarakkeet: järjestys, *numero*, valmistaja, huoltoliike, osoite, puh., email/www, palautusnumero. Järjestys ja numero sarakkeiden tietotyyppi on numeric(18, 0) ja muiden sarakkeiden tietotyyppi on text.

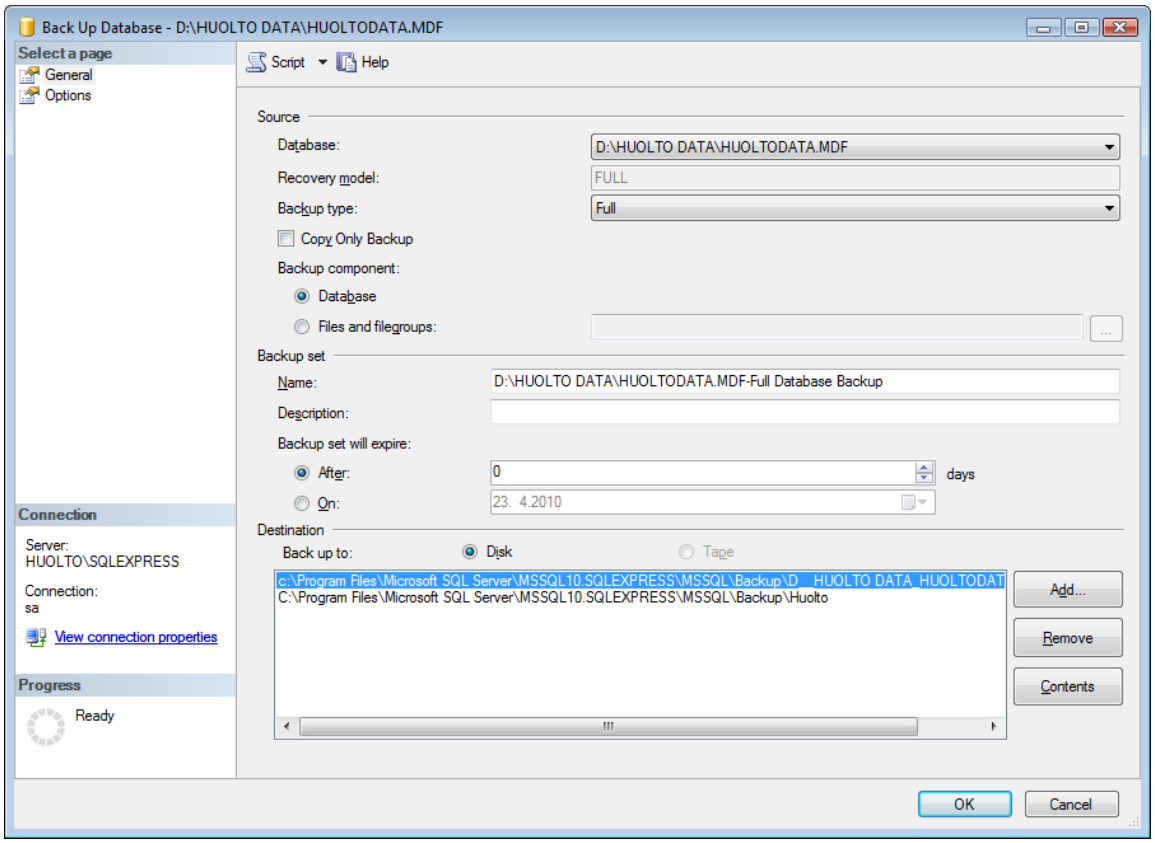
7.3 Tietokannan varmuuskopiointi

Tietokanta täytyy varmuuskopioida manuaalisesti johtuen Express Editionin rajoituksista. Varmuuskopio otetaan SQL Server Management Studiossa valitsemalla haluttu tietokanta hiiren oikealla painikkeella avautuvasta valikosta valitsemalla Tasks --> Back Up...(Kuva 7).



Kuva 7. Varmuuskopioinnin aloittaminen.

Avautuvassa ikkunassa näkyy kolme pääkohtaa: Source, Backup set ja Destination. Source kohdassa on Database kohta, jossa näkyy varmuuskopioitava tietokanta ja Backup type kohdassa voi valita erilaisia varmuukopiointi tyyppejä, kuten Full, Differential ja Transaction Log. (Kuva 8).



Kuva 8. Tietokannan varmuuskopiointi

Full tyyppinen varmuuskopio luo täydellisen kopion koko tietokannasta. Täydellinen tietokannan varmuuskopio on itsenäinen ja sen voi palauttaa joko samaan tai uuteen tietokantaan samalle tai eri palvelimelle. Tämä antaa paljon joustavuutta tilanteessa, jolloin tietokanta täytyy palauttaa. Se voidaan suorittaa tietokannoille, joissa on mikä tahansa palautusmalli. Täydellistä palautusta käytetään yleensä, kun aletaan palautua koko tietokantaa vahingoittaneesta tai koko tietokannan hävittäneestä katastrofista. On suositeltavaa, että koko tietokanta varmuuskopioidaan säännöllisin väliajoin. Tässä tietokannassa käytetään tätä mallia, sillä tietokanta on suhteellisen pieni ja varmuuskopioiden koko ei kasva valtavan suureksi.

Differentiaalinen malli varmuuskopioi vain ne datasivut jotka ovat muuttuneet edellisen täydellisen varmuuskopion jälkeen. Differentiaali varmuuskopiot sisältävät aikaisempien differentiaali varmistusten tiedot ja sitä voidaan käyttää suurissa ja muuttuvissa tietokannoissa. Näihin kuuluvat esimerkiksi tietovarasto malliset tietokannat.

Transaction log eli lokivarmistus varmistaa viimeisimmän lokivarmistuksen jälkeen tapahtuneet muutokset, mutta se vaatii toimiakseen täydellisen tietokannan varmistuksen.

Backup set kohdassa käyttäjä määrittelee varmuuskopion nimen, johon järjestelmä ehdottaa automaattisesti polkua jossa tietokanta sijaitsee ja lisää polun perään palautusmallin. Description kohtaan voi halutessa kirjoittaa kuvauksen varmuuskopiosta. Backup set will expire kohdassa määritellään milloin varmuuskopio vanhenee ja sen päälle voi kirjoittaa.
Destination kohdassa määritellään minne varmuuskopio luodaan, ja mille medialle varmuuskopio tehdään. (Teratrax 2010.)

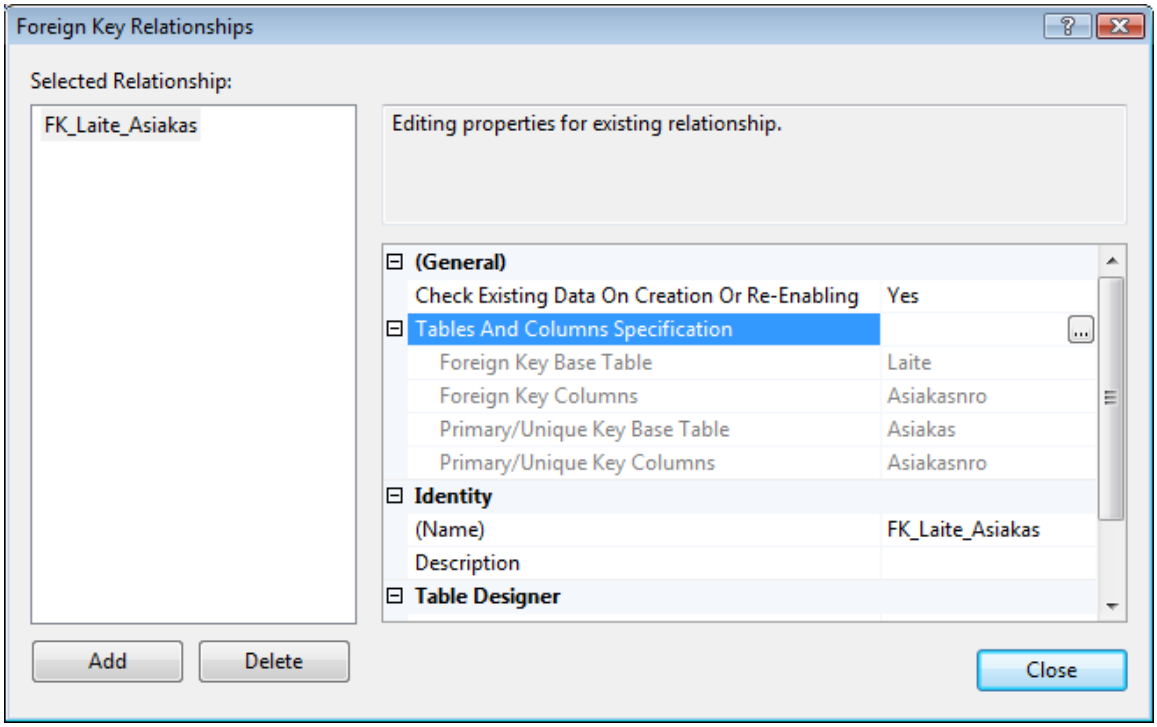
7.4 Yhteyksien luominen

Tietokannan eheys (integrity) on tärkeää relaatiomallissa. Tietokanta on silloin eheä, kun sen tiedot ovat oikein, ristiriidattomia ja vastaavat reaalimaailmaa. Eheys vaarantuu, jos esimerkiksi sama asiakas tallennetaan kahteen kertaan tai jos asiakkaasta löytyy useita eri osoitteita eikä tiedetä, mikä on oikea. Codd määritteli relaatiomalliin eheysrajoitteita. Coddin ensimmäinen eheyssääntö on avaineheys (entity integrity). Tämän säännön mukaan perusavaimen arvo ei saa olla NULL-arvo. (Hovi, Huotari & Lahdenmäki, 2005, 11.)

Toinen eheyssäännöistä on viite-eheys (referential integrity). Tämän säännön mukaan isätaulusta ei saa poistaa tietoja, jos lapsitaulussa on kyseessä olevaan isään liittyviä lapsirivejä. Muutoin lapsitaulun tiedot jäisivät orvoiksi. Tätä kutsutaan viite-eheyden särkymistilanteeksi, jota ei saisi tapahtua. Useimmat relaatiomallit mahdollistavat viite-eheyden valvomisen määrittelemällä tietokantaan eheysrajoitteet, joita ei voida ohittaa. (Hovi, Huotari & Lahdenmäki, 2005, 12.)

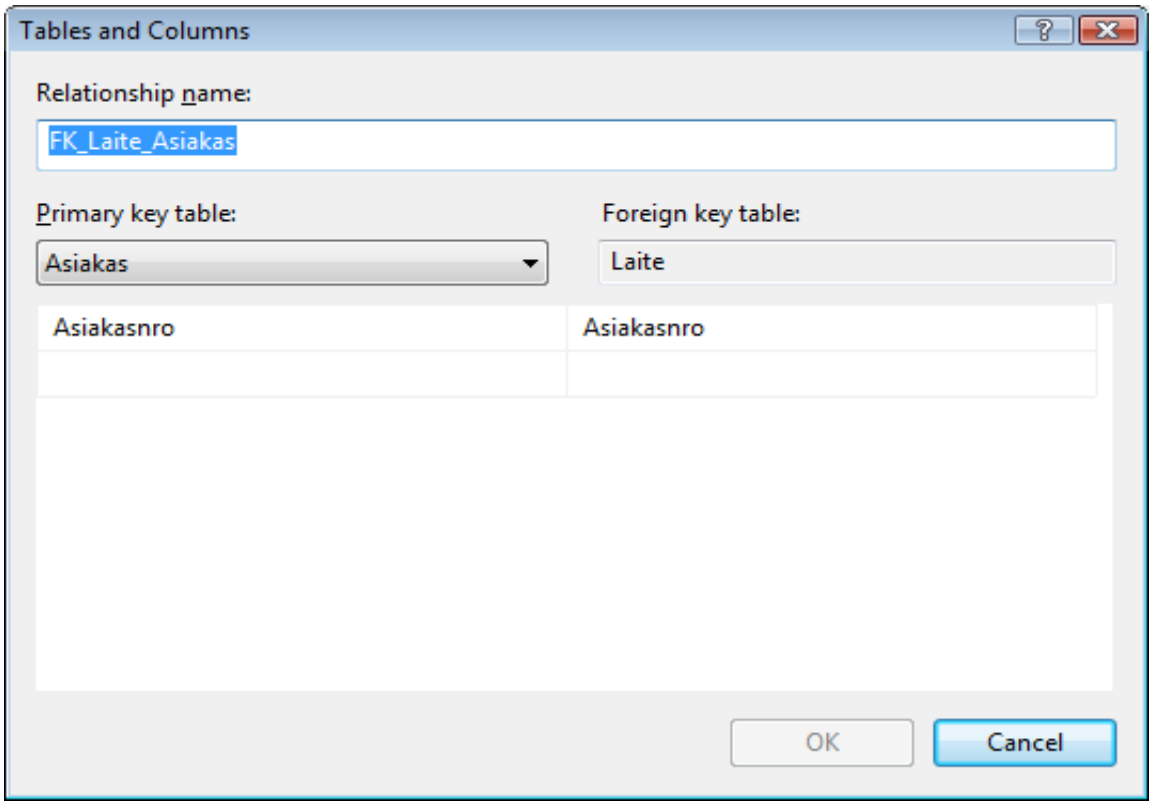
Tässä työssä on jo suunnitteluvaiheessa pyritty luomaan tietokannasta sellainen, että avaineheys sekä viite-eheys eivät särkyisi. Ja kuten kappaleessa Taulujen luominen ja niiden rakenne on kerrottu, mikään perusavain ei salli NULL-arvoa. Eli tältä osin avaineheys säilyy.

Microsoft SQL Server Management Studiossa taulujen yhteyksien luonti tapahtuu seuraavanlaisesti. Kun halutut taulut on ensin luotu ja niille on määritelty perusavaimet ja viiteavaimet, löytyy ohjelman yläpalkista relationships-nappi, jota klikkaamalla käyttäjä saa näkyviin Foreign Key Relationships -ikkunan. Tässä ikkunassa käsitellään siis viiteavaimia. Taulu, joka sisältää lapsi-tietueen, tulee olla aktiivisena Desing-näkymässä, jotta yhteyden luonti onnistuu. Uuden yhteyden luonti tapahtuu klikkaamalla Add-nappia. Seuraavaksi täytyy muokata Tables And Columns Specifications kohtaa (kuva 9.). Kun se aktivoidaan, ilmestyy samalle riville oikealle laidalle nappi, jossa on kolme pistettä.



Kuva 9. SQL Server Management Studion yhteyksien luonti, vaihe 1.

Tätä klikkaamalla pääsee Tables and Columns -ikkunaan (kuva 10.), jossa voi määritellä yhteyden nimen, perusavain-taulun, sekä isä-taulun perusavain sarakkeen ja lapsi-taulun viiteavain sarakkeen. Viiteavain-taulua ei voi muuttaa tässä tilassa ja siksi taulun tulee olla aktiivisena design-tilassa, jotta yhteyksien luonti onnistuu.



Kuva 10. SQL Server Management Studion yhteyksien luonti, vaihe 2.

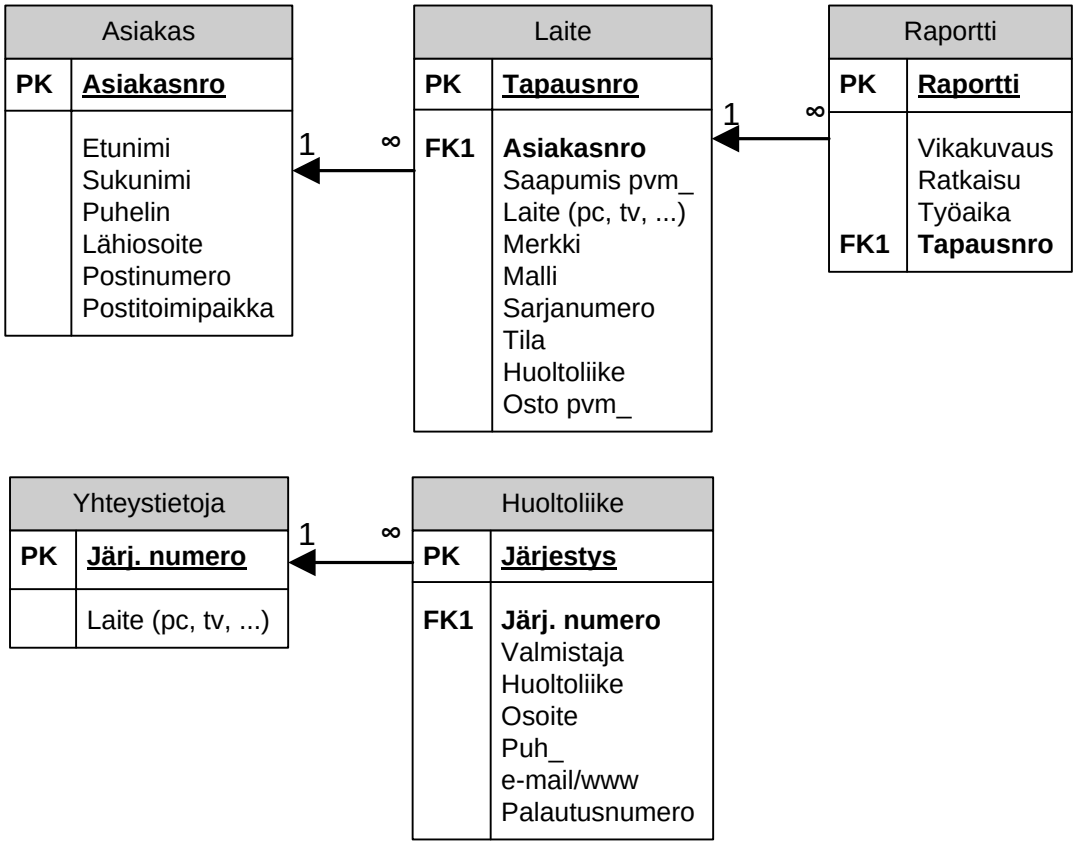
Tässä esimerkissä on mallina Asiakas- ja Laite-taulut, joista asiakas on isä-taulu ja laite on lapsi-taulu. Laitetaulu perii automaattisesti isä-taulusta asiakasnumeron, jolloin yhdellä asiakkaalla voi olla useita laitteita. Tämä sama kuvio toistuu myös laite- ja raportti-taulujen välillä, joista laite on isä-taulu ja raportti on lapsi-taulu. Eli laite-taulu perii asiakas-taulusta asiakkaan yksilöivän asiakasnumeron ja raportti perii laite-taulusta huoltotapauksen yksilöivän tapausnumeron.

Coddin viite-eheyssäännön mukaisesti asiakas-taulusta ei voida poistaa asiakasta, jos tälle on lisätty huoltotapaus ja huoltotapausta ei voida poistaa laite-taulusta, jos sille on kirjattu raportti-tauluun esimerkiksi vikakuvaus. Ainut tapa poistaa näitä tietoja on lähteä poistamaan niitä alhaalta ylöspäin. Tosin tietojen poistamiselle ei pitäisi olla minkäänlaista tarvetta.

7.5 Tietokannan rakenne

Seuraavan sivun mallissa (Kuva 11.) havainnollistetaan tietokannan rakennetta, josta käy ilmi taulut ja niiden Primary Key ja Foreign Key. Kuvassa näkyy myös taulujen suhteet ja periy-

tyminen. Yhteystieto- ja huoltoliike-taulut ovat erillään muista tauluista, koska niitä ei tarvita asiakkaan tietoja ja laitteita käsiteltäessä.



Kuva 11. Tietokannan rakenne.

Yllä olevassa kuvassa PK tarkoittaa Primary Key:tä, FK1 tarkoittaa Foreign Key:tä. Joiden avulla taulujen välinen relaatio muodostuu, esimerkiksi yhdellä asiakkaalla voi olla monta laitetta ja yhdellä laitteella voi olla monta raporttia. Yhdellä laitteella voi olla monta raporttia, sellaisissa tapauksissa joissa sama laite tuodaan useamman kerran huoltoon.

8 TIETOKANNAN HALLINTAJÄRJESTELMÄ

Microsoft Visual Basic 2008 Express Editionin käyttöönotto on hyvin yksinkertaista. Sen asentaminen on erittäin suoraviivainen toimenpide ja se ei poikkea mitenkään muitten Microsoftin tuotteiden asentamisesta. Tässä luvussa käydään läpi uuden projektin luominen Microsoft Visual Basic 2008 Express Edition:lla, ja selvitetään kuinka toteutettu hallintajärjestelmä on rakennettu.

8.1 Uuden projektin luominen

Kun uutta projektia luodaan, antaa ohjelma valmiita malleja tulevan ohjelman rakenteeksi. Näitä ovat: Windows Forms Application, Class Library, WPF Application, WPF Browser Application, Console Application sekä Internetistä haettavat mallit. Tässä työssä on käytetty Windows Form Applicationia, koska se vastaa työn tarkoitusperää. On kuitenkin tärkeää tietää miksi tähän työhön on valittu juuri tämä malli.

8.1.1 Visual Basic projektien mallit

Windows Forms Application, eli Windows lomake sovellus. Tällä voi luoda perinteisen itsenäisen windows sovelluksen tai käyttöliittymän jollekkin Web sovellukselle. (Microsoft. 2010 c.)

Class Library, eli luokka kirjasto. Sillä voi luoda nopeasti muihin projekteihin jaettavia ja uudelleenkäytettäviä luokkia sekä komponentteja. (Microsoft. 2010 c.)

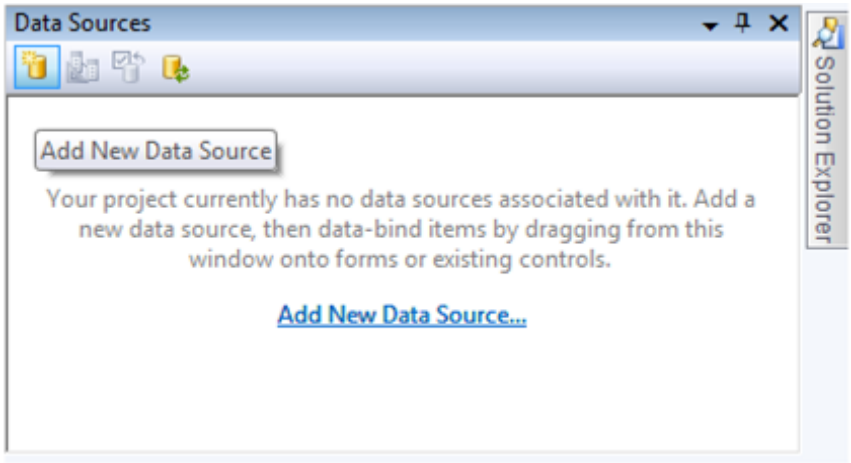
WPF Application, eli WPF sovellus. Tämä malli luo projektin joka käyttää Windows Presentation Foundationia. (Microsoft. 2010 c.)

WPF Browser Application, eli WPF selainsovellus. Tämä malli luo WPF sovelluksen, joka toimii selaimessa. Tätä mallia kutsutaan XAML-selainsovellukseksi. Se yhdistää Web-sovellusten ja asiakasohjelmien ominaisuuksia. Kuten Web-sovellukset, XAML-selainsovellukset voidaan julkaista Web-palvelimelle ja käynnistää selaimen kautta. (Microsoft. 2010 d.)

Console Application, eli konsolisovellus. Nämä suunnitellaan yleensä ilman graafista käyttöliittymää (GUI) ja ne kootaan .exe -tiedostoon. Tätä käytetään kirjoittamalla käskyjä komentokehotteeseen. (Microsoft. 2010 c.)

8.1.2 Tietokantayhteyden luominen

Sitten kun uusi projekti on luotu, valitaan ikkunan oikeasta ylälaidasta Data Sources kentästä Add New Data Source. Se avaa uuden ikkunan, josta valitaan Database vaihtoehto (Kuva 12). Sen jälkeen valitaan New Connection, ja avautuvasta ikkunasta valitaan olemassaolevan tietokannan sijainti ja Data Source malliksi valitaan Microsoft SQL Server Database File (SqlClient), joka käyttää paikallista tietokantaa.



Kuva 12. Data Sources kenttä.

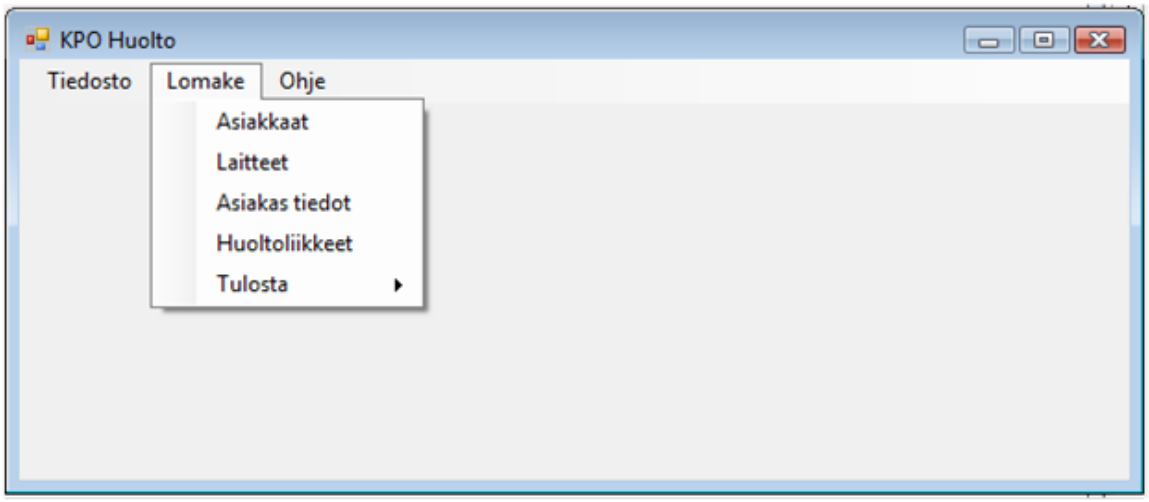
Tämän jälkeen ohjelma pyytää nimeämään yhteyden, ja sitten käyttäjä voi valita, mitä tauluja haluaa käyttää projektissa. Sen jälkeen viimeistellään yhteyden luonti klikkaamalla Finish-painiketta.

9 KÄYTTÖLIITTYMÄ

Tässä luvussa esittelemme, mitä käyttäjälle näkyvään ohjelmaan olemme tehneet. Esittelemme tässä kappaleessa myös miten ohjelmaa käytetään, ja selitämme miksi olemme mitäkin tehneet.

9.1 KPO Huolto

Ensimmäiseksi käyttäjän näkyvillä on tämä KPO Huolto ikkuna (Kuva 13).



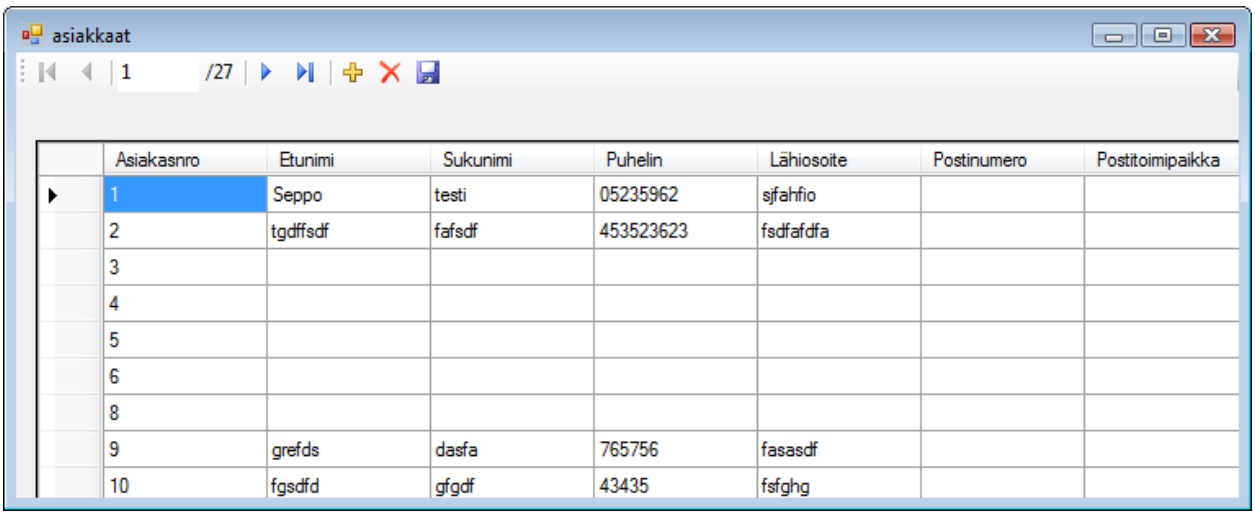
Kuva 13. KPO Huolto lomakenäkymä.

Tämän ikkunan kautta käyttäjä pääsee käsiksi kaikkiin lomakkeisiin, mitä ohjelma sisältää. Tiedostovalikon alta ei tässä ikkunassa löydy mitään muuta kuin sulje toiminto, joka nimensä mukaisesti sulkee ikkunan. Lomakevalikon alta löytyy useampia toimintoja, jotka ovat näkyvissä yllä olevassa kuvassa. Lomakevalikon toiminnoista lisää myöhemmin. Ohjevalikon alta löytyy valikko tiedot, jota painamalla saadaan näkymiin tietoja (Kuva 14) kenen ohjelma on ja niin edelleen. Kaikissa aloitus sivun valikoista avautuu uusi ikkuna ja koodina näihin on käytetty *.show() komentoa, esimerkiksi asiakkaat.Show() avaa asiakkaat taulun ja samalla tavoin avautuvat kaikki muutkin sivut.

Kuva 14. Tietoja-ikkunan näkymä.

9.2 Asiakkaat

Asiakkaat-lomakkeessa pystytään tarkastelemaan nykyisten asiakkaiden tietoja, sekä voidaan lisätä uusia asiakkaita, poistaa asiakkaiden ja muokata nykyisten asiakkaiden tietoja. Asiakas-lomakkeessa (Kuva 15) on lomakkeen ylälaidassa painikkeita jotka sisältävät erilaisia toimintoja.



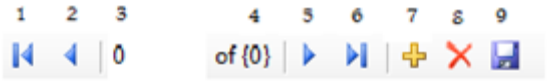
Asiakasnro	Etunimi	Sukunimi	Puhelin	Lähiosoite	Postinumero	Postitoimipaikka
1	Seppo	testi	05235962	sjfahfio		
2	tgdffsdf	fafsdf	453523623	fsdfafdfa		
3						
4						
5						
6						
8						
9	grefds	dasfa	765756	fasasdf		
10	fgsdfd	gfgdf	43435	fsfghg		

Kuva 15. Asiakkaat-lomakkeen näkymä.

Painikkeita, jotka ovat lomakkeen yläreunassa, kutsutaan BindingNavigator:iksi (Kuva 16). Painikkeet vasemmalta oikealla tekevät seuraavia toimintoja:

1. Painike siirtää valinnan ensimmäiseen asiakkaaseen.

2. Painike siirtää valintaa yhden askeleen ylöspäin.
3. Tämä näyttää numeron, monennessako asiakkaassa valinta on.
4. Tämä näyttää asiakkaiden kokonaismäärän.
5. Painike siirtää valintaa yhden askeleen alaspäin.
6. Painike siirtää valinnan viimeiseen asiakkaaseen.
7. Tämä painike lisää uuden asiakkaan taulun loppuun, johon voidaan syöttää tietoja.
8. Tämä painike poistaa valitun asiakkaan tiedot.
9. Tällä painikkeella tallennetaan kaikki muutokset tietokantaan.

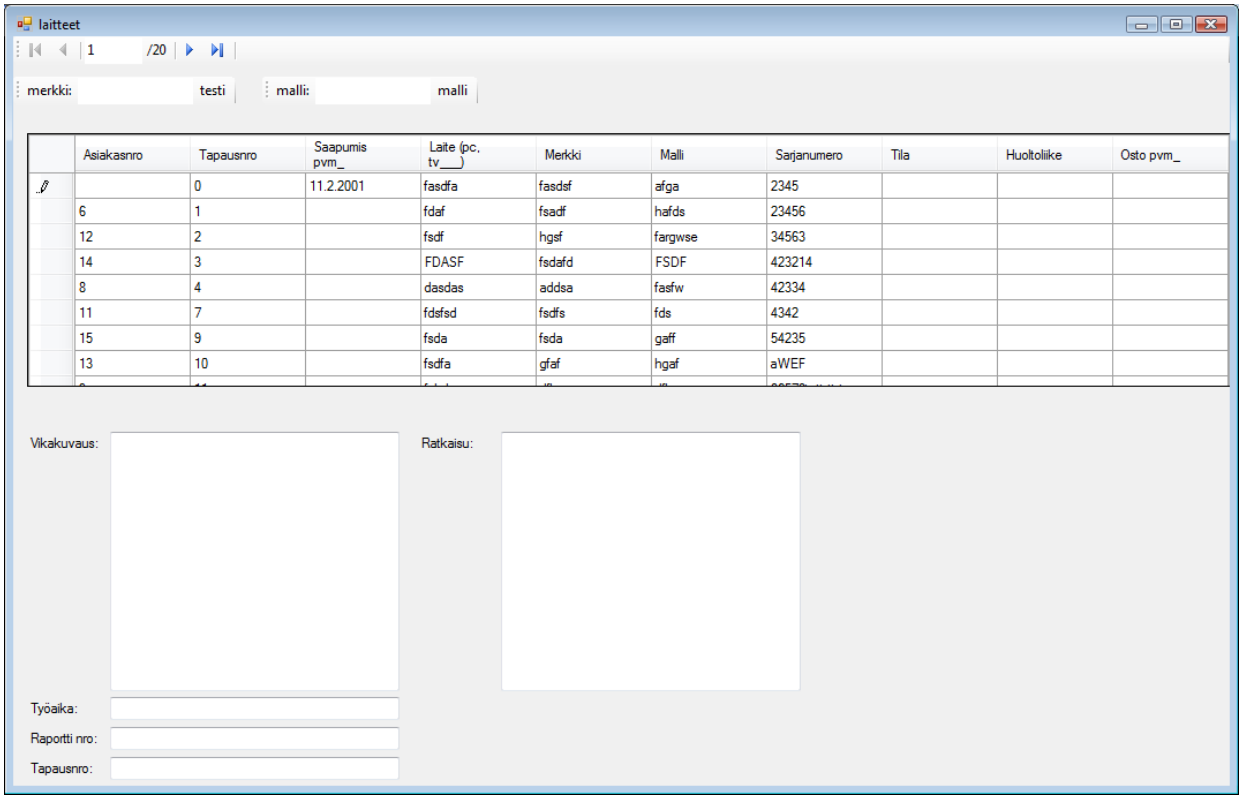


Kuva 16. Painikkeet asiakkaat-lomakkeessa.

Asiakkaat-lomakkeella on tietueita, joihin syötetään tietoja asiakkaista. Ensimmäisenä, kun uusi asiakas syötetään saa tämä yksilöllisen asiakasnumeron automaattisesti. Loput syötettävistä tiedoista ovat etunimi, sukunimi, puhelinnumero, lähiosoite, postinumero ja postitoimipaikka. Näiden kaikkien tietojen perusteella asiakkaat yksilöityvät, ja kahta samanlaista asiakas tietoa ei löydy. Monilla ihmisillä voi olla täysin sama nimi ja kenties osoitekin ja puhelinnumerokin, jos käytetään esimerkiksi samaa puhelinta. Tällaisissa tapauksissa on apua asiakasnumerosta, joka ei voi olla kahdella asiakkaalla sama.

9.3 Laitteet

Laitteet-taulusta pystytään tarkastelemaan asiakkailta tulleita laitteita ja niiden vikakuvauksia sekä ratkaisuja. Tähän tauluun olemme mahdollistaneet ainoastaan tarkastelu toiminnon, joten tästä taulusta ei voida poistaa eikä muokata tietoja. Taulusta on tarkoituksella rajattu pois poisto ja muokkaus mahdollisuus, jotta ei tulisi vahingossa tehtyä muutoksia joita ei ole tarkoittanut tehtäväksi sillä taulussa ei ole näkyvissä asiakkaan nimeä vaan pelkästään asiakasnumero (Kuva 17).



Kuva 17. Laitteet-taulun näkymä.

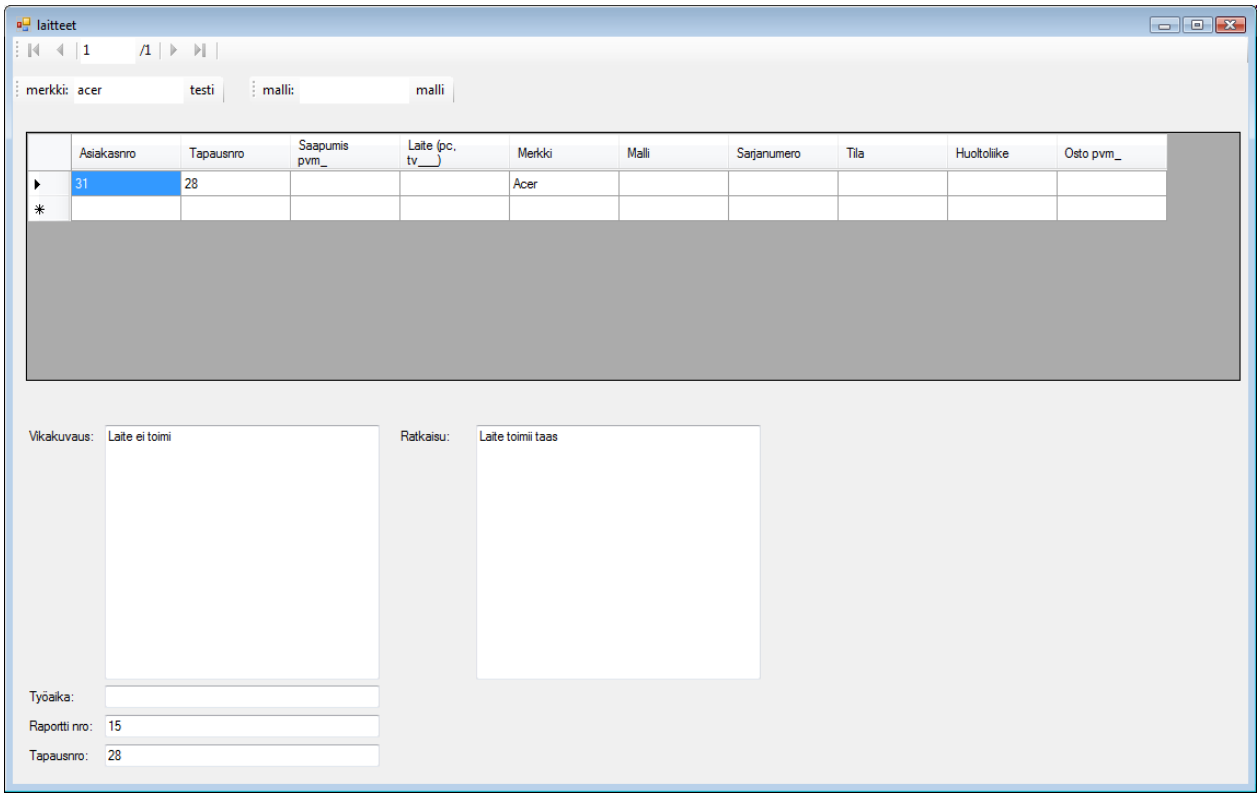
Tässä taulussa on samoin kuin asiakkaat taulussa ylälaidassa navigointipainikkeita, jotka ovat käytännöllisiä kun laitteita kertyy enemmän. Toiminnot vasemmalta oikealle (Kuva 18):

1. Tämä painike siirtää valinnan ensimmäiseen laitteeseen.
2. Tämä painike siirtää valintaa yhden ylöspäin.
3. Tämä valinta näyttää monennessako laitteessa valinta on tällä hetkellä.
4. Tämä näyttää kaikkien laitteiden kokonaismäärän.
5. Tämä painike siirtää valintaa yhden.
6. Tämä painike siirtää valinnan viimeiseen laitteeseen.

Kuva 18. Painikkeet laitteet-lomakkeesta.

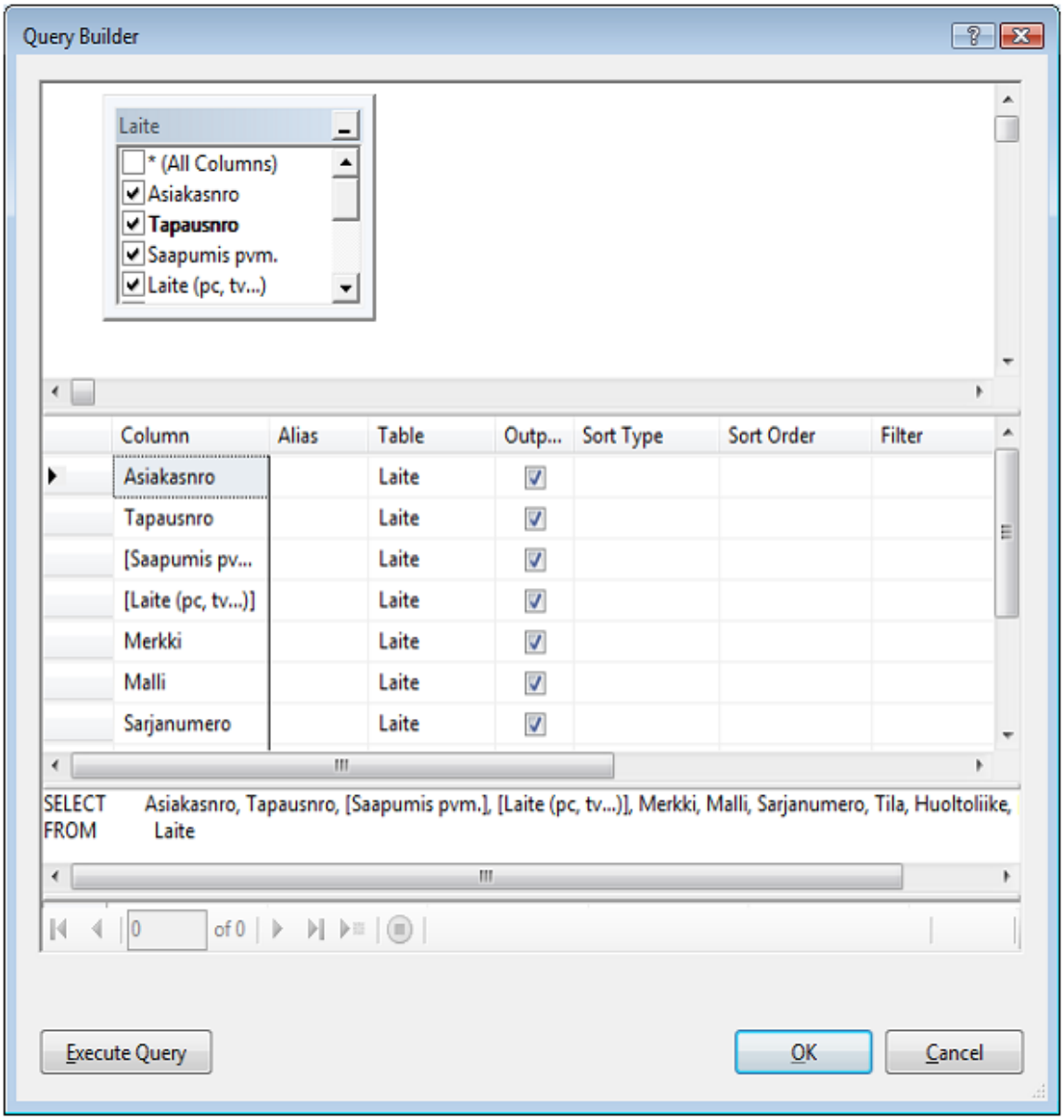
Laite-taulussa on monia tietoja, joita laitteesta kerätään. Laitteet taulusta löytyy monia tietoja, joita syötetään talteen. Ensimmäisenä syötetään laitteen kaikki tiedot, ja sen jälkeen aletaan syöttämään vikakuvausta laitteesta. Kun laite on saatu korjattua, kirjoitetaan ratkaisu. Myöhemmin kuitenkin lisää laitteet taulun toiminnoista. Sillä kuten edellä mainittiin, on tämä taulu vain tietojen selailua varten eikä niiden muokkaamista, lisäämistä tai poistamista varten.

Myöhemmin tulee esille, että lomakkeen tiedoille voidaan tehdä muutkin toiminnot. Laitteet sivulta löytyy myös hakuominaisuudet. Hakeminen tapahtuu merkin tai mallin mukaan. Esimerkiksi, jos merkki kohtaan laitetaan Acer, listaa se kaikki Acer merkkiset laitteet näkyville (Kuva 19).



Kuva 19. Hakukenttä esimerkki.

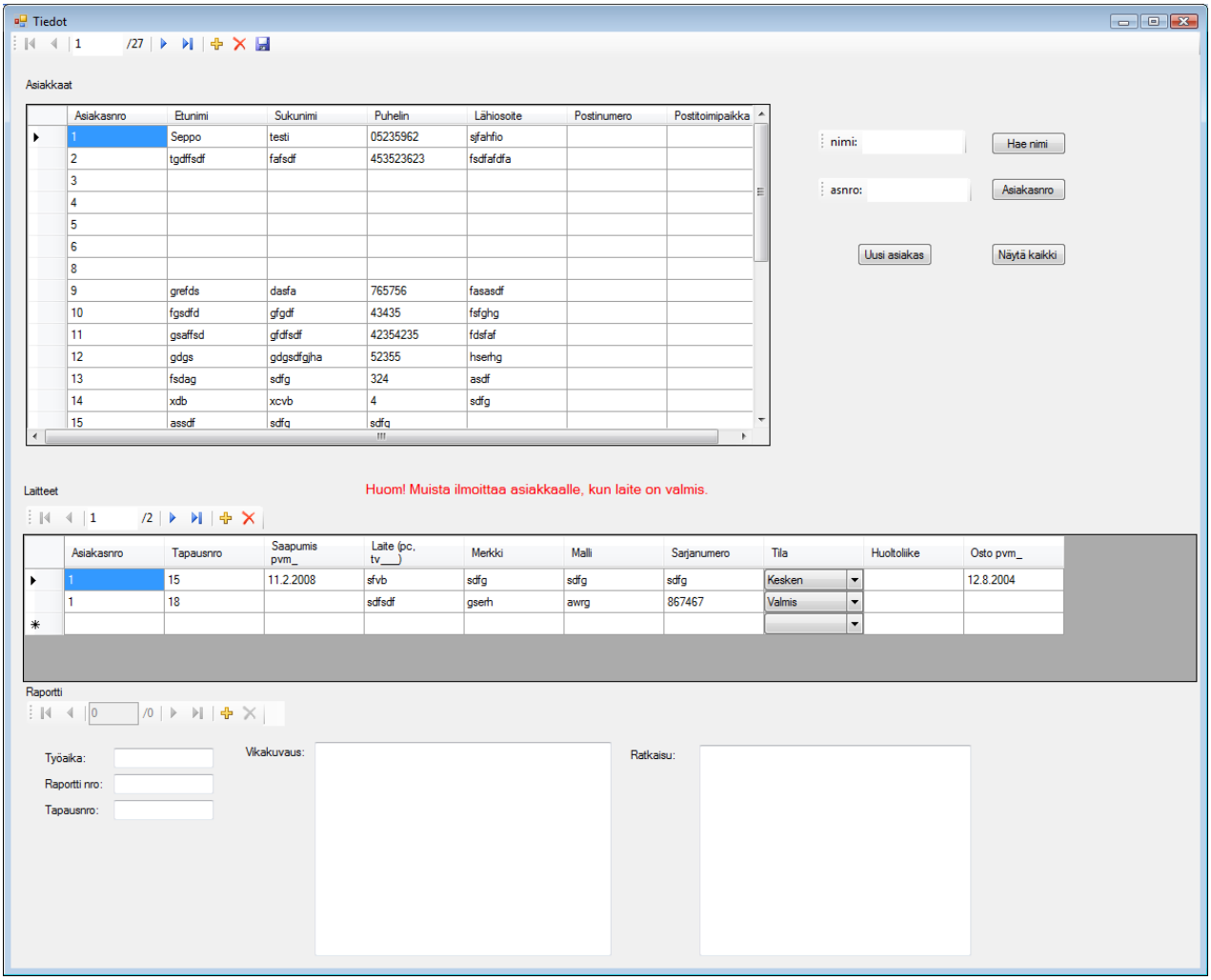
Haut on luotu käyttämällä Search Criteria Builder työkalua, jonka saa näkyviin kun valitsee taulun, johon haluaa tehdä hakuja ja painaa taulun oikeasta yläkulmassa pientä nuolta ja valitsee sieltä Add Query. Avautuvasta ikkunasta valitaan New query name, mikäli halutaan tehdä uusi hakuehto. Nimen antamisen jälkeen painetaan Query Builder näppäintä, joka aukaisee Query Builder näkymän (Kuva 20). Output kohdasta valitaan mitkä rivit hakutuloksessa näytetään ja filter kohtaan annetaan hakuehto. Lisää hakuehtoja voi syöttää or kohtiin, jotka löytyvät filter kohdan oikealta puolelta.



Kuva 20. Query Builder näkymä.

9.4 Tiedot

Tiedot-taulu on se taulu, jota pääsääntöisesti käytetään. Tähän tauluun on yhdistetty molemmat laitteet sekä asiakkaat taulut, sillä tätä taulua on tarkoitus käyttää pääsääntöisesti tietojen syöttämiseen. Samalla näkee kaikki tarpeelliset tiedot laitteista ja asiakkaista yhdellä kertaa eikä tarvitse hyppiä useampien taulujen välillä. Tässä lomakkeessa on laitteet tauluun otettu ominaisuuksiksi tietojen lisääminen, poistaminen ja muokkaaminen. Minkä takia vasta tässä taulussa eikä aikaisemmin johtuu siitä, kun asiakas on valittuna, laitteet tulevat varmasti oikealle asiakkaalle (Kuva 21).

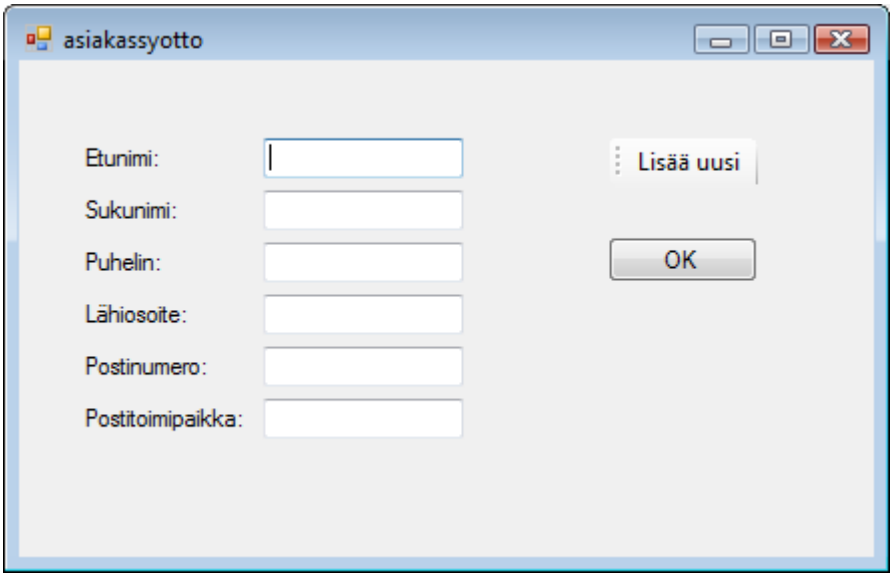


Kuva 21. Tiedot-taulun näkymä.

Tiedot-taulussa löytyy samanlainen painikerivistö ylälaidasta, mikä oli myöskin asiakkaat taulussa, ja tekee tässäkin samat toiminnot kuin aikaisemminkin. Ainoa muutos painikkeiden toiminnoissa on se, että tallennusnappi tallentaa kaikki muutokset tiedot-taulusta eikä pelkästään asiakkaisiin kohdistuneita muutoksia. Asiakkaisiin on liitetty haku asiakkaan nimellä ja asiakasnumero (Kuva 22).

Kuva 22. Asiakastaulun hakukentät

Kainuun Puhelinosuuskunnan pyynnöstä teimme vielä erillisen asiakkaan lisäystoiminnon, jonka tarkoituksena on helpottaa asiakkaan tietojen syöttöä. Uuden asiakkaan tietojen syöttäminen aloitetaan painamalla uusi asiakas painiketta, jonka jälkeen aukeaa ikkuna, johon tiedot syötetään (Kuva 23).



Kuva 23. Uuden asiakkaan lisäämis-ikkuna.

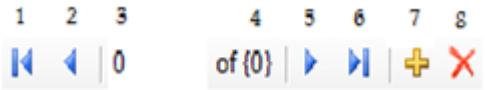
Asiakassyöttö toimii siten, että kun ikkuna aukeaa, painetaan Lisää uusi-painiketta, jonka jälkeen kentät vapautuvat tietojen syöttämistä varten. Tietojensyötön päätyttyä painetaan Oknappia ja asiakas on lisätty tietokantaan.

Syöttökentät ovat oletuksena lukittuja ja ne avautuvat lisää uusi -painikkeella.

```
Private Sub BindingNavigatorAddNewItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BindingNavigatorAddNewItem.Click

        EtunimiTextBox.ReadOnly = False
        SukunimiTextBox.ReadOnly = False
        PuhelinTextBox.ReadOnly = False
        LähiosoiteTextBox.ReadOnly = False
        PostinumeroTextBox.ReadOnly = False
        PostitoimipaikkaTextBox.ReadOnly = False
End Sub
```

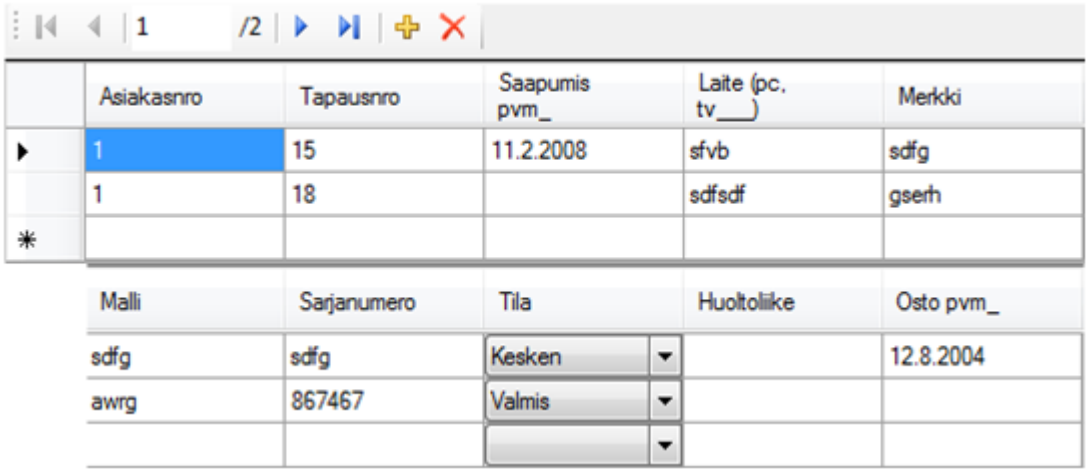
Edellä olevassa koodissa Private Sub kohdassa esitellään lisää uusi -nappi ja myös sen toiminto, joka tässä tapauksessa on lisätä uusi asiakas. Samalla napinpainalluksella tekstikenttien ReadOnly arvo muuttuu Falseksi, jolloin tietoja voi syöttää.

Laitteet-taulun yläpuolella on hiukan erilainen painikerivistö laitteiden omalla sivulla olevaan painikerivistöön verrattuna (Kuva 24). Painikkeet yhdestä kuuteen ovat samat kuin aiemminkin, mutta nyt on tullut lisäksi 7. laitteiden lisäys ja 8. laitteiden poisto. Laitteiden poiston yhteydessä on kuitenkin huomioitava, että jos laitteelle on määritelty raporttitauluun tietoja, on ne poistettava ennen kuin ohjelma antaa poistaa laitetta.



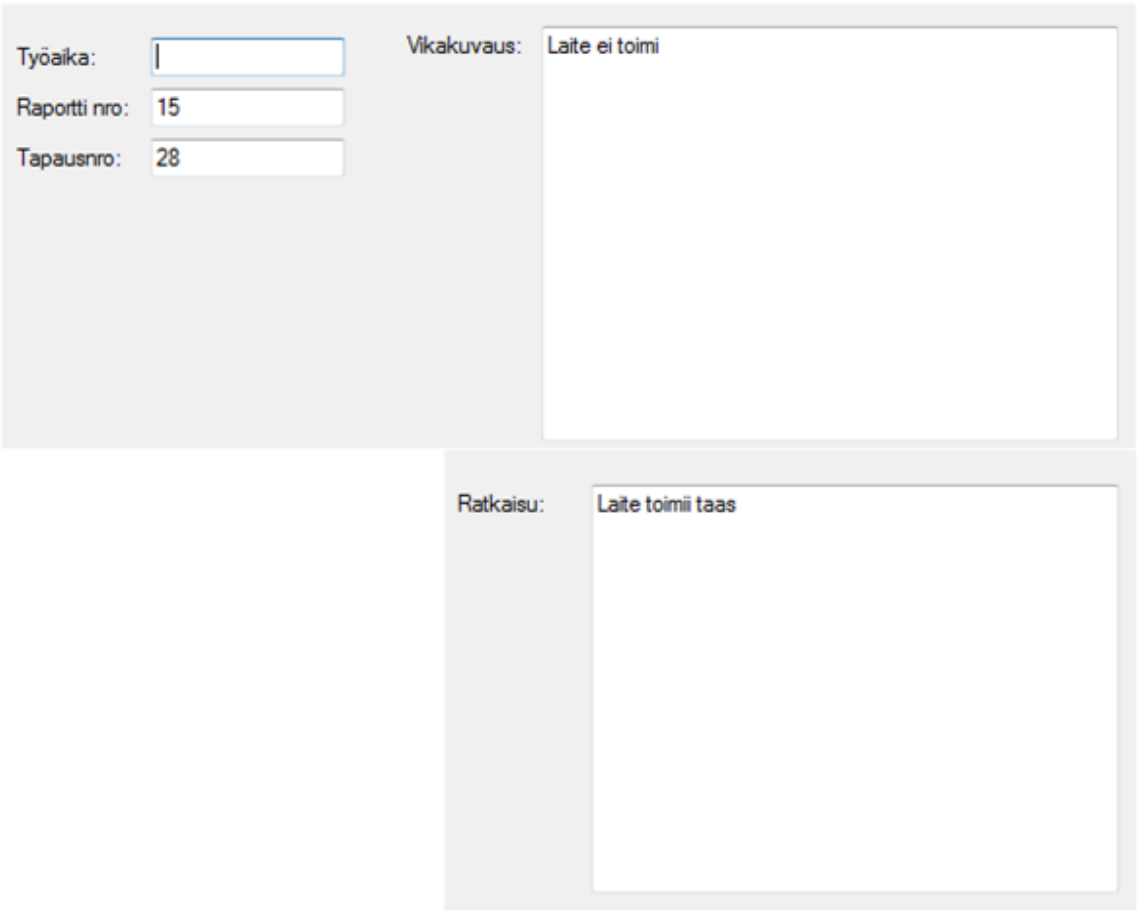
Kuva 24. Tiedot-sivulla oleva laitteisiin linkitetyt painikkeet.

Laitteet-sivulla on monia kohtia, johon tietoja voi syöttää, ja osa saa arvon automaattisesti (Kuva 25). Asiakasnumero tulee automaattisesti, sen mukaan mikä asiakas on valittuna. Tapausnumero on vastaavasti automaattisesti täyttyvä. Tässä tulee arvo juoksevalla numeroinnilla, jotta pystytään tarkastelemaan laitteita omina töinään vaikka sama laite kävisikin useamman kerran huollossa. Voihan yhdessä laitteessa olla useamman kerran vikoja kuin kerran. Saapumispäiväkohtaan kirjataan ylös päivä, jolloin laite on tuotu huollettavaksi. Näin voidaan seurata, että laitteet eivät ole hyllyissä kuukausikaupalla. Seuraavana on kohta Laite. Tähän kirjataan vain ylös onko kyseessä esimerkiksi tietokone, televisio tai jokin muu laite. Merkki kohtaan taas kirjataan laitteen merkki, esimerkiksi Acer. Malli kohtaan taas kirjataan laitteen tarkka malli, koska esimerkiksi pelkkä Fujitsu merkkinä ei välttämättä kerro laitteesta mitään. Fujitsu valmistaa monia eri laitteita. Sarjanumero otetaan myöskin ylös. Jos kysymykseen tulee esimerkiksi takuuhuoltoon lähetys, kysyvät huoltoliikkeet lähes poikkeuksetta laitteen sarjanumeron. Seuraavana on Tila. Tähän valitaan pudotusvalikosta vaihtoehto kesken tai valmis sen mukaan onko laite vielä työn alla vai onko se jo valmistunut. Huoltoliike kohtaan tarvitsee kirjata tietoa vain siinä tapauksessa, jos laite lähetetään jollekin huoltoliikkeelle. Tällöin muistetaan mille huoltoliikkeelle laite on mennyt mahdollisia yhteydenottoja varten. Viimeisenä kohtana on laitteen ostopäivämäärä. Siitä nähdään, onko takuuta jäljellä vai ei ja joka tapauksessa huoltoliikkeet sitä useasti kysyvät.



Kuva 25. Laitteet-taulun sisältö.

Raportti-taulu tarvitaan, että laitteille, joita asiakkaat tuovat, saadaan kirjattua tietoja työnteosta (Kuva 26). Raporttitaulusta löytyy viisi erillistä kohtaa joihin tietoja tallennetaan. Ensimmäisenä on Työaika kohta. Tähän kirjataan työhön käytetty aika, joka meni laitteen tai vian korjaamiseen. Myöhemmin sieltä voidaan katsoa, jos asiakas haluaisi saada arvion kauanko työ kestää. Raporttinumero on juokseva numerointi, joka yksilöi jokaisen raportin. Tapausnumeroon taas tulee automaattisesti sama luku kuin on laite taulussa tapausnumerona. Näin saadaan vikaraportit yhdistettyä oikeaan laitteeseen. Vikakuvauskohtaan kirjataan asiakkaan ilmoittama vika. Ratkaisukohtaan kirjataan miten vika on korjattu, ja mahdollisesti voidaan hieman kertoa enemmän viasta, joka laitteessa oli.



Kuva 26. Raportti-taulun näkymä.

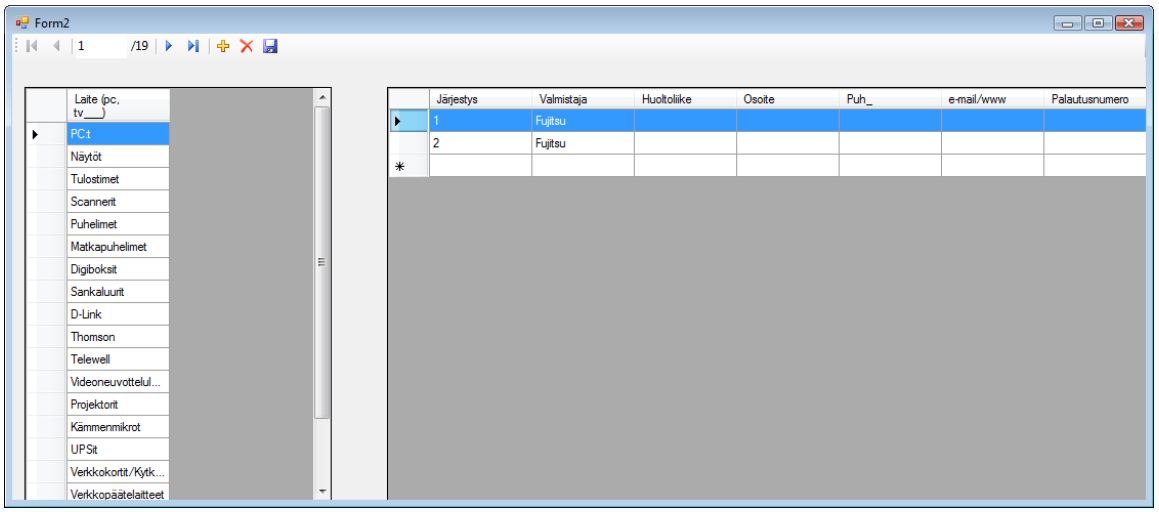
Raportti-taululla on samoin oma painikerivistö, jolla voidaan selailla eri raportteja. Myös painikkeet uuden raportin lisäämiselle ja poistamiselle (kuva 27).

1. Painike siirtää valinnan ensimmäiseen asiakkaaseen.
2. Painike siirtää valintaa yhden askeleen ylöspäin.
3. Tämä näyttää numeron, monennessako asiakkaassa valinta on.
4. Tämä näyttää asiakkaiden kokonaismäärän.
5. Painike siirtää valintaa yhden askeleen alaspäin.
6. Painike siirtää valinnan viimeiseen asiakkaaseen.
7. Tämä painike lisää uuden asiakkaan taulun loppuun, johon voidaan syöttää tietoja.
8. Tämä painike poistaa valitun asiakkaan tiedot.

Kuva 27. Raportti-taulun hallinta painikkeet

9.5 Huoltoliikkeet

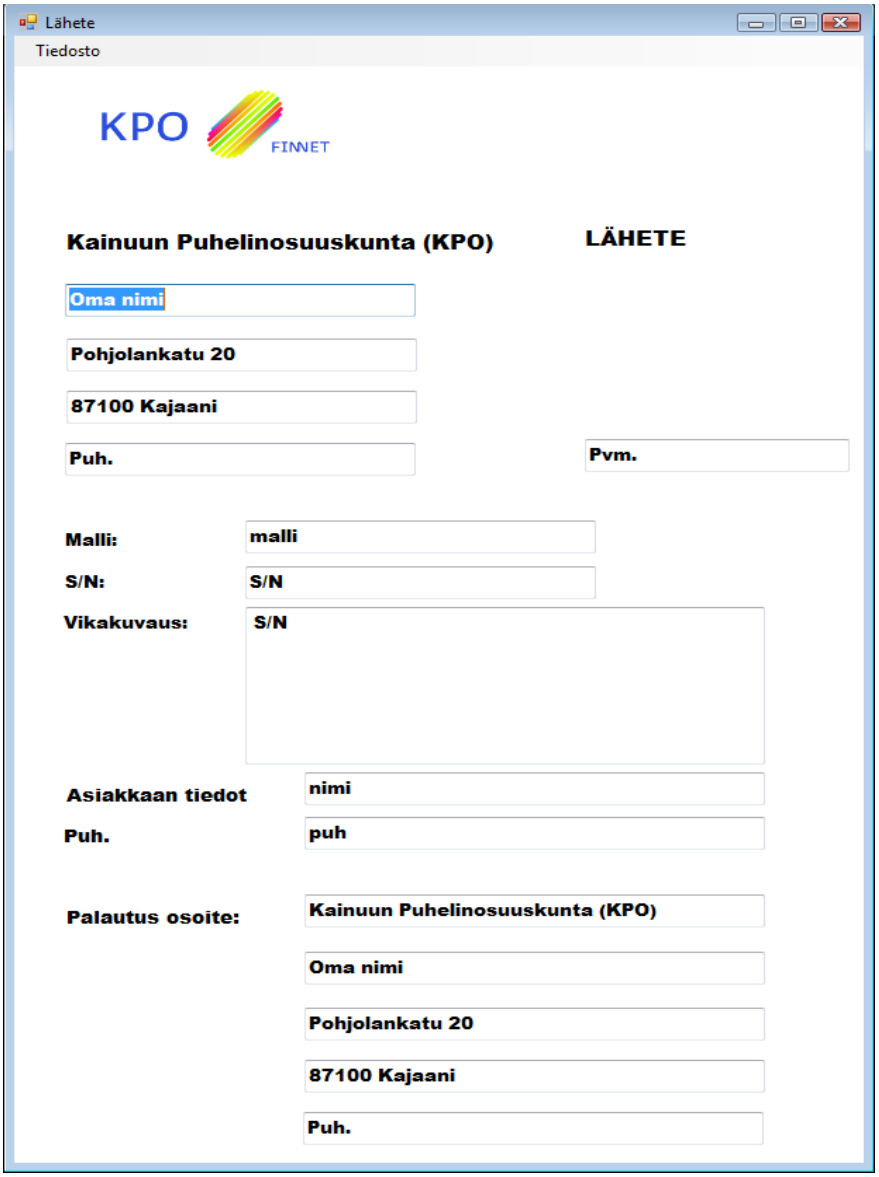
Huoltoliikkeet sivulla on listattu huoltoliikkeet ja niiden tiedot, kuten puhelinnumero, sähköpostiosoite ja niin edelleen, jotta saataisiin helposti eri laitteiden takuukorjaajien tiedot yhdestä paikasta (Kuva 28).



Kuva 28. Huoltoliikkeet-näkymä.

9.6 Tuloste

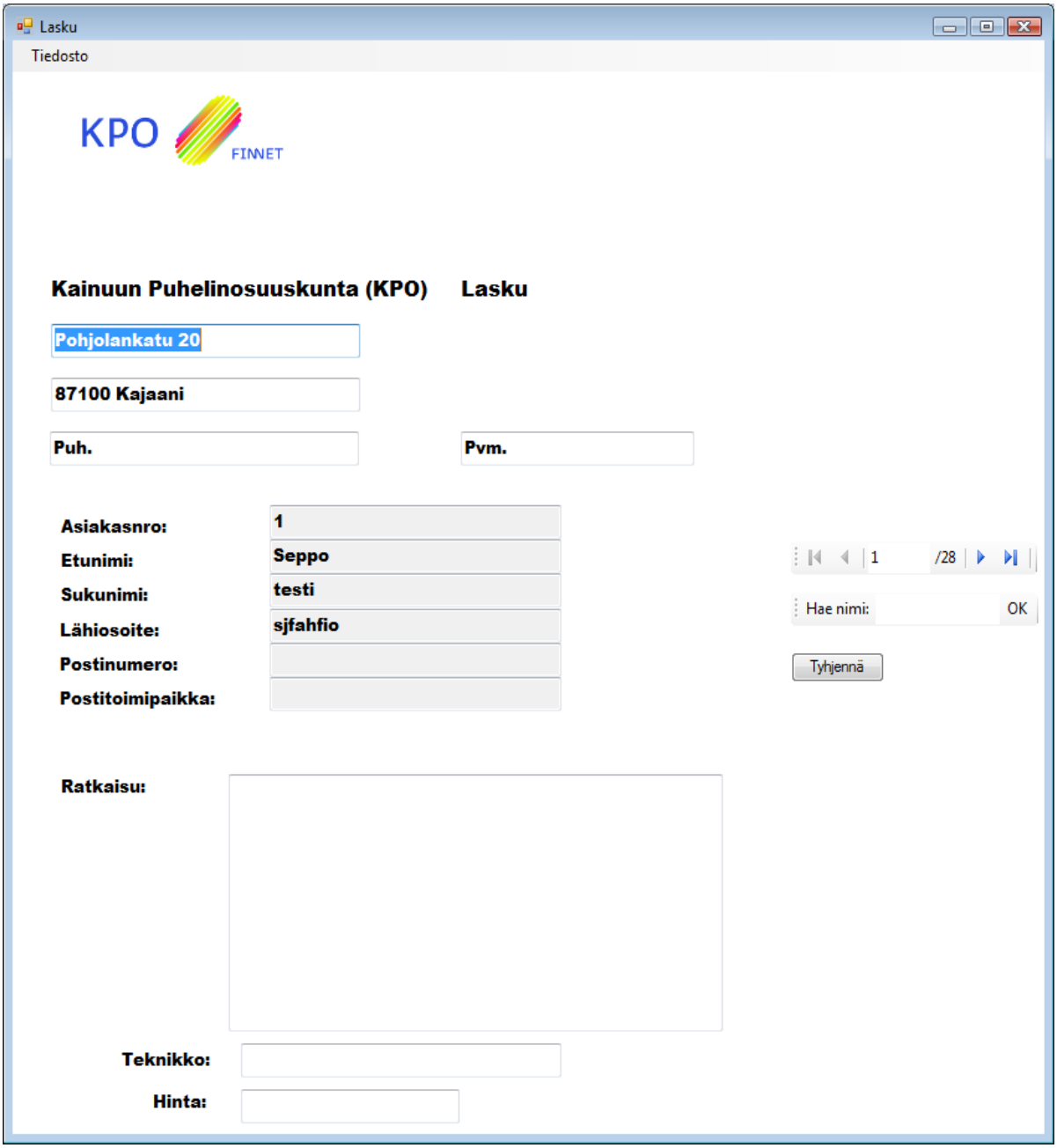
Lomake-valikossa on alivalikko tulosta, joka sisältää kaksi tulostettavaa sivua. Ensimmäisenä tulostettavissa on lähete, jos laitteita joudutaan lähettämään huoltoon (Kuva 29). Tämä on malli, johon täytetään kaikki tarvittavat tiedot, jonka jälkeen lomake tulostetaan ja lähetetään laitteen mukana esimerkiksi maahantuojalle.



Kuva 29 Kuva lähetelomakkeesta

Lähete-sivun valikosta löytyy myös kohta tallenna, joka tallentaa lomakkeen tiedot tekstinä .txt tai .doc muotoon (Liite 1). Koska lomakkeen tiedot tallennetaan, voidaan tietoja tarkistella jälkeenpäin, jos tuntuu että laite on ollut ikuisuuden matkalla, eikä laitteesta kuulu mitään. Voidaan tarkistaa palautus osoitteet ja muutakin tarpeellista tietoa. Tallenna-painikkeen painamisen jälkeen aukeaa tallenna nimellä ikkuna, johon syötetään tiedoston nimi ja valitaan pääte joko .doc tai .txt ja tallennuspaikka. Tallennus tapahtuu vasta sen jälkeen, kun on syötetty tiedoston nimi ja painettu tallenna, ja tämän jälkeen ohjelma tallentaa tiedot tiedostoon myöhempää tarkastelua varten.

Toinen lomake, joka on tarkoitettu tulostettavaksi on lasku (Kuva 30). Laskusta käy ilmi kaikki tarpeellinen, mitä asiakas tarvitsee. Asiakkaan tiedot tulevat laskulle hakukenttää avuksi käyttäen. Asiakas haetaan nimellä ja asiakkaan tiedot täyttyvät automaattisesti niillä tiedoilla, jotka on tietokantaan syötetty. Vaikkakin hakukenttä on lomakkeella, on se määritelty siten, että se ei tulostuksessa näy, vaan tulostuksessa käy ilmi vain kaikki oleelliset asiat.



Kuva 30. Lasku-lomake.

10 POHDINTAA

Opinnäytetyön lähtökohtana oli tehdä KPO:n yksityisasiakkaita palvelevalle huolto-osastolle asiakas- ja huoltotietokanta, tälle on tarvetta, koska huollossa on töissä paljon työharjoittelijoita. Näin huollon toimintaa voitaisiin nopeuttaa sen työntekijöiden suuren vaihtuvuuden vuoksi. Tarkoituksena oli myös kehittää omaa osaamista uuden SQL Server 2008 järjestelmän osalta sekä tietokantakäyttöliittymän rakentamisen suhteen. Alun vaikeuksia olivat SQL Server 2008 järjestelmän uutuus, sillä se ei ole ollut vielä työn aloittamisen aikaan markkinoilla kovin pitkään. Tämän takia jouduimme opettelemaan paljon uutta asiaa. Myös Visual Basic 2008 Express Edition oli meille uusi kehitystyökalu, johon oli lisätty paljon työskentelyä helpottavia ominaisuuksia aikaisempiin versioihin nähden.

Käytimme Express versioita siitä syystä, että KPO:lla ei ollut tarvetta täydelle versiolle, koska tietokannalle ja käyttöliittymälle riittää Express version ominaisuudet. Työtä voi tulevaisuudessa kehittää tarpeiden mukaan, esimerkiksi lisäämällä verkko-ominaisuuksia. Tietoturvaa voisi parantaa varmuuskopioimalla tiedot fyysisesti toiselle tallennusmedialle nykyisen yhden kovalevyn sijaan.

Työharjoittelussa tutustuimme huollon toimintatapoihin ja aloimme pohtia, kuinka niitä voisi tehostaa. Koska huollossa työskentelee paljon harjoittelijoita, on myös työntekijöiden vaihtuvuus hyvin suuri, tämä taas johtaa suoraan vajeeseen tietopääomassa, siksi huoltohistoriatiedot halutaan säilyttää. Tietopääomalla tarkoitetaan tässä tapauksessa harjoittelijoiden saamia tietoja ja taitoja. Koska jokainen uusi harjoittelija täytyy perehdyttää tehtäviin, syntyi ajatus keskitetystä järjestelmästä, mihin käyttäjä kirjaisi asiakkaiden tiedot ja huoltotapaukset. Järjestelmään merkitään valmiista tapauksista raportit, jolloin vikojen korjaaminen nopeutuu. Käyttäjällä on nyt valmiit ohjeet korjausta varten.

Ensimmäisiä ongelmia tuotti SQL Server järjestelmän asentaminen, joka johtui lähinnä sen monipuolisuudesta. Asennettaessa piti suunnitella tarkkaan mitä ominaisuuksia haluaisimme juuri tätä projektia varten. Sitten oli valittava sopivimmat vaihtoehdot. Meidän täytyi myös suunnitella mihin varsinainen tietokanta asennetaan. Ideaali tilanne olisi ollut, jos käytössä olisi ollut useampia kiintolevyjä. Näin olisi saatu varmuuskopiot fyysisesti eri kiintolevylle, joka olisi lisännyt tietojen säilymisvarmuutta vikatilanteissa.

Tietokannan suunnitteluun saatiin KPO:lta suuntaa antavat tiedot, mutta itse suunnittelu ja toteutus olivat täysin meidän harteilla. Tässä auttoi erityisesti Hovin, Huotarin ja Lahdenmäen kirja Tietokantojen suunnittelu & indeksointi. Asiassa auttoivat myös omat kokemuksemme, joita saimme harjoittelun aikana tekemiemme asioiden kautta. Vaikka näkemyksemme tietokannan rakenteesta oli selvillä, oli suuri työ saada se toteutettua käytännössä, koska SQL Server 2008 oli meille täysin uusi järjestelmä. Työtä tehdessä aikaa kului paljon pelkästään kehitystyökalujen perusasioiden ja ominaisuuksien opetteluun.

Visual Basic 2008 Express Editionilla työskentely sujui vähän jouhevammin, koska meillä oli ollut muutama kurssi tästä aiheesta, tosin 2005 versiolla. Ongelmia tuotti lähinnä tuttujen ominaisuuksien etsiminen ja uusiin ominaisuuksiin tutustuminen. Tietokannan liittäminen Visual Basic projektiin oli täysin uusi kokemus, kuten oli myös tietokantahakujen rakentaminen Visual Basic ympäristössä.

Käytimme Visual Basic 2008:aa, koska Visual Basicissa on kätevät työkalut käyttöliittymän tekemiseen. Koska ohjelmointi ei ollut pääosassa tässä työssä, Visual Basic oli käytännöllisin vaihtoehto graafisen käyttöliittymän rakentamiseen. Näin ei tarvinnut ohjelmoida lähinnä muuta kuin käyttöliittymän toiminnallisuus. Suurimman ohjelmointi ongelman aiheutti lomakkeen tietojen tallentaminen tiedostoon, ongelma saatiin ratkaistua perehtymällä Visual Basic ohjelmointiin syvällisemmin.

Opinnäytetyötä oli mielenkiintoista ja haastavaa tehdä. Kaiken kaikkiaan työ onnistui hyvin ja saavutimme suunnitteluvaiheessa asettamamme tavoitteet, ja opimme paljon uutta Microsoft SQL Server 2008 Express järjestelmästä sekä Microsoft Visual Basic 2008 Express kehitystyökalusta.

LÄHTEET

Ekonoja, A., Lahtonen, T. & Mäntylä, J. 2004 a. Tietokannat. Saatavilla: http://appro.mit.jyu.fi/doc/tiedonhallinta/tietokannat/index0.html (Luettu 3.5.2010).

Ekonoja, A., Lahtonen, T. & Mäntylä, J. 2004 b. Relaatiotietokannat. Saatavilla: http://appro.mit.jyu.fi/doc/tiedonhallinta/tietokannat/index1.html (Luettu 3.5.2010).

Ekonoja, A., Lahtonen, T. & Mäntylä, J. 2004 c. Relaatiotietokantojen peruskäsitteet. Saatavilla: http://appro.mit.jyu.fi/doc/tiedonhallinta/tietokannat/index2.html (Luettu 3.5.2010).

Ekonoja, A., Lahtonen, T. & Mäntylä, J. 2004 d. Tietokannan hallintajärjestelmät. Saatavilla: http://appro.mit.jyu.fi/doc/tiedonhallinta/tietokannat/index3.html (Luettu 3.5.2010).

Ekonoja, A., Lahtonen, T. & Mäntylä, J. 2004 e. Suunnittelun vaiheet. Saatavilla: http://appro.mit.jyu.fi/doc/tiedonhallinta/suunnittelu/index1.html (Luettu 3.5.2010).

Ekonoja, A., Lahtonen, T. & Mäntylä, J. 2004 f. Käsitteellinen mallintaminen. Saatavilla: http://appro.mit.jyu.fi/doc/tiedonhallinta/suunnittelu/index2.html (Luettu 3.5.2010).

Ekonoja, A., Lahtonen, T. & Mäntylä, J. 2004 g. ER-kaavion piirtäminen. Saatavilla: http://appro.mit.jyu.fi/doc/tiedonhallinta/suunnittelu/index3.html (Luettu 3.5.2010).

Hovi, A., Huotari, J. & Lahdenmäki, T. 2005. Tietokantojen suunnittelu & indeksointi. 2. laitos, 1. painos. Porvoo: WS Bookwell Oy.

Huotari, J. & Hovi, A. 2008. Tietokannan suunnittelu. Saatavilla: http://student.labranet.jamk.fi/~huojo/opetus/IIO30100/IIO30100m2.pdf (Luettu 15.4.2010).

Kettunen, S. 2002. Tietojärjestelmän ostaminen – käytännön opas yrityksille. Porvoo: WS Bookwell Oy.

Korhonen, P. 2008. SQL Server 2008 sovelluskehitys – uudet ominaisuudet. Saatavilla:

http://download.microsoft.com/download/4/7/a/47ae4e2b-c576-4ba9-b2d7-5b93ef30206d/DevDays_SQL2008_1.ppt (Luettu 6.5.2010).

KPO, 2009 a. KPO yrityksenä. Saatavilla:
http://www.kpo.fi/index_kfin.asp?pid=103 (Luettu 13.05.2010).

KPO, 2009 b. Historia. Saatavilla:
http://www.kpo.fi/index_kfin.asp?pid=210&level_id=1 (Luettu 13.05.2010).

Laaksonen, A. a. Visual Basic -opas: Osa 1.
Saatavilla: http://www.ohjelmointiputka.net/opas.php?tunnus=vbo_1 (Luettu 20.11.2009).

Laaksonen, A. b. Visual Basic -opas: Osa 2.
Saatavilla: http://www.ohjelmointiputka.net/opas.php?tunnus=vbo_2 (Luettu 20.11.2009).

Microsoft. 2009 a. SQL Server 2008 Express. Saatavilla:
http://www.microsoft.com/sqlserver/2008/en/us/express.aspx (Luettu 20.1.2010)

Microsoft. 2010 b. Recovery Model Overview. Saatavilla: http://msdn.microsoft.com/en-us/library/ms189275.aspx (Luettu 3.1.2010).

Microsoft. 2010 c. Visual Basic and C# Windows Templates. Saatavilla: http://msdn.microsoft.com/en-us/library/0fyc0azh.aspx (Luettu 20.1.2010).

Microsoft. 2010 d. Windows Presentation Foundation XAML Browser Applications Overview. Saatavilla: http://msdn.microsoft.com/en-us/library/aa970060.aspx (Luettu 25.1.2010).

Niteshi, R., Subnivis, P. B. & Wadekar, V. 2008. SQL Server 2008 BI Features. http://www.infosys.com/microsoft/resource-center/documents/bi-sqlserver-2008.pdf (Luettu 6.5.2010).

Pelland, P. 2008. Microsoft Visual Basic 2008 Express Edition. Redmond, Washington: Microsoft Press. Saatavilla: http://blog.mynetx.net/wp-content/uploads/files/mspress-vb2008expr.pdf (Luettu 6.5.2010).

Pitkänen, J. 2009. Microsoft Visual Basic 2008 Express Edition – Ilmainen kehitysympäristö Visual Basicille. Saatavilla: http://www.tietokone.fi/softa/windows/microsoft_visual_basic_2008_express_edition (Luettu 4.12.2009).

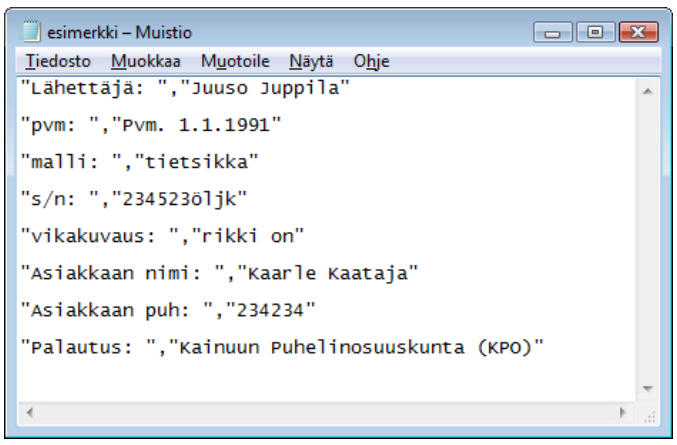
Teratrax, 2010. Understanding SQL Server backup types. Saatavilla: http://www.teratrax.com/articles/sql_server_backup_types.html (Luettu 20.4.2010).

LIITTEIDEN LUETTELO

Tallenna napin koodi:

```
Private Sub TallennaToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles TallennaToolStripMenuItem.Click

    SaveFileDialog1.ShowDialog()

    If SaveFileDialog1.FileName <> "" Then
        FileOpen(1, SaveFileDialog1.FileName, OpenMode.Output) ' Avaa tiedoston, tiedostoon tulostusta
                                                                 varten
        WriteLine(1, "Lähettäjä: ", TextBox1.Text)
        WriteLine(1) ' Tulostaa tyhjän rivin tiedostoon, luettavuuden holpottamiseksi

        WriteLine(1, "pvm: ", TextBox5.Text)
        WriteLine(1) '

        WriteLine(1, "malli: ", TextBox6.Text)
        WriteLine(1) '

        WriteLine(1, "s/n: ", TextBox7.Text)
        WriteLine(1) '

        WriteLine(1, "vikakuvaus: ", TextBox8.Text)
        WriteLine(1) '

        WriteLine(1, "Asiakkaan nimi: ", TextBox9.Text)
        WriteLine(1) '

        WriteLine(1, "Asiakkaan puh: ", TextBox10.Text)
        WriteLine(1) '

        WriteLine(1, "Palautus: ", TextBox11.Text)
        WriteLine(1) '

        FileClose(1)
    End If

End Sub
```

SaveFileDialog1.Show.Dialog() avaa tallenna nimellä ikkunan ja If SaveFileDialog1.FileName <> "" Then tarkastaa onko tiedostonimi kenttään syötetty jotain ja jos on niin ohjelma suorittaa if lauseen, joka tallentaa lomakkeelle syötetyt tiedot tiedostoon.



Esimerkki lähete lomakkeen tallennetusta tiedosta.