LAHDEN AMMATTIKORKEAKOULU
Lahti University of Applied Sciences

TIETOVARASTOSOVELLUKSET

Myynnin ennustussovellus

LAHDEN
AMMATTIKORKEAKOULU
Tekniikan ala
Tietotekniikka
Ohjelmistotekniikka
Opinnäytetyö
Kevät 2012
Jere Aunola

Lahden ammattikorkeakoulu
Tietotekniikka

AUNOLA, JERE:

Tietovarastosovellukset
Myynnin ennustesovellus

Ohjelmistotekniikka opinnäytetyö, 52 sivua

Kevät 2012

TIIVISTELMÄ

Business intelligence tarkoittaa tiedon esittämistä selkeästi ymmärrettävällä tavalla. Ennen kuin tietoa voidaan esittää tai analysoida, pitää se olla saatavilla yhtenäisessä muodossa. Tähän ratkaisuna ovat tietovarastot, joihin kerätään yrityksen tieto. Se ladataan yleensä useasta eri lähdejärjestelmästä, joissa se voi olla monessa eri muodossa. Tietoa ladattaessa tietovarastoon se muutetaan yhtenäiseksi työalueella. Työalueelta se sijoitetaan tietovarastoon. Tietovaraston tietokanta koostuu dimensio- ja faktatauluista.

Tietovaraston tietoa pitää tämän jälkeen esittää selkeällä ja ymmärrettävällä tavalla, jotta siitä on hyötyä. Tietoa voidaan analysoida esimerkiksi käyttäen OLAP -kuutiota, johon voidaan esilaskea arvoja. OLAP mahdollistaa moniuloitteisen tiedon analysoinnin ja tietoon porautumisen. Nykypäivänä yksi tärkeimmistä asioista yrityksille on niillä oleva tieto. Yritykset voivat sen avulla tarkkailla tilannettaan ja esimerkiksi ennustaa tulostaan.

Työssä toteutettiin myynnin ennustussovellus. Ennustesovelluksella asiakas ennustaa tuotetasolla myyntiä. Asiakkaalla oli aiemmin käytössään web-portaali ennustamista varten, mutta se oli liian hidas käytettäväksi. Sovellus toteutettiin C#-ohjelmointikielellä ja .NET-ohjelmointikehyksellä. Sovellus koostuu kahdesta selkeästä osasta, jotka ovat käynnistyslatain ja ennustesovellus. Käynnistyslatain hoitaa automaattiset päivitykset, ja se on toteutettu käyttäen ClickOnce:a. Ennustesovellus toimii Microsoft Excelin työkirjana, joka on tehty käyttäen Visual Studio Tools for Officea. Se mahdollistaa liitännäisten ja kolmannen osapuolen komponenttien kehittämisen Office-ohjelmiin.

Avainsanat: tietovarasto, myynnin ennustaminen, OLAP, C#, .NET, VSTO, Visual Studio Tools for Office, ClickOnce, dimensio, fakta, tähtimalli, lumihiutalemalli, business intelligence

Lahti University of Applied Sciences
Degree Programme in information technology

AUNOLA JERE: Data warehouse software
Sales forecasting software

Bachelor's Thesis in software engineering 52 pages

Spring 2012

ABSTRACT

Business intelligence means visualizing data in a clear and understandable way. Before the data can be visualized or analyzed it has to be available in a coherent form. A solution to this issue is a data warehouse where a company's data is stored. Data is loaded from many different source systems where it can be in many different forms. When data is loaded to a data warehouse it will be transformed to a standard form in the work area. The database of Data warehouse commonly consists of dimensio and fact tables.

The data in a data warehouse must be presented clearly so it can be used in effective decision making. Data can be analyzed for example with OLAP cubes where it is possible to pre-calculate values. OLAP makes it possible to analyze the multidimensional data and drilldown. One of the main assets for companies today is the information they posses. They can use it to monitor their performance and for example forecast their turnover.

A sales forecasting application was created in this project. The customer uses the software to forecast sales in product level. Before this software the customer had web-based forecasting software, which was too slow to be used. The software was created using the C# programming language and .NET framework. It consists of multiple pieces but two of them are clearly visible: the startup loader and the forecasting program. The startup loader handles the updates using ClickOnce. The forecasting program runs as a workbook on top of Microsoft Excel. It is created using Visual Studio Tools for Office which makes it possible to write plugins and third-party components to be created for Office programs.

Key words: data warehouse, sales forecasting, OLAP, C#, .NET, VSTO, Visual Studio Tools for Office, ClickOnce, dimension, fact, star model, snowflake model, business intelligence

SISÄLLYS

1 JOHDANTO

Yrityksille kertyy paljon tietoa erilaisista järjestelmistä eri puolilta maailmaa. Onkin tärkeää, että yritykset voivat analysoida ja seurata kehitystään kertyneen tiedon avulla. Yleisimpinä ongelmina kuitenkin on, että tieto on peräisin monesta erilaisesta järjestelmästä, eikä se aina välttämättä ole kovinkaan yksiselitteistä ja ymmärrettävää.

Tässä työssä käsitellään tietovarastosovelluksia ja Business Intelligenceä sekä tiedon keräämiseen ja tallentamiseen käytettäviä menetelmiä. Työssä perehdytään myös Microsoftin tarjoamiin ratkaisuihin, joilla voidaan toteuttaa tietovarastosovelluksia.

Logica on IT-alan yritys, joka toimii 36 eri maassa ja työllistää Suomessa 3200 henkilöä ja maailmanlaajuisesti 41000 henkilöä. Logica tarjoaa asiakkailleen konsultointia ja tietojärjestelmien integrointia. (Logica 2012.)

Työ tehtiin Logica Suomi Oy:lle, joka myy sovelluksen asiakkaalle. Tarkoituksena oli toteuttaa myynin ennustussovellus, jolla asiakas voi ennustaa myyntiä tuotetasolla. Asiakkaalla oli aiemmin käytössään web-portaali, joka tarjosi ominaisuudet ennustamiseen. Portaali toimi kuitenkin hitaasti ja vaati aina internetyhteyden. Tästä syystä aloitettiin uuden version kehitys.

Kehitys alkoi keväällä 2011 toteuttamalla niin sanottu POC (Proof Of Concept) –versio, jonka toteutti Logica. POC -version tarkoitus oli selvittää, onko mahdollista toteuttaa uusi versio käyttäen Microsoft Office Exceliä käyttöliittymänä vanhan web-portaalin sijaan. Lisäksi haluttiin tietää, saavutettaisiinko tällaisella ratkaisulla huomattavasti nopeammin toimiva toteutus.

Tämä opinnäytetyö kertoo ennustesovelluksen toteuttamisesta edellä mainitun POC -version pohjalta. Sovellus toteutettiin 2011 kesällä, POC –version hyväksymisen jälkeen. Asiakas otti sovelluksen käyttöönsä alkusyksystä 2011.

Toisessa luvussa käsitellään tietovarstosovelluksia, tietovarastoja sekä Microsoftin tarjoamia ohjelmistokehyksiä ja ratkaisuja, joita käytettiin sovelluksen toteuttamisessa. Kolmannessa luvussa esitellään myynnin ennustussovellus ja tarkastellaan sen toteutusta, rakenteita ja ominaisuuksia. Sovelluksen toteuttamisessa käytettiin luvussa kaksi esiteltyjä tekniikoita ja menetelmiä. Luvussa kerrotaan myös erinäisistä ongelmista, joihin toteutuksessa kohdattiin. Muutamia kehitysideoita sovelluksen tulevaisuuden kannalta on myös esitelty. Neljännessä luvussa kerrotaan kehitysideoita, joiden avulla sovellus voidaan tuotteistaa. Luvussa käsitellään myös ylläpitoa ja laajennettavuutta edesauttavia kehitysehdotuksia. Viides luku sisältää yhteenvedon, sovelluksen tulevaisuuden näkymät ja johtopäätökset.

2 TIETOVARASTOSOVELLUKSET

Tietovarastosovelluksilla pyritään esittämään ja analysoimaan tieovarastoissa olevaa tietoa. Business Intelligence eli BI tarkoittaa tiedon esittämistä selkeässä ja ymmärrettävässä muodossa. Yleisesti BI sisältää raportointia, analysointia ja mallintamista. (Logica 2009, 34.)

Raportointiominaisuudet mahdollistavat yrityksen suorituskyvyn tarkkailun. Analysointi auttaa yritystä mahdollisuuksien tunnistamisessa sekä huonon tai hyvän suorituskyvyn syiden selvittämisessä. Mallintaminen helpottaa suorituskyvyn ennakoimista yrityksessä. (Logica 2009, 34.)

BI ei ole vain faktoja ja lukuja tietokoneen näytöllä. Suuri määrä rivejä numeroineen esittämässä tarkkoja myynti- tai tuotantolukuja voi olla erittäin tarkkaa tietoa. Se ei kuitenkaan ole Business Intelligenceä ennen kuin tiedot on laitettu muotoon, jota niitä tarvitseva päätöksentekijä voi ymmärtää. (Larson 2009, 11.) Ytimekkäät yhteenvedot asiakastyytyväisyydestä tai kokoonpanolinjan tehokkuudesta voivat olla helposti ymmärrettäviä. Nekään eivät kuitenkaan ole Business Intelligenceä, ennen kuin ne voidaan toimittaa niin ajoissa, että ne ehtivät vaikuttaa päivittäiseen päätöksentekoon. (Larson 2009, 11.) Business Intelligencen on siis tarkoitus tukea tehokasta päätöksentekoa. Se tarjoaa perusteellista tietoa, jolle päätöksenteon voi perustaa. (Larson 2009, 12.)

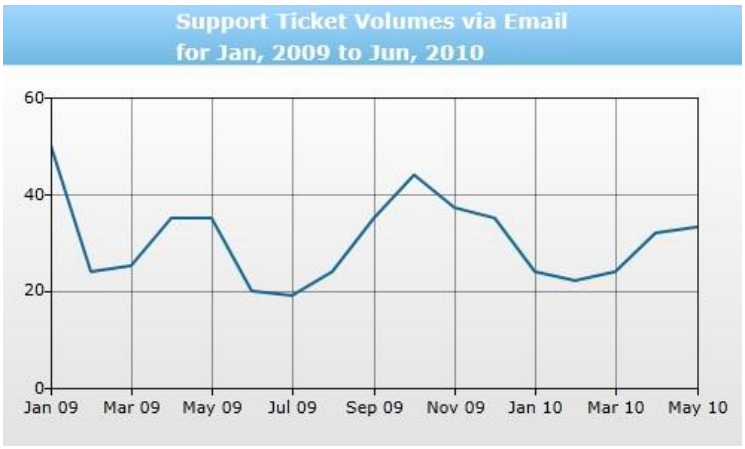
Vaikka BI on tärkeää koko yritykselle, samaa informaatiota ei yleensä tarvita yrityksen eri tasoilla. Yrityksen eri tasot vaativat erilaista Business Intelligenceä tehokkaan päätöksenteon tueksi. Yrityksen korkealla tasolla olevat päätöksentekijät katsovat kokonaiskuvaa ja asettavat pitkän aikavälin tavoitteet. Päätöksentekijöillä pitäisi olla laaja yleisnäkymä heidän vastuualueeltaan. (Larson 2009, 16.)

2.1 Business Intelligencen historia ja tulevaisuus

1970- ja 1980-luvuilla markkinoille alkoi ilmestyä sovelluspaketteja, jotka tarjosivat tiedon analysointia. Laskentatehon puute ja huono käyttäjäystävällisyys kuitenkin rajoittivat BI-työkalujen leviämistä laajaan käyttöön. Taulukkolaskentasovelluksien julkaiseminen 1980-luvulla avasi loppukäyttäjille mahdollisuuden luoda omia tietomalleja liiketoiminnan analysoimiseksi. Taulukkolaskentasovelluksia käytetään edelleen paljon ja tullaan varmasti käyttämään myös tulevaisuudessa liiketoiminnan analysoimiseen. 1990-luvulla yleistynyt SQL-tietokantojen ja tietovarastojen käyttö on mahdollistanut nopean kasvun BI-työkalujen käytössä. (Rasmussen, Goldy & Solli 2002, 3-4.)

Nykyään moni BI-ohjelmistoja tarjoava yritys on julkaissut web-pohjaisia ratkaisuja. Yritykset voivat helposti ja edullisesti antaa käyttäjille pääsyn yrityksen suureen tietomäärään ja hienostuneisiin analysointityökaluihin. Internet- tai intranetyhteydellä henkilö voi tutkia ja analysoida tietoa analysointisovelluksilla lähes mistä tahansa, vaikka omasta kodistaan tai linja-auton penkiltä. (Rasmussen, Goldy & Solli 2002, 4.)

Yksi tärkeimmistä BI:hin vaikuttavista trendeistä on toimittajien siirtyminen kohti kaupallisia tietokanta-alustoja BI-sovelluksissaan. BI-sovellus on korkeintaan yhtä hyvä kuin sen saatavilla oleva tieto. On siis tärkeää, että tieto on helposti luettavissa analysointityökaluilla tai siirrettävissä tietovarastoihin. (Rasmussen, Goldy & Solli 2002, 5.) Kuviossa 1. on esitetty Dundas sovelluksella toteutettu graafi, joka esittää sähköpostin kautta tulleiden tukipyyntöjen määrää.



KUVIO 1. Tiedon esittäminen graafina (Dundas 2012)

Nykyään lähes kaikki modernit analysointityökalut käyttävät Online Analytical Processing (OLAP) -kuutioita tietokantanaan. Tämä mahdollistaa joustavat näkymät ja nopean porautumisen. (Rasmussen, Goldy & Solli 2002, 13-14.)

Porautuminen tarkoittaa sitä, että käyttäjä voi halutessaan tarkastella valittua tietoa tarkemmin. Tällöin järjestelmä automaattisesti generoi uuden raportin tai näkymän, joka sisältää tarkemman tiedon. (Rasmussen, Goldy & Solli 2002, 18.)

Business Intelligence -ratkaisut ovat jo nykyään lähes aina web-pohjaisia portaaleja, joissa voidaan analysoida ja tulkita raportteja. Nykyään on olemassa paljon BI-ratkaisuja, joissa käytetään niin sanottuja dashboardeja. Dashboardilla esitetään yhdellä näytöllä useita erilaisia tietoja. Tälläinen dashboard on esiteltynä kuviossa 2. siinä esiteteään myynnin tunnuslukuja. Myös eLearning on noussut keskeiseksi osaksi BI:tä. eLearning eli verkko-oppiminen mahdollistaa käyttäjille esimerkiksi BI-järjestelmän itseopiskelun. eLearningin avulla käyttäjät voivat opiskella järjestelmän käyttöä koska tahansa. He voivat myös suorittaa kurssit niin monta kertaa kuin näkevät sen tarpeelliseksi. Järjestelmän opetus voidaan myös aloittaa ennen kuin järjestelmä on edes otettu käyttöön. Lisäksi käyttäjät voivat myöhemmin kerrata kursseja, jos he kokevat siihen tarvetta. (Rasmussen, Goldy & Solli 2002, 25-29.)

KUVIO 2. Myynnin dashboard Dundas web-sovelluksesta (Dundas 2012)

BI-sovelluksista tulee aina vain tehokkaampia, ja ne tarjoavat yhä enemmän ja enemmän toiminnallisuuksia. Tällaisia ovat esimerkiksi porautuminen, graafit, taulukot, pivotointi, kommentit, dashboardit ja muut yleiset analysointiominaisuudet, kuten suodatus ja lajittelu. (Rasmussen, Goldy & Solli 2002, 33.)

2.2 Tietovarastot

Yksi yrityksen tärkeimmistä asioista on sen tieto. Se yleensä säilytetään operationaalisissa järjestelmissä ja tietovarastossa. Karkeasti sanottuna operationaalinen järjestelmä on järjestelmä, johon tiedot kirjataan. Tietovarastosta puolestaan luetaan tietoa. (Kimball & Moss 2002, 2.)

Operationaalisen järjestelmän käyttäjät ovat niitä henkilöitä, jotka ottavat tilauksia vastaan, rekisteröivät uusia asiakkaita ja kirjaavat valituksia. He myös yleensä

käsittelevät vain yhtä tallennetta kerrallaan ja toistavat samaa operaatiota uudelleen ja uudelleen. (Kimball & Moss 2002, 2.)

Tietovaraston käyttäjät seuraavat ja tarkkailevat yrityksen toimintaa esimerkiksi laskemalla tilaukset ja vertaamalla niitä edellisen viikon tilauksiin. He eivät myöskään lähes koskaan käsittele vain yhtä tapahtumaa kerrallaan vaan haluavat usein vastauksen kysymykseen, jonka ratkaisu vaatii satojen tai tuhansien tapahtumien kokoamisen yhteen. (Kimball & Moss 2002, 2.)

2.2.1 Tietovarastojen historia

Monet ihmiset eivät ymmärrä tietovarastojen perusperiaatetta heti kättelyssä, vaan he saattavat ihmetellä, miksi tietoa kopioidaan tietokannasta toiseen. Heidän mielestään siinä ei ole mitään järkeä, ja he näkevät sen vain ajan hukkana. Miksi tietoa ei vain haeta suoraan lähdejärjestelmästä kun sitä tarvitaan? Tämän ymmärtämiseksi on tärkeää tutusta tietovarastojen historiaan. (Hammergren & Simon 2009, 14.)

Tietovarastot johtavat juurensa aikaan, jolloin tietokoneet eivät vielä olleet laajalti saatavilla. Markkinointitutkimus johtaa juurensa aina 1800-luvun lopulle. Sitä voidaan myös pitää omalla tavallaan tietovarastojen syntymisenä. Markkinointitutkimus vaatii korkeatasoisen informaation keräämistä, jotta pystytään laatimaan trendejä ja tekemään päätöksiä liikemailmassa. (Hammergren & Simon 2009, 14.)

1970-luvulla keskustietokoneet olivat yleisiä. Niissä suoritettiin tiedonkäsittelyohjelmia, jotka käsittelivät monimutkaisia tiedostoja tai ensimmäisiä tietokantoja. Nämä kannat eivät kuitenkaan muistuttaneet lähellekään relaatiotietokantoja, jotka ovat nykyään yleisimpiä kantoja. Keskustietokoneet tekivät yllättävän hyvää työtä tiedon prosessoinnissa, mutta ongelmana oli tiedon välittäminen ja vertaileminen. Erittäin hankalaa, ellei lähes mahdotonta, oli verrata kahdella eri alueella sijaitsevien yritysten tietoja. Parhaimmillaan saattoi joutua lähettämään kirjeen tietojenkäsittelyosastolle, jossa pyysi haluamiaan tietoja. Vastauksen sai noin kuukauden kuluttua, jos sai vastausta ollenkaan. (Hammergren & Simon 2009, 15.)

1980-luvulla PC-tietokoneet alkoivat yleistyä, eivätkä tietokoneohjelmat enää olleet vain keskustietokoneilla. Ongelmaksi nousi se, miten organisaatio saattoi kilpailla, kun sen tieto oli nyt hajautettuna organisaation sisällä eri tietokoneilla. Vaikka ongelma oli ollut olemassa jo aiemmin, silloin kun tiedot olivat vielä keskustietokoneella, ne olivat kuitenkin usein eristettyinä eri tiedostoissaan. Yhtenä ratkaisuna oli DDBMS (distributed database management system), eli hajautettu tietokantojen hallintajärjestelmä, jonka tarkoitus oli hakea halutut tiedot eri tietokannoista organisaation sisällä käyttäjälle. DDBMS-konsepti oli hyvä ja tutkimukset olivat lupaavia, mutta lopputulos oli kuitenkin toinen. Tämä tekniikka ei vain toiminut oikeassa elämässä. (Hammergren & Simon 2009, 16.)

Teradata aloitti maailman ensimmäisen kaupallisen relaatiotietokantahallintajärjestelmän (RDBMS, relational database management system) testiversion toimittamisen vuonna 1983. (Hammergren & Simon 2009, 16.)

Vuonna 1986 Ralph Kimball perusti yrityksen nimeltä Red Brick Systems. Yritys mainosti erikoistunutta relaatiotietokantajärjestelmää, jonka etuna oli suorituskyky monimutkaisissa kyselyissä. Se oli jopa kymmenenkertainen muiden sen aikaisten kilpailijoiden järjestelmiin verrattuna. Syynä suorituskykyyn olivat indeksit. 1980-lukua voidaankin pitää tietovarastojen syntymävuotena. (Hammergren & Simon 2009, 17.)

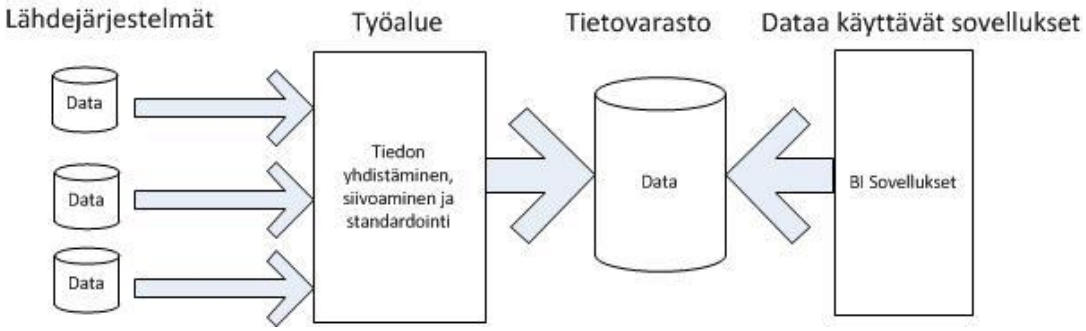
1990-luvun aikana syntyi käsite nykyisestä tietovarastoinnista. Silloin otettiin mallia 1970-luvulla käytössä olleesta mallista, jossa tietoa kopioitiin paikasta toiseen. Tällä kertaa se toteutettiin oikein ja 1993 Bill Inmon kirjoitti kirjan Building the Data Warehouse. Monet tuntevatkin Inmonin tietovarastoinnin isänä. Muitakin julkaisuja ilmestyi 1990-luvun aikana, kuten Ralph Kimballin vuonna 1996 julkaisema The Data Warehouse Toolkit. (Hammergren & Simon 2009, 17-18.)

2000-luvulla tietovarastointiin on tullut uusia haasteita ja tekniikoita kehitetään jatkuvasti. Ihmiset ovat yhteydessä toisiinsa enemmän ja helpommin kuin koskaan aiemmin. Tietomäärät kasvavat valtavasti ja asettavat uusia haasteita tietovarastoinnille. Suuret yhtiöt, kuten Microsoft, Oracle ja IBM, ovat tuoneet omia BI-

ratkaisujaan markkinoille ja ostaneet 90-luvun tietovarastointiyrityksiä itselleen. (Hammergren & Simon 2009, 18-19.)

2.2.2 Tietovaraston periaatteet

Tietovarasto koostuu erilaisita elementeistä, jotka toteuttavat jonkin tietyn toiminnon. Yleisesti tietovarastossa on neljä erillistä elementtiä, jotka on esitetty kuviossa 3. Elementteihin kuuluvat yleensä lähdejärjestelmät, työalue, tietovarasto ja dataa käyttävät työkalut. (Kimball & Ross 2002, 6-7.)

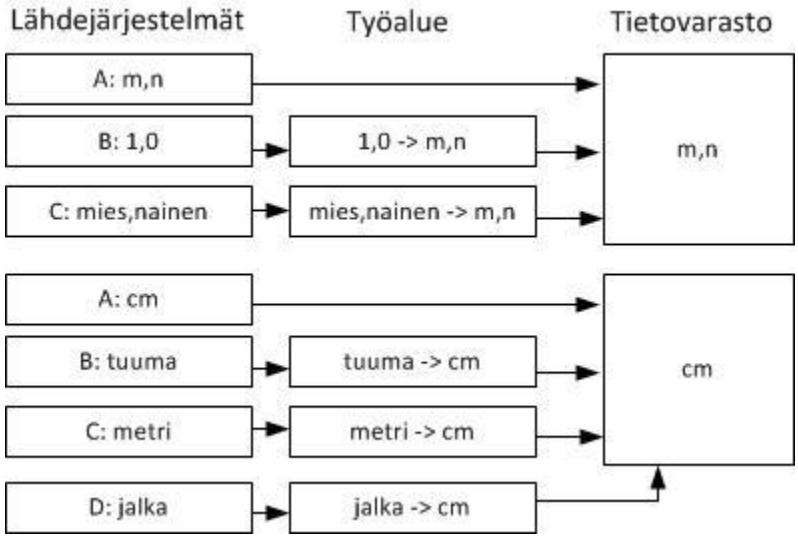


KUVIO 3. Tietovaraston peruselementit

Operationaaliset lähdejärjestelmät sisältävät tiedot yrityksen tapahtumista. Niiden tärkeimmät vaatimukset ovat nopea suorituskyky ja saatavuus. Yleensä näihin järjestelmiin tehtävät kyselyt ovat pieniä ja koskevat yhtä tai muutamaa riviä. Lähdejärjestelmät sisältävät yleensä vain hyvin vähän historiatietoa. Lähdejärjestelmät koostuvat yleensä useista erilaisista sovelluksista ja niiden tietokannoista. Niiden sisältämä tieto on harvoin samanlaista. Yleensä sama tieto on eri tavalla tallennettuna eri järjestelmiin. Ihannetilanteessa lähdejärjestelmien tiedot olisivat aina samanlaisia, mikä helpottaisi suuresti tietovarastojen toteuttamista. (Kimball & Ross 2002, 7.)

Työalue, josta käytetään myös termiä ETL (Extraction-Transformation-Load), on lähdejärjestelmien ja tietovaraston välissä sijaitseva alue. Sinne ladataan ja tallen-

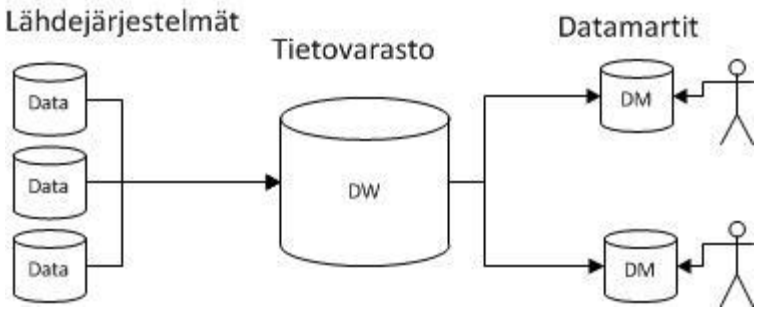
netaan tiedot lähdejärjestelmistä käsiteltäväksi. Työalueelle ladattu tieto käsitellään työalueella yhtenäiseksi. (Kimball & Ross 2002, 8.) Tämä on esiteltynä kuviossa 4, jossa myös ilmenee, miten sama tieto on eri lähdejärjestelmissä tallennettu eri tavalla. Siinä kolmesta eri lähdejärjestelmästä tuodaan työalueelle sukupuoli ja pituus. Kaikissa lähdejärjestelmissä on käytössä eri yksiköt, joten ne muutetaan työalueella yhdeksi. Esimerkiksi sukupuolen ilmaisemiseen on valittu käytettäväksi merkintää m (mies) tai n (nainen), jolloin vain yhdestä lähdejärjestelmästä tieto tulee oikein. Muissa se on esitetty joko 1,0 tai mies, nainen, ja ne muutetaan työalueella muotoon m,n.



KUVIO 4. Tiedon yhtenäistäminen

2.2.3 Datamartit

Yrityksen ollessa iso, sen tietovarasto saattaa kasvaa hyvinkin suureksi. Suuri koko hidastaa raportointia ja kyselyitä. Yleensä yrityksen osastoilla tai muilla toiminnoilla on tarve katsella vain omia tietojaan. Tällaisissä tilanteissa voidaankin muodostaa tietovarastosta pienempiä tietokantoja, kuten kuviossa 5 on esitetty. Näitä kantoja kutsutaan datamarteiksi. (Hovi 1997, 35.)



KUVIO 5. Datamartit tietovarastossa

Datamarttien ansiosta tietokannan käyttö nopeutuu huomattavasti, kun raportointi ja kyselyt käyttävät niitä. Tietovarasto voidaan myös rakentaa pelkästään datamarteista. (Hovi 1997, 36.)

2.2.4 Faktataulut

Faktataulu on niin sanottu päätaulu moniuloitteisessa mallissa. Tauluun talletetaan numeeriset liiketoiminnan suorituskykyä kuvaavat arvot. Kuviossa 6. on esimerkki siitä, mille myynnin faktataulu voisi näyttää. (Kimball & Ross 2002, 17.) Faktataulujen perusavain (Primary Key) koostuu dimensiotaulujen perusavaimista (Hovi 1997, 73).

Sales Fact	
PK	OrderNumber
PK	OrderLineNumber
PK	ProductKey
PK	OrderDateKey
PK	CustomerKey
PK	EmployeeKey
	OrderQuantity
	ProductTotalCost
	SalesAmount

KUVIO 6. Esimerkki faktataulusta

Faktatauluissa aika on yleensä aina yhtenä dimensiona. Faktataulujen koko kasvaa järjestelmässä jatkuvasti, kun niihin lisätään uusia rivejä. Vanhoja tietoja ei päivitetä, koska halutaan säilyttää myös historiatiedot. Faktataulun rivi on kaikkien siihen liittyvien dimensioiden risteyskohta. (Hovi 1997, 76.) Kuviossa 7. on esitelty sellaista dataa, jota kuvion 6. faktataulu voisi pitää sisällään.

Sales Fact

OrderNumber	OrderLineNumber	ProductKey	OrderDateKey	CustomerKey	EmployeeKey	OrderQuantity	ProductTotalCost	SalesAmount
10001	1	2	5	3	4	1	10	10
10002	2	2	2	5	4	2	20	20
10003	3	5	15	10	8	5	250	250

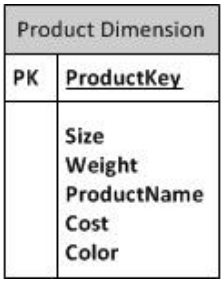
KUVIO 7. Esimerkki faktataulun sisällöstä

Faktatauluissa on yleensä useita viiteavaimia (Foreign Key), jotka viittaavat dimensioihin. Faktatauluihin lisätään ainoastaan rivit, joilla on tapahtumia. Tästä syystä voidaankin sanoa, että faktataulu on niin sanotusti harva. Faktataulut vievät kuitenkin noin 90 % moniuloitteisen tietokannan tilasta. (Kimball & Ross 2002, 18-19.)

2.2.5 Dimensiotaulut

Toisin kuin faktatauluissa, dimensiotauluissa saattaa tapahtua ajan myötä muutoksia riveillä. Esimerkiksi jonkin tuotteen nimi saattaa muuttua. Tämän jälkeen joudutaan pohtimaan, muutetaanko myös historiassa olevien tuotteiden nimet. Jos nimet muutetaan, voidaan verrata nykyisiä tietoja edellisiin aivan kuin mitään muutosta ei olisi koskaan tapahtunutkaan. Jos kuitenkin päätetään, ettei muutosta tehdä historiaan, ei nykyisiä tietoja voida helposti verrata historian kanssa. Tällöin historia pysyy muuttumattomana ja näkyy oikein. Dimensiotauluissa on useita kenttiä, joita kutsutaan attribuuteiksi. Dimensiotaulut myös ovat yleensä hyvin pieniä faktatauluihin verrattuna. (Hovi 1997, 73-77.) Alla olevassa kuviossa 8. on

esimerkki tuotedimensiotaulusta, joka sisältää attribuutit: koko (size), paino (weight), tuotenimi (product name), hinta (cost) ja väri (color).



KUVIO 8. Esimerkki dimensiotaulusta

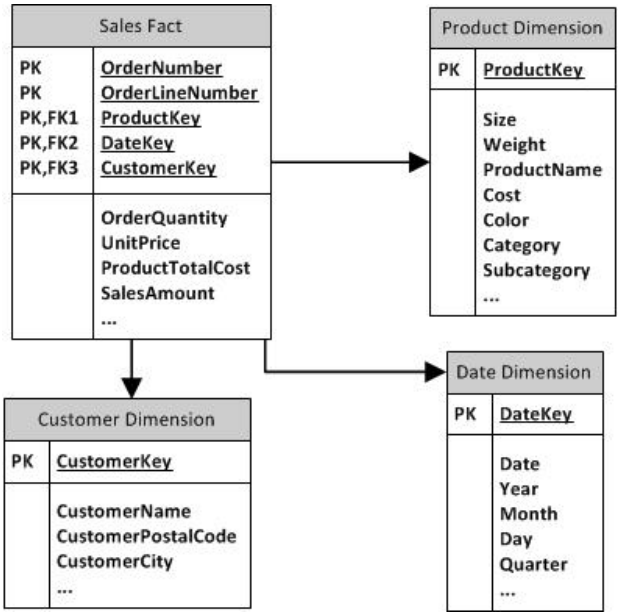
Dimensiotaulut sisältävät yleensä tekstimuotoisia kuvauksia. Dimensiot ovat tärkeä osa tietovarastoa, koska ne ovat lähes ainoa lähde kiinnostaviin teksteihin ja kuvauksiin. Dimensiot tekevät tietovarastosta ymmärrettävän ja käytettävän. Mitä enemmän aikaa käytetään attribuuttien ja oikean terminologian käyttöön dimensioita tehdessä, sitä parempi tietovarastosta tulee. (Kimball & Ross 2002, 19-21.) Esimerkki kuvion 8. dimensiotaulun sisällöstä on nähtävillä kuviossa 9.

Product Dimension					
ProductKey	Size	Weight	ProductName	Cost	Color
10000	8	1,5	Ball	15	Yellow
10001	5	1	Ball	10	Red
10002	10	2	Ball	20	Green
10003	18	3,5	Ball	50	Gray
10004	15	5,2	Ball	25	Black

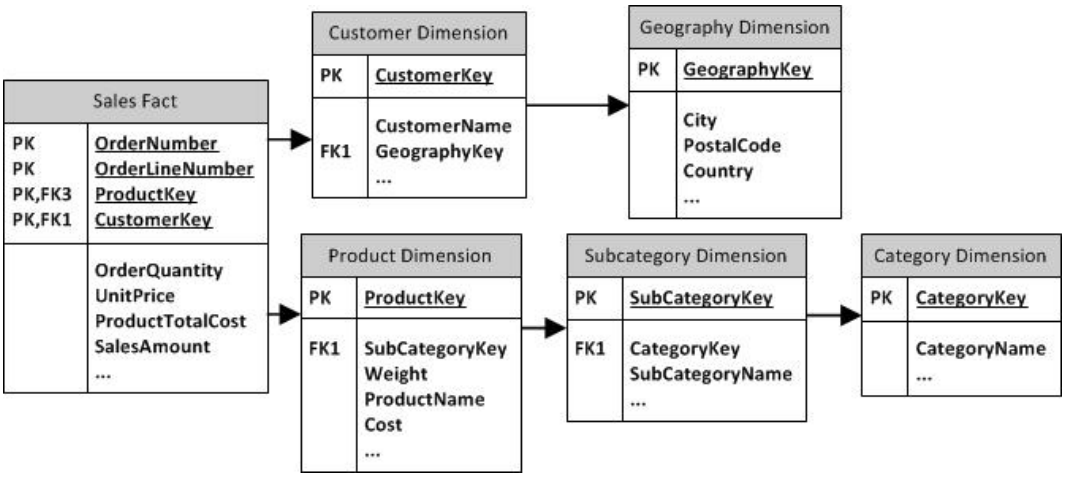
KUVIO 9. Esimerkki tuote dimensiotaulun sisällöstä

2.2.6 Tähti- ja lumihiutalemalli

Faktataulut ovat yleensä normalisoituja, kun taas dimensiotaulut ovat denormalisoituja. Tällöin tauluista syntyy yhteen liitettäessä niin sanottu tähtimalli, joka on esitetty kuviossa 10. Normalisointi tarkoittaa sitä, ettei taulussa ole kahteen kertaan samaa avaintietoa, eli niin kutsuttuja tuplarivejä. Denormalisoidussa taulussa puolestaan tietoa toistetaan. Dimensiotaulujen tieto voidaan myös normalisoida, jolloin tähtimalli muuttuu lumihiutalemalliksi. Tämä johtuu siitä, että dimensiotaulut jakautuvat useammaksi peräkkäisiksi dimensioiksi. (Kimball & Ross 2002, 21-22.) Kuviossa 11. on nähtävillä lumihiutalemalli, joka pohjautuu kuvion 10. tähtimalliin.



KUVIO 10. Tähtimalli (Star)

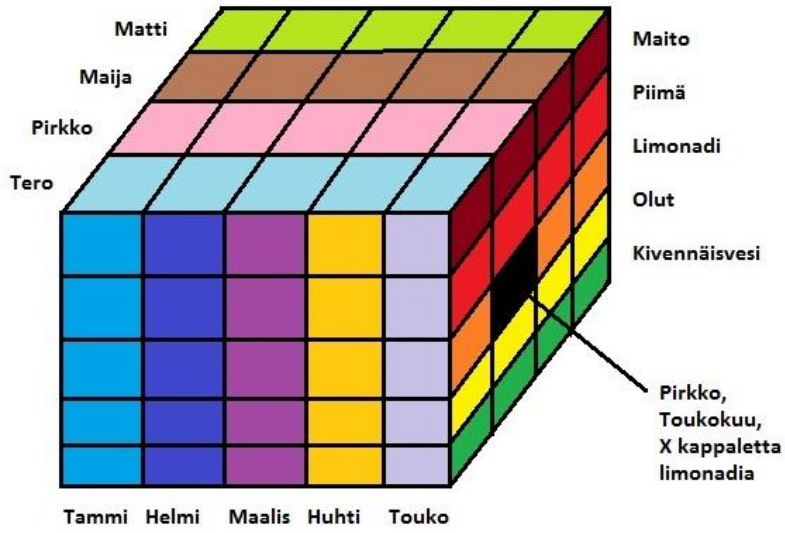


KUVIO 11. Lumihiutalemalli (Snowflake)

2.3 OLAP

OLAP eli OnLine Analytical Processing tarkoittaa tietojen moniulotteista analysointia. Yrityksen tietoa voidaan usein kuvata moniulotteisesti. Esimerkiksi myyntiä voidaan tarkastella eri näkökulmista, kuten tuotteittain, myyntipaikoittain, myyjittäin tai ajanjaksoina. Vaikka OLAP:sta puhutaankin yleensä kuutiona, on siinä todellisuudessa enemmän kuin kolme ulottuvuutta. Kuutiossa sen sijaan niitä on vain kolme. Yksi ulottuvuuksista on yleensä aina aika. (Hovi 1997, 57-59.)

Kuviossa 12. on esimerkkinä kolmiuloitteinen kuutio. Sen dimensioina ovat asiakas, aika ja tuote. Faktatietona on se, montako kappaletta tuotetta on myyty. Esimerkissä mustalla väritetty kuution osa esittää Pirkko-nimisen asiakkaan toukokuussa ostamaa X määrää limonadia.



Kuvio 12. Esimerkki kolmiuloitteisesta kuutiosta

OLAP-pohjainen liiketoiminnan analysointi ja tietovarastointi tekivät toisistaan sen, mitä ne ovat nykyään. Se on yleinen tapa analysoida tietoa. Lähes jokainen organisaatio joko käyttää tai harkitsee käyttävänsä sitä. (Hammergren & Simon 2009, 136.)

Liiketoiminnan analysointi on OLAP-alustojen tarjoaman tiedon visualisointia. Tarkoituksena on vastata kysymykseen, mitä tapahtui ja miksi. Toisin kuin kyselyt ja raportointi, OLAP mahdollistaa tietoon porautumisen, joka voi vastata miksi-kysymykseen. (Hammergren & Simon 2009, 136.)

OLAP siis mahdollistaa moniuloitteisen tiedon analysoinnin. Käyttäjät yleensä haluavat tarkastella asioita eri näkökulmista. Aikadimensiota käytettäessä voidaan esimerkiksi tehdä trendianalysointia, kun taas esimerkiksi sijaintidimensiota käyttämällä voidaan seurata, millä alueella myynti on suurinta. (Hammergren & Simon 2009, 136.)

2.4 Porautuminen

Porautuminen on OLAP:in ominaisuus, jota käyttäjät todennäköisesti eniten käyttävät. Porautumisen ideana on se, että käyttäjä voi halutessaan nähdä haluamansa tiedon tarkemmalla tasolla, porautumalla tietoon. (Hammergren & Simon 2009, 140.) Kuviossa 13. on esiteltynä kuvakaappaus Microsoftin Business Intelligence Development Studion OLAP-kuution selaustyökalusta, jolla on porauduttu internet-myyntiin sijainnin perusteella.

Drop Filter Fields Here

			Drop Column Fields Here	
Country	**State-Province**	**City** **Postal Code**	Internet Order Quantity	Internet Sales Amount
⊟ Australia	⊟ New South Wales	⊞ Coffs Harbour	387	235,454.97 $
		⊞ Darlinghurst	286	155,010.38 $
		⊞ Goulburn	370	310,875.90 $
		⊞ Lane Cove	296	220,083.58 $
		⊞ Lavender Bay	297	195,122.90 $
		⊞ Malabar	287	176,905.61 $
		⊞ Matraville	305	216,564.44 $
		⊞ Milsons Point	279	187,075.60 $
		⊞ Newcastle	344	245,936.52 $
		⊞ North Ryde	300	208,082.62 $
		⊞ North Sydney	295	159,227.11 $
		⊞ Port Macquarie	405	278,084.50 $
		⊞ Rhodes	295	200,418.56 $
		⊞ Silverwater	285	175,222.51 $
		⊞ Springwood	291	165,100.66 $
		⊞ St. Leonards	307	185,423.08 $
		⊞ Sydney	399	280,983.31 $
		⊞ Wollongong	411	338,913.47 $
		Total	5,839	3,934,485.73 $
	⊞ Queensland		2,903	1,988,415.03 $
	⊞ South Australia		867	618,255.86 $
	⊞ Tasmania		393	239,937.90 $
	⊞ Victoria		3,343	2,279,906.06 $
	Total		13,345	9,061,000.58 $
⊞ Canada			7,620	1,977,844.86 $
⊞ France			5,558	2,644,017.71 $

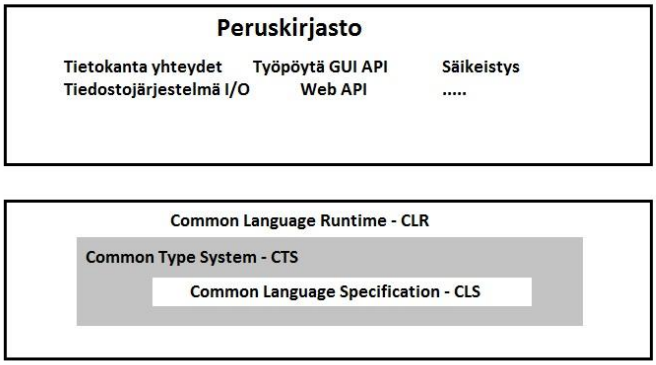
KUVIO 13. Adventureworks OLAP-kuution porautuminen sijainnin mukaan

2.5 .NET -ohjelmistokehys ja C#

.NET -ohjelmistokehys on Microsoftin tarjoama ohjelmistokehys Windows-käyttöjärjestelmän ohjelmien toteuttamiseen. Sitä voidaan kuitenkin käyttää myös muissa käyttöjärjestelmissä kuten joissain Linux-jakeluissa ja Mac OS X:ssä. (Troelsen 2010, 6.)

.NET -kehyksen tärkeimpiä ominaisuuksia ovat sen tuki monelle eri ohjelmointikielelle, kuten C#, VB, F# ja S#. Se myös tukee eri ohjelmointikielten välistä niin sanotusti ristikkäistä perintää, poikkeusten käsittelyä ja ohjelmakoodin debuggausta. Ristikkisellä perinnällä tarkoitetaan sitä, että esimerkiksi C# ohjelmakielellä tehdyn luokan voi periyttää jollain muulla .NET:iä tukevalla ohjelmointikielellä toteutetusta luokasta. (Troelsen 2010, 7.)

.NET -alusta koostuu kolmesta avainasemassa olevasta osasta: CLR, CTS ja CLS. Ajonaikaista kerrosta CLR:ää kutsutaan *Common Language Runtimeksi*, ja sen tehtävänä on etsiä, ladata ja hallita .NET:in tarjoamia tyyppejä ohjelmoijan puolesta. CLR myös huolehtii monista alemman tason yksityiskohdista, kuten muistin ja säikeiden hallinnasta. (Troelsen 2010, 7.) Kuviossa 14. on esiteltynä .NET:in perusosat.



KUVIO 14. .NET perusluokan, CLR, CLS ja CTS suhteet

CTS eli *Common Type System* on toinen .NET:in osista. Se kuvaa täysin kaikki mahdolliset ajonaikaiset datatyypit ja ohjelmoinnin rakenteet, kuten esimerkiksi luokat, rajapinnat, enumeraatiot ja tietueet. CTS määrittelee, miten nämä entiteetit voivat vuorovaikuttaa toisiinsa ja kuvaa myös, kuinka ne on esitetty .NET metadataformaatissa. Kaikki .NET -kielet eivät kuitenkaan välttämättä tue kaikkia CTS:n tarjoamia ominaisuuksia. (Troelsen 2010, 7.)

Viimeisenä .NET:in osana on CLS eli *Common Language Specification*. CLS on määrittely, joka määrittelee yleiset tyypit ja ohjelmoinnin rakenteet, joita kaikki .NET -kielet voivat käyttää. Kirjastoa kirjoittaessa voi olla varma, että se toimii kaikilla .NET –kielillä, jos käyttää vain CLS:n datatyyppejä ja rakenteita. Tätä varten voidaankin asettaa esim C# kääntäjä tarkistamaan CLS-yhteensopivus käännöksen aikana. CLS siis pitää huolen siitä, että .NET -kääntäjä pystyy tuottamaan sellaista ohjelmakoodia, jota CLR voi isännöidä. (Troelsen 2010, 7-8.)

Perusluokkakirjastot ovat CLR/CLS/CTS:n lisäksi yksi .NET -kehyksen tarjoamista ominaisuuksista. Tämä kirjasto tarjoaa helpon tavan käyttää esimerkiksi säikeitä, tiedostojärjestelmää ja joitakin laitteita. Lisäksi se myös tarjoaa tietokantayhteydet, XML-dokumenttien manipuloinnin ja käyttöliittymien toteutuksen. (Troelsen 2010, 8.)

C# on Microsoftin kehittämä ohjelmointikieli, joka toteutettiin .NET ohjelmistokehystä varten. Se muistuttaa syntaksiltaan Java-ohjelmointikieltä, mutta sitä ei kuitenkaan voida kutsua Javan kopioksi. Sekä Java että C# pohjautuvat C-perheen ohjelmointikieliin (esimerkiksi: C, C++), ja tästä syystä ne jakavatkin samankaltaisen syntaksin. (Troelsen 2010, 8.)

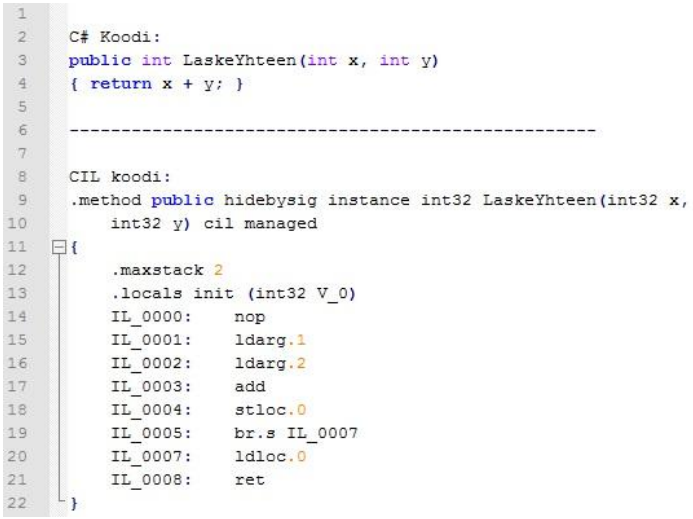
C# tukee operaattoreiden ylikuormitusta, enumeraatioita, tietueita ja takaisinkutsuja delegaattien avulla (Troelsen 2010, 8). Delegaatit siis mahdollistavat takaisinkutsujen toteuttamisen. Delegaatti on tyyppi turvallinen olio, joka osoittaa toiseen metodiin tai listaan metodeita. Se pitää sisällään tärkeää informaatiota kuten osoitteen metodiin, jotka kutsutaan, parametrit ja paluuarvon tyypin. (Troelsen 2011, 398.) C# tukee myös lambda-ilmaisuja ja anonyymejä tietotyyppejä. C# on monen eri ohjelmointikielen hybridi, minkä tuloksena se on syntaksiltaan hy-

vin puhdas, kenties jopa puhtaampi kuin Java. Se on myös yhtä yksinkertainen kuin VB (Visual Basic) ja sillä saavutetaan yhtä paljon joustavuutta ja tehokkuutta kuin C++:lla. (Troelsen 2010, 8.)

Ohjelmoitaessa C#:lla ei tarvita osoittimia, mutta niitä voi halutessaan käyttää. Myös muistin hallinta on automaattista roskienkeruun ansiosta (Garbage Collection), eli delete ei ole käytettävissä C#:ssa. Lisäksi C#:ssa on paljon muitakin ominaisuuksia, kuten esimerkiksi tuki anonyymeille metodeille ja valinnaisille metodin parametreille. (Troelsen 2010, 8-9.)

Yksi tärkeimmistä asioista ymmärtää C# -ohjelmointikielestä on se, että se voi tuottaa ainoastaan ohjelmia, jotka toimivat .NET -kehyksessä. C#:lla ei siis voi koskaan tehdä unmanaged C/C++ -ohjelmaa. Virallisesti käytetään termiä managed, kun ohjelmakoodi suoritetaan .NET -ajoympäristössä. Binääristä tiedostoa, joka sisältää tämän managed-ohjelmakoodin, kutsutaan assemblyksi. Toisin sanoen ohjelmakoodia, jota ei voida suorittaa .NET –ajoympäristössä, kutsutaan unmanaged-koodiksi. (Troelsen 2010, 10.)

Nämä assemblyt, esimerkiksi C# -koodi joka on käännetty .NET –kääntäjällä, sisältää CIL (Common Intermediate Language) -koodia. CIL-koodi on ikään kuin Javan tavukoodi, joka käännetään alustakohtaisiksi käskyiksi vasta sitten, kun on aivan pakko. Kuviossa 15. on esimerkki C# -funktiosta nimeltään YhteenLasku. Kuviossa esitetään miltä se näyttää CIL-koodina. CIL-käskyjen lisäksi assemblyt pitävät sisällään metatietoa, joka kuvaa jokaisen assemblyn sisällään pitämän tietotyypin. Metatieto kertoo perusluokan, sekä sen mitä rajapintoja luokka toteuttaa ja kuvauksen luokan kaikista jäsenistä. Assemblyt itse on myös kuvattu käyttäen metatietoa, jota tässä tapauksessa kutsutaan manifestiksi. Manifesti pitää sisällään tiedon assemblyn versiosta, kulttuuritiedot lokalisointia varten sekä listan kaikista ulkopuolisista assembly-referensseistä, jotka tarvitaan assemblyn suorittamista varten. (Troelsen 2010, 13.)



```
C# Koodi:
public int LaskeYhteen(int x, int y)
{ return x + y; }

-------------------------------------------------

CIL koodi:
.method public hidebysig instance int32 LaskeYhteen(int32 x,
    int32 y) cil managed
{
    .maxstack 2
    .locals init (int32 V_0)
    IL_0000:    nop
    IL_0001:    ldarg.1
    IL_0002:    ldarg.2
    IL_0003:    add
    IL_0004:    stloc.0
    IL_0005:    br.s IL_0007
    IL_0007:    ldloc.0
    IL_0008:    ret
}
```

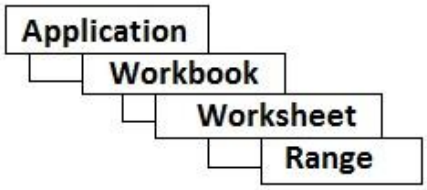
KUVIO 15. Esimerkki C# ja CIL koodista

CIL-koodi käännetään ajon aikana automaattisesti laitteistokohtaisiksi käskyiksi. Tämän käännöksen tekee JIT-kääntäjä. Se on alustakohtainen ja optimoi CIL-koodin kyseiselle alustalle. JIT-kääntäjä myös säilöö jo käännetyn koodin. Toisin sanoen ensimmäisellä kerralla, kun kutstutaan metodia LaskeYhteen, JIT-kääntäjä kääntää sen. Toisella kutsukerralla funktio on jo käännettynä muistissa. On myös mahdollista kääntää CIL-koodi valmiiksi alustakohtaisiksi käskyiksi. Tämä saattaa nopeuttaa ohjelman käynnistymistä, jos se sisältää esimerkiksi paljon grafiikkaa. (Troelsen 2010, 17.)

2.6 Visual Studio Tools for Office

Visual Studio Tools for Office eli VSTO tarjoaa .NET -ohjelmointituen Office-ohjelmille Visual Studiolla. VSTO kääntää Word- tai Excel-dokumentin .NET –luokaksi, johon voidaan asettaa kontrolleja ja käyttää muita .NET:in ominaisuuksia. Eniten ominaisuuksia ja tukea VSTO tarjoaa Wordille, Excelille ja Outlookille. VSTO on integroitu Visual Studio Professionaliin ja sitä korkeampiin versioihin. VSTO ohjelmien kanssa voidaan myös käyttää ClickOnce julkaisua, joka tekee ohjelmien jakelusta entistäkin helpompaa. (Carter & Lippert 2009, 5.)

Office-ohjelmien kirjoittamisessa käytetään lähes aina ohjelmakoodia, joka käyttää Office-ohjelman oliomallia. Oliomallit ovat olioita, jotka Office-ohjelma tarjoaa. Näillä olioilla voidaan kontrolloida Office-ohjelmaa. Jokaisen Office-ohjelman oliomallia voidaan kuvata hierarkisesti. Juurioliona on Application, eli sovellus. (Carter & Lippert 2009, 7.) Kuviossa 16. on esiteltynä Excel-ohjelman oliomallien hierarkia.



Kuvio 16. Excelin oliomallin hierarkia

Sovellusolio ja työkirjaolio ovat yhteydessä, koska työkirjoihin päästään käsiksi sovellusolion avulla. Muihin olioihin ei päästä suoraan käsiksi sovellusoliosta, vaan täytyy kulkea polkua alemmas sovellusoliosta kohti alueoliota. (Carter & Lippert 2009, 8.)

Office-ohjelmien oliomalli koostuu useista olioista. Esimerkiksi Excelissä niitä on noin 300. Oliot kuvaavat pääsääntöisesti Office-ohjelman ominaisuuksia. Esimerkiksi Excel-ohjelmassa on seuraavat oliot: työkirja, sarja ja kaavio. Eniten käytetty on yleensä Application eli sovellusta kuvaava olio. Nämä eroavat tavallisista .NET -olioista kuitenkin siinä, ettei suurinta osaa niistä voida luoda dynaamisesti käyttäen new-operaatiota. (Carter & Lippert 2009, 8.)

Office-ohjelmat tarjoavat ohjelmoijalle tapahtumia eli eventtejä. Ne ovat ohjelman tapa kommunikoida takaisin ohjelmoijan koodiin, jolloin ohjelmoija voi suorittaa haluamansa koodin vastauksena tällaiseen tapahtumaan. Office-oliomallissa on vähemmän eventtejä kuin metodeita tai asetuksia. Esimerkiksi Excelissä tälläisiä eventtejä on noin 85 kappaletta. Excelissä ainoat oliot, jotka laukaisevat eventtejä

ovat Application, Workbook, Worksheet, Kaavio, OLEObject ja QueryTable. (Carter & Lippert 2009, 28-29.)

Ohjelmoijan täytyy luoda eventille käsittelijämetodi koodissaan, joka voidaan rekisteröidä eventtiin. Tätä metodia kutsutaan automaattisesti aina kun siihen rekisteröitynyt tapahtuma tapahtuu. Kuviossa 17. on esimerkki Excelin Open-tapahtuman delegaatista. Ohjelmoijan täytyy siis luoda metodi, joka vastaa tätä delegaattia. (Carter & Lippert 2009, 31.) Kuviossa 18. on tapahtuman käsittelijämetodi, joka vastaa kuvion 17. delegaattia.

```
public delegate void AppEvents_WorkbookOpenEventHandler(Workbook wb);
```

KUVIO 17. Excelin Open-tapahtuman delegaatti

```
public void OpenKasittelija(Excel.Workbook wb)
{
    // Koodi joka suoritetaan tapahtuman tapahtuessa
}
```

KUVIO 18. Excelin Open-tapahtuman käsittelijä

Tapahtuman käsittelijän kirjoittamisen lisäksi pitää vielä rekisteröidä tapahtumankäsittelijä olioon, joka laukaisee tapahtuman. Tämä tapahtuu luomalla uusi delegaattiolio, minkä Excelin oliomalli määrittelee. Tälle delegatille annetaan muodostinfunktiioon parametrina luotu käsittelijämetodi. Tämä luotu delegaattiolio rekisteröidään tapahtuman kuuntelijaksi käyttäen += -operaattoria. (Carter & Lippert 2009, 31.) Tämä toimenpide voidaan suorittaa yhdellä ohjelmakoodilauseella, joka on esiteltynä Kuviossa 19.

```
application.WorkbookOpen +=
    new AppEvents_WorkbookOpenEventHandler(OpenKasittelija);
```

KUVIO 19. Tapahtuman käsittelijän rekisteröinti

VSTO tarjoaa kolme erilaista tasoa tehdä Office-ohjelmointia: Ohjelmataso, Liitännäistaso ja dokumenttitaso. Ohjelmatasolla toimiva ohjelma on erillinen sovellus Officesta, joka ohjaa ja automatisoi Office-ohjelmaa. Tällaisiä sovelluksia voidaan tehdä esimerkiksi Visual Studiolla. Tällaisissä sovelluksissa käyttäjä käynnistää sovelluksen, joka sitten käynnistää Office-ohjelman ja yleensä automatisoi jotain tehtävää. Ohjelmatason sovelluksissa ainutlaatuista on se, että ohjelmakoodia ei ajeta Office-ohjelman prosessissa toisin kuin muun tason VSTO -sovelluksissa. Liitännäistason sovellukset ovat DLL-tiedostomuotoisia sovelluksia, jotka Office-ohjelma lataa ja luo tarvittaessa. Liitännäiset suoritetaan Officen prosessissa. Käynnistääkseen tällaisen sovelluksen käyttäjä avaa Office-ohjelman, joka tunnistaa rekisteröidyt liitännäiset käynnistyessään ja lataa ne. Dokumenttitason ohjelma taas poistuu käytöstä heti kun dokumentti, johon se on sidottu, suljetaan. Liitännäinen taas voi olla ladattuna koko Office-ohjelman suorituksen ajan. Dokumenttitason ohjelmat tulivat tunnetuksi VBA:n (Visual Basic for Applications) kautta. Dokumenttitason ohjelman suorittamiseksi käyttäjän täytyy siis avata dokumentti, johon tämä ohjelma on sidottuna. (Carter & Lippert 2009, 51-52.)

2.7 ClickOnce

ClickOnce on Microsoftin .NET -ohjelmien julkaisuun tarkoitettu tekniikka. Se mahdollistaa automaattisilla päivityksillä varustettujen Windows-ohjelmien julkaisun. ClickOnce tarjoaa helpon ja nopean tavan julkaista ohjelmia. Automaattiset päivitykset päivittävät sovelluksesta vain ne komponentit, jotka ovat muuttuneet versioiden välillä. Lyhyesti sanottuna ClickOnce -sovellus on mikä tahansa Windows Presentation Foundations, Windows Forms tai konsolisovellus, joka on julkaistu käyttäen ClickOnce -tekniikkaa. (Microsoft 2012.)

ClickOnce -sovellus voidaan julkaista kolmella erilaisella tavalla: internetissä, verkkojaolla tai joltakin medialta kuten CD-ROM. ClickOnce -sovellus voidaan asentaa käyttäjän tietokoneelle, jolloin se on käytettävissä myös silloin kun internet-yhteyttä ei ole saatavilla. Toinen vaihtoehto on suorittaa se internet-tilassa. Internet-tilassa suoritettuna sovellusta ei asenneta pysyvästi käyttäjän tietokoneelle. ClickOnce:lla julkaistut sovellukset voivat siis automaattisesti myös tarkistaa käynnistyessään, onko uudempaa versiota saatavilla ja tarvittaessa ladata uudemmat tiedostot. (Microsoft 2012.)

ClickOnce -sovellukset ovat niin sanotusti eristettyjä sovelluksia, mikä aiheuttaa sen, ettei niiden asentaminen voi estää muiden sovellusten toimintaa. Käyttäjät voivat myös asentaa ClickOnce -ohjelmat ilman järjestelmänvalvojan tunnuksia. Tämä helpottaa huomattavasti ohjelman jakelua ja asentamista isoissa yrityksissä ja konserneissa. (Microsoft 2012.)

ClickOncen julkaisun arkkitehtuuri koostuu pääsääntöisesti kahdesta XML-manifestitiedostosta: ohjelmamanifesti ja julkaisumanifesti.

Ohjelmamanifesti kuvaa ohjelman, joka sisältää assemblyt, riippuvuudet, tiedostot, joista ohjelma koostuu, vaadittavat oikeudet ja tiedon siitä, mistä päivitykset ovat saatavilla. Ohjelmoija luo manifestin käyttäen joko Visual Studion Publish Wizardia tai generoi manifestin generointityökalulla Mage.exe, joka sisältyy Windows Software Development Kit:iin (SDK). (Microsoft 2012.)

Julkaisumanifesti kuvaa sitä, kuinka sovellus on julkaistu, ja kuviossa 20. on esitettynä osa ennustesovelluksen julkaisumanifestin sisällöstä. Se sisältää tiedon ohjelmamanifestin sijainnista (rivillä viisi) ja versiotiedon siitä, mikä versio käyttäjien tulisi suorittaa (rivillä 15). (Microsoft 2012.)

```
<assemblyIdentity name="Forecasting.application" version="1.0.10.117" publicKeyToken="45e455cc1b6b2a93"
    language="en" processorArchitecture="x86" xmlns="urn:schemas-microsoft-com:asm.v1" />
<description asmv2:publisher="Logica" asmv2:product="Forecasting" xmlns="urn:schemas-microsoft-com:asm.v1"
<deployment install="true" mapFileExtensions="true" co.v1:createDesktopShortcut="true">
<deploymentProvider codebase="http://192.168.XXX.XXX:8080/forecasting/Forecasting.application" />
</deployment>
<compatibleFrameworks xmlns="urn:schemas-microsoft-com:clickonce.v2">
<framework targetVersion="4.0" profile="Client" supportedRuntime="4.0.30319" />
<framework targetVersion="4.0" profile="Full" supportedRuntime="4.0.30319" />
</compatibleFrameworks>
<dependency>
<dependentAssembly dependencyType="install"
    codebase="Application Files\Forecasting_1_0_10_117\Forecasting.exe.manifest"
    size="19953">
  <assemblyIdentity name="Forecasting.exe" version="1.0.10.117"
    publicKeyToken="45e455cc1b6b2a93" language="en" processorArchitecture="x86" type="win32" />
  <hash>
    <dsig:Transforms>
      <dsig:Transform Algorithm="urn:schemas-microsoft-com:HashTransforms.Identity" />
    </dsig:Transforms>
    <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
    <dsig:DigestValue>EQON9DfZTABSG2QgXj82HLdIoPc=</dsig:DigestValue>
  </hash>
</dependentAssembly>
</dependency>
```

KUVIO 20. Ennustesovelluksen julkaisumanifesti

3 MYYNNIN ENNUSTAMISSOVELLUS

Myynnin ennustamissovellus tehdään Logican asiakkaalle korvaaman vanha web-pohjainen ennustesovellus. Keväällä 2011 toteutettiin yksinkertainen POC–versio (Proof Of Concept). Sen tarkoitus oli todistaa, että saataisiin huomattava nopeus- ja käytettävyysetu, jos uusi sovellus toteutettaisiin Microsoft Office Excelin päälle. Aiempi ennustesovellus oli erittäin hidas ja epävarma. Se myös vaati toimiakseen aina internet-yhteyden. Asiakas hyväksyi Proof Of Concept –version, ja uuden ennustesovelluksen kehitys aloitettiin kesällä 2011.

Excel valittiin käyttöliittymäksi, koska se on monille jo entuudestaan tuttu. Se taas vähentää koulutuksen tarvetta. Lisäksi tarkoituksena on se, että käyttäjät voivat ennustaa ja tarkastella ennusteita myös ilman internet-yhteyttä. Tämä tietenkin vaatii sen, että käyttäjät ovat ladanneet ennusteet omalle koneellensa.

Ennustesovelluksen toteutuksessa huomiota kiinnitettiin erityisesti seuraaviin asioihin: toiminnan nopeus, käytettävyys ja yksinkertaisuus. Sovelluksesta haluttiin käyttäjälle yksinkertainen ja tehokas tapa ennustaa myyntiä.

Sovelluksella käyttäjät ennustavat tuotteiden myyntiä. Näiden ennusteiden perusteella asiakas valmistaa tuotteet. Ennusteet on jaettu markkina-alueittain ja kuukausittain.

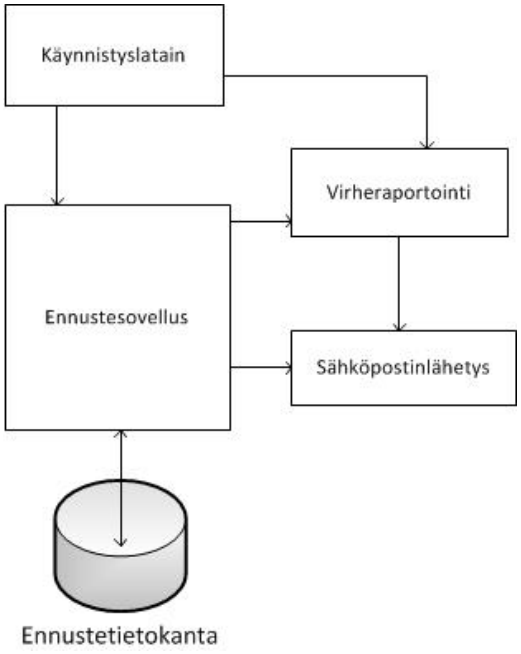
Ennusteiden ylläpitolomakkeet jätettiin ainoana toiminnallisuutena vielä web-toteutukseen. Siellä käyttäjät hallinoivat ennusteita ja jakavat oikeuksia sille, kuka ennustaa mitäkin markkina-aluetta.

3.1 Ennustesovelluksen toiminta

Ennustesovellus koostuu useasta eri komponentista. Sovellus voidaan jakaa kahteen pääkomponenttiin: web-sovellukseen ja itse ennustesovellukseen.

Web-sovellus eli vanha ennustesovellus sisältää toimintoja, jotka on jätetty toistaiseksi käytettäväksi. Itse ennustamistoiminto on poistettu käytöstä web-sovelluksesta. Web-sovelluksen ainoa tarkoitus onkin hallita ja luoda ennusteita, sekä hallita käyttäjiä ja heidän oikeuksiaan. Web-sovelluksella voidaan myös hallita ennusteiden tiloja.

Uusi ennustesovellus (tähän viitataan jatkossa tässä työssä nimellä ennustesovellus) voidaan myös pilkkoa useampaan selkeään osaan: käynnistyslatain, virheraportointi, sähköpostinlähetys, itse ennustesovellus ja ennustetietokanta. Kuviossa 21. on esitelty ennustesovelluksen eri osat ja niiden väliset suhteet. Ennustessovellus koostuu useasta eri ominaisuudesta sisältäen raportoinnin ja ennusteiden tallentamisen tietokoneelle.



Kuvio 21. Ennustesovelluksen tärkeimmät komponentit

Ennustesovelluksella asiakas ennustaa myyntiä tuotetasolla. Tuotteita voi olla ennusteesta riippuen jopa kymmeniä tuhansia. Sovelluksen pitää tästä huolimatta toimia nopeasti ja olla samalla yksinkertainen käyttää.

Ennustetiedot ovat kuukausikohtaisia, mutta sovelluksella voidaan ennustaa kerralla useampaa kuin yhtä kuukautta. Jokaisella tuotteella on mittareita, kuten kappalemäärä ja kappalehinta. Nämä mittarit ovat saatavilla vain tuotetasolla, ja ne myös tallennetaan ennustekantaan vain tuotetasolle.

Tuoteryhmätasolle ennustesovellus summaa tuotetasojen mittarien arvot. Ennustearvoja voidaan syöttää tuotetasolle tai tuoteryhmätasolle. Tuotetasolle syötettäessä lasketaan tuoteryhmän mittarien arvot uudestaan yksinkertaisesti summaamalla kaikkien tuotteiden arvot.

Tuoteryhmätasolle ennustetta syötettäessä ohjelma toimii huomattavasti monimutkaisemmin. Ensimmäisenä tarkistetaan, onko alatasoilla jo olemassa arvoja. Jos alatasoilla on arvoja, niin jaetaan syötetty arvo näille alatasoille noudattaen jo olemassa olevaa jakaumaa.

Jos alatasoilla ei ole arvoja, käytetään arvon jakamiseen joko ennusteen oletusreferenssikautta, joka on edellisen vuoden saman kuukauden toteutuma. Käyttäjä voi kuitenkin vaihtaa referenssikauden johonkin muuhun halutessaan. Tällöin käytetään valitun kauden ennustetta referenssinä. Jos alatasoilla ei ole arvoja ja oletusreferenssi on valittuna, mutta sillä ei ole toteumaa, arvot jaetaan tasan jokaiselle alatasolle.

3.2 Tietokanta

Ennustesovellus käyttää Oraclen tietokantaa, jonka kehitysversio on palvelimella, jossa on Windows-käyttöjärjestelmä. Tuotanto- ja testitietokanta sijaitsevat asiakkaan palvelimilla. Ennustesovelluksella on oma tietokanta ennustetiedoille. Tähän tietokantaan ladataan tietoja asiakkaan tietovarastosta. Tiedot ladataan tietovaras-

tosta ennustesovelluksen tietokantaan käyttäen PL/SQL koodia. Tiedon siirtämisessä käytetään DELETE, INSERT ja SELECT SQL-lauseita ja MERGE PL-lauseita.

Ennustetietokannasta takaisin tietovarastoon siirrettäessä siirto tapahtuu erillisen SQL-tiedoston avulla. Tietojen lataus käyttää ennustekannassa olevia näkymiä.

Ennustetietokanta sisältää dimensiotauluja, kaksi faktataulua ja asetustauluja. Dimensiotaulut sisältävät esimerkiksi tuotteet ja markkina-alueet. Faktataulut sisältävät referenssitiedot ja itse ennusteiden rivitiedot. Asetustaulut sisältävät käyttäjätietoja, ennusteiden tilatietoja ja rahayksikkötietoja.

3.3 Tiedon esittäminen ja lataaminen

Ennustesovellus siis lataa tietoja ennustetietokannasta ja esittää tiedot käyttäjälle Microsoft Office Excelissä toimivan liitännäisen avulla. Sovellus lataa tietokannasta käyttäjätiedot, tiedot ennusteista, ennustetiedot, dimensiotietoja ja referenssitietoja.

3.3.1 Tiedon lataaminen

Ennustesovellus yrittää ottaa yhteyttä ennustetietokantaan käynnistyessään. Jos yhteyttä ei saada, kysytään käyttäjältä, haluaako hän työskennellä offline- eli yhteydettömässä tilassa. Työskennellessä online-tilassa sovellus tarkastaa käynnistyessään käyttäjän oikeudet asetustauluista. Tietokannasta haetaan käyttäjälle saatavilla olevien ennusteiden otsikkotiedot oikeuksien mukaan.

Käyttäjän valittua haluamansa ennuste ja aikaväli, haetaan ennusteen tiedot tietokannasta. ForecastValues-luokka hoitaa fakta- ja referenssitietojen latauksen. ProductHierarchy-luokka lataa tuotedimensiotaulun koko sisällön.

Ennustetiedot ladataan ennustekannasta väliaikaiseen .NET:in säiliöön. Tuotehierarkia ladataan myös tällaiseen säiliöön. Nämä säiliöt ovat hajautustauluja, ja niiden tyyppi on Dictionary. Kuviossa 22. on nähtävillä molempien kontainereiden luontilauseet.

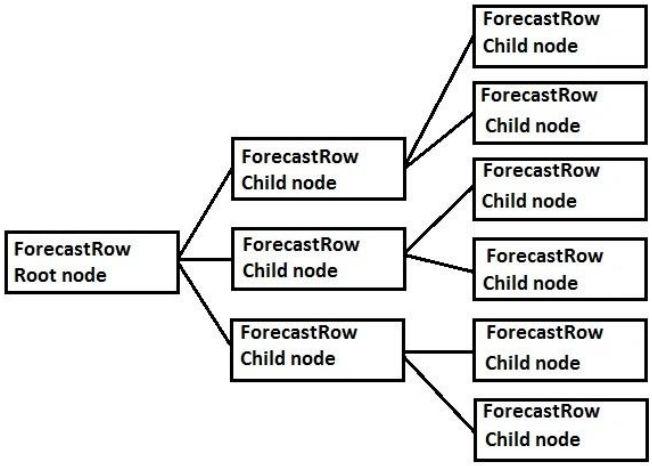
```
// Ennustetietojen väliaikainen sijoitus kontaineri
forecastRows = new Dictionary<string, Row>();

// Tuotehierarkian väliaikainen sijoitus kontaineri
productRelations = new Dictionary<string, LinkedList<string>>();
```

KUVIO 22. Dictionary-tyyppisten kontainereiden luonti

Ennustetiedot ladataan forecastRows-hajautustauluun, jonka avaimena on tuotteen id, ja sen tyyppi on string. Tämän hajautustaulun arvona ovat ennustekausien tiedot, jotka on sijoitettu Row-tyyppiseen olioon. Row-luokka on ennustesovellusta varten toteutettu, ja se kuvastaa yhtä ennustettavaa riviä. Tämä luokka pitää sisällään ennusterivin tuotteen id:n, arvot, tuoteryhmän ja tuotteen referenssitiedot.

3.3.2 Tietorakenteet

Ennustesovellus käyttää pääsääntöisesti .NET:in geneerisiä säiliöitä. Tärkein rakenne on kuitenkin puurakenne, joka pitää sisällään ennustettavat rivit. Tämä ForecastRow-niminen puurakenne rakennetaan tuotteiden ja ennustetietojen hajautustauluista. Yksi ForecastRow-olio esittää yhtä solmua puurakenteessa. Solmulla voi olla monta lapsisolmua, mutta vain yksi emosolmu. Solmun tiedot tallennetaan List-tyyppiseen säiliöön. Tideot ovat samassa järjestyksessä, kuin missä ne näkyvät käyttäjälle. Tiedoissa ovat kaikki arvot kaikille periodeille, jotka ovat samalla rivillä. Jokaisella ennusterivillä, eli ForecastRow-solmulla, on myös yhtesiä attribuutteja, kuten ennusteen id, ennustuskuukaudet ja mittarit. Alla olevassa kuviossa 23. on nähtävillä ennustesovelluksen rakentama puurakenne.



KUVIO 23. Ennusterivien puurakenne

3.3.3 Tiedon tallentaminen

Ennustetiedot on sovelluksen avulla mahdollista tallentaa joko ennustetietokantaan tai tiedostoon. Tiedostoon tallennettaessa ennustetietoja voidaan käyttää myös silloin, kun internet-yhteyttä ei ole saatavilla. Tiedosto voidaan myös lähettää toiselle henkilölle käyttäen sähköpostin lähetystoimintoa, ja sen voi avata ennustesovelluksessa, vaikkei olisikaan muuten oikeuksia ladata ennusteita tietokannasta.

Tämä mahdollistaa sen, että esimerkiksi tietyn alueen johtaja voi ladata ja lähettää ennusteen ennustettavaksi eri osaston vastaaville, kuitenkaan antamatta heille oikeuksia ladata tai tallentaa ennusteita tietokantaan.

Tiedostoon tallennettaessa käytetään hyödyksi serialisointia ja tiedoston pakkaamista. .NET tukee näitä ominaisuuksia suoraan. Kuviossa 24. on esillä ennustesovelluksen ennusteiden tallennusfunktio. Ensiksi kopioidaan ennustetiedot ForecastFileData olioon, se on esitettynä kuviossa 25., joka on serialisoitava luokka. ForecastFileData olio serialisoidaan tiedostoon, ja tämän jälkeen tiedosto pakataan

käyttäen .NET:in GZipStrean luokkaa. Tiedostoa avatessa toiminnallisuus on lähes sama, mutta käänteisessä järjestyksessä.

```
public static void saveToFile(string fileName, string version, string user, ForecastRow root,
    ForecastValues values)
{
    ProductHierarchyData productData = new ProductHierarchyData();
    productData.productNames = ProductHierarchy.productNames;
    productData.productRelations = ProductHierarchy.productRelations;
    productData.rootNodes = ProductHierarchy.rootNodes;
    ForecastFileData data = new ForecastFileData();
    data.productHierarchyData = productData;
    data.forecastValues = values;
    data.rootRow = root;
    data.user = user;
    data.version = version;

    FileStream filestream = File.Open(fileName, FileMode.Create);
    GZipStream zipstream = new GZipStream(filestream, CompressionMode.Compress);
    BinaryFormatter formatter = new BinaryFormatter();
    formatter.Serialize(zipstream, data);
    zipstream.Close();
    filestream.Close();
}
```

KUVIO 24. Ennusteiden tallennusfunktio

```
[Serializable()]
class ForecastFileData
{
    public string version;
    public string user;
    public ForecastRow rootRow;
    public ForecastValues forecastValues;
    public ProductHierarchyData productHierarchyData;
}
```

KUVIO 25. Tallennuksessa käytetty apuluokka

3.3.4 Tiedon esittäminen

Ennustetiedot esitetään puumaisena rakenteena. Yksi rivi näytöllä kuvastaa joko tuotetta tai tuoteryhmää, jonka alle voi kuulua tuotteita tai tuoteryhmiä. Alimmalla tasolla on aina pelkkiä tuotteita.

3.4 Ominaisuudet ja niiden toteutus

Ennustesovellus sisältää useita ominaisuuksia, jotka helpottavat käyttäjiä. Tärkeimpiä ominaisuuksia itse ennustamisen lisäksi ovat automaattiset päivitykset, sähköpostin lähetys, virheraportointi, käynnistyslatain ja raportointiominaisuudet.

Käyttäjille tärkein ominaisuus on ennustaminen ja raportointi. Muut ominaisuudet helpottavat esimerkiksi ohjelman jakelua ja virheiden paikantamista.

3.4.1 Automaattiset päivitykset

Automaattiset päivitykset helpottavat ja nopeuttavat uuden version jakelua asiakkaalle. Ne myös nopeuttavat erilaisten virheiden korjausta. Ennustesovellus päivittää itsensä automaattisesti aina, kun internet-yhteys ja uudempi versio ovat saatavilla. ClickOnce tekniikka tarjoaa päivitysten tarkastamiseen ja lataamiseen täysin automaattisen toiminnon. Tämä toiminto on kuitenkin ennustesovelluksessa korvattu käynnistyslataimessa sijaitsevalla koodilla, koska normaalisti päivitysten lataaminen ei ole pakollista.

Päivitys on tehty pakolliseksi siksi, ettei vanhaa versiota voida käyttää, jos uudempi versio on saatavilla. Ohjelma myös sisältää tiedon tiedostoversiosta. Tätä versiota kasvatetaan silloin, jos tehdään oleellisia muutoksia tiedostorakenteeseen, jolloin vanhoja tiedostoja ei voi avata uudemmalla versiolla.

3.4.2 Sähköpostin lähetys

Ennustesovelluksessa on myös mahdollista lähettää sähköposteja ohjelmallisesti käyttäen MAPI-kirjastoa. Sähköposteja lähetetään vain kahdessa eri tapauksessa. Ensinnäkin käyttäjä voi jakaa muokkaamansa ennusteen muille käyttäen valikosta löytyvää painiketta Send as mail. Painikkeen painamisen jälkeen ohjelma pyytää käyttäjää tarvittaessa tallentamaan ennusteen ja avaa tämän jälkeen oletussähköpostiohjelman, jossa ennuste on valmiiksi asetettuna liitetiedostoksi. Sen lisäksi sähköpostiominaisuutta käyttää hyväkseen virheraportointi.

3.4.3 Virheraportointi

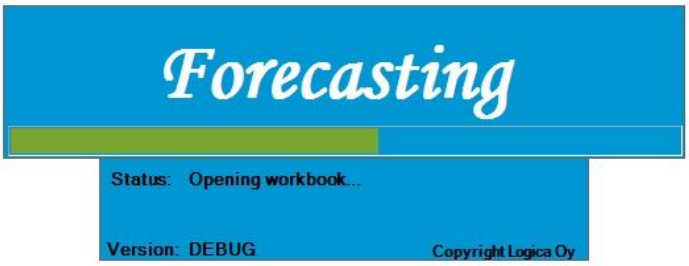
Virheraportointi toteutettiin omana moduulinaan ennustesovellukseen, koska haluttiin saada ongelmatilanteissa tarkka kuvaus virheestä ja muita tietoja, kuten Oracle Clientin rekisteri tiedot. Virherportointimoduulia voidaan kutsua, kun tapahtuu hallittu poikkeus. Hallitulla poikkeuksella tarkoitetaan poikkeusta, joka käsitellään ohjelmakoodissa. Tällaiset poikkeukset ohjelmassa eivät aiheuta ohjelman jumiutumista tai kaatumista.

Virheraportointimoduuli kysyy käyttäjältä aina kuviossa 26 esillä olevalla ikkunalla, haluaako hän lähettää virheraportin. Käyttäjän valitessa kyllä, joka on tietenkin suositus, ohjelma avaa sähköpostiohjelman, johon on valmiiksi asetettu vastaanottaja sekä virheen kuvaus. Käyttäjä voi myös halutessaan kirjoittaa lisätietoja virheestä viestiin merkittyyn kohtaan ja kertoa, mitä oli tekemässä, kun virhe tapahtui.

KUVIO 26. Ennustesovelluksen virheikkuna

3.4.4 Käynnistyslatain

Sovellukseen toteutettiin erillinen pieni ohjelma, käynnistyslatain. Käynnistyslatainta käytetään korvaamaan Microsoft Office:n oletus käynnistysikkuna, kun Excel käynnistyy. Tämä näkymä on nähtävillä kuviossa 27.



KUVIO 27. Ennustesovelluksen käynnistysikkuna

Käynnistyslatain suorittaa splash-näkymän aikana muutamia tärkeitä toimintoja. Tärkein tehtävistä on päivityksien tarkastaminen. Päivitykset tarkastetaan käyttäen ClickOncen luokkaa ApplicationDeployment. Yksinkertaistettu päivitysten tarkastukseen vaadittava ohjelmakoodi on kuvattu kuviossa 28. Siinä haetaan ensin sovelluksen käyttämä ilmentymä ApplicationDeployment-luokasta, minkä jälkeen sitä käyttäen tarkistetaan, onko uusia päivityksiä saatavilla. Jos uusi versio on saatavilla, ladataan ja asennetaan se.

```
ApplicationDeployment deployment = ApplicationDeployment.CurrentDeployment;
if (deployment.CheckForDetailedUpdate().UpdateAvailable)
{
    deployment.Update();
}
```

KUVIO 28. Päivityksien tarkastaminen

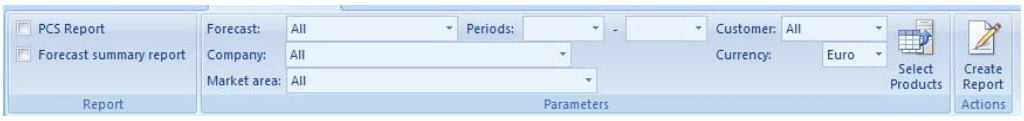
Käynnistyslatain luo myös kuviossa 29. esillä olevan ohjelmakoodin avulla uuden Excel-ohjelmaluokan ilmentymän. Tätä käyttäen rekisteröidään käsittelijä tapahtumalle WorkbookOpen, joka tapahtuu kun avataan uusi työkirja. Käsittelijässä voidaan sen jälkeen suorittaa ohjelmakoodia kun työkirja avataan. Seuraavaksi avataan uusi työkirja, jonka sijainti on tallennettuna muuttujaan nimeltä path. Tämän jälkeen asetetaan Excelin ikkuna näkyväksi käyttäjälle.

```
Microsoft.Office.Interop.Excel.Application _excel;

_excel = new Microsoft.Office.Interop.Excel.Application();

_excel.WorkbookOpen += new AppEvents_WorkbookOpenEventHandler(_excel_WorkbookOpen);

Workbook wb = _excel.Workbooks.Open(path, Type.Missing, Type.Missing,
    Type.Missing, Type.Missing, Type.Missing, Type.Missing, Type.Missing,
    Type.Missing, Type.Missing, Type.Missing, Type.Missing, Type.Missing,
    Type.Missing, Type.Missing);

_excel.Visible = true;
```

KUVIO 29. Excel ilmentymän luominen ja työkirjan avaaminen

3.4.5 Raportointi

Ennustesovelluksella on myös mahdollista generoida kahdenlaisia raportteja ennusteista. Nämä raportit ovat kappalemäärä- ja ennusteiden yhteenvetoraportit. Raportoinnille luotiin oma välilehti rengasvalikkoon (Ribbon bar), joka on nähtävillä kuviossa 30. Tämä välilehti on näkyvissä vain pääkäyttäjille.

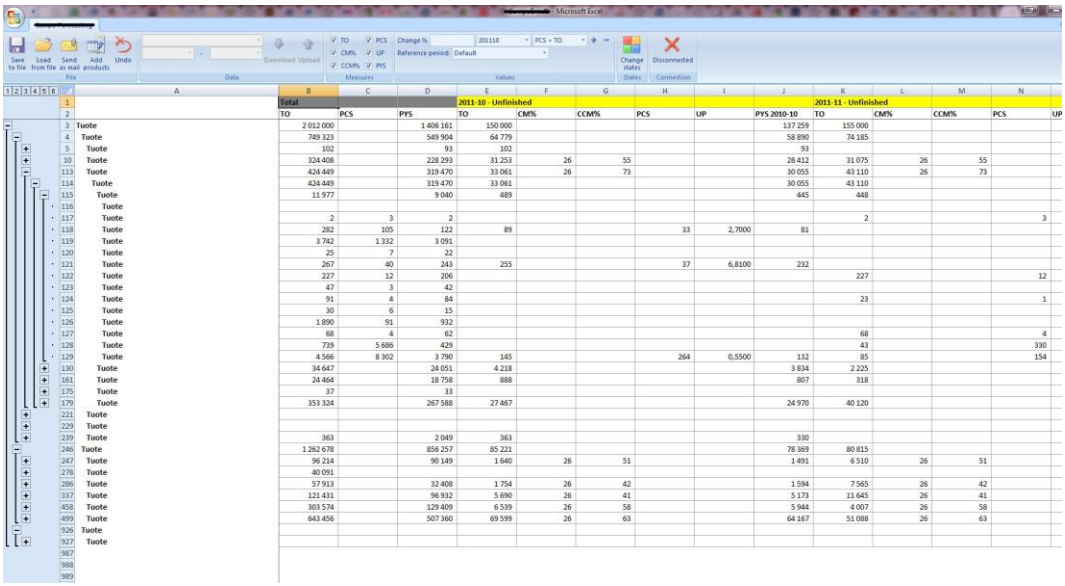


KUVIO 30. Raportoinnin rengasvalikko

Raporteille voidaan valita vain tietty osa tuotehierarkiasta tuotteidenvalintadialogin avulla. Raportteja voidaan myös rajata erilaisten valintojen avulla, kuten markkina-alue ja ennuste.

3.5 Ulkoasu

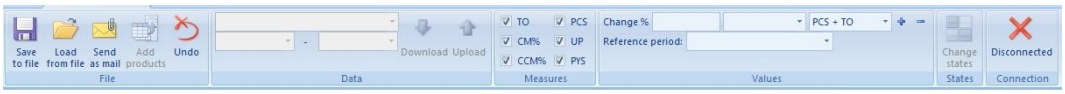
Ulkoasua suunniteltaessa ja toteuttaessa pyrittiin luomaan siitä mahdollisimman yksinkertainen ja selkeä. Kuviossa 31 on nähtävillä ennustesovelluksen käyttöliittymä. Ylhäällä on Microsoftin Office -ohjelmista tuttu rengasvalikko, joka on muokattu ennustesovelluksen tarpeisiin. Työskentelytilassa on ennusterivit ja periodit esitetty hierarkisesti. Yksi rivi kuvastaa siis yhden tuotteen tai tuoteryhmän ennustetietoja eri periodeilta. Periodit on värikoodattu eri tilojen mukaan. Keltainen kertoo, että periodin ennustaminen on aloitettu, mutta sitä ei ole vielä tehty loppuun. Harmaa kertoo sen, että kyseessä on tietoa, jota ei voida muokata. Punaisella värillä ilmoitetaan periodit, joiden ennustamista ei ole vielä aloitettu ja vihreällä ne periodit, joiden ennustaminen on valmis.



KUVIO 31. Ennustesovelluksen ulkoasu

3.5.1 Rengasvalikko

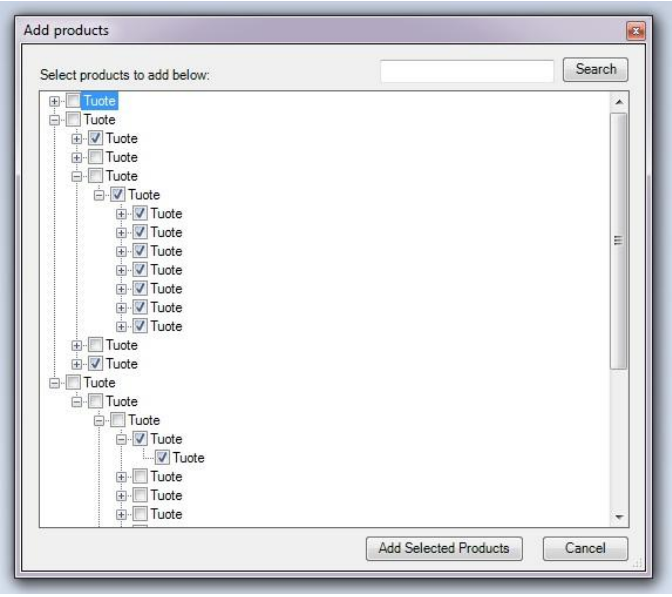
Rengasvalikko on selkeä tapa tehdä ohjelmaan valikot, ja se on jo entuudestaan tuttu käyttäjille. Kuviossa 32. on esitetty ennustesovellukseen toteutettu rengasvalikko.



KUVIO 32. Ennustesovelluksen rengasvalikko

Rengasvalikkoon on sijoitettu kaikki ennustesovelluksen toiminnot. Valikosta käyttäjä voi tallentaa, avata, lähettää ja ladata ennusteita. Käyttäjä voi myös piilottaa ja näyttää mittareita. Mittareiden arvoja voidaan myös kasvattaa tai vähentää tietyn prosenttiosuuden verran käyttäen massamuokkaustyökalua.

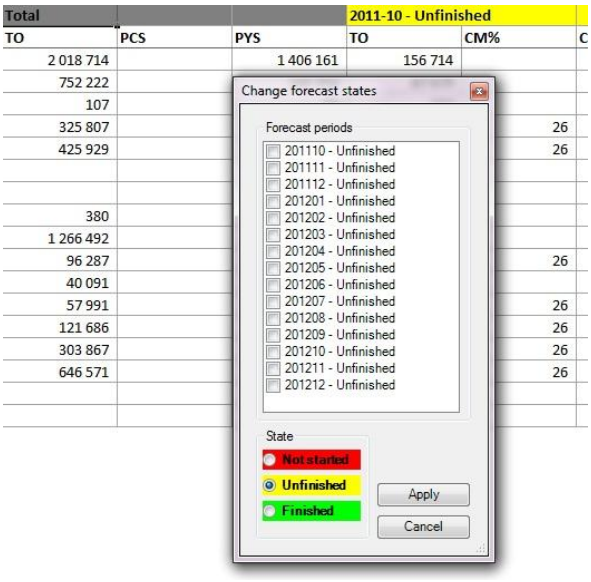
Tuotteita lisättäessä aukeaa kuvion 33. esittämä dialogi, josta käyttäjä voi valita haluamansa tuotteet. Dialogista käyttäjä voi myös etsiä tuotteita hakutoiminnon avulla.



KUVIO 33. Tuotteidenlisäämisdialogi

Rengasvalikosta löytyy myös Undo- eli peruutustoiminto, joka peruuttaa viimeisimmän arvon muokkauksen ja palauttaa sen edelliseksi. Valikosta on myös mahdollista luoda yhteys ennustekantaan, jos yhteys ei jostakin syystä onnistunut sovellusta käynnistättäessä.

Valikon avulla käyttäjät voivat myös vaihtaa periodeiden tiloja erillisen dialogin avulla. Tämä dialogi on nähtävillä kuviossa 34. Käyttäjälle näytettävät tilat ovat vain visuaalisena apuna käyttäjälle.



KUVIO 34. Ennusteperiodien tilojenvaihtodialogi

Ennustetietokannassa on myös jokaiselle periodille ja ennusteelle sisäiset tilat, jotka eivät näy suoraan käyttäjälle. Nämä tilat muuttuvat automaattisesti ennusteita ladattaessa ja lähetettäessä tietokantaan. Kun ennuste ladataan tietokannasta, se lukitaan tietylle käyttäjälle, ja merkitään tieto siitä, että ennustaminen kyseiselle ennusteelle on aloitettu. Tämän jälkeen ennustetta ei voi kukaan muu enää ladata kuin se käyttäjä, joka sen latasi ensimmäisenä. Sen jälkeen, kun käyttäjä on saanut ennusteen valmiiksi ja lähettää sen ennustekantaan, se lukitaan pysyvästi, minkä jälkeen ennustetta ei voi enää ladata ennustekannasta. Pääkäyttäjät voivat kuitenkin muuttaa ennusteiden tilaa tarvittaessa web-portaalin avulla.

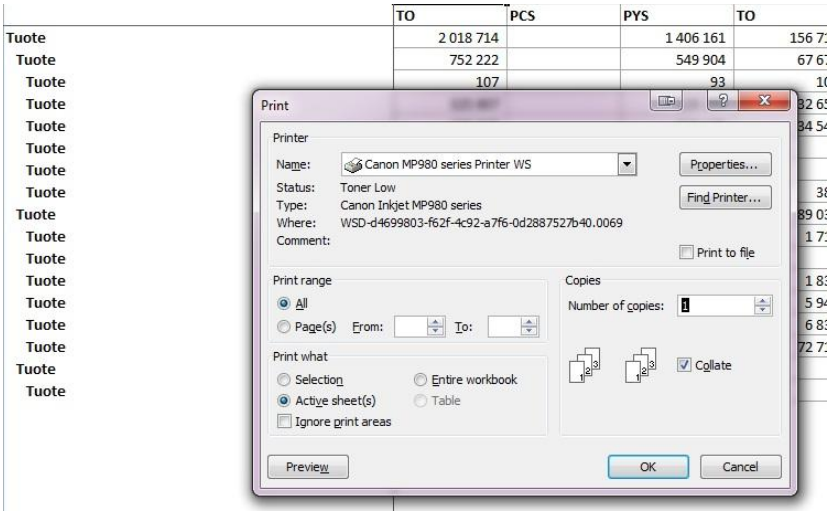
3.5.2 Työskentelytila

Ennustetietoa voi olla näytöllä kerralla paljon. Tuotteita voi olla jopa kymmeniä tuhansia ja ennustettavia kuukausia useita. Tästä syystä työtila haluttiin mahdollisimman suureksi, eli kaikki toiminnallisuus pyrittiin siirtämään rengasvalikkoon, jossa ne eivät ole tiellä. Työtilan ainoana toiminnallisuutena on arvojen esittäminen ja syöttäminen ennusteriveille sekä raporttien esittäminen.

Työskentelytilassa näkyvä tieto voidaan myös tulostaa. Tulostamisessa on hyödynnetty Excelin tarjoamaia ominaisuuksia. Kuviossa 35. on nähtävillä ohjelmakoodi, joka suoritetaan kun käyttäjä painaa Excelin valikosta tulosta-painiketta. Ohjelmakoodissa kutsutaan Excelin tarjoamaa valmista tulostusdialogia, joka on nähtävillä kuviossa 36.

```
private void print_btn_Click(object sender, RibbonControlEventArgs e)
{
    Globals.ThisWorkbook.Application.Dialogs[Microsoft.Office.Interop.Excel.XlBuiltInDialog.xlDialogPrint].Show();
}
```

KUVIO 35. Tulostusdialogin avaaminen



KUVIO 36. Tulostusdialogi

3.6 Ongelmat ja niiden ratkaisut

Työssä eniten ongelmia aiheutti sovelluksen suorituspolku ja Excelin tapa ilmoittaa käyttäjälle, että käytössä on kolmannen osapuolen komponentteja. Nämä kaksi ongelmaa nousivat esille vasta sovelluksen toteutuksen loppuvaiheilla. Muitakin

ongelmia sovelluksen toteutuksessa oli. Ne aiheutuivat pääsääntöisesti siitä, että VSTO:n avulla ei voida kontrolloida Exceliä täysin.

3.6.1 Ohjeikkuna

Microsoft Excel lisää automaattisesti kaikkien kolmannen osapuolen komponenttien kontrollien päälle kuviossa 37. esitetyn ponnahdusikkunan, joka kehottaa käyttäjää painamaan F1-näppäintä saadakseen lisätietoja. Tähän niin sanottuun ponnahdusikkunaan ohjelmoija voi lisätä lyhyen kuvauksen kontrollista, mutta ei muuta. Käyttäjän painaessa F1-näppäintä minkä tahansa kolmannen osapuolen konrollin päällä, avaa Excel kuvion 38. kaltaisen ohjeikkunan. Tässä ohjeikkunassa Microsoft kertoo ohjeet siihen, miten käyttäjä voi ottaa/poistaa käytöstä Officen liitännäisiä.

KUVIO 37. Excelin lisäämä ponnahdusikkuna

KUVIO 38. Excelin oma ohjeikkuna

Microsoft on lisännyt tämän ponnahdusikkunan, koska se sai viestejä toimimattomista liitännäisistä, joita se ei kuitenkaan ollut tehnyt. Tämä ponnahdusikkuna ja sen avaama ohjeikkuna ovat aina pakotetusti kolmannen osapuolen luomissa kontrolleissa, eikä niitä voida ylikirjoittaa. Niiden ainut tarkoitus on kertoa käyttäjälle, ettei kyseinen toiminnallisuus ole Microsoftin tekemä, eikä Microsoft täten ota mitään vastuuta niiden toimivuudesta.

Ennustesovellukseen asiakas kuitenkin halusi, että F1-näppäintä painettaessa käyttäjä saisi ohjeita sovelluksesta. Tämä ei kuitenkaan ole mahdollista VSTO:n tarjoamien eventtien ja olioiden avulla, joten piti keksiä jotakin muuta. Päädyttiin erikoiseen ja ehkä hieman kyseenalaiseen ratkaisuun. Sovellukseen toteutettiin näppäimistön painalluksia kuunteleva luokka, jonka tehtävänä on selvittää, koska käyttäjä painaa F1-näppäintä. Tämä luokka sijaitsee ennustesovelluksessa ja saattaa virhetilanteissa jäädä aktiiviseksi, jos ennustesovellus ei sammu oikein. Näppäimistön kuuntelijaluokan näppäinpainalluksen käsittelijäfunktio on esitettynä kuviossa 39. Näppäimistön kuuntelija kuitenkin kuuntelee näppäimistöä, vaikka ennustesovellus ei olisi aktiivinen, joten lisäksi piti selvittää, onko ennustesovellus aktiivinen vai onko jokin muu sovellus aktiivisena. Ennustesovelluksen ollessa

aktiivinen ikkuna ja käyttäjän painaessa F1-näppäintä näppäinkuuntelija kaappaa näppäimen painalluksen ja avaa tehdyn ohjeikkunan. Jos taas ennustesovellus ei ole aktiivinen ikkuna, näppäinkuuntelija ei reagoi mihinkään näppäimistön painalluksiin vaan välittää ne eteenpäin käsiteltäviksi. Lisäksi poistettiin F1-näppäimen toiminnallisuus ennustesovelluksesta, ettei Excel avaisi lisäksi omaa ohjeikkunaansa.

```
private static IntPtr HookCallback(int nCode, IntPtr wParam, IntPtr lParam)
{
    if (nCode >= 0 && wParam == (IntPtr)WM_KEYUP)  // Tarkisteaan näppäintapahtuman tyyppi
    {
        if (IsActiveWindow()) // Tarkastetaan onko ennustesovellus aktiivinen
        {
            int vkCode = Marshal.ReadInt32(lParam); // Luetaan näppäinkoodi parametreistä
            if ((Keys)vkCode == Keys.F1)    // Tarkistetaan onko F1 näppäin
                if (!string.IsNullOrEmpty(helpFilePath))  // Tarkistetaan ennustesovelluksen ohjeen polku
                    System.Diagnostics.Process.Start(@helpFilePath);  // Avataan ohjeikkuna uutena prosessina
        }
    }
    return CallNextHookEx(_hookID, nCode, wParam, lParam); // Lähetetään tapahtuma eteenpäin
}
```

KUVIO 39. Näppäinpainallusten käsittelijäfunktio

3.6.2 Suorituspolku

Ennustesovelluksen suorituspolku on sama kuin Office Excelin. Tämä aiheutti yllättäen ongelmia vanhan Oracle Client -version kanssa, joka on käytössä asiakkaalla. Vanha Oracle Client ei toimi oikein, jos sovelluksen suorituspolussa on erikoismerkkejä. Ennustesovelluksen oletussuorituspolku on Officen asennuskansio. Yleensä Office on asennettuna C:\Program Files\Microsoft Office\ -hakemistoon. Tässä hakemistopolussa ei kuitenkaan ole erikoismerkkejä, ellei kyseessä ole 64-bittinen käyttöjärjestelmä ja 32-bittinen Office-asennus, jolloin Officen oletusasennushakemisto on: C:\Program Files (x86)\Microsoft Office\. Tälläisessä tilanteessa suorituspolku sisältää erikoismerkkejä, jotka ovat tässä tapauksessa siis sulkumerkit. Seurauksena on se, ettei Oracle Client toimi oikein, eikä yhteyttä ennustekantaan voida muodostaa, jolloin ennustesovellusta ei voida käyttää online-tilassa tällaisilla kokoonpanolla.

Tähän ongelmaan on lähes mahdotonta tehdä korjausta ennustesovelluksen puolelta. Helpoin tapa päästä eroon ongelmasta on päivittää Oracle Client uudempaan versioon, jossa tämä ongelma on korjattu. Tätä ei kuitenkaan voitu tehdä asiakkaan järjestelmien toimivuuden takaamisen takia. Seuraavana vaihtoehtona on asentaa Office-hakemistoon, jossa ei ole erikoismerkkejä. Tämä vaihtoehto on kuitenkin jokseenkin työläs, koska käyttäjillä ei välttämättä ole oikeuksia itse asentaa sovelluksia koneelleen.

Nämä vaihtoehdot voi asiakas itse tehdä korjatakseen ongelman. Ennustesovellusta muuttamalla voidaan myös ratkaista tämä ongelma. Yksi monista muutoksista, joilla voitaisiin korjata ongelma, on luopua kokonaan Excelistä. Tällöin ennustesovellus voitaisiin asentaa hakemistoon, jossa ei ole erikoismerkkejä. Tämä kuitenkin vaatisi ainankin ennustesovelluksen käyttöliittymäosuuden uudelleen kirjoittamisen. Sen lisäksi joitakin muutoksia myös ennustetietojen käsittelyyn saatettaisiin joutua tekemään.

Toisena vaihtoehtona voitiasiin ennustetietojen lähetys ja hakeminen ennustekannsta siirtää erilliseen sovellukseen, joka toimisi palvelimena ennustesovelluksella. Tällöin ennustesovellus ei koskaan ottaisi suoraan tietokantayhteyttä vaan välipalvelin hoitaisi tämän. Tämä vaihtoehto taas lisäisi sovelluksen monimutkaisuutta ja hankaloittaisi ylläpitoa jossain määrin.

Kolmantena vaihtoehtona olisi tietokantayhteyteen käyttää kolmannenosapuolen kehittämää kirjastoa. Tämä on ratkaisuista omalla tavallaan huonoin. Kolmannenosapuolen kirjastot ovat maksullisia, eikä ole tietoa siitä, ratkaisisivtko ne lopulta ongelmaa.

4 SOVELLUKSEN TUOTTEISTAMINEN

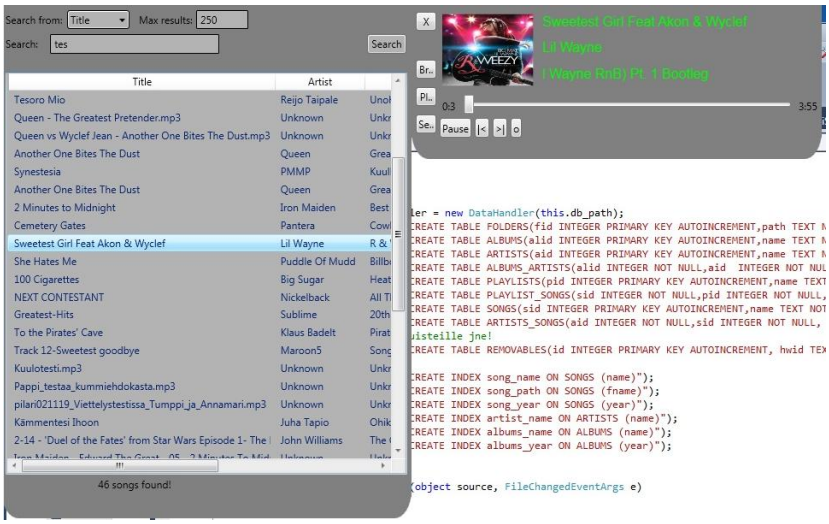
Ennustesovellusta voidaan kehittää itsenäisemmäksi ja monipuolisemmaksi sovellukseksi, jolloin se voidaan tuotteistaa helposti. Tällä hetkellä se on riippuvainen Microsoft Office Excelistä, eikä toimi ilman sitä. Sovellus on myös pitkälle räätälöity vain yhden asiakkaan tarpeiden ja järjestelmien mukaan. Tämän takia ennustesovelluksen käyttöönotto muilla asiakkailla on työlästä. Ylläpito vaikeutuu entisestään kun sovelluksesta on useita eri versioita eri asiakkailla. Ennustesovellus pitäisi kirjoittaa uusille asiakkaille lähes tyhjästä kokonaan uusiksi, jotta se toimisi heidän järjestelmissään. Lisäksi ennustesovellus vaatisi vanhan web-portaalin, jossa voidaan hallita ennusteita ja käyttöoikeuksia.

4.1.1 Käyttöliittymä

Microsoft Excel on todellisuudessa täysin turha ennustesovelluksen toiminnallisuuden kannalta. Samat toiminnallisuudet ja vielä enemmän voitaisiin saavuttaa Windows Presentation Foundationin eli WPF:n avulla. WPF mahdollistaa monipuolisen ja näyttävän käyttöliittymän toteuttamisen. Siihen on myös saatavilla samanlainen rengasvalikko kuin Officessa.

WPF sisältää DataGrid komponentin, jota voitaisiin hyvin käyttää ennustetietojen esittämiseen. Siinä onnistuvat samat asiat kuin Excelissäkin ja sitä voidaan myös kustomoida. WPF tarjoiaisi myös enemmän joustavuutta ja korjaisi kaikki nykyisen sovelluksen ongelmat.

WPF-sovelluksien käyttöliittymät voidaan toteuttaa myös Microsoft Blend -sovelluksella, joka mahdollistaa näyttävien ja kustomoitujen käyttöliittymien toteuttamisen. Microsoft Blend:llä voidaan myös suunnitella aivan uusia kontrolleja käytettäväksi WPF-sovelluksissa, tai muokata jo olemassa olevia. Kuviossa 40. on esimerkki WPF:llä toteutetusta sovelluksen käyttöliittymästä.



KUVIO 40. MiSe Instant Music sovelluksen käyttöliittymä

4.1.2 Liitännäisrajapinta

Ennustesovelluksesta saataisiin monikäyttöinen toteuttamalla asiakaskohtaiset vaatimukset liitännäisten avulla. Tämä vaatisi tarkat rajapinta- ja moduulisuunnittelut, mutta oikein ja hyvin toteutettuna se mahdollistaisi ennustesovelluksen helpon ylläpitämisen. Tälläinen liitännäisrajapinta voidaan tehdä esimerkiksi käyttäen Managed Extensibility Framework:iä eli MEF:iä.

Jokaista asiakasta kohden voitaisiin tehdä oma liitännäinen, joka toteuttaisi tietyn rajapinnan. Liitännäinen olisi vastuussa tietojen lataamisesta, lähettämisestä ja tallentamisesta. Itse ennustesovellus olisi lähinnä käyttöliittymä, joka toteuttaisi liitännäisten määrittelemät toiminnallisuudet. Jos toteutusta ei suunnitella tarkkaan, riskinä on, että loppujen lopuksi liitännäiset sisältävät suurimman osan toiminnallisuudesta. Tällöin uuden liitännäisen tekeminen voi olla työlästä.

Liitännäisten avulla olisi helppo hallita ja erottaa eri asiakkaiden ratkaisut toisistaan. Niiden avulla myös eri asiakkaiden ratkaisut voivat poiketa suuresti toisistaan. Toteutuksessa liitännäinen voi ottaa yhteyttä tietokantajärjestelmään ilman

rajoituksia tai se voi käyttää esimerkiksi WCF (Windows Communication Foundation) -välipalvelinta.

4.1.3 Ylläpito

Ylläpidon kannalta liitännäispohjainen ratkaisu helpottaisi ennustesovellukseen tehtävien korjausten ja ominaisuuksien toteutusta. Liitännäisiin tehtäisiin asiakaskohtaiset muutokset, kun taas itse ennustesovellukseen tehtäisiin kaikkia asiakkaita koskevat korjaukset ja muutokset.

Asiakaskohtaiset tiedot sijaitsisivat dll-kirjastossa ennustesovelluksen alihakemistossa. Ennustesovellus lataisi kirjaston automaattisesti käynnistyessään. Jos kirjastoja olisi useampia, sovellus voisi kysyä, mikä niistä ladataan. Tämä helpottaisi sovelluksen jatkokehitystä ja ylläpitoa. Sovellus voitaisiin helposti kääntää jokaiselle asiakkaalle käyttäen Visual Studion käännöskonfiguraatioita. Tälläinen konfiguraatio luotaisiin jokaiselle asiakkaalle. Lisäksi luotaisiin yksi konfiguraatio kehitystä ja testausta varten. Asiakaskohtainen konfiguraatio rakentaisi sovelluksesta asiakkaan vaatiman paketin ja liittäisi siihen oikean dll-kirjaston. Kehityskonfiguraatio liittäisi sovellukseen kaikkien asiakkaiden dll-kirjastot ja kysyisi käynnistyessään mitä niistä käytetään.

Lisäksi voitaisiin tehdä yksikkötestit, jotka tarkistaisivat kaikkien dll-kirjastojen toteuttamat rajapinnat virheiden välttämiseksi ja yhtenäisen toiminnan tarkastamiseksi.

4.1.4 Suunnittelu ja toteutus

Uuden ja joustavamman version toteuttaminen vaatii hyvin huolellisen suunnittelun. Arkkitehtuuri on suunniteltava tarkasti, ja siihen on kulutettava huomattavasti enemmän aikaa kuin edellisten versioiden suunnitteluun. Suunnittelussa voitaisiin käyttää apuna Visual Studion luokkasuunnitteluominaisuuksia. Sen avulla on

mahdollista generoida automaattisesti luokkien ja metodien rungot. Testitapaukset on suunniteltava myös ajoissa ja raja-arvot metodeille määriteltävä niin, että yksikkötestit voidaan kirjoittaa ja ottaa käyttöön kehityksen alusta alkaen.

Tälläisen sovelluksen toteuttaminen olisi myös huomattavasti helpompaa ja tehokkaampaa kunnon versionhallinnalla ja virheseurannalla. Tähän soveltuisi hyvin Visual Studio Team Foundation System versionhallinta. Se sisältää hyvän versionhallinnan ja virheseurannan. Tämän avulla voitaisiin seurata, kuka tekee ja milloin tekee mitäkin. Samalla nähtäsiin eri toiminnallisuuksien ja korjausten toteuttamiseen kulunut aika.

5 YHTEENVETO

Ennustesovelluksen lopputuloksena saavutettiin haluttu nopeus ja yksinkertainen käytettävyys. Sovelluksen toteutuksen alussa ei kuitenkaan tehty kunnollista suunnittelua, mikä näkyy selkeästi ohjelman rakenteessa. Sovelluksen monet luokat ovat tiukasti riippuvaisia toisistaan. Selkeimpänä kahtena osiona ovat käyttöliittymä ja ennustedataa käsittelevät toiminnot. Sovelluksesta olisi saatu huomattavasti selkeämpi rakenteiltaan ja helpommin laajennettava, jos kunnollinen arkkitehtuuri- ja moduulisuunnittelu olisi tehty. Tästä voidaankin todeta, kuinka suuri merkitys huolellisella suunnittelulla on ohjelmistoprojekteissa.

Asiakas oli ennustesovellukseen erittäin tyytyväinen, ja koko projektiryhmä saikin sovelluksesta paljon kiitosta. Sovellus on nyt ollut asiakkaalla käytössä noin puoli vuotta ja muutamia esiintyneitä virheitä lukuun ottamatta se on toiminut hyvin ja luotettavasti. Sovelluksesta ei ole pystytty korjaamaan tietyillä kokoonpanoilla ilmaantuvaa ongelmaa, jonka aiheuttavat erikoismerkit Officen asennushakemistossa.

Ennustesovellus täytti kaikki asiakkaan vaatimukset ja hieman enemmänkin. Asiakkaalle sovellus näkyy laadukkaana, yksinkertaisena ja selkeänä ratkaisuna. Se on helppokäyttöinen ja nopea. Tästä voidaan päätellä, että sovellus onnistui suunnitelmien mukaisesti, ja se täyttää kaikki sille asetetut vaatimukset. Sovellus on kuitenkin ohjelmakoodiltaan ja rakenteiltaan yllättävän monimutkainen ja paikoin hieman sekava.

Ennustesovellusta on tarkoitus tulevaisuudessa tarjota useammalle kuin yhdelle asiakkaalle. Tästä syystä onkin tärkeää miettiä tarkaan ja harkiten, miten uusi versio sovelluksesta tulisi suunnitella ja toteuttaa, jotta siitä saataisiin mahdollisimman joustava ja laajennettava. Uuden version toteutuksessa voitaisiin hyödyntää yksikkötestausta. Web-portaali pitäisi myös kokonaan jättää pois ja sen toiminnallisuus siirtää ennustesovellukseen.

LÄHTEET

Carter, E. & Lippert, E. 2009. Visual Studio tools for Office 2007. 2. painos. Boston: Pearson Education.

Dundas. 2012. Executive Dashboard Demo [viitattu 3.4.2012]. Saatavissa: http://www.dundas.com/dashboard/online-examples/demos/Executive-Dashboard.aspx

Hammergren, T. & Simon, R. 2009. Data Warehousing For Dummies. 2. painos. Indiana: Wiley publishin.

Hovi, A. 1997. Tietovarastotekniikka. Espoo: Suomen Atk-kustannus Oy.

Kimball, R. & Ross, M. 2002. The Datawarehouse Toolkit. 2. painos. USA: John Wiley and Sons.

Brian Larson. 2009. Delivering Business Intelligence. USA: McGraw-Hill.

Logica. 2009. The BI Framework How to Turn Information into a Competitive Asset. Logica.

Logica. 2012. Logica lyhyesti [viitattu 14.3.2012]. Saatavissa: http://www.logica.fi/we-are-logica/about-logica/

Microsoft. 2012. ClickOnce Deployment Overview [viitattu 14.3.2012]. Saatavissa: http://msdn.microsoft.com/en-us/library/142dbbz4%28v=vs.90%29.aspx

Rasmussen, N, Goldy P. & Solli, P. 2002. Financial Business Intelligence. New York: John Wiley and Sons.

Troelsen, A. 2010. Pro C# and the .NET 4 Platform. 5. Painos. New York: Springer Science+Business Media.