# ENTITY FRAMEWORK

## BETTY-tietojärjestelmässä

Olli Luukas

Opinnäytetyö
Joulukuu 2011

Ohjelmistotekniikan koulutusohjelma
Tekniikan ja liikenteen ala

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖN
KUVAILULEHTI

| Tekijä(t) | Julkaisun laji | Päivämäärä |
|---|---|---|
| LUUKAS, Olli | Opinnäytetyö | 7.12.2011 |
| | Sivumäärä<br>28 | Julkaisun kieli<br>Suomi |
| | Luottamuksellisuus<br>( ) saakka | Verkkojulkaisulupa myönnetty<br>( X ) |

Työn nimi
ENTITY FRAMEWORK
BETTY-tietojärjestelmässä

Koulutusohjelma
Ohjelmistotekniikan koulutusohjelma

Työn ohjaaja(t)
SALMIKANGAS, Esa

Toimeksiantaja(t)
Jyväskylän ammattikorkeakoulu,
BETTY-projekti

Tiivistelmä

Opinnäytetyössä toteutettiin tietojärjestelmä osana Jyväskylän ammattikorkeakoulun BETTY-projektia. Tietojärjestelmän päätarkoituksena oli JAMK:n betonilaboratorioon toimitettavien betonikoekappaleiden toimitusketjun digitalisointi. Toteutettu tietojärjestelmä koostui tietokannasta ja tietokantaan yhteydessä olevista sovelluksista. Sovellukset olivat kaksi betonilaboratorion työntekijöille kehitettyä työpöytäsovellusta ja betonilaboratorion asiakkaille kehitetty internetselaimessa toimiva asiakassovellus.

Työssä kuvataan Entity Frameworkin käyttöä tietojärjestelmää toteuttaessa, sen tuomia hyötyjä ja käydään läpi vastaan tulleita ongelmia. Työssä esitellään myös muita tietojärjestelmän toteutuksessa hyödynnettyjä teknologioita ja sovelluksia. Pääpaino työssä on kuitenkin Entity Frameworkissa ja siihen kuuluvassa Entity Data Modelissa.

Työn tuloksensa saatiin yleiskuva Entity Frameworkin ja Entity Data Modelin soveltamisesta MySQL-tietokantaa käyttävässä tietojärjestelmässä. Myös Entity Frameworkin mahdollistamat erilaiset lähestymistavat tietorakenteisiin pohjautuvien sovellusten kehittämiseen tuovat tietynlaista joustavuutta sovelluskehitykseen. Entity Framework mahdollistaa tietorakenteen helpon ja nopean päivittämisen tietorakenteesta luotujen olioiden dokumentaation hinnalla.

Entity Frameworkille tarjotaan sekä hyvät kehitys- ja hallintaympäristöt että selkeä dokumentaatio ja kattava esimerkkivalikoima, miten sitä voidaan hyödyntää. Se toimii myös moitteetta yhteen MySQL:n kanssa.

Avainsanat (asiasanat)
Entity Framework, Entity Data Model, MySQL, .NET Framework, tietojärjestelmä

Muut tiedot

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

DESCRIPTION

| Author(s) | Type of publication | Date |
|---|---|---|
| LUUKAS, Olli | Bachelor´s Thesis | 07122011 |
| | Pages<br>28 | Language<br>Finnish |
| | Confidential<br><br>( ) Until | Permission for web publication<br>( X ) |

Title
ENTITY FRAMEWORK
in BETTY information system

Degree Programme
Software Engineering

Tutor(s)
SALMIKANGAS, Esa

Assigned by
JAMK University of Applied Sciences,
Project BETTY

Abstract

In this bachelor's thesis an information system was implemented as a part of project BETTY, a project of JAMK University of Applied Sciences. The main purpose of the information system was to digitalize the supply chain of concrete samples coming to JAMK's concrete laboratory. The implemented information system consisted of a database and applications connecting to the database. The applications were two desktop applications for the employees in the concrete laboratory and a web browser application for the customers of the concrete laboratory.

In this thesis the use of Entity Framework is described during the implementation of the information system. The gains the Entity Framework brought and problems that stood out are also processed. There are additional other technologies and applications which were used during the implementation of the information system. The main focus of this thesis is still on Entity Framework and Entity Data Model.

This thesis resulted in a general view of applying Entity Framework and Entity Data Model to an information system using a MySQL database. Entity Framework allows different kind of design approaches which allow certain sort of flexibility towards for data-oriented applications. Entity Framework enables easy and fast updating of data structure to created objects with the price of documentation.

Entity Framework is provided with both good development and management environments and clear documentation and comprehensive selection of tutorials. It also works faultlessly with MySQL.

Keywords
Entity Framework, Entity Data Model, MySQL, .NET Framework, information system

Miscellaneous

# SISÄLTÖ



# KUVIOT

# KÄSITTEET JA LYHENTEET / TERMISTÖ

**Entiteetti**

Itsenäinen kokonaisuus esim. ER-kaaviossa ja Entity Data Modelissa.

**ER-kaavio**

Entity-relationship –model, vapaasti suomennettuna entiteetti-relaatiokaavio. Sisältää entiteetit ja niiden väliset relaatiot.

**GPL-lisenssi**

GNU General Public License, GNU-hankkeen yleinen lisenssi. Tarkoitettu vapaiden ohjelmistojen julkaisemiseen, sillä se antaa kenelle tahansa oikeuden käyttää, kopioida, muuttaa ja jakaa edelleen ohjelmia ja niiden lähdekoodia.

**MSSQL**

Microsoft SQL Server. Microsoftin kehittämä relaatiotietokantaohjelmisto.

**ORM**

Object-relational Mapping. Ohjelmointitekniikka jossa luodaan esim. tietokannasta tiedot oliomuotoon ja täten käytettäviksi suoraan ohjelmointikielessä.

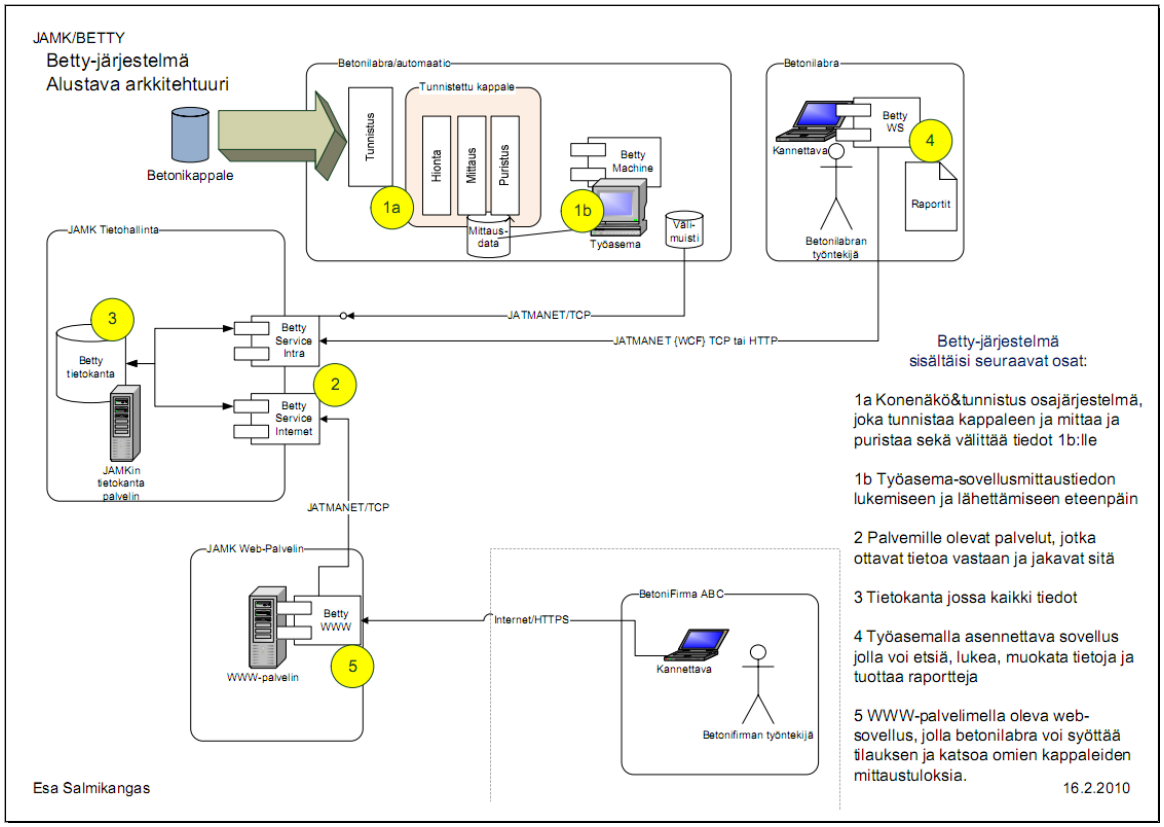
**Relaatio**

Entiteettien välinen suhde.

# 1 TYÖN LÄHTÖKOHDAT

Opinnäytetyön toimeksiantajana toimi Jyväskylän ammattikorkeakoulun BETTY -projekti, jonka tarkoituksena oli betonilaboratorion tutkimus- ja kehitysympäristön kehittäminen. Yhtenä projektin osana oli betonintestauslaboratorioon saapuvien koekappaleiden toimitusketjun digitalisointi, eli fyysisessä muodossa olevan aineiston siirtäminen digitaaliseen muotoon. Toimitusketju sisälsi paljon mm. erilaisten lomakkeiden, kuten koekappalekorttien (liite 1), täyttämistä ja tietojen siirtämistä edellä mainituista lomakkeista erillisiin seloste- ja laskutuspohjiin. Lisäksi betonilaboratoriolla on useita asiakkaita, joille toimitetaan koekappaleiden mittauksista saatuja tietoja. Tämän kokonaisuuden digitalisointi toimi opinnäytetyön lähtökohtana.

Toimitusketjun digitalisoiminen toteutettiin kehittämällä Betty-tietojärjestelmä (ks. kuvio 1). Betty-tietojärjestelmän tavoitteena oli kerätä betonikoekappaleiden toimitusketjun aikana koekappaleita koskevat tiedot keskitettyyn tietokantaan, josta betonilaboratorion työntekijät ja asiakkaat pystyvät tarkastelemaan koekappaleisiin liittyviä tietoja. Tietojärjestelmää varten kehitettiin erilliset sovellukset niin betonilaboratorion työntekijöille kuin sen asiakkaille. Betonilaboratorion laboratoriopuolella laboratoriotyöntekijöiden täytyi pystyä tarkistamaan tiettynä päivänä testattavien koekappaleiden tunnisteet ja lisätä testattavien koekappaleiden mittaus-, punnitus-, puristus- ja tiiveyskoetuloksia tietojärjestelmään. Betonilaboratorion toimistopuolella laboratoriotyöntekijät tarvitsivat sekä mahdollisuuden seurata koekappaleiden kulkua toimitusketjussa että hallinnoida useiden koekappaleiden tiedoista luotavia selosteita ja laskutustietoja. Betonilaboratorion asiakkaille oli luotava mahdollisuus seurata omien koekappaleidensa tilaa, niiden koetuloksia ja sijaintia toimitusketjussa.




KUVIO 1. Betty-järjestelmän alustava arkkitehtuuri

Yksi tietojärjestelmän keskeisimmistä osista oli keskitetty tietokanta, jonka tuli olla vakaa ja johon tuli olla pääsy sekä betonilaboratorion työntekijöillä että asiakkailla. Jotta tietokanta olisi helposti käytettävissä, kehitettiin betonilaboratorion työntekijöille erilliset työpöytäsovellukset laboratorio- ja toimistotiloihin ja asiakkaille erillinen selainsovellus.

# 2 TEKNIIKAT

## 2.1 Yleistä

BETTY-projektissa kehitettävän järjestelmän tietokantavaihtoehdot olivat MSSQL ja MySQL, sillä molemmat olivat toteuttajille ennestään tuttuja ja molemmille löytyi *Entity Framework*-tuki. Tietokannaksi valittiin MySQL lähinnä sen ilmaisuuden perusteella.

MySQL-tietokannan lisäksi BETTY-projektin digitalisoinnissa käytettiin .NET Frameworkiin pohjautuvia tekniikoita. Tärkeimmät BETTY-projektiin liittyvät .NET-tekniikat olivat C#, ASP.NET, Windows Presentation Foundation ja Entity Framework, joista Entity Framework osoittautui myös hyväksi tutkimuskohteeksi Microsoftin ulkopuolisen tahojen tarjoaman tietokantaohjelmiston kanssa.

## 2.2 MySQL

MySQL on tunnettu tietokannan hallintajärjestelmä, joka on saatavilla sekä maksullisena että GPL-lisenssin alaisena. Tämän lisäksi MySQL tarjoaa useille eri ohjelmointikielille rajapintoja, jotka ovat saatavilla ilmaiseksi MySQL:n internetsivuilta (MySQL Connectors n.d.). Yksi näistä rajapinnoista on ADO.NET-rajapinta, joka on tarpeellinen sekä BETTY-projektin työpöytä- että asiakassovelluksia varten. BETTY-projektin MySQL-tietokanta toimi erillisellä Linux-palvelimella.

Koska MySQL on pääsääntöisesti tietokannan hallintajärjestelmä, se toimitetaan tavanomaisesti ilman graafista käyttöliittymää. MySQL:n tietokantojen ja niiden rakenteen hallinnointiin voidaan käyttää komentorivityökaluja, tai vaihtoehtoisesti erilaisia käyttöliittymillä varustettuja sovelluksia, kuten MySQL Workbench, jota hyödynnettiin BETTY-projektissa.

## 2.3 .NET Framework

.NET Framework on Microsoftin kehittämä sovelluskehys ja samalla yhtenäinen osa Windows-käyttöjärjestelmää (.NET Framework Conceptual Overview n.d.). Sen tärkeimmät komponentit ovat ajoympäristö eli Common Language Runtime (CLR) ja luokkakirjasto, joka sisältää mm. ADO.NETin. (.NET Framework 4 n.d.) .NET Frameworkin vahvuus on mahdollistaa sovelluskehittäjän keskittymisen businesslogiikkaan ja se tarjoaa myös valmiit luokkakirjastot Microsoft Office –tuotteisiin.

BETTY-projektissa käytettiin .NET Frameworkin versiota 4.0, ja hyödynnettiin sen tarjoamia Microsoft Officen luokkakirjastoja. Betonilaboratorion työntekijöiden käyttöön suunnitellut kaksi työpöytäsovellusta toteutettiin WPF:ää ja asiakkaille tarkoitettu asiakassovellus ASP.NETiä hyödyntäen. Suurimpana etuna näissä tekniikoissa BETTY-projektin kannalta oli mahdollisuus käyttää samaa businesslogiikan sisältävää kirjastoa sekä työpöytä- että selainsovelluksissa.

## 2.4 Entity Framework

### 2.4.1 Yleistä

Tietokantasovellusten kehityksessä on yleistä käsitellä tietokantataulun tietuetta yhtenä kokonaisuutena, esimerkiksi oliona. Tällöin oliolla kulkee mukanaan kaikki tieto, joka voi olla sille mm. businesslogiikan kannalta tärkeää. Sovelluskehityksessä puhutaan ns. kääreluokista (wrapper class), jotka sisällyttävät toisen luokan tai komponentin toiminnallisuuden itseensä (Sonier 2009.). Tietokantataulun tietueesta luotava olio on hyvä esimerkki kääreluokasta, sillä tällöin kyseiseen olioon sisällytettäisiin mahdollisesti toiminnallisuus kyseisen tietokantataulun tietueiden luomiseen, lukemiseen, päivittämiseen ja poistamiseen. Tällaisten kääreluokkien tekeminen on kuitenkin työlästä, ja ne ovat

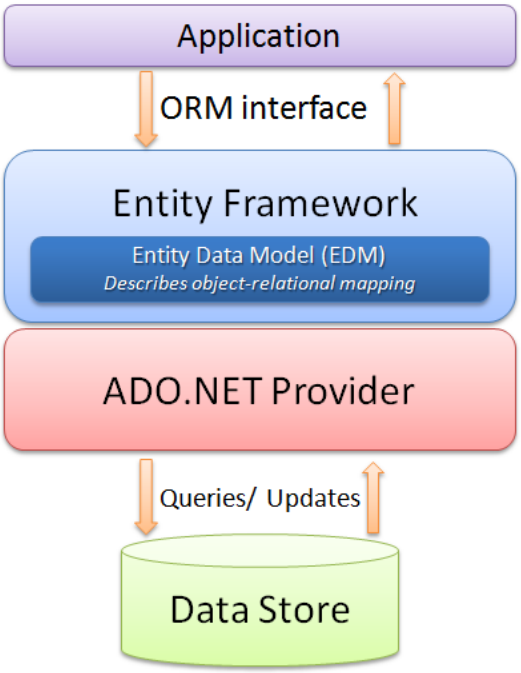
useimmiten projekti- tai tietokantakohtaisia ratkaisuja. Tähän ongelmaan Entity Framework tarjoaa ratkaisun.

Entity Framework on .NET Frameworkia varten suunniteltu ORM-kehys, mutta ei kuitenkaan ensimmäinen .NET Frameworkille tehty ORM. Tunnettuja .NET Frameworkille ennen Entity Frameworkia suunniteltuja ORM:eja ovat mm. NHibernate ja LINQ to SQL. ORM yleistää tietokantaan tallennetun tiedon loogisen mallin ja esittää sen käsitteellisenä mallina sovellukselle. Sovelluskehityksessä tämä näkyy siten, että esimerkiksi tietokantaohjelmistoa vaihdettaessa tietokantaa käsitteleviä toimintoja ei tarvitse uudelleenkirjoittaa, ja vältetään ns. kovakoodaus. Myös jo olemassa olevaan tietokantaan tehtävät muutokset voidaan siirtää ohjelmakoodiin. Entity Frameworkilla kaikkia muutoksia ei kuitenkaan saada suoraan siirrettyä, kuten tietokantataulun pääavaimen muutosta. Uudet, muutoksissa lisätyt, tietokantataulut taas siirtyvät ohjelmakoodiin ilman ongelmia.

Entity Frameworkin ensimmäinen versio julkaistiin vuonna 2008 .NET Framework Service Pack 1:n ja Visual Studio 2008 Service Pack 1:n yhteydessä. Ensimmäinen versio kohtasi paljon kritiikkiä, ja siitä seurauksena se sai hieman yli 900 ohjelmistokehittäjän allekirjoittaman epäluottamuslauseen (ADO.NET Entity Framework Vote of No Confidence n.d.; Entity Framework Vote of No Confidence Signatories 2008). Entity Frameworkin toinen versio, nimeltään Entity Framework 4.0, julkaistiin osana .NET Framework 4.0:aa vuoden 2010 ensimmäisellä puoliskolla. Versio 4.0 korjasi monia vikoja, joita ensimmäisessä versiossa oli ja jotka olivat johtaneet epäluottamuslauseeseen. Kolmas versio, jota BETTY-projektissa käytettiin, versio 4.1, julkaistiin vuonna 2011.

Entity Framework on kokoelma tietopainoitteisten ohjelmistosovellusten kehittämistä tukevia teknologioita ADO.NETissä ja se mahdollistaa sovelluskehittäjien työskentelyn toimialuekohtaisten olioiden ja ominaisuuksien kanssa ilman huolta siitä, millaiseen tietokantaan tiedot on tallennettu. Sovelluskehittäjien ei tarvitse enää olla täysin tietoisia tietokannan rakenteesta, mutta he pääsevät käsiksi tietokannassa oleviin tietoihin ja relaatioihin ohjelmointirajapinnasta (ks. kuvio 2). Tämän johdosta sovelluskehittäjät voi-

vat luoda ja ylläpitää tietopainotteisia sovelluksia vähemmällä määrällä ohjelmistokoodia kuin perinteisissä sovelluksissa, joita kehittäessä sovelluskehittäjien täytyi olla tietoisia tiedon sijainnista tietokannassa sijaitsevissa tauluissa ja sarakkeissa. Entity Framework on .NET Frameworkiin tehty ORM, jonka Entity Data Model määrittelee. Entity Framework vaatii vähintään .NET Frameworkin version 3.5 SP1 toimiakseen. (Entity Framework Overview n.d.)




KUVIO 2. Entity Frameworkin sijainti sovellusarkkitehtuurissa (ks. alkuperäinen kuvio: ADO.NET Entity Framework At-a-Glance n.d.)

### 2.4.2 Entity Data Model

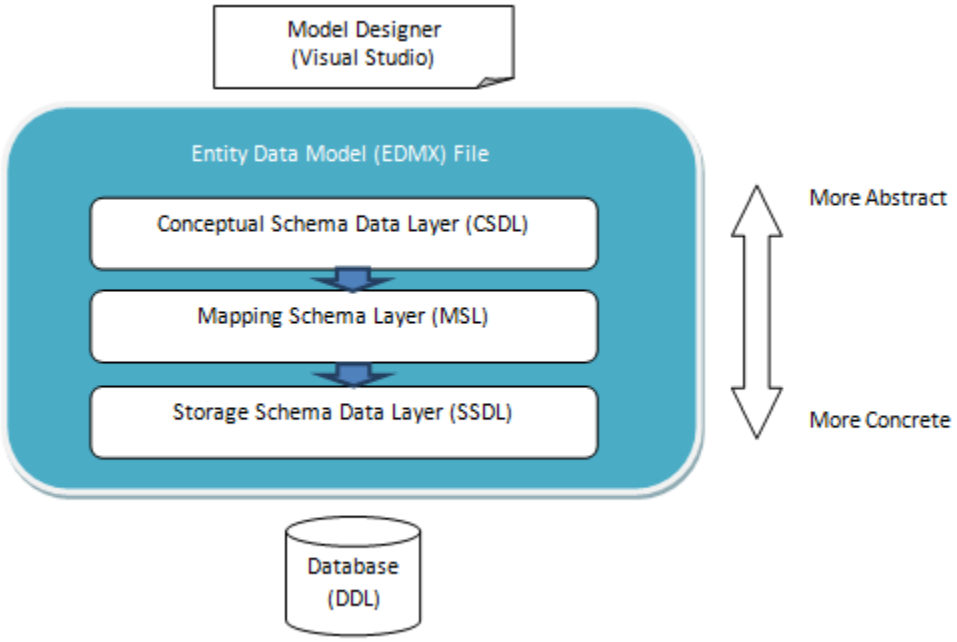
#### 2.4.2.1. Yleistä

Entity Data Model koostuu käsitteistä, jotka kuvaavat tietorakenteen sen tallennusmuodosta riippumatta. Tavanomaisissa tapauksissa tieto voi olla tallennettuna useaan eri tallennusmuotoon, jolloin tietopohjaisen sovelluksen suunnittelussa haasteeksi koituu tasapainoilu sekä ohjelmakoodin kirjoittamisen ja ylläpidon että tietoyhteyksien, tiedon

varastoinnin ja skaalautumisen välillä. EDM käsittelee tämän haasteen kuvaamalla tietorakenteen entiteetteinä ja relaatioina, jotka ovat riippumattomia tietovarastomallista. Seurauksena tästä tiedon muoto on epäoleellista sovellussuunnittelun ja -kehityksen suhteen, ja entiteetit ja relaatiot voivat kehittyä sovelluksen kehittyessä niiden kuvatessa tietorakennetta siten, kuinka sitä sovelluksessa käytetään. EDM siis mahdollistaa sovelluskehittäjien keskittymisen ohjelmakoodin kirjoittamiseen ja ylläpitoon. (Entity Data Model n.d.)

EDM rakentuu kolmesta eri kerroksesta, joiden toisessa päässä sijaitsevat entiteetit ja toisessa tieto määrittelemättömässä muodossa (ks. kuvio 3). EDM:ia luodessa jokaiselle näille kolmelle kerrokselle luodaan omat määrittelytiedostot, ja jokaisen tiedoston sisältö määritellään mallia vastaavalla XML-pohjaisella määrittelykielellä:

- käsitteellisen mallin määrittelykieli (CSDL)
- kartoituksen määrittelykieli (MSL) ja
- varastomallin määrittelykieli (SSDL).




KUVIO 3. Entity Data Modelin rakenne (ks. alkuperäinen kuvio Fancey 2010)

CSDL määrittelee entiteetit ja niiden väliset relaatiot, kun taas SSDL on esitysmuoto tietokannasta tai sen osasta. MSL määrittelee kartoituksen käsitteellisen ja varastokerroksen välillä. Entity Framework muodostaa tietokantakyselyt näiden määrittelytiedostojen perusteella.

Suurimmassa osassa tapauksista EDM:n tarkoitus on luoda yksi yhteen-suhteessa oleva kartoitus olemassa olevan tietokantakaavion ja siitä luotavan käsitteellisen mallin välille. Tietokantakaaviot koostuvat pääosin tauluista, jotka on mahdollisesti linkitetty toisiinsa relaatioilla, kun taas entiteettityypit määrittelevät tiedon käsitteellisen mallin. Entiteettityypit ovat kooste useasta kentästä, joista jokainen viittaa esimerkiksi tiettyyn tietokannassa olevaan sarakkeeseen. Entiteettityypit voivat olla relaatiossa toistensa kanssa, riippumatta tietokantakaavion relaatioista. Relaatioissa olevat entiteetit ovat esillä relaation osoittavan kentän kautta, joka mahdollistaa kokonaisen entiteetin tai entiteettikokoelman palauttamisen tietokantasarakkeen arvon sijaan. Käytännössä entiteetit ovat olio-ohjelmoinnin luokkailmentymiä tietokantataulujen riveistä, joissa tietokantataulun sarakkeet ja relaatiot näkyvät luokkaolion ominaisuuksina. Luokkailmentyminä entiteettejä voidaan laajentaa lisäämällä niille esimerkiksi metodeja ja logiikkaan perustuvia ominaisuuksia.

#### 2.4.2.2. Avainkäsitteet

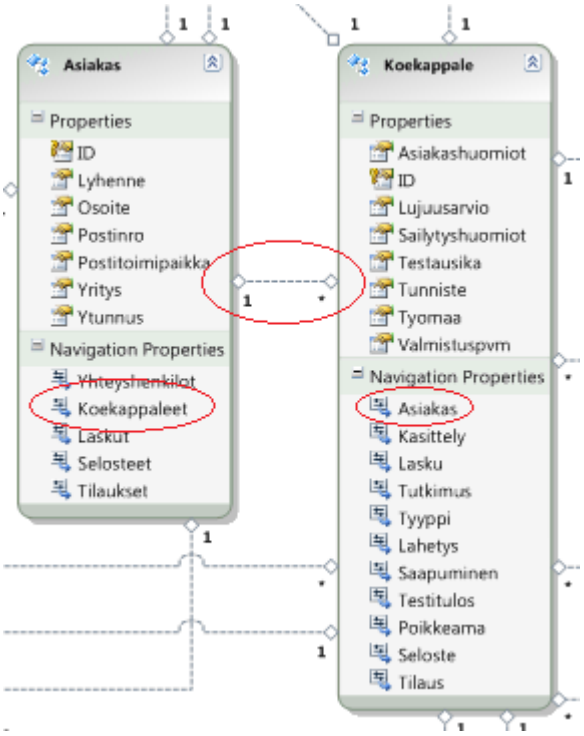
Entity Data Modelin tietorakenteen määrittelee kolme avainkäsitettä:

- entiteettityyppi
- yhteystyyppi ja
- ominaisuus.

**Entiteettityyppi**

Entiteettityyppi on EDM:n tietorakenteen näkyvin käsite. Se koostuu ominaisuuksista ja kuvaa ylätason käsitteiden rakenteen, kuten esimerkiksi koekappale ja asiakas BETTY-projektin tietojärjestelmässä (ks. kuvio 4). Entiteetti esittää tiettyä määritettyä objektia,

kuten koekappaletta tai asiakasta. Entiteettityyppi on entiteetin mallinne (entity type (Entity Data Model) n.d.), samalla tavalla kuin ohjelmakoodissa luokan määritelmä on kyseisestä luokasta luotavan olion mallinne. Toisinpäin ilmaistuna entiteetti on entiteettityypin ilmentymä. Entiteeteillä täytyy olla uniikki avain entiteettijoukossa, joka on kokoelma tietyn entiteettityypin ilmentymiä, kuten tietokannassa tietokantataulun riveillä täytyy olla pääavain. Entiteettityyppi voi periytyä toisesta entiteettityypistä, mutta perittyjä ominaisuuksia ei voi ylikirjoittaa.




KUVIO 4. Betty-projektin asiakas- ja koekappaletauluista luodut entiteetit ja niiden välinen yhteys ja toisiinsa viittaavat navigointiominaisuudet korostettuna

**Yhteystyyppi**

Yhteystyyppi, lyhyemmin yhteys, kuvaa EDM:n entiteettien välisiä suhteita, kuten relaatiot relaatiotietokannassa. Yksi yhteyden keskeisimmistä eduista ovat sen avulla luodut navigointiominaisuudet, jotka mahdollistavat kokonaisen entiteetin sijoittamisen toisen entiteetin ominaisuudeksi. Esimerkiksi tietokannassa koekappaleella on tiedossa asiakkaan pääavain, mutta EDM:ssa yhteyden avulla koekappale-entiteetillä kyseinen asiakasentiteetti navigointiominaisuutena (ks. kuvio 4). Kyseisestä navigointiominaisuudesta

pääsee käsiksi kaikkiin kyseisen asiakas-entiteetin ominaisuuksiin ja mahdollisiin laajennuksissa määritettyihin metodeihin. Vastaavasti asiakas-entiteetillä on navigointiominaisuutena kokoelma koekappale-entiteettejä.

**Ominaisuus**

Ominaisuudet määrittelevät entiteettityyppien rakenteen, esimerkiksi asiakas-entiteettityypin rakenteen määrittelevät sen ominaisuudet (ks. kuvio 4). EDM:n ominaisuudet ovat täysin verrannollisia luokkaolioiden ominaisuuksiin.

#### 2.4.2.3. Lähestymistavat

Kun Entity Frameworkia käytetään sovelluskehityksessä, se mahdollistaa erilaisia lähestymistapoja EDM:n luomiseen. Käytettävissä olevat lähestymistavat ovat:

- tietokanta ensin (Database First)
- malli ensin (Model First) ja
- ohjelmakoodi ensin (Code First).

Tietokanta ensin on yleinen lähestymistapa sovelluskehityksessä. Tässä periaatteessa luodaan ensin tietokanta, joka sitten sovitetaan ohjelmakoodiin. Tämä on mahdollista Entity Frameworkissa luomalla EDM jo olemassa olevasta tietokannasta. BETTY-projektissa käytettiin tätä lähestymistapaa.

Malli ensin mahdollistaa EDM:n määrittämisen ilman taustalla toimivaa tietokantaa, jolloin sovelluskehittäjät voivat vapaasti luoda uusia entiteettejä ja käyttää niitä ohjelmakoodissa. Tietokantaa ei tarvitse olla olemassa sovelluskehitystä tehdessä, mutta sen voi luoda sovelluskehityksen aikana kehitetystä EDM:sta.

Ohjelmakoodi ensin on vielä testivaiheessa oleva lähestymistapa, jossa tietokannan rakenteen muodostaminen tehdään ohjelmakoodissa EDM:n mallieditorin sijaan. Tällöin

ohjelmakoodissa olevat luokat ja niiden ominaisuudet merkitään tarpeen mukaan määritteillä, jotka kertovat mm. tietyn luokan kuvaavan tietokantataulua, ja saman luokan tietyn ominaisuuden kuvaavan edellä mainitun taulun pääavainta. (Joshi 2011.)

### 2.4.3 Entity Frameworkin hyödyt

Entity Framework vähentää sovelluksen tietorakenteen luomiseen ja sen päivittämiseen käytettävää aikaa. Se korvaa kääreluokat toiminnallisuudellaan ja suorittaa tietorakennemuutoksia seuraavat ohjelmakoodiin tehtävät muutokset automaattisesti. Entity Frameworkin käyttöönotto on yksinkertaista, eikä se vaadi sovelluskehittäjältä olemassa olevaa tietokantapalvelinta.

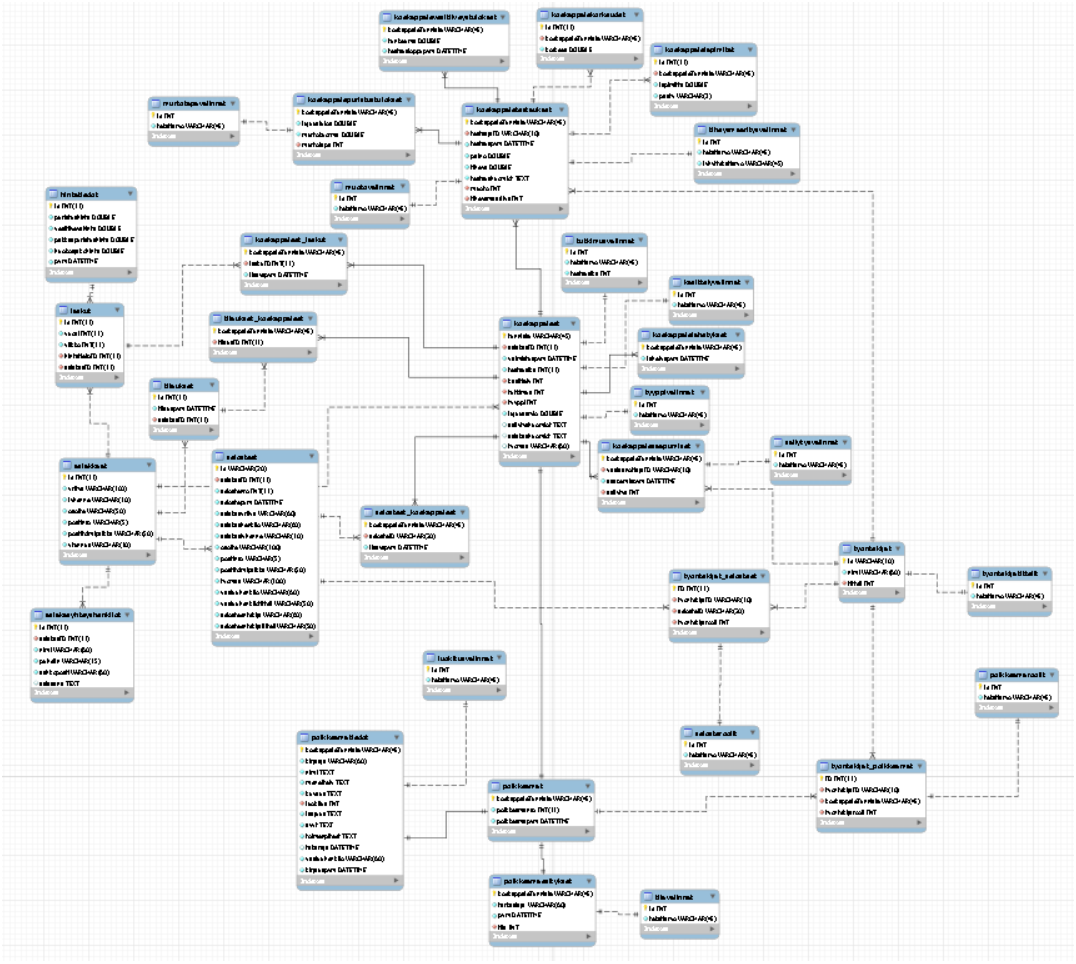
Entity Frameworkin vaihtoehtoiset lähestymistavat mahdollistavat myös erilaisen tavan luoda sovelluksen tietorakenne. Sovelluksen tietorakenteen voi suunnitella ohjelmakoodin rinnalle, josta sovelluksen käyttämä tietokanta sitten luodaan. Entity Frameworkia hyödyntävän sovelluskehittäjän ei myöskään tarvitse huolehtia SQL-kyselyistä, vaan Entity Framework muodostaa ne luodun Entity Data Modelin pohjalta.

## 2.5 Työkalut

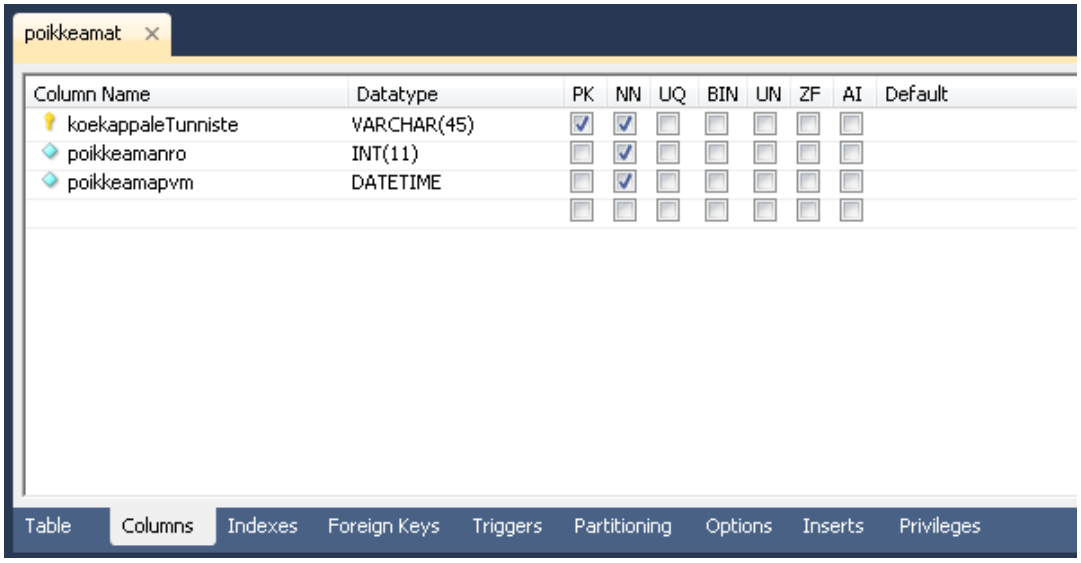
### 2.5.1 MySQL Workbench

MySQL Workbench on useille käyttöjärjestelmille kehitetty työkalu tietokantasuunnittelua, -kehitystä ja ylläpitoa varten (MySQL Workbench 5.2 n.d.). MySQL Workbench on julkaistu sekä kaupallisena että GPL-lisenssin alaisena, eli sitä voi käyttää vapaasti ja ilmaiseksi. Kaupallisessa versiossa on vain muutama ominaisuus, joita ei GPL-lisenssin alaisessa versiossa ole (MySQL Workbench Features n.d.).

MySQL Workbenchin toiminnallisuudesta BETTY-projektin kannalta keskeisimmiksi osoittautui mahdollisuus luoda ER-kaavio, sen takaisinmallinnus sekä tietokannan määrittely ja ylläpito suoraan tietokantapalvelimella. Tietokannan suunnittelu visuaalisen ER-kaavion avulla (ks. kuvio 5) on käytännöllistä isoissakin tietokannoissa, kun suunnittelijalla säilyy kokonaiskuva tietokannan tauluista ja niiden välisistä relaatioista. Myös tietokantataulujen luominen on yksinkertaista ja kenttien ja niiden tietotyyppien ja ominaisuuksien asettaminen hoituu helposti (ks. kuvio 6).



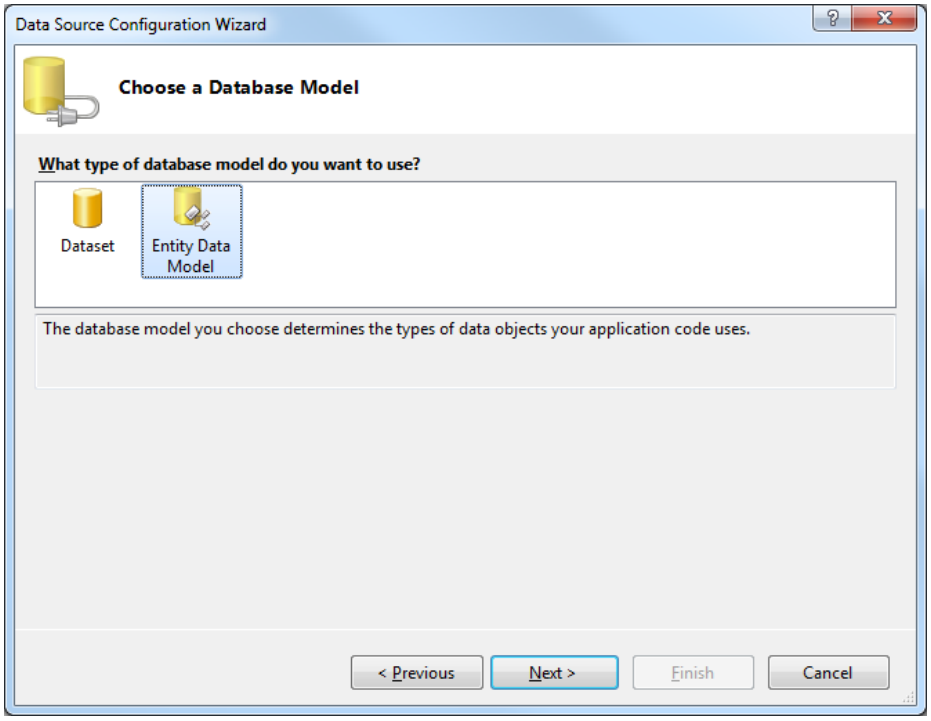
KUVIO 5. Betty-projektin tietokannan ER-kaavio



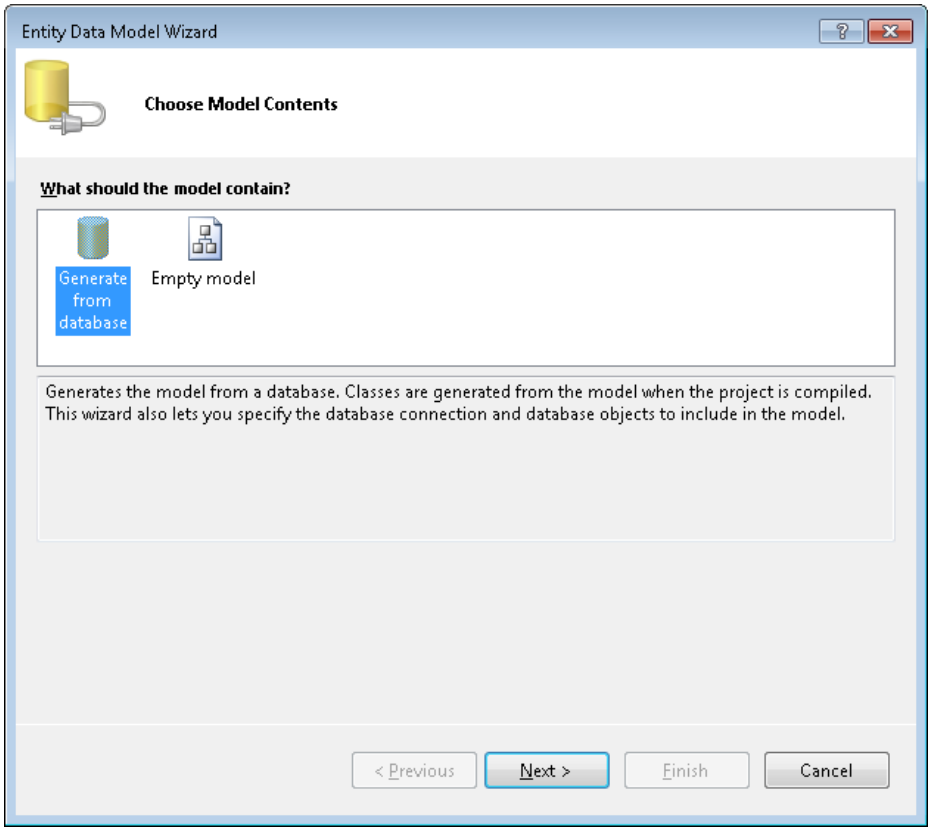

KUVIO 6.Tietokantataulun rakenteen luonti- ja muokkausnäkymä MySQL Workbench-sovelluksessa.

### 2.5.2 Visual Studio 2010 Entity Frameworkin tukena

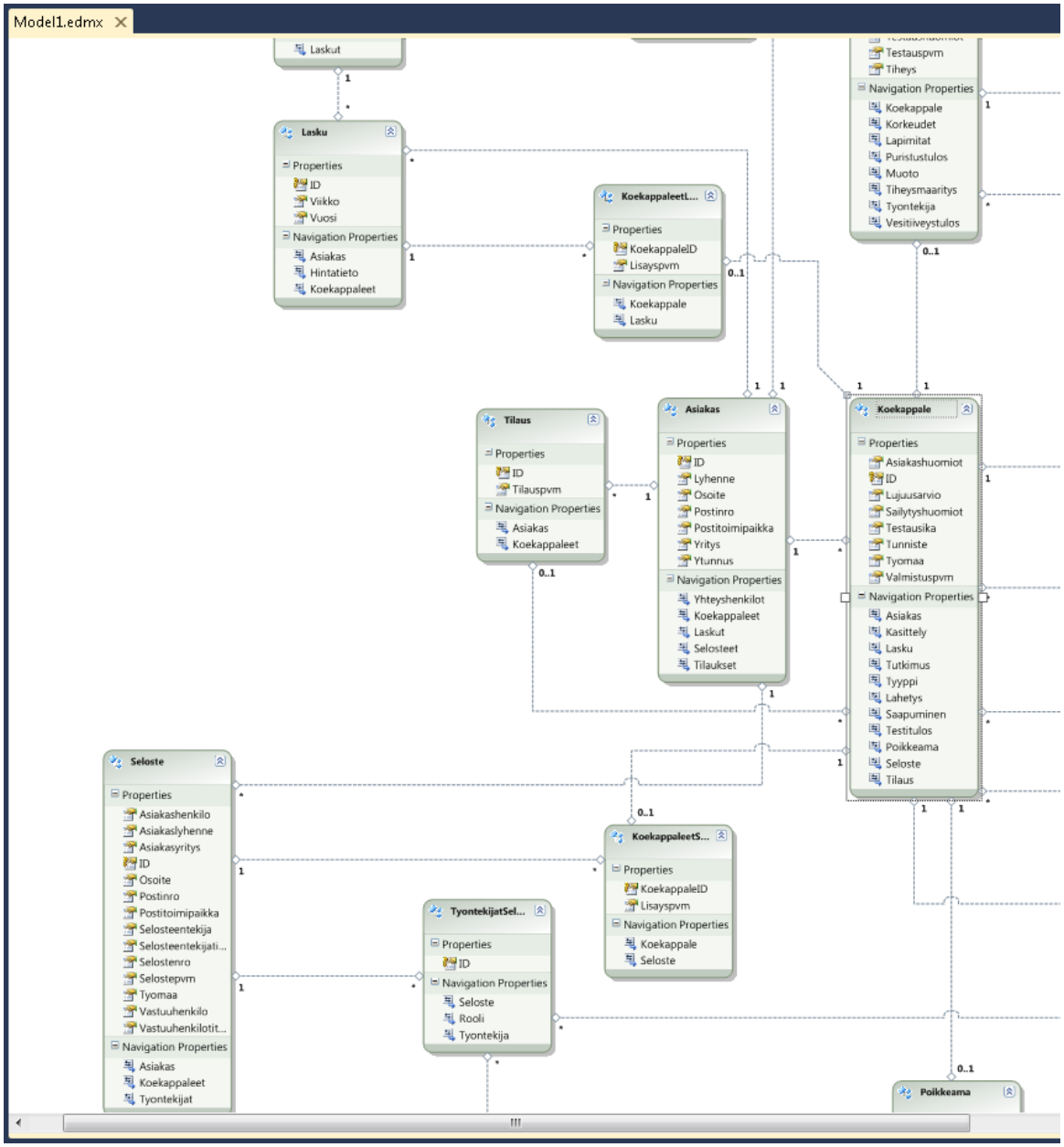
.NET-sovellusten pääkehitysympäristö on Microsoftin Visual Studio. Visual Studio 2010 sisältää Entity Frameworkin kannalta käytännöllisen tietolähteen kokoonpanovelhon, jonka avulla voi luoda EDM:n suoraan tietokannasta. EDM:n luominen velhon avulla ei eroa juurikaan tavasta, jolla ADO.NETin DataSet-oliot luodaan samaisella velholla (ks. kuvio 7). Erona DataSetin luomiseen EDM:ia luodessa uutena vaihtoehtona käyttäjä voi valita, luodaanko malli jo olemassa olevasta tietokannasta, vai haluaako käyttäjä luoda uuden tyhjän mallin (ks. kuvio 8), jonka hän voi suunnitella Visual Studiossa. Olemassa olevasta tietokannasta luodessa sovelluskehittäjä saa käyttöönsä suoraan tietokantatauluista ja niiden välisistä relaatioista luodut entiteetit ja niiden navigaatio-ominaisuudet (ks. kuvio 9).




KUVIO 7. Entity Data Modelin luonti tietolähteen kokoonpanovelhon avulla

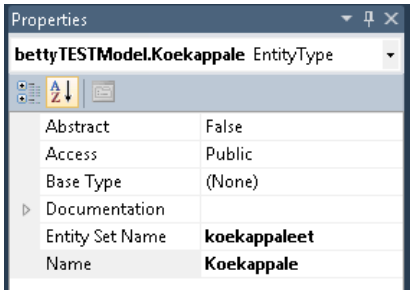



KUVIO 8. Entity Data Modelin sisällön valitseminen, tietokannasta luonti valittuna




KUVIO 9.Kuvankaappaus Visual Studio 2010:n EDM-näkymästä

Visual Studiossa entiteettien muokkaaminen on suhteellisen vaivatonta. Visual Studio näyttää entiteettien ominaisuudet selkeästi (ks. kuvio 10), jolloin myös sovellussuunnittelija saa helposti selville, mihin tietokantatauluun mikäkin entiteetti viittaa. Näkyvien entiteettien nimet on muutettu tietokantataulujen nimistä vastamaan hyviä nimeämiskäytänteitä (Capitalization Conventions n.d.), joissa luokkien nimet aloitetaan isolla kirjaimella ja olio nimetään yksikkömuodossa (ks. kuvio 9).




KUVIO 10. Koekappale-entiteetin ominaisuudet

Visual Studiossa EDM näkyy projektissa *.edmx –päätteisenä tiedostona, ja sen yhteydessä olevana *.Designer.cs –päätteisenä tiedostona. Visual Studiossa .edmx-tiedosto näkyy oletuksena ER-kaaviona (ks. kuvio 9), kun taas jälkimmäinen näkyy tavallisena ohjelmakooditiedostona. Ohjelmakooditiedostossa on jokaisesta entiteetistä luotu oma luokka, joka sisältää konstruktorina toimivan tehdasmetodin, jokaista entiteetin kenttää vastaavan ominaisuuden ja attribuutin, ominaisuuksien muuttamiseen liittyvät tapahtumametodit sekä navigaatio-ominaisuudet. Luokan attribuutit ja ominaisuudet ovat tyypeiltään primitiivisiä, mutta navigaatio-ominaisuudet ovat tyypiltään kompleksisia, tyypillisesti muista entiteeteistä luotuja luokkia.

# 3 ENTITY FRAMEWORKIN JA MYSQL:N YHTEISKÄYTTÖ

## 3.1 Järjestelmävaatimukset

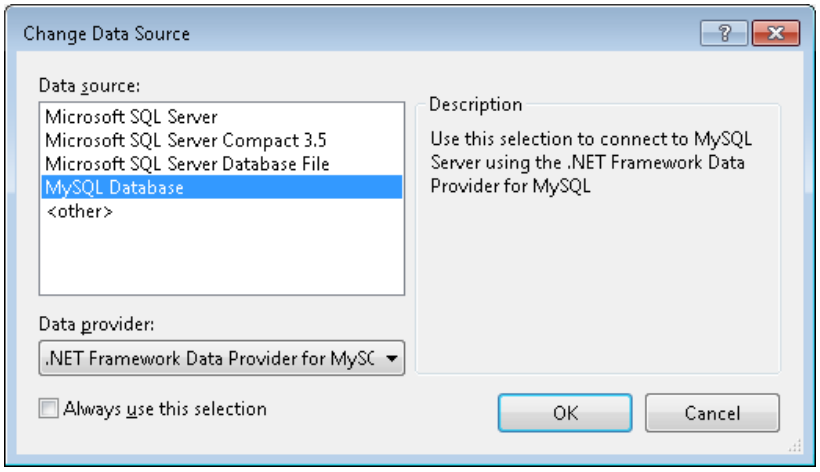
Ensimmäinen askel MySQL -tietokannan ja Entity Frameworkin yhteiskäyttöön on tarvittavien ajureiden ja ohjelmistojen asennus, eli järjestelmävaatimusten täyttäminen. Entity Frameworkin vaatii .NET Framework 3.5 Service Pack 1:n ja Visual Studio 2008 Service Pack 1:n toimiakseen. Suositeltavaa on kuitenkin käyttää Entity Frameworkin versiota 4.0 tai uudempaa ensimmäisen version suurten puutteiden takia (ADO.NET Entity Framework Vote of No Confidence n.d.). Tällöin järjestelmävaatimuksena on .NET Framework 4.0 ja Visual Studio 2010.

.NET Framework ei tue oletuksena MySQL-tietokantojen käyttöä, joten MySQL-tietokantaa käyttävää sovellusta varten on asennettava oikeat ajurit. MySQL tarjoaa tähän tarkoitukseen Connector/Net-nimisen ADO.NET-ajurin, joka mahdollistaa MySQL-tietokantayhteyttä vaativien .NET sovellusten kehittämisen (MySQL Connector/Net n.d.). Kaikki ajurin uusimpien versioiden ominaisuudet, versiosta 6.3.2 eteenpäin, eivät ole käytössä Visual Studio 2010 aiemmissa Visual Studion versioissa, mutta ajuri itsessään vaatii toimiakseen ainoastaan .NET Frameworkin version 2.0 tai uudemman. Huomion arvoista on, että Visual Studion Express-versio voi ottaa yhteyden vain tietokantatyyppiseen tiedostoon, ei palvelimeen (Entity Data Model Wizard n.d.). Täten Connector/Net-ajuri ei toimi Visual Studion Express-versiolla.

## 3.2 Sovelluskehitys

### 3.2.1 Entity Data Modelin käyttöönotto

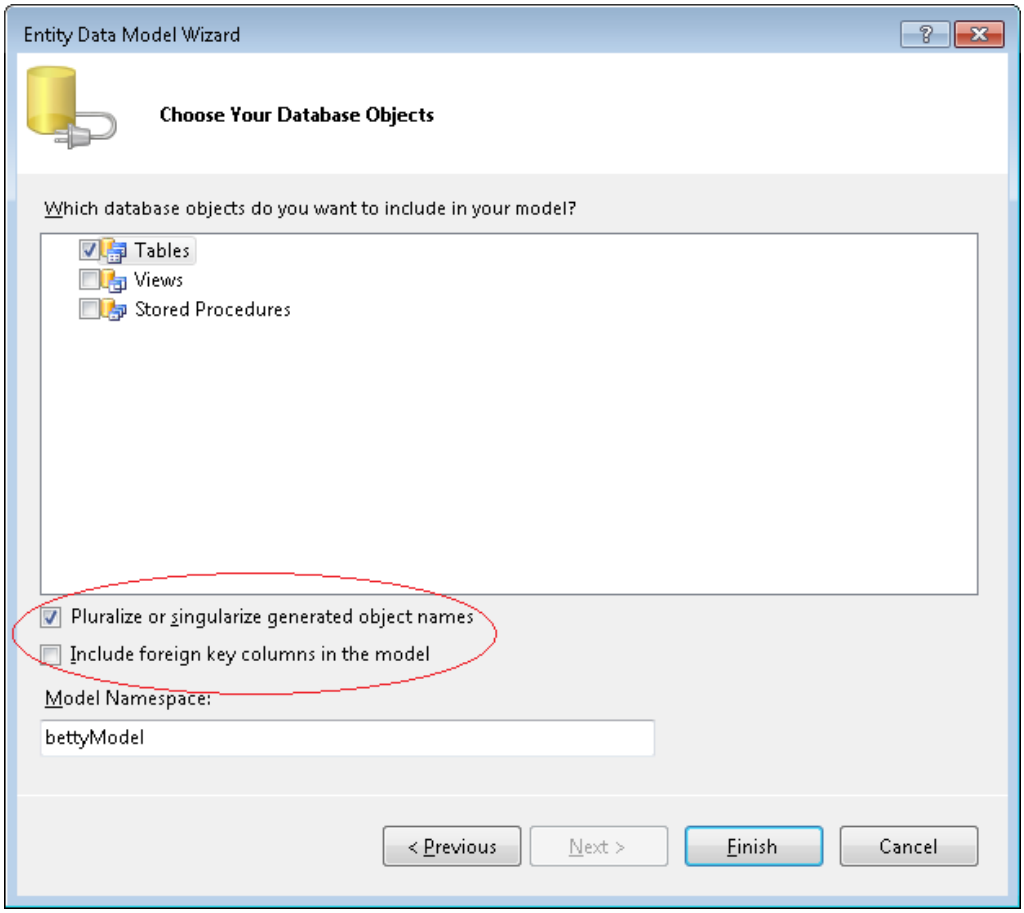
Tietokantaa käyttävän sovelluksen kehityksessä on määriteltävä tietolähde, jos EDM on päätetty luoda tietokannasta (ks. kuvio 8). Kun MySQL Connector/Net –ajuri on asennettu, näkyy tietolähteiden listassa myös MySQL-tietokanta (ks. kuvio 11). Tietolähteen valinnan ja tietokantapalvelimen asetusten määrittäminen ovat ainoat kohdat EDM:n luonnissa, jotka eroavat jonkin verran muilla tietokannoilla.




KUVIO 11. MySQL-tietokanta valittuna tietolähteiden listassa.

Tietokantataulujen nimeämiskäytäntöjä on useita erilaisia. BETTY-projektissa tietokantataulut nimettiin suomeksi, pienillä kirjaimilla ja monikkomuodossa, esimerkiksi koekappaleita sisältävä taulu oli nimeltään ’koekappaleet’. .NET:n nimeämiskäytänteet kuitenkin poikkeavat edellisestä, sillä .NET:ssä käytetään paljon isolla alkukirjaimella alkavia nimiä (Capitalization Conventions n.d.). EDM mahdollistaa kuitenkin erilaisten nimeämiskäytänteiden käytön tietokannassa ja ohjelmakoodissa, sillä tietokantataulujen nimiä ja niiden sarakkeita vastaavat entiteettien nimet ja ominaisuudet voidaan muuttaa vastaamaan .NET:n käytänteitä (ks. kuvio 9). Mikäli tietokantataulut on nimetty englanniksi, voi EDM:n luonnissa määrittää muuttamaan tietokantataulujen nimet yksiköstä

monikkoon ja toisinpäin (ks. kuvio 12), jolloin entiteettien nimet ovat muutetussa muodossa. Suomenkielisillä taulunimillä tämä toiminnallisuus ei toimi.




KUVIO 12. Entity Data Modelin luonnissa entiteettien nimien muuttamisen monikkoon tai yksikköön valittuna, ja vierasavaimia ei sisällytetä EDM:iin mukaan.

EDM:n sisältämistä entiteeteistä on hyvä saada mahdollisimman selkeitä, jolloin sovelluskehityskin on helpompaa. Entiteettien mahdollisesti turhilta ominaisuuksilta vältytään, jos tietokantataulujen vierasavainsarakkeita vastaavat ominaisuuskentät karsitaan automaattisesti pois EDM:n luonnissa (ks. kuvio 12). Tällöin entiteeteistä luoduista luokista tulee johdonmukaisia, ja sovelluskehitys on helpompaa.

### 3.2.2 Esille tulleet ongelmat

Käytettäessä kahta tekniikkaa, jotka tallentavat tietoa erilaisissa formaateissa, ovat ongelmat vähintäänkin odotettavissa. MySQL:n ja .NET Frameworkin tapauksessa aika- ja päivämäärätiedon tallennusformaatti voi olla erilainen, sillä MySQL tukee päivämäärien esitystapoja, joita ei voi esittää semmoisenaan .NETin tietotyypeillä. (Handling Date and Time Information in Connector/Net n.d.) Osaratkaisuna voidaan MySQL:n asetuksista määritellä vain paikkansapitävät päivämäärät kelpaamaan, jolloin esimerkiksi päivämäärä 29.2.2011 ei kelpaa, koska vuonna 2011 ei ollut karkauspäivää. Tällöin täytyy ottaa huomioon muut mahdolliset ympäristöt joista tietokantaa käytetään, ettei niistä pystytä syöttämään virheellisiä päivämääriä tietokantaan. BETTY-projektissa tietokantaa käytetään vain .NET –ympäristöstä, jolloin tämä määritys on MySQL-tietokannassa kannattava.

Connector/Net säilöö oletusarvoisesti välimuistiinsa 25 viimeisintä proseduuria. Käytännössä tämä tarkoittaa sitä, että esimerkiksi yli 25 taulun kokoinen tietokanta ei mahdu välimuistiin. EDM:ia luodessa ongelma tulee esille siten, että kaikki tietokannassa olevat taulut eivät päädy EDM:iin, vaan luotu malli on vajaa. Tällöin luotu malli ei sisällä kaikkia tietokantatauluja vastaavia entiteettejä ja niiden välisiä relaatioita. Ongelman voi kuitenkin ratkaista uudelleenmäärittelemällä välimuistin koon. BETTY-projektissa välimuisti poistettiin käytöstä, kun tietokantatauluja oli käytössä yli 30 kappaletta.

Tietokantamuutoksia tehdessä EDM pysyi muutoksissa mukana, paitsi jos tietokantataulujen pääavaimiin tehtiin muutoksia. Tällöin EDM:n päivittäminen ei riittänyt, vaan se piti luoda täysin uudestaan tietokannasta. Vaikkakin prosessi itsessään oli nopea ja vaivaton, tulivat suurimmat ongelmat esille EDM:n uudelleenluonnin jälkeen. Tällöin entiteetit täytyi nimetä uudelleen, ja tietokantarakenteen päivittäminen osoittautui BETTY-projektissa odotettua työläämmäksi.

Entity Data Modelia päivittäessä sen luomiin entiteettiluokkiin lisätyt dokumentaatiokommentit poistuvat. Jokaisen päivityksen tai uudelleenluonnin yhteydessä EDM:n *.Designer.cs-päätteinen ohjelmakooditiedosto luodaan uudelleen, jolloin entiteeteistä automaattisesti luotujen luokkien ja niiden ominaisuuksien dokumentaatio katoaa. Tämä on ongelmallista siksi, että se poistaa entiteettiluokkien dokumentaation hyödyn lähes kokonaan. Tähän ongelmaan ratkaisua ei löytynyt.

# 4 TULOKSET JA JOHTOPÄÄTÖKSET

EDM:n erilaiset lähestymistavat mahdollistavat tietynlaisen joustavuuden sovelluskehityksessä. Luomalla mallin ensin, tai sovelluskehityksen aikana, ei kulu aikaa tietokantaan tehtäviin muutoksiin, vaan muutokset voi tehdä suoraan EDM:iin. Tällöin muutokset näkyvät välittömästi myös ohjelmakoodissa, ja tarpeellisten muutoksien jälkeen voi mallista luoda tietokannan. Mikäli tietokanta on luotu ensin, on sovelluskehityksen aloittaminen taas vaivatonta, kun sovelluksen tietorakenteen saa suoraan tuotua ohjelmakoodiin, eikä relaatiologiikkaa tarvitse alkaa erikseen rakentamaan.

Tavanomaisilla tavoilla tietokannan muokkaaminen ja sen jälkeen ohjelmistokoodin päivittäminen vastaamaan uutta tietokantarakennetta on työlästä, etenkin jos tietokannan rakenne muuttuu usein. Entity Framework mahdollistaa tietorakenteen helpon ja nopean päivittämisen dokumentoinnin hinnalla, kun entiteeteistä luodut luokat ja niiden dokumentointi generoituu jokaisella päivityskerralla uudestaan. Huomion arvoista kuitenkin, että dokumentoinnin uudelleengeneroiminen koskee vain entiteettien ja niille kuuluvien ominaisuuksien dokumentointia. Koska entiteeteistä luotuja luokkia voidaan laajentaa, esimerkiksi lisäämällä luokkaan metodeja, voidaan laajennukset määritellä EDM:n ohjelmakooditiedoston ulkopuolella, eikä niiden dokumentointi siten kärsi EDM:n uudelleengeneroinnista.

Entity Framework ja MySQL ovat hyvin toimiva yhdistelmä, ja varteenotettava vaihtoehto ilmaista tietokantaohjelmistoa suosiville .NET-sovelluskehittäjille. Entity Framework mahdollistaa tietojärjestelmän nopeamman ja joustavamman kehittämisen, ja koska MySQL tarjoaa hyvän dokumentaation Connector/Net-ajurille, on sitä mieleistä käyttää .NET-sovellusten kehittämisessä. Lisäksi molemmille, sekä Entity Frameworkille että MySQL:lle, tarjotaan hyvät kehitys- ja hallintaympäristöt, jotka tukevat tietojärjestelmän ylläpitoa. Microsoft tarjoaa Entity Frameworkille myös kattavan aloittajan oppaan, jossa käydään läpi ensiaskeleet Entity Frameworkin käyttöönotossa.

# LÄHTEET

ADO.NET Entity Framework At-a-Glance. n.d. Microsoft Developer Network, Data Developer Center. Viitattu 13.11.2011. http://msdn.microsoft.com/en-us/data/aa937709

ADO.NET Entity Framework Vote of No Confidence. n.d. Wufoo.com. Viitattu 30.10.2011. http://efvote.wufoo.com/forms/ado-net-entity-framework-vote-of-no-confidence/

Capitalization Conventions. n.d. Microsoft Developer Network. Viitattu 9.11.2011. http://msdn.microsoft.com/en-us/library/ms229043.aspx

Entity Data Model. n.d. Microsoft Developer Network. Viitattu 31.10.2011. http://msdn.microsoft.com/en-us/library/ee382825.aspx

Entity Data Model Wizard. n.d. Microsoft Developer Network. Viitattu 12.11.2011. http://msdn.microsoft.com/en-us/library/bb399247.aspx

Entity Framework Overview. n.d. Microsoft Developer Network. Viitattu 30.10.2011. http://msdn.microsoft.com/en-us/library/bb399567.aspx

Entity Framework Vote of No Confidence Signatories. 2008. Wufoo.com. Viitattu 30.10.2011. http://efvote.wufoo.com/reports/entity-framework-vote-of-no-confidence-signatories/#public

entity type (Entity Data Model). n.d. Microsoft Developer Network. Viitattu 1.11.2011. http://msdn.microsoft.com/en-us/library/ee382837.aspx

Fancey, J. 2010. Model-First in the Entity Framework 4. Microsoft Developer Network, Data Developer Center. Viitattu 13.11.2011. http://msdn.microsoft.com/en-us/data/ff830362

Handling Date and Time Information in Connector/Net. n.d. MySQL 5.0 Reference Manual. Viitattu 13.11.2011. http://dev.mysql.com/doc/refman/5.0/en/connector-net-programming-datetime.html

Joshi, B. 2011. Introduction to Entity Framework Code First. codeguru.com. Viitattu 11.11.2011. http://www.codeguru.com/csharp/article.php/c19233/Introduction-to-Entity-Framework-Code-First.htm

MySQL Connector/Net. n.d. MySQL 5.0 Reference Manual. Viitattu 11.11.2011. http://dev.mysql.com/doc/refman/5.0/en/connector-net.html

MySQL Connectors. n.d. MySQL.com. Viitattu 2.11.2011.
http://www.mysql.com/products/connector/

MySQL Workbench 5.2. n.d. MySQL.com. Viitattu 2.11.2011.
http://www.mysql.com/products/workbench/

MySQL Workbench Features. n.d. MySQL.com. Viitattu 2.11.2011.
http://www.mysql.com/products/workbench/features.html

.NET Framework 4. n.d. Microsoft Developer Network. Viitattu 30.10.2011.
http://msdn.microsoft.com/en-us/library/w0x726c2.aspx

.NET Framework Conceptual Overview. n.d. Microsoft Developer Network. Viitattu 28.11.2011. http://msdn.microsoft.com/library/zw4w595w.aspx

Sonier, P. 2009. What is a wrapper class? StackOverflow 20.5.2009. Viitattu 29.10.2011.
http://stackoverflow.com/questions/889160/what-is-a-wrapper-class

# LIITTEET

## Liite 1. Koekappalekortti

JYVÄSKYLÄN AMMATTIKORKEAKOULU
TEKNIIKKA JA LIIKENNE
**RAKENNUSLABORATORIO T029**

koekappalekortti

| | | |
|---|---|---|
| asiakas<br>valmistuspäivä<br><br>tunnus<br>testauspäivä | koekappale<br>_____ 150 kuutio<br>_____ valulieriö<br>_____ rakennekpl<br>_____ muu | vastaanotettu<br><br>vastaanottaja |
| testi<br>_____d puristuslujuus ($k_s$ _____) | testin tulos (voima kN) | paino [g] |
| muu testaus | testattu/testaaja/huomiot | mitat [mm]<br>$d_1$<br>$d_2$<br>$d_3$<br>h |
| muoto/säilytys/käsittely/huomiot | | |