SAVONIA

OPINNÄYTETYÖ - **AMMATTIKORKEAKOULUTUTKINTO**
TEKNIIKAN JA LIIKENTEEN ALA

WEB-POHJAINEN TIETO-KANNAN HAKUTYÖKALU

TEKIJÄ/T: Sonja Manninen

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Koulutusohjelma Tietotekniikan koulutusohjelma	
Työn tekijä(t) Sonja Manninen	
Työn nimi Web-pohjainen tietokannan hakutyökalu	
Päiväys 1.9.2015	Sivumäärä/Liitteet 38 / 0
Ohjaaja(t) lehtori Jussi Koistinen, lehtori Sami Lahti	
Toimeksiantaja/Yhteistyökumppani(t) TCD Consulting and Research Oy	

Tiivistelmä

Opinnäytetyön aiheena oli tehdä työkalu, jonka avulla voi luoda ja suorittaa hakulauseita käyttöliittymän kautta. Sovellus kehitettiin Consulting and Research Oy:lle.

Työkalu muodostaa valitun tietokannan rakenteen visuaalisena käyttöliittymään ja luo hakulauseen käyttäjän valintoihin pohjautuen. Hakutyökalun pääasiallinen tarkoitus on mahdollistaa hakulauseiden muodostaminen myös käyttäjille, joilla ei ole SQL-osaamista.

Tämän työn tavoitteena oli kehittää helppokäyttöinen ja toimiva sovellus, jonka avulla kuka tahansa kykenee suorittamaan tietokantahakuja. Työ rakennettiin Microsoftin Visual Studio:lla ohjelmistokehyksenä ASP.NET MVC, hyödyntäen erilaisia Javascript- kirjastoja. Tietokantapalvelimena toimi Microsoftin SQL Server 2014. Käyttäjän tunnistamiseen käytettiin Windows- tunnistautumista. Hakutyökalun lisäksi tehtiin hallintasivusto, jonka kautta pääkäyttäjä määrittää käytettävät tietokantayhteydet sekä roolien oikeudet luotuihin yhteyksiin.

Työn tuloksena kehitettiin vaatimusmäärittelyn mukainen sovellus aikataulun mukaisesti. Työ oli yrityksen näkökulmasta onnistunut.

Avainsanat
MVC, ASP.NET, C#, SQL server

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author(s) Sonja Manninen			
Title of Thesis Web-Based Database Query Tool			
Date	1 September 2015	Pages/Appendices	38 / 0
Supervisor(s) Mr Jussi Koistinen, Lecturer, Mr Sami Lahti, Lecturer			
Client Organisation /Partners TCD Consulting and Research Oy			

Abstract

The purpose of this thesis was to build a tool to create and execute queries through the interface. The application was developed for a company called TCD Consulting and Research Oy.

The tool shows the visual presentation of the chosen database structure and generates the query based on the user's choices. The main purpose of this tool is to enable querying to users without any knowledge of SQL.

The practical goal of this project was to build a user friendly and practical tool for anyone to excute queries. The project was built using Visual Studio with the ASP.NET MVC development framework and utilizing some JavaScript libraries. SQL Server 2014 was the database server. The tool uses Windows authentication. Alongside the query tool, there is also an administrator panel. The administrator defines database connections for the application and role permissions for these connections.

The result of this thesis was that the requirements of the project was completed on time and the corporation was pleased with the end product.

Keywords
MVC, ASP.NET, C#, SQL Server

SISÄLTÖ

1 JOHDANTO

Opinnäytetyön aiheena on luoda web-pohjainen tietokannan hakutyökalu TCD Consulting and Research Oy:lle. Työkalu näyttää tietokannan rakenteen visuaalisena käyttäjälle ja luo hakulauseen käyttäjän tekemien valintojen mukaan. Käyttäjä valitsee taulurakenteesta haluamansa kentät. Lisäksi hän voi määrittää hakuehtoja ja aggrekaattifunktioita taulurakenteesta. Hakulauseen muodostamisen jälkeen käyttäjä voi suorittaa haun ja hänelle näytetään hakutulos käyttöliittymässä.

Työ toivotaan rakennettavan Visual Studiolla ohjelmistokehyksenä ASP.NET MVC5. Lisäksi kriteereihin kuuluu, että työ olisi täysin dynaaminen, jolloin sitä voi käyttää minkä tahansa tietokannan kanssa. Työkalun pääasiallinen tarkoitus on mahdollistaa tietokantahaut käyttäjille, joilla ei ole tietämystä SQL:stä.

Lisäksi kokonaisuuteen kuuluu hallintapuoli, jossa pääkäyttäjä voi määrittää työssä käytettävät tietokantayhteydet. Lisätyille yhteyksille määritetään käyttöoikeudet roolien mukaan. Tietokannan hakutyökalu käyttää käyttäjien tunnistamiseen Windows-tunnistautumista. Työn tavoitteena on rakentaa helppokäyttöinen ja toimiva sovellus.

2 KÄYTETYT TEKNIIKAT JA TYÖKALUT

Työ on MVC-mallin mukaan rakennettu verkkosivusto .NET-ympäristöön. Ohjelmistokehitysympäristönä käytettiin Microsoftin Visual Studiota. Ohjelmointikielenä on pääasiassa C# ja tietokantana toimii Microsoftin SQL Server.

C#

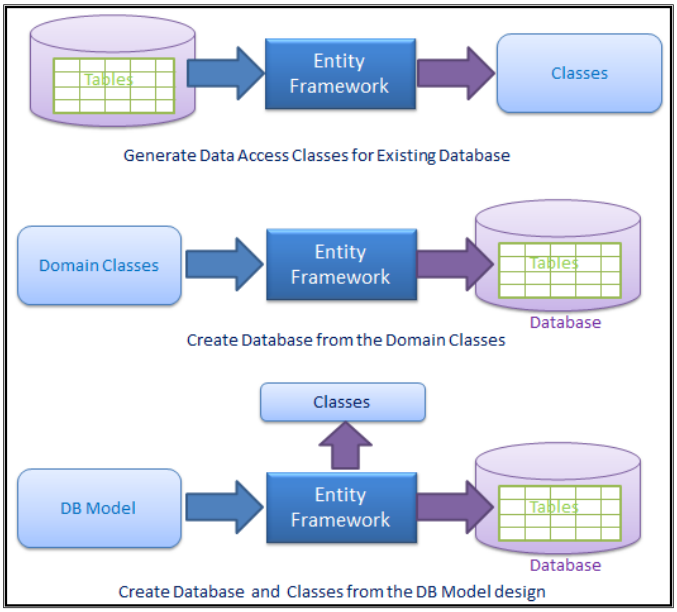
C# on oliopohjainen ohjelmointikieli, joka perustuu C++-kieleen ja sisältää myös piirteitä Java-kielestä. Se on kehitetty .NET:ä varten. (Moghadampour 2009, 14.) C# on opinnäytetyön pääasiallinen ohjelmointikieli.

Entity Framework

Entity Frameworkin avulla voidaan käsitellä tietokannan dataa olioimaisesti. Tietokannan tietoihin pääsee käsiksi LINQ-kyselyiden avulla. Kuvassa 1 esitellään erilaisia käyttötapauksia. (What is Entity Framework?)

LINQ

Language Integrated Query on .NET-sovellluskehyksen komponentti, jonka avulla voidaan suorittaa datakyselyitä .NET-ohjelmointikielessä. (What is Entity Framework?)



KUVA 1. Luokkarakenne voidaan luoda jo olemassa olevista tauluista tai luoda taulut luokista (What is Entity Framework?)

HTML (HyperText markup language)

HTML:llä luodaan web-sivujen sisältö ja visuaalisuus, esimerkiksi teksti ja kuvat. Se kuvastaa siis web-sivuston sisältöä, mutta ei toiminnallisuuksia. (HTML)

JavaScript ja jQuery

Javascript on oliopohjainen kieli, jota käytetään usein web-ympäristössä. Sitä käytetään yleensä lisäämään dynaamista toiminnallisuutta sivustolle. (What is JavaScript, really...?) jQuery on helposti ymmärrettävä avoimen lähdekoodin Javascript-kirjasto. Sen avulla voidaan kirjoittaa JavaScriptiä lyhyemmin ja selkeämmin. (What is jQuery)

Ajax

Ajax on lyhenne sanoista Asynchronous JavaScript and XML. Opinnäytetyössä Ajaxia on hyödynnetty vaihtamaan dataa palvelimen kanssa niin, ettei koko sivua tarvitse päivittää uudelleen.

ASP.NET

ASP.NET on ohjelmistokehys, jonka avulla voi luoda dynaamisia web-sivuja. Se tukee kolmea erilaista kehitysmallia: Web pages, MVC sekä Web forms (ASP.NET MVC Tutorial).

MVC

MVC on ohjelmistoarkkitehtuurityyli, joka jakaa ohjelmaan kolmeen osaan, Model-View-Control. MVC on yksi ASP.NET:n ohjelmointimalleista. Model (malli) edustaa sovelluksen ydintä. Se hakee sovelluksen datan ja varastoi sen myös takaisin tietokantaan. View (näkymä) on vastuussa datan näyttämisestä sivustolla. Controller (käsittelijä) toimii välikätenä mallin ja näkymän välissä. Se käsittelee mallin datan ja lähettää sen näkymään. (ASP.NET MVC Tutorial)

SQL

SQL (Structured Query Language) on ohjelmointikieli, jota käytetään datan käsittelyyn tietokannassa.

SQL Server

Microsoftin SQL Server on tietokantojen hallintajärjestelmä.

Aggregate function

Aggrekaattifunktioilla voidaan suorittaa laskentaa hakulauseen kenttiin. Aggrekaattifunktio palauttaa luvun. Sen avulla voidaan laskea esimerkiksi summa tai rivien lukumäärä tietyn kentän mukaan.

Active Directory

Active Directory on Microsoftin keskitetty hakemistopalvelu, joka automatisoi tietoverkon hallinnan. (Active Directory definition) Active Directory Domain Services on taas palvelinrooli Active Directoryssa. Sen avulla voidaan määrittää Active Directoryn käyttäjien ja tietokoneiden oikeuksia ja ryhmitellä niitä. (Microsoft Active Directory Domain Services..)

3 SUUNNITTELU JA MÄÄRITTELY

TCD Consulting and Research Oy on perustettu vuonna 2013. Yritys on erikoistunut tiedon keräämiseen ja analysointiin. Yrityksellä oli tarve työkalulle, jonka avulla tavallinen käyttäjä voi hakea dataa tietokannasta.

3.1 Työn suunnittelu

Työ haluttiin tehtävän MVC-mallin mukaisena ASP.NET sivustona. Työtä saatiin suunnitella melko vapaasti itse. Työn kehittämisen aikana opiskeltiin, mikä olisi hyvä toteutustapa ominaisuuksille.

Aikataulu määriteltiin kahden viikon mittaisiin jaksoihin. Vaatimusmäärittelyn mukaiset tehtävät jaettiin tasaisesti jaksoihin. Työ oli laaja kokonaisuus, joten aikataulun avulla se pysyi helpommin hallinnassa.

3.2 Määrittely

Taulukossa 1 esitellään hakutyökalun vaatimusmäärittely. Arvon kolme toiminnot ovat tärkeimpiä. Toiminnot on jaettu kolmeen kokonaisuuteen: hallintapuoli, hakutyökalu sekä tietojen ulosvienti. Työn kehittäminen aloitettiin pääasiassa ensin hallintapuolesta työstämällä samanaikaisesti myös hakutyökalua. On järkevää rakentaa hallintapuoli ensin, jotta varsinaisen hakutyökalun puolella voidaan käyttää oikein määritetyttyjä tietokantayhteyksiä.

Työn edetessä pidettiin säännöllisesti yhteyttä yrityksen kanssa sähköpostitse. Lisäksi välillä pidettiin palavereja. Suunnitelmat elivät hieman työn edetessä, kun saatiin uusia ideoita tai todettiin, että jokin lähestymistapa oli liian hankala. Sovellusta rakennettaessa oli tärkeintä pitää mielessä, että käyttäjä ei välttämättä tiedä hakulauseista tai tietokantarakenteesta mitään. Työn piti olla niin helppokäyttöinen ja itsestäänselvä, että sitä pystyy käyttämään kuka tahansa.

TAULUKKO 1 Projektin vaatimusmäärittely alkuperäisessä muodossaan

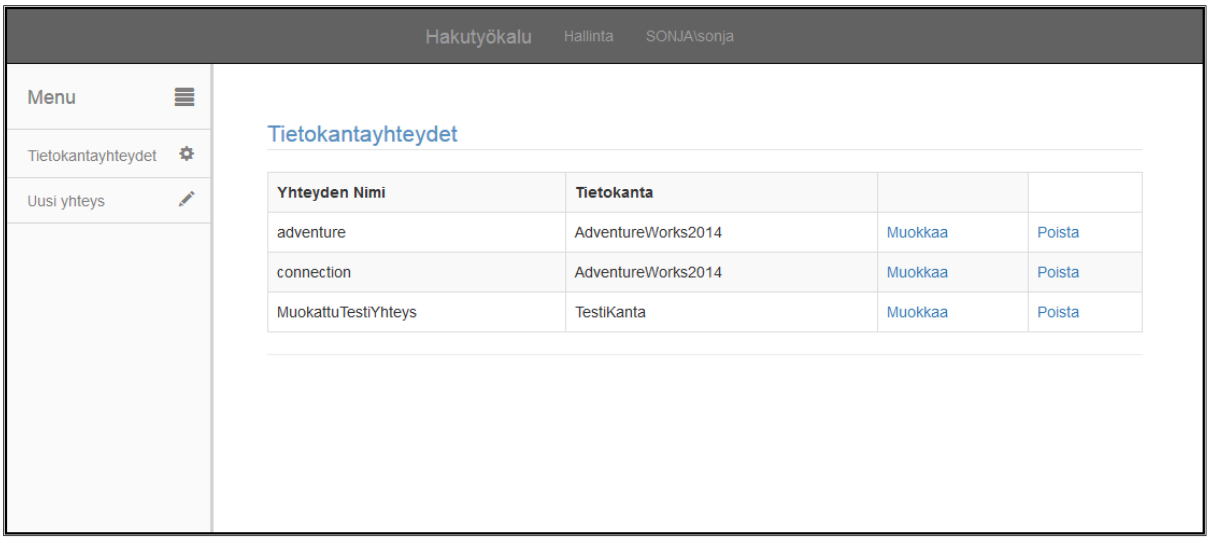
Kokonaisuus	Toiminto	Tärkeys
Hallintapaneeli	Windows authentication	3
	Windows roolien määrittely	3
	Roolien oikeudet yhteyksiin	2
	Tietokantayhteydet	3
Hakutyökalu	Tietokannan rakenteen visuaalinen esittäminen (relaatiomalli)	3
	Haettavien kenttien poiminta	3
	Taulujen välinen liitostyyppi	2
	Hakutuloksen esittäminen käyttöliittymässä	3
	Hakulauseen tallentaminen roolikohtaisesti	2
	Generoidun hakulauseen näyttäminen	1
	Tallennettujen hakujen selaaminen	2
	Hakutuloksen rajaaminen	3
	GROUP BY hakulauseen tekeminen	2
	COUNT haun tekeminen	2
	SUM/AVG haun tekeminen	1
	Useampi hakuehto	2
	Hakutuloksen tiedon korvaaminen (BIT 1 muunnetaan tuloksessa ”Kyllä”)	1
Tietojen ulos-vienti	Haun tulos esitetään käyttöliittymässä	3
	Haun tiedot voidaan tallentaa CSV/Excel muodossa	3

4 HALLINTAPUOLI

Hallintapuolen käyttöoikeus on ainoastaan pääkäyttäjällä. Tämä osio sovelluksesta on tarkoitettu peruskäyttäjien tietokantayhteyksien määrittämiseen.

4.1 Pääkäyttäjän näkymä

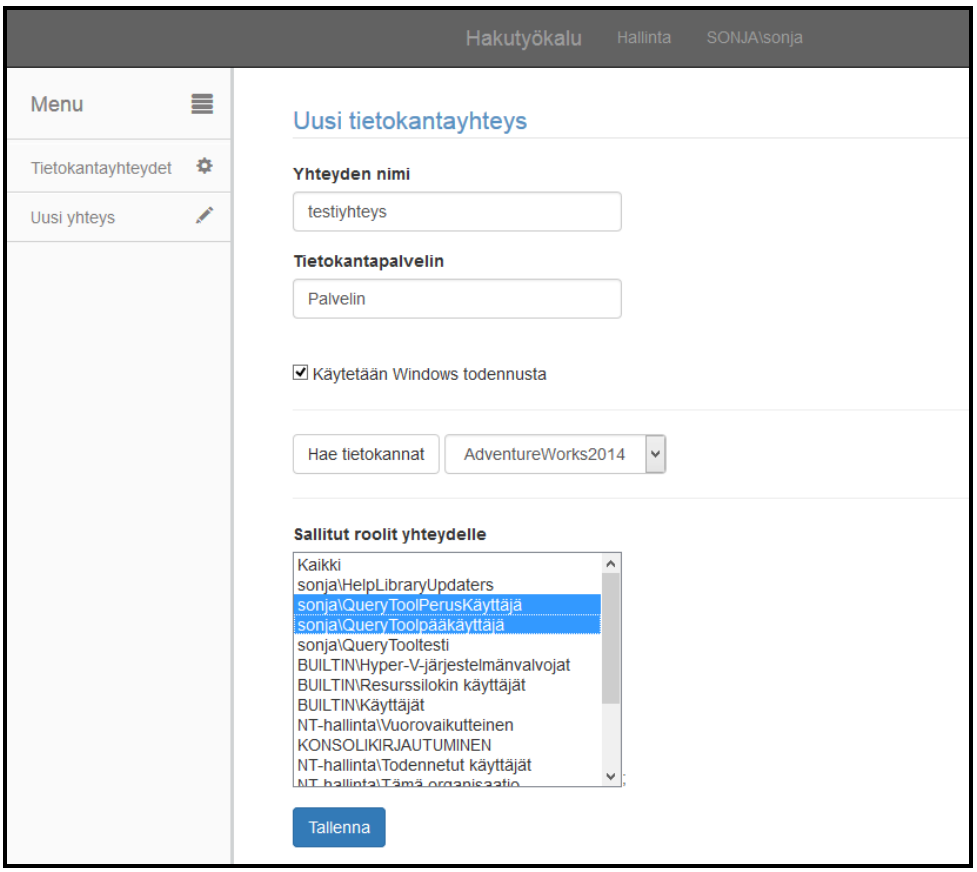
Pääkäyttäjä määrittää käytettävät tietokantayhteydet ja roolien oikeudet näihin yhteyksiin hallintapuolella. Hallintapuoli koostuu yhteyksien listauksesta, lisäämisestä, muokkaamisesta sekä poistamisesta. Hallintapuolen pääsivulla näytetään listaus jo luoduista yhteyksistä. Pääsivulla pääkäyttäjä voi määrittää, haluaako hän muokata tai poistaa jonkin näistä yhteyksistä. Lisäksi on tietenkin mahdollisuus luoda uusia yhteyksiä. Kuvassa 2 esitellään tietokantayhteyksien listaus.



KUVA 2 Tietokantayhteyksien listaus näytetään alkunäkymänä hallintasivustolla

Kuvassa 3 esitellään uuden yhteyden lisäämisen lomake. Uutta yhteyttä luotaessa on ensin määritettävä, mitä tietokantapalvelinta käytetään. Lisäksi yhteyden luomiseen tarvitaan tieto, otetaanko palvelimelle yhteys Windows-tunnistautumisen avulla vai erillisillä tunnuksilla. Luonnollisesti myös tunnukset on annettava, mikäli Windows-tunnistautumista ei käytetä.

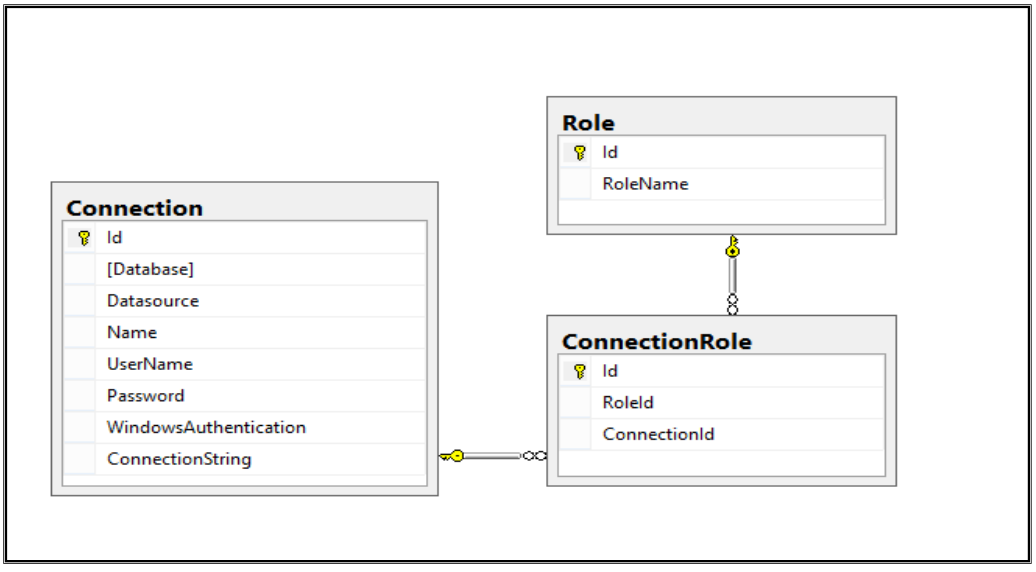
Tietokantapalvelimen määrittämisen jälkeen haetaan alasvetolistaan ne tietokannat, jotka tältä palvelimelta löytyvät. Lopuksi pääkäyttäjä määrittää, kenellä on käyttöoikeus uuteen yhteyteen. Sovellus käyttää Windows-tunnistautumista käyttäjien tunnistamiseen, joten käyttäjäryhmät haetaan Active Directoryn toimialuepalveluista.



KUVA 3 Uudelle tietokantayhteydelle määritetään käyttöoikeudet

4.2 Hallintapuolen toiminta käytännössä

Yhteyksien hallintaan rakennettiin oma tietokanta, joka pitää sisällään kaikki yhteydet ja niihin kuuluvat roolit. Yhteys tähän tietokantaan on tallessa projektin Web.config-tiedostossa. Kaikki pääkäyttäjän määrittämät yhteydet haetaan suoraan tästä tietokannasta hakutyökalua varten. Kuvassa 4 esitellään tietokannan rakenne.



KUVA 4 Microsoft SQL Serverin diagrammi hallintapuolen tietokannasta sisältää kolme taulua

Tietokannan tauluista luotiin omat mallit (Model). Lisäksi rakennettiin vielä erillinen luokka, jota kutsutaan nimellä DbContext. Tämän luokan avulla tietokannan dataa voidaan käsitellä oliomaisesti. Datalle voidaan siis suorittaa CRUD-operaatiot. CRUD-operaatioilla tarkoitetaan Luo (Create), Lue (Read), Päivitä (Update) sekä Poista (Delete) -operaatioita. Entity FrameWorkin avulla kaikki toiminnot tietokantaan pystyttiin toteuttamaan muutamalla lauseella.

Yhteys- ja roolitaulun tiedoista luotiin erillinen malli, joka yhdistää näiden kahden tiedot. Yhdistävä malli luotiin, koska näkymä (View) tarvitsee molempien taulujen tietoja lomakkeelle. Tämän tyyppistä luokkaa kutsutaan näkymän malliksi (ViewModel). Sen avulla voidaan luoda mallista/malleista kustomoitu versio, joka pitää sisällään vain ne tiedot, joita näkymässä tarvitaan. Näin voidaan helposti palauttaa lomakkeen tiedot takaisin käsittelijälle (Controller) ja käsittelijältä alkuperäisen mallin (Model) kautta tietokantaan. Kuvassa 5 esitellään näkymän mallin rakenne.

```
using System.ComponentModel.DataAnnotations;

namespace QueryTool.Models
{
    public class CreateConnectionViewModel
    {
        public Connection Connection { get; set; }
        public long Id { get; set; }

        [Required]
        public string[] Roles{ get; set; }

    }
}
```

KUVA 5 Näkymän malli pitää sisällään Connection-luokan kaikki ominaisuudet

Malli pitää sisällään oman uniikin id:n, yhteystaulun kaikki ominaisuudet ja roolitaulusta vain taulukolistauksen rooleista. Kuvassa 6 esitellään esimerkkinä koodi, jossa luodaan näkymän malli olemassa olevasta datasta tietokannassa. Ensin haetaan id:n mukaisen yhteyden tiedot sekä lisätään siihen kuuluvat roolit. Tämän avulla saadaan esitäytettyä lomake, kun pääkäyttäjä aikoo muokata yhteyttä.

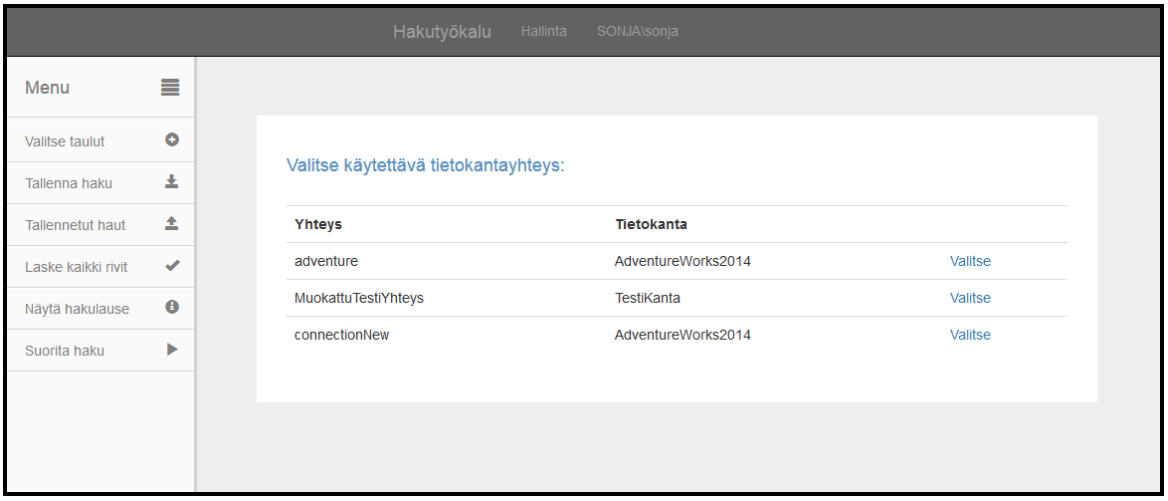
```
public CreateConnectionViewModel GetViewModel(int conId)
{
    CreateConnectionViewModel model;
    using (var db = new RoleContext())
    {
        Connection con = db.Connection.Find(conId);

        model = new CreateConnectionViewModel { Connection = con };

        model.Roles = (from c in db.Connection
                       join cr in db.ConnectionRole on c.Id equals cr.ConnectionId
                       join r in db.Role on cr.RoleId equals r.Id
                       where (c.Id == conId)
                       select r.RoleName).ToArray();
    }

    return model;
}
```

KUVA 6 Näkymän mallin luominen tehdään tietokantayhteyden id:n mukaan

Lomake validoidaan asynkronisesti JavaScriptin avulla, jolloin sivua ei tarvitse ladata uudelleen, jos lomakkeen lähetys epäonnistuu. Käyttäjälle annetaan virheilmoitus, mikäli validointi epäonnistuu. Tarjolla olevien tietokantojen hakeminen tapahtuu myös JavaScriptin Ajax-kutsujen avulla. Jos kaikki tiedot ovat oikein, lomake lähetetään käsittelijälle (Controller), jossa suoritetaan tallennus tietokantaan.

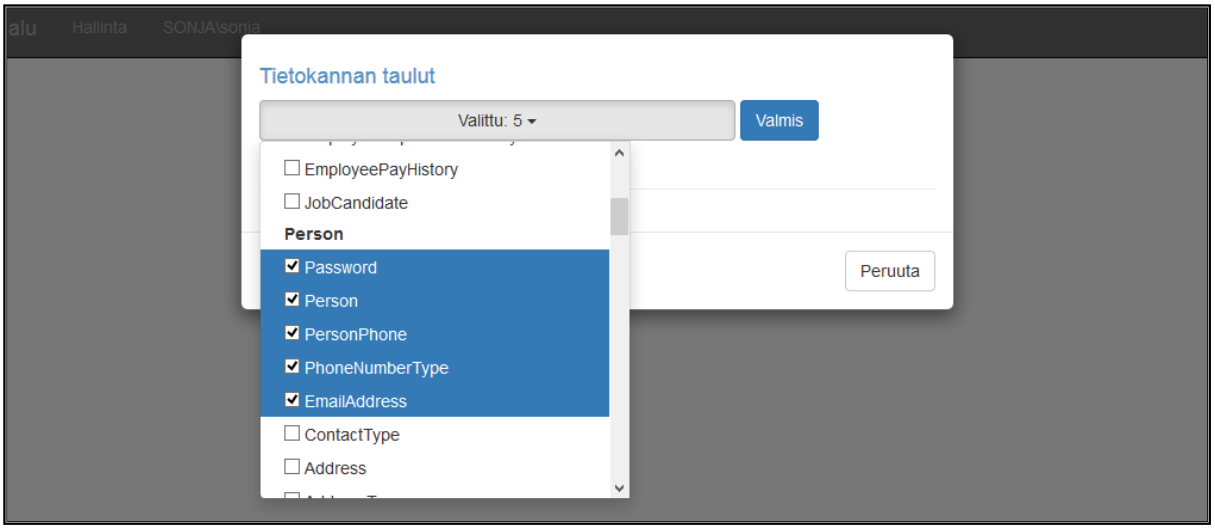
5 HAKUTYÖKALU KÄYTTÄJÄN NÄKÖKULMASTA

Kun peruskäyttäjä avaa sovelluksen, ensimmäiseksi tunnistetaan käyttäjä ja haetaan näytölle sen mukaisesti rooleille kuuluvat tietokantayhteydet. Käyttäjä valitsee haluamansa yhteyden kuvan 7 mukaisesti.



KUVA 7 Tarjolla olevat yhteydet riippuvat käyttäjän roolista

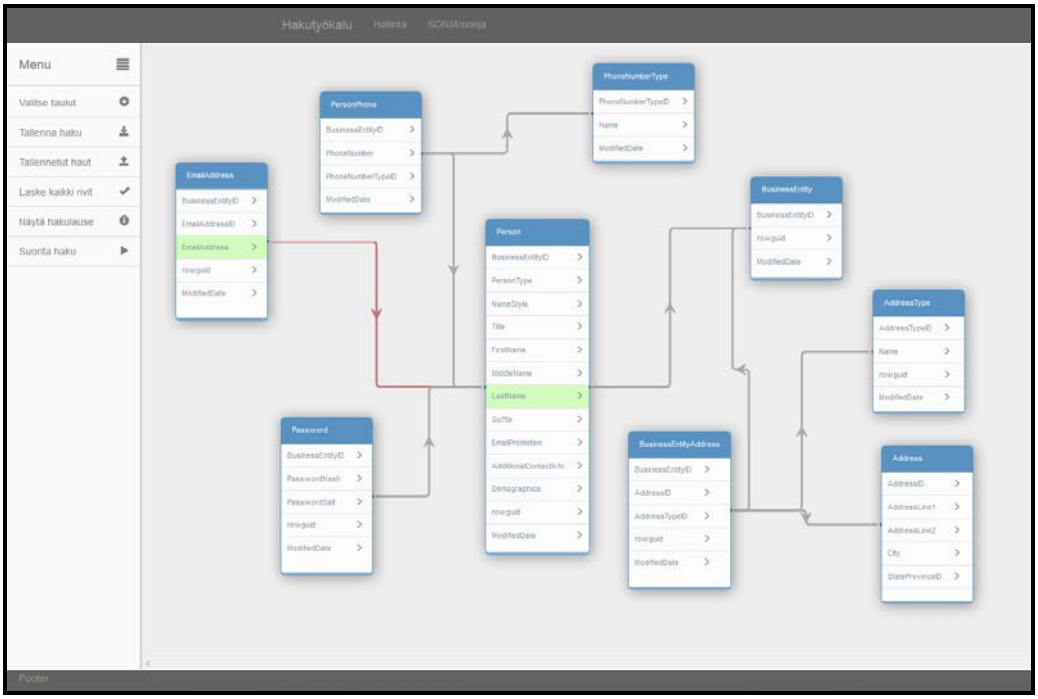
Tämän jälkeen tulee valita, mitkä taulut halutaan tietokantarakenteeseen mukaan kuvan 8 mukaisesta dialogista. Tietokannan taulut valitaan itse, koska niitä voi olla tietokannassa jopa satoja. Tämän vuoksi ei ole järkevää muodostaa automaattisesti koko rakennetta näytölle. Käyttäjä todennäköisesti tarvitsee hakulauseeseensa vain murto-osaa tietokannan tauluista. Jotta tauluille saadaan mahdollisimman paljon tilaa käyttöliittymään, myös sivuvalikon pystyy piilottamaan menu-kuvaketta klikkaamalla.



KUVA 8 Valitun tietokantayhteyden mukaisesti haetaan kaikki taulut valittavaksi

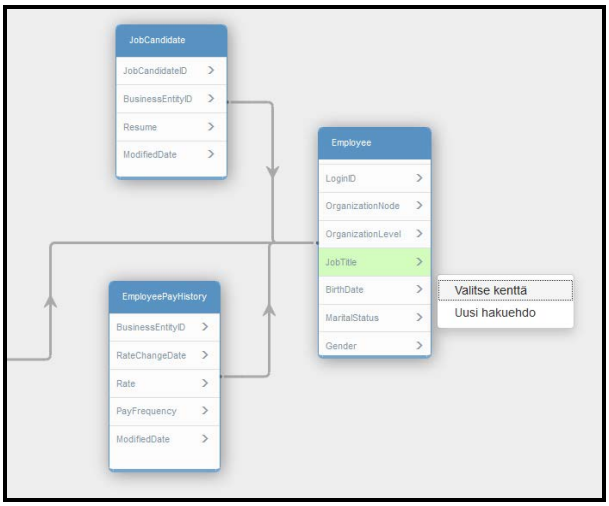
5.1 Kenttien valitseminen hakulauseeseen

Taulujen valitsemisen jälkeen luodaan taulut sekä niiden väliset yhteydet näytölle. Jos dataa on paljon, taulun rivejä voi kelata. Tauluja on mahdollista myös liikutella sekä venyttää korkeammaksi. Taulut luodaan vierekkäin näytölle, jonka jälkeen ne vedetään irralleen toisistaan, jotta taulujen väliset liitokset on helpompi nähdä. Kuvassa 9 esitellään näkymä sen jälkeen, kun käyttäjä on asetellut taulut näytölle ja muuttanut niiden oletuskokoa.



KUVA 9 Valitut taulut on generoitu käyttöliitymään ja on valittu pari kenttää

Käyttäjä voi valita haluamansa kentän klikkaamalla kentän nimeä ja valitsemalla sen aukeavasta valikosta. Samaan tapaan hän voi valita kentän poistettavaksi hakulauseesta sekä lisätä hakuehdon tai aggrekaattifunktion kentän mukaan. Kuvassa 10 on valittu jo yksi kenttä, jonka taustaväri ilmoittaa valinnasta. Käyttäjä on klikannut toista kenttää avatakseen valikon.



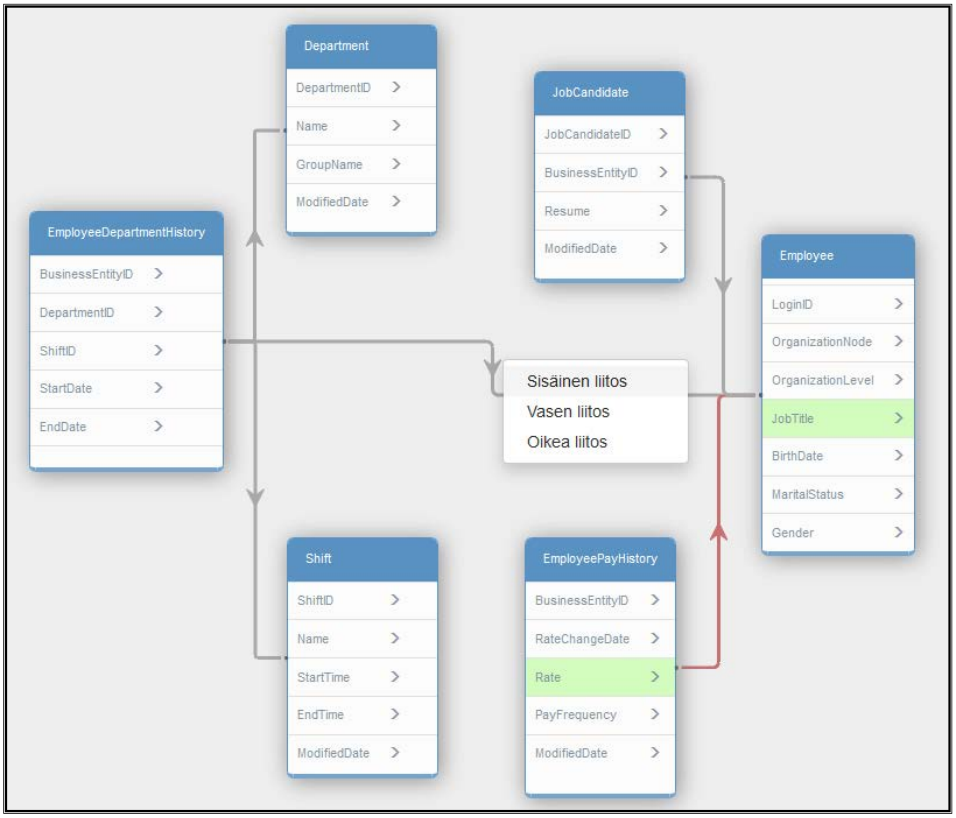
KUVA 10 Employee-taulusta ollaan valitsemassa BirthDate-kenttää

5.2 Taulujen väliset liitokset

Mikäli halutut taulut eivät ole suorassa yhteydessä toisiinsa, niiden välille on määriteltävä liitos. Jos taulut ovat suoraan yhteydessä toisiinsa, liitos määritellään automaattisesti. Kuvassa 11 liitos on määritelty automaattisesti kahden vierekkäisen taulun välille, koska molemmista on valittu kenttä.

Taulujen välistä yhteysviivaa klikkaamalla voidaan joko määrittää liitos uuteen tauluun, tai muuttaa liitostyyppiä minkä tahansa jo valittujen taulujen välillä. Liitosviivoissa vierasavaimen omaava taulu osoittaa nuolella tauluun, josta avain on kotoisin. Pääasiallinen tarkoitus nuolelle on yksinkertaisesti helpottaa viivan klikkaamista käyttöliittymässä taulujen välisiä yhteyksiä muokattaessa.

Samassa kuvassa määritetään yhteyttä Employee-taulun ja EmployeeDepartmentHistory-taulun välille. Jos haluttujen taulujen välissä on useampia tauluja, määritellään liitokset kaikille välitauluille.

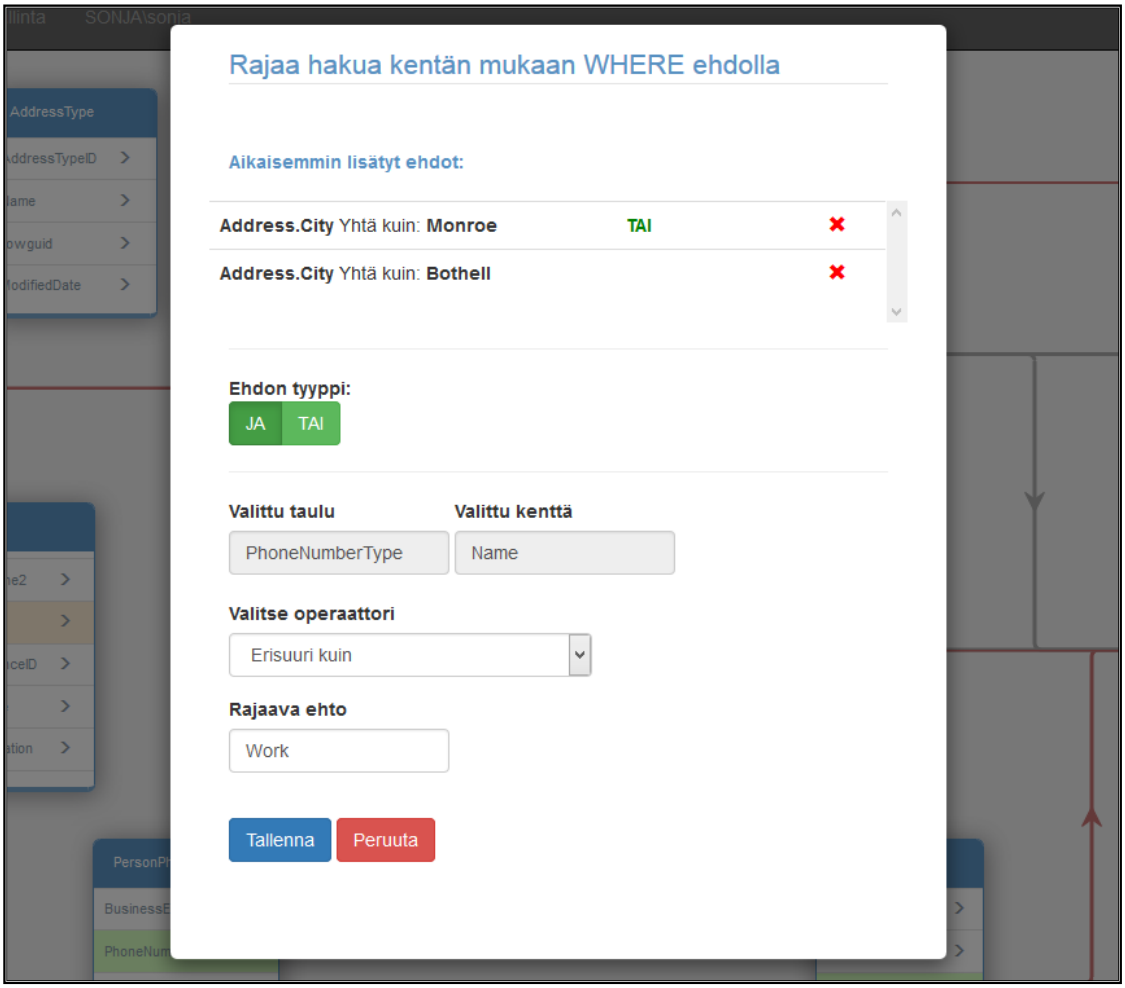


KUVA 11 Liitoksen määrittäminen kahden taulun välille

Oletuksena käytetään sisäistä liitosta (INNER JOIN), joka valitsee siis vain ne tulokset, joissa molemmista tauluista löytyy vastine sarakkeiden välillä. Muita vaihtoehtoja ovat vasen liitos (LEFT JOIN, valitsee kaikki vasemmasta taulusta ja oikeasta ne jotka täsmää vasempaan) sekä oikea liitos (RIGHT JOIN, valitsee taas kaikki oikeasta taulusta, ja vasemmasta vain ne jotka täsmää oikeaan).

5.3 Haun rajaaminen ja aggrekaattifunktiot

Hakulauseeseen on mahdollista lisätä erilaisia rajauksia (WHERE ehto). Taulun, jonka kentän mukaan hakuehto asetetaan, täytyy olla mukana hakulauseessa. Jos kyseisestä taulusta ei haluta mitään kenttiä mukaan hakulauseeseen, taulu vain liitetään mukaan määrittämällä liitos siihen. Hakuehdon lisääminen tapahtuu saman valikon kautta kuin kentän lisääminen. Tällöin aukeaa dialogi, jossa voidaan määrittää haluttu ehto. Kuvassa 12 on lisätty jo kaksi ehtoa Address-taulun kenttään City. Hakuehdot poistetaan myös saman dialogin kautta.

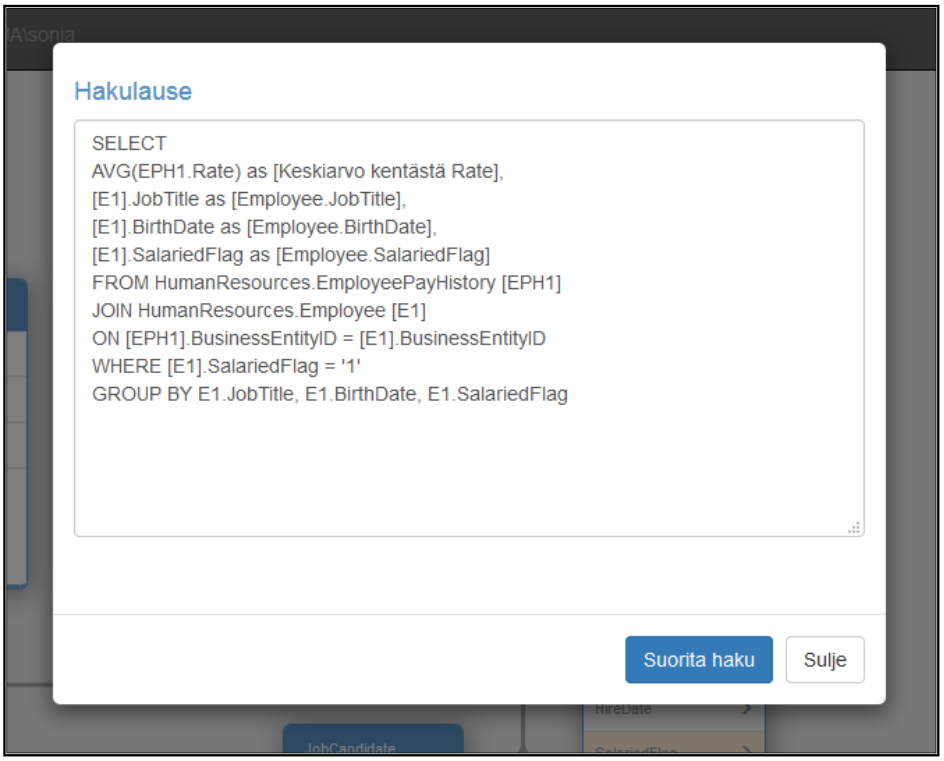


KUVA 12 Hakuehdon lisääminen

Hakulauseeseen voi lisätä myös aggrekaattifunktioita. Numeerisille kentille voidaan suorittaa summa - ja keskiarvolaskelmia tuloksen rivien mukaan. Aggrekaattifunktion lisääminen tapahtuu saman valikon kautta kuin kentän lisäys, poisto tai hakuehdon lisääminen. Myös koko tuloksen kaikki rivit voidaan laskea. Käyttöliittymän sivuvalikossa on valinta, jolla saadaan kaikkien rivien laskenta mukaan tai pois päältä.

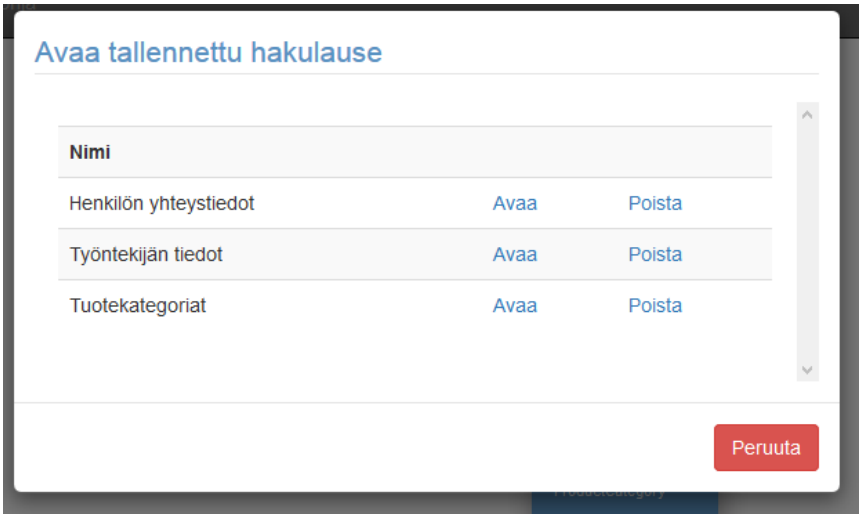
5.4 Hakulauseiden toiminnot

Valintojen pohjalta muodostunutta hakulausetta on mahdollista tarkastella omassa dialogissaan. Jos käyttäjä osaa SQL:ää, hän voi myös käsin muokata tai kirjoittaa lauseen mieleisekseen. Lauseen voi suorittaa samaan tapaan kuin käyttöliittymän kautta muodostetun hakulauseen. Kuvassa 14 on esimerkki hakulausedialogista.



KUVA 14 Valintojen pohjalta muodostunut hakulause

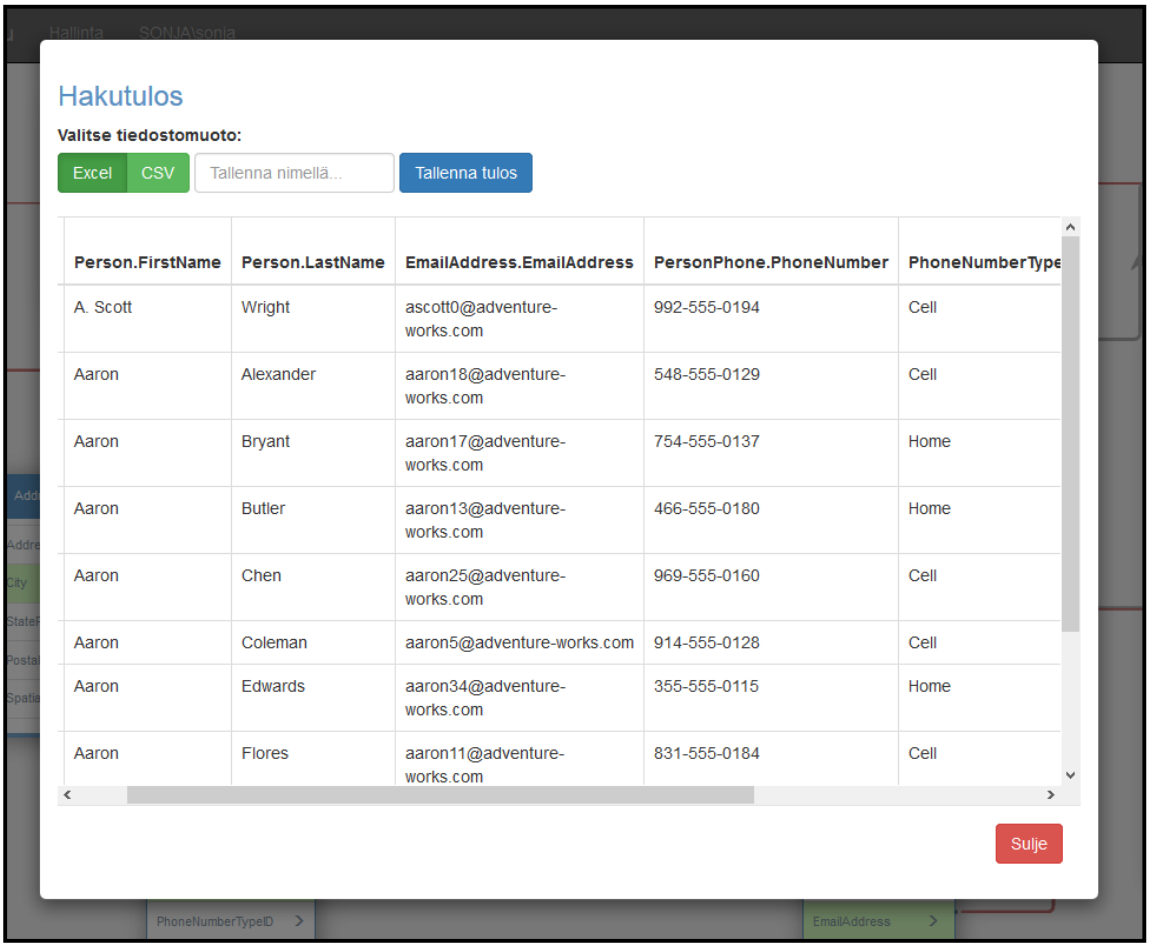
Muodostetun hakulauseen voi myös tallentaa. Hakulauseiden tallentaminen ja selaaminen tapahtuvat sivuvalikosta klikkaamalla, jolloin aukeaa dialogi. Kaikki valittuun tietokantayhteyteen tallennetut hakulauseet on mahdollista suorittaa uudelleen. Lisäksi lauseita voi poistaa. Kuvassa 15 esitellään tallennettujen hakulauseiden selausdialogi.



Kuva 15 Tallennettujen hakulauseiden selaaminen

5.5 Hakutuloksen tallentaminen

Hakulauseen tuloksia voidaan tarkastella omassa dialogissaan. Dialogiin haetaan ensimmäiset kymmenen tulosriviä. Käyttäjä voi palata valintojen tekemiseen sulkemalla dialogin tai tallentaa koko hakulauseen tuottamat rivit Excel- ja CSV-muodossa dialogin kautta. Kuvassa 13 esitellään käyttöliittymän hakutulosdialogi.



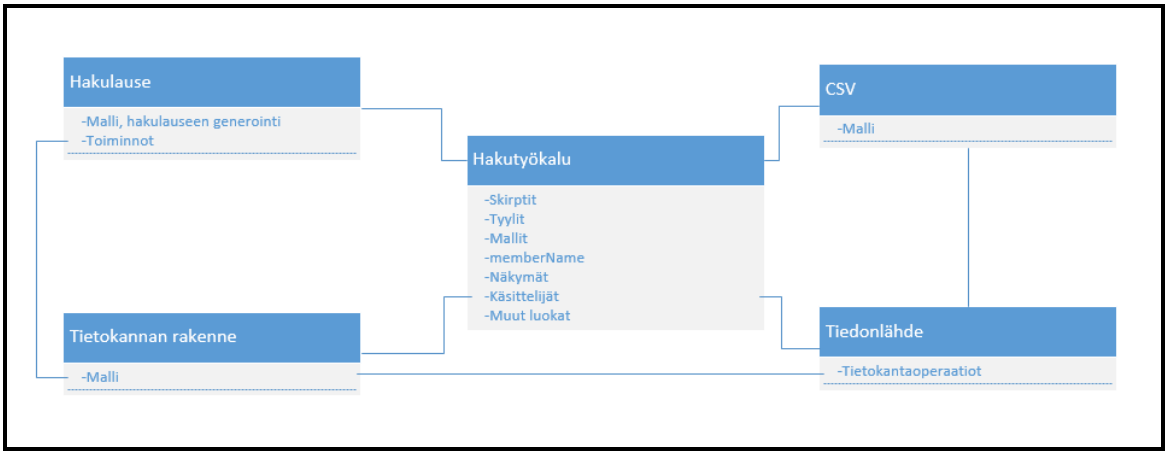
KUVA 13 Suoritetun hakulauseen tulokset näytetään dialogissa

6 HAKUTYÖKALUN TOIMINTA KÄYTÄNNÖSSÄ

Hakutyökalun rakentamiseen käytettiin pääasiassa JavaScriptiä ja Ajax-kutsuja käsittelijään (Controller), sillä sivua ei voida päivittää aina, kun käyttäjä tekee valintoja. Testitietokantana oli Microsoftin mallitietokanta AdventureWorks2014. Kaikki toiminta tapahtuu samalla sivulla, ja tarvittava data näytetään dialogeissa.

6.1 Projektin rakenne

Hakutyökalun toiminnot on jaettu luokkakirjastoihin, jotka ovat yhteydessä pääprojektiin. Kuvassa 16 esitellään karkeasti rakennetta. ASP.NET MVC-projektiin pystyy generoimaan useita toimintoja automaattisesti, mutta tämä projekti kehitettiin alusta alkaen itse. Kuvassa 16 on diagrammi projektin luokkarakenteesta.



KUVA 16 Projektin rakenne karkeasti esitettynä

1. Kaikki luokkakirjastot ovat yhteydessä hakutyökaluun. Hakutyökalu itsessään sisältää myös luokkia.
2. Tietokannan rakenteen sisältävässä kirjastossa on ne valitut taulut, joiden rakenteen mukaisesti luodaan pääsivulle visuaalinen esitys tietokannasta.
3. Hakulause-kirjasto koostuu kahdesta osiosta. Malli sisältää tietorakenteet, joihin käyttäjän valinnat tallennetaan. Lisäksi luokka huolehtii hakulauseen muodostamisesta näiden perusteella. Toinen luokka taas vie käyttäjän valinnat rakenteisiin. Se ottaa vastaan Tietokannan rakenteen mukaiset parametrit käyttöliittymän puolelta. Se huolehtii valinnoista, poistoista sekä hakuehtojen ja aggrekaattien lisäämisestä malliin.
4. CSV-luokkakirjastossa on rakenne, johon hakulauseen tuottama data siirretään CSV- ja Excel–tallennusta varten.
5. Tiedonlähde huolehtii kaikista tavallisista yhteydenotoista tietokantaan. Se lukee hakutuloksen datan CSV-malliin.

Entity Frameworkia varten on projektin sisällä omat luokat, jotka ovat vastuussa hallintapuolen yhteyksien sekä tallennettujen hakulauseiden operaatioista. Entity frameworkia ei voitu hyödyntää hakutyökalun operaatioihin, koska se vaatii tietoonsa mallin (Model) tarkan rakenteen. Tietokannan rakenne taas luodaan vasta käyttäjän valinnan mukaan.

6.2 Käyttäjän tunnistaminen

Käyttäjän avatessa sivusto haetaan ensin kaikki käyttäjäryhmät Active Directorysta ja tutkitaan, mihin ryhmiin käyttäjä kuuluu. Näitä ryhmiä verrataan hallintapuolen tietokannan roolitauluun ja määritetään, mitkä yhteydet haetaan käyttäjälle valittavaksi. Kuvassa 17 esitellään koodi, joka hakee Active Directorysta käyttäjäryhmät ja hakee roolien perusteella oikeat yhteydet.

```
public ActionResult Index()
{
    List<string> userRoles = new List<string>();

    var id = User.Identity as System.Security.Principal.WindowsIdentity;

    foreach (var g in id.Groups)
    {
        var name = g.Translate(typeof(System.Security.Principal.NTAccount)).Value;
        userRoles.Add(name.ToUpper());
    }

    ViewBag.conn = FormControl.GetUserConnections(userRoles);

    return View();
}
```

KUVA 17 Sovelluksen avaamisen yhteydessä määritetään oikeudet tietokantayhteyksiin

6.3 Dialogit

Kaikkia dialogeja varten on luotu omat osittaiset näkymät (PartialView), jotka avataan pääsivun päälle modaaliseksi dialogiksi. Kuvassa 18 on jQuery-funktio, jonka avulla avataan dialogi. Kuvassa 19 näytetään pääsivun koodi, jonka sisään osittainen näkymä ladataan modaalisena. Lopuksi vielä esimerkkinä kuvan 20 koodi, joka avaa valintojen pohjalta muodostuneen hakulauseen sisällön dialogiin. Samalla tyylillä avataan muitakin osittaisia näkymiä dialogeihin. Käsittelijässä (Controller) haetaan tarvittava data, joka palautetaan oikean näkymän mukana jQuery-funktiolle, joka generoi näkymän sisällön pääsivun koodin sisälle.

```
function openDialog(url)
{
    $.get(url, function (data) {

        $('#modal-container').modal({ backdrop: 'static' });
        $('.modal-content').html(data);
        $('#modal-container').modal('show');
    });
}
```

KUVA 18 Dialogille määritellään backdrop: 'static', jotta se ei sulkeudu jos käyttäjä klikkaa dialogin ulkopuolelle. Parametri data sisältää käsittelijän palauttaman näkymän sisällön.

```
<div id="modal-container" class="modal fade" role="dialog" tabindex="-1">
    <div class="modal-dialog">
        <div class="modal-content"></div>
    </div>
</div>
```

KUVA 19 Näkymän sisältö ladataan jQueryn avulla tämän sisälle

```
//näytä hakulause:
public ActionResult ShowQuery()
{
    string query = string.Empty;

    var sql = GetSqlSession();
    sql.TopCount = 0;

    if (sql.FirstTable != null)
    {
        query = sql.GetQuery();
    }

    return PartialView("_ShowQueryPartial", query);

}
```

KUVA 20 Käsittelijän koodi hakulausedialogin avaamiseen

6.4 Tietokantarakenteen muodostaminen

Kun käyttäjä on valinnut haluamansa tietokantayhteyden, avataan Ajax-kutsulla dialogi, josta käyttäjä voi valita haluamansa taulut. Taulujen listaukseen käytettiin Bootstrapin MultiSelect-lisäosaa, jonka avulla saa selkeän alasvetolistan tarjolla olevista tauluista. Taulut ryhmitellään tauluskeeman mukaisesti. Yhteyden mukaiset taulut haetaan Information Schemasta tietokantayhteyden mukaisesti. Information Schema koostuu SQL Serverin näkymistä. Ne sisältävät tietoa muun muassa tietokannan tauluista, sarakkeista, näkymistä, pääavaimista sekä vierasavaimista.

Käyttäjän valittua haluamansa taulut suoritetaan hakulause jälleen Information Schemasta, joka hakee valittujen taulujen nimet, sarakkeet ja tiedot liitoksista. Kuvassa 21 esitellään esimerkkilause, jonka avulla saadaan tarvittavat tiedot kolmen taulun liitoksista.

SQLQuery2.sql - not connected*

```
USE AdventureWorks2014

SELECT
  KCU1.TABLE_NAME AS 'FK_TABLE_NAME'
, KCU1.COLUMN_NAME AS 'FK_COLUMN_NAME'
, KCU2.TABLE_NAME AS 'UQ_TABLE_NAME'
, KCU2.COLUMN_NAME AS 'UQ_COLUMN_NAME'
FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS RC
JOIN INFORMATION_SCHEMA.KEY_COLUMN_USAGE KCU1
ON KCU1.CONSTRAINT_CATALOG = RC.CONSTRAINT_CATALOG
    AND KCU1.CONSTRAINT_SCHEMA = RC.CONSTRAINT_SCHEMA
    AND KCU1.CONSTRAINT_NAME = RC.CONSTRAINT_NAME
JOIN INFORMATION_SCHEMA.KEY_COLUMN_USAGE KCU2
ON KCU2.CONSTRAINT_CATALOG = RC.UNIQUE_CONSTRAINT_CATALOG
    AND KCU2.CONSTRAINT_SCHEMA = RC.UNIQUE_CONSTRAINT_SCHEMA
    AND KCU2.CONSTRAINT_NAME = RC.UNIQUE_CONSTRAINT_NAME
    AND KCU2.ORDINAL_POSITION = KCU1.ORDINAL_POSITION
WHERE KCU2.TABLE_NAME IN ('Person', 'PersonPhone', 'PhoneNumberType')
ORDER BY KCU2.TABLE_NAME
```

100 %

Results | Messages

	FK_TABLE_NAME	FK_COLUMN_NAME	UQ_TABLE_NAME	UQ_COLUMN_NAME
1	Employee	BusinessEntityID	Person	BusinessEntityID
2	Customer	PersonID	Person	BusinessEntityID
3	PersonCreditCard	BusinessEntityID	Person	BusinessEntityID
4	BusinessEntityContact	PersonID	Person	BusinessEntityID
5	EmailAddress	BusinessEntityID	Person	BusinessEntityID
6	Password	BusinessEntityID	Person	BusinessEntityID
7	PersonPhone	BusinessEntityID	Person	BusinessEntityID
8	PersonPhone	PhoneNumberTypeID	PhoneNumberType	PhoneNumberTypeID

KUVA 21 Yhdistävien avaimien hakeminen SQL-lauseella

Taulujen tiedot tallennetaan tietokannan rakenteen luokkaan, josta kerrottiin aikaisemmin. Luokkaan tallennetaan lista käyttäjän valitsemista tauluista, joiden sisällä on lista kunkin taulun sarakkeista ja niiden datatyypistä, vierasavaimista sekä pääavaimesta. Tämän pohjalta muodostuu taulurakenne käyttöliittymään. Kuvassa 22 esitellään luokan ominaisuudet.

QueryTool.DBSchema

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DBSchema
{
    public class DBschema
    {
            public List<Table> Table { get; set; }

             public DBschema()
             {
                 Table = new List<Table>();
             }
    }

    public class Table
    {
        public string TableName { get; set; }
        public string TableSchema { get; set; }
        public List<Column> Columns { get; set; }
        public List<TableReference> ForeignKeys { get; set; }
    }

    public class Column
    {
        public string Name { get; set; }
        public string ColumnType { get; set; }
    }

    public class TableReference
    {
        public string PKcolumn { get; set; }
        public string FKtable { get; set; }
        public string FKcolumn { get; set; }
    }
}
```

KUVA 22 Taulujen rakenneluokka

6.5 Taulurakenteen generoiminen näytölle

Taulurakenteen luomiseksi näytölle hyödynnettiin JavaScript-kirjastoa nimeltään jsPlumb. JsPlumb on työkalu erilaisten diagrammien muodostamiseen. JsPlumbin avulla voi yhdistää erilaisia HTML-elementtejä toisiinsa viivoilla määrittämällä jokaiselle elementille oman id:n. Lisäksi elementit saa liikuteltaviksi. Elementeille määritetään lähde (source) ja kohde (target) elementit id:n mukaan, joiden mukaan piirretään yhteysviivat. Tämän vuoksi kirjasto soveltui erinomaisesti tähän tarkoitukseen.

Kun data on haettu, lähetetään se JavaScript funktiolle, jotta taulut voidaan luoda näytölle. Taulurakenne on nyt tallessa myöhempää käyttöä varten.

Jokaisesta taulusta luotiin oma division-elementti (div) silmukassa (loop). Jokaisen divisionin sisällä on taulukko, joka sisältää kyseisen taulun sarakkeet. JsPlumbilla taulut luotiin liikuteltaviksi ja piirrettiin taulujen väliset yhteydet. Taululementteihin on piilotettu tarvittavaa tietoa data-attribuuttiin. Data-attribuutti on HTML5:n ominaisuus, jonka tarkoitus on mahdollistaa datan tallentaminen elementtiin, niin että siihen pääsee helposti käsiksi.

Taulujen kelaamiseen käytettiin JavaScript-kirjastoa Perfect Scrollbar. Kirjasto otettiin käyttöön, koska tavallinen vierityspalkki on liian iso taulun reunaan. Vierityspalkki tuodaan esiin ainostaan, kun käyttäjä vie hiiren taulun päälle. Palkki on pieni ja kapea, jolloin taulujen ulkoasu säilyy hyvänä.

Taulujen koon muuttaminen ja vierityspalkki piti myös lisätä jQueryn avulla. Koska kaikki generoidaan suoraan ohjelman ajon aikana, nämä ominaisuudet piti lisätä vasta taulujen muodostamisen aikana.

Vierityspalkki lisättiin jQuery Trigger-ominaisuudella. Trigger-ominaisuus lisää tapahtuman (event) HTML-elementtiin. Taulun koon muuttaminen on taas jQueryUI-kirjaston toiminto. Kuvassa 23 esitellään funktio, joka lisää sen tauluihin niiden luomisen jälkeen. Koon muuttaminen piti lisätä uloimman div-elementin lisäksi myös sen sisällä olevaan table-elementtiin, jotta ne molemmat vaihtavat kokoaan yhtenevästi.

```
function setResize() {

    $('.db-table').each(function () {

        $(this).resizable({

            resize: function (event, ui) {
                jsPlumb.repaint(ui.helper);

            }
          , alsoResize: "#" + $(this).find('.table-body').attr('id')
          , minHeight: 100
          , minWidth: 100

        });
    });
}
```

KUVA 23 Lisätään luotuihin elementteihin koon muuttaminen hiiren avulla

6.6 Valinnat tietokantarakenteesta

Kun tietokantarakenne on luotu, käyttäjä voi liikutella taulut haluamallaan tavalla näytölle ja muodostaa hakulauseen. Käyttöliittymässä on määritetty värikoodit kertomaan käyttäjälle hakulauseen tilasta. Vihreä tausta kertoo valitusta kentästä. Oranssi tausta kertoo hakuehdosta ja punainen aggrekaatista. Käytännössä kaikki toiminnot noudattavat samaa kaavaa:

1. Käyttäjä klikkaa HTML-tauluelementin riviä käyttöliittymässä.
2. Tarpeelliset tiedot otetaan talteen. Käyttäjän klikatessa taulun kenttää, taulurivin data-attribuutissa on tallessa sarakkeen nimi ja tyyppi. Koko tauluelementin data-attribuuttiin on taas tallennettu taulun nimi. Avataan valikko, josta käyttäjä valitsee haluamansa toiminnon.
3. Etsitään kentän ja taulun nimeä vastaava olio. Näin saadaan tauluoliosta myös tieto esimerkiksi siitä, mihin muihin tauluihin valittu taulu on yhteydessä.
4. Suoritetaan Ajax-kutsu käsittelijän (Controller) funktioon ja lähetetään parametrinä tarvittava data.
5. Suoritetaan toiminnot käsittelijässä ja tallennetaan muutokset Sessioon. Sen jälkeen palautetaan tarvittava tieto takaisin JavaScript-funktiolle.
6. Tehdään tarvittavat muutokset käyttöliittymään riippuen suoritetusta toiminnosta.

Eli käytännössä esimerkiksi kentän lisääminen tapahtuu seuraavasti: Käyttäjä klikkaa haluamaansa kenttää. Javascript muuttujiin tallennetaan sarakkeen ja taulun nimi. Etsitään taulurakenteesta näitä tietoja vastaava esiintymä. Käännetään olio JSON-merkkijonoksi ja suoritetaan Ajax-kutsu käsittelijän SelectColumn-funktioon. Funktio ottaa parametrinä vastaan merkkijonomuodossa (string) oliot, minkä jälkeen ne muutetaan takaisin oikean tyyppiseksi. Kuvassa 24 esitellään kääntäminen.

```
public ActionResult SelectColumn(string chosenTable, string column)
{
    JavaScriptSerializer jss = new JavaScriptSerializer();

    Table table = jss.Deserialize<Table>(chosenTable);
    Column chosenColumn = jss.Deserialize<Column>(column);
```

KUVA 24 Merkkijonon kääntäminen takaisin olioksi

Kun oliot on muutettu takaisin oikean tyyppiseksi, validoidaan liitokset, suoritetaan tarvittavat metodit uuden kentän lisäämiselle ja tallennetaan kenttä oikeaan tietorakenteeseen. Lopuksi muutokset tallennetaan Sessioon ja palautetaan käyttöliittymään tarvittavat tiedot muutoksista. Jos validointi ei mene läpi, annetaan virheilmoitus käyttäjälle.

Aina kun käyttäjä klikkaa elementtiä, tutkitaan valitun sarakkeen status. Jos kenttä on valittu, näytetään valikossa vaihtoehto kentän poistamiselle. Jos kenttä on numeerinen, näytetään myös vaihtoehdot aggrekaattien lisäämiselle ja/tai poistamiselle.

6.7 Liitoksien määrittäminen

Käyttäjä määrittää liitoksen klikkaamalla taulujen välistä liitosta. JsPlumbin avulla pystyy luomaan tapahtuman(event) tälle. Kun käyttäjä klikkaa liitosviivaa, avataan jQueryn avulla valikko, josta voi valita liitostyypin. Käytännössä Ajax-kutsun mukana lähetetään tieto taulurakenteesta liitoksen toiselta puolen ja toisen taulun nimi.

Valinnan jälkeen määritellään, ovatko molemmat taulut jo olemassa hakulausetta varten. Taulut ovat olemassa, jos molemmista tauluista on valittu jotakin. Tässä tapauksessa päivitetään olemassa oleva liitos. Kuvassa 25 esitellään koodi, joka vaihtaa olemassaolevaa liitostyyppiä. Jos toinen tauluista puuttuu, luodaan se samalla hakulausetta varten ja määritetään oikea liitostyyppi tietorakenteeseen. Jos käyttäjä on klikannut liitosta, josta kumpikaan taulu ei ole mukana valinnoissa, näytetään virheilmoitus.

```
private JoinType UpdateJoin(string jointype)
{
    JoinType join = JoinType.Inner;

    switch (jointype)
    {
        case "right-join": join = JoinType.Right;
            break;
        case "left-join": join = JoinType.Left;
            break;
    }

    return join;
}
```

KUVA 25 Päivitetään olemassa oleva liitostyyppi valinnan mukaan

6.8 Hakuehtojen muodostaminen

Hakuehtoa ei voida lisätä samalla tavoin kun muita valintoja, koska hakuehdolle tulee määrittää ominaisuudet. Hakuehtoa varten avataan dialogi, jossa määritetään, kuinka lausetta rajataan.

Ehdon tarjolla olevat operaattorit riippuvat valitun kentän datatyypistä. Datatyypit voidaan jaotella numeerisiin, päivämäärä- sekä tekstityyppisiin kenttiin. Numero- ja päivämäärätyyppisille kentille voidaan suorittaa esimerkiksi ”suurempi kuin”-tyyppinen hakulauseke. Tekstityyppiselle taas voidaan suorittaa jokerimerkkihaku LIKE, jossa haetaan kaikki esiintymät, joissa esiintyy hakuehdon mukainen merkkijono. Taulukossa 2 esitellään hakutyökalussa mukana olevat operaattorit.

TAULUKKO 2 Työkalun tukemat operaattorit

Datatyyppi	Selite	Merkintä
Yhteiset	Yhtä kuin	=
	Erisuuri kuin	<>
	Sisältää nämä arvot	IN
Text	Kuten	LIKE
Numeric, Date	Pienempi kuin	<
	Suurempi kuin	>
	Pienempi tai yhtäsuuri kuin	<=
	Suurempi tai yhtäsuuri kuin	>=
Date	Tältä väliltä	BETWEEN
	Ei tältä väliltä	NOT BETWEEN

Ennen dialogin avaamista luodaan oliolista, joka pitää sisällään operaattorin nimen selkokielellä sekä operaattorin todellisen arvon. Tästä listasta muodostetaan lomakkeelle alasvetolista.

Myös tätä lomaketta varten muodostettiin oma malli (Model), joka sisältää tarvittavat tiedot. Peruskontrollien lisäksi dialogille tarvitaan tiedot aikaisemmin lisätyistä hakuehdoista. Ehdot on haettu listaan merkkijonoina selkokielisessä muodossa, ja niillä kaikilla on oma id. Käyttäjä voi määrittää luodessaan uuden ehdon, liitetäänkö uusi ehto mukaan AND vai OR mukaisesti.

Myös hakuehtojen poisto tapahtuu dialogilla. Käyttäjä voi klikata haluamansa ehdon pois hakulauseesta. Poisto suoritetaan Ajax-kutsulla, joka lähettää valitun ehdon id:n käsittelijälle (Controller) ja suorittaa poistofunktion.

Lisäksi lomakkeella on validointi, joka estää käyttäjää muodostamasta virheellistä ehtoa. Tekstityyppiselle ehdolle voi käytännössä antaa myös numero- tai päivämääräarvoja. Mutta jos datatyyppi on päiväys tai numeerinen, hakulause epäonnistuu, jos arvo on eri muodossa.

6.9 Hakulauseen luominen

Hakulauseen suorittamista varten on oma funktio, joka muodostaa käyttäjän valinnoista hakulauseen. Kuvassa 26 esitellään tästä funktiosta aggrekaattifunktioiden lisääminen hakulauseeseen. Hakulauseluokka ottaa ensimmäisen valitun taulun erilliseen olioon ja kaikki muut taulut tulevat listaan. Valittujen kenttien lisääminen hakulauseeseen aloitetaan tästä ensimmäisestä taulusta.

```
query += string.Format("SELECT {0}", TopCount > 0 ? "TOP " + TopCount + " " : "");

if (RowCount)
{
    query += " COUNT(*) as [Kaikki rivit], ";
}

if (Aggregates != null)
{
    foreach (var a in Aggregates)
    {
        query += string.Format(" {0}", a.GetAggregateString());

        if (Aggregates.IndexOf(a) <= Aggregates.Count - 1 && (FirstTable.Columns.Count != 0 || OtherTables.Count != 0)) query += ", ";
    }
}
```

KUVA 26 Aggrekaattien lisäys hakulauseeseen

Lause lähetetetään lopulta suoritettavaksi ja siitä muodostetaan oma rakenne tuloksen esittelyä ja tallentamista varten. Tulosten tarkastelua varten hakulauseeseen lisätään TOP 10, jolloin saadaan vain ensimmäiset 10 tulosriviä. Jos käyttäjä haluaa tallentaa tuloksen, haetaan kaikki rivit. Kuvassa 27 havainnollistetaan hakulauseen suorittamista. Kutsutaan GetData-funktiota ja lähetetään sille muodostunut hakulause. Käsinkirjoitettu haku on tallennettu taas sellaisenaan luokkaan. Siihen joudutaan lisäämään TOP 10 tietojen lukemisen aikana, jos käyttäjä ei ole lisännyt sitä itse.

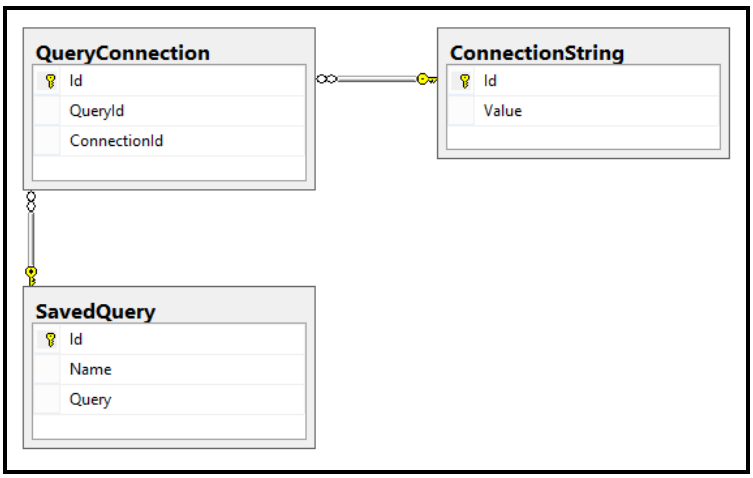
```
if (sql.CustomQuery == null)
{
    sql.TopCount = 10;
    csv = db.GetData(sql.GetQuery());
    ViewBag.CustomQuery = false;
}
else
{
    csv = db.GetData(sql.CustomQuery, true);
    ViewBag.CustomQuery = true;
}
```

KUVA 27 Hakulauseen generointi

6.10 Hakulauseen tallentaminen

Hakulauseiden tallentamista varten on luotu tietokanta, joka pitää sisällään tarvittavat taulut. Tämän tietokannan pohjalta on luotu mallit (Model) sekä oma erillinen DbContext-luokka, joka suorittaa operaatiot tietokantaan Entity FrameWorkin avulla. Kuvassa 28 esitellään tietokannan rakenne. Hakulauseiden selaaminen tapahtuu aina tietokantayhteyskohtaisesti, jolloin ei tarvitse huolehtia käyttäjän oikeuksista.

Tallennetun hakulauseen suorittaminen tapahtuu aivan kuten normaalin hakulauseen. Haetaan vain tietokannasta valittu hakulause ja suoritetaan se käyttöliittymän kautta muodostuneen hakulauseen sijasta.



KUVA 28 Tallennettu hakulause liittyy aina tiettyyn tietokantayhteyteen

6.11 Hakulauseen muodostamisen säännöt

Hakulauseen muodostamisessa täytyy ottaa huomioon muutamia seikkoja, jotta kaikki toimii odotusten mukaisesti jokaisessa tilanteessa. Seuraavaksi käydään läpi joitakin oleellisia asioita.

6.11.1 Liitokset

Ensimmäinen huomion arvoinen asia on taulujen väliset liitokset. Alunperin ajatuksena oli automaattisesti etsiä liitokset kahden tauluvalinnan välille, vaikka välissä olisi useita välitauluja.

Lopulta päädyttiin kuitenkin liitosten pakolliseen määrittämiseen. Jossain tapauksissa toiseen tauluun voi olla montakin liitosta taulujen kautta, eikä voida mitenkään määrittää, minkä näistä liitoksista käyttäjä haluaa. Tämän takia aina lisättäessä kenttää uudesta taulusta, tutkitaan onko tauluun määritetty liitosta. Jos liitosta ei löydy, ohjataan käyttäjää määrittämään liitos ennen kentän valintaa.

6.11.2 Taulualiakset ja Aggrekaatit

Tauluille ja kentille täytyy olla alias mukana, koska sama taulu voidaan valita kahteen kertaan. Jos sama taulu liitetään uudelleen täytyy kertoa aliaksien avulla, kumpaan tauluun viitataan. Seuraavassa esimerkissä (Using a self-join to find the products...) etsitään tietokannasta ne tuotteet, joita tarjoaa useampi tavarantoimittaja:

```
SELECT DISTINCT pv1.ProductID, pv1.VendorID
FROM Purchasing.ProductVendor pv1
INNER JOIN Purchasing.ProductVendor pv2
ON pv1.ProductID = pv2.ProductID
AND pv1.VendorID <> pv2.VendorID
ORDER BY pv1.ProductID
```

Jos hakulauseeseen lisätään aggrekaattifunktio, on kaikki muut mukana olevat valinnat asetettava hakulauseen loppuun GROUP BY-merkinnän jälkeen. GROUP BY-järjestys tulee automaattisesti kenttien valintajärjestyksen mukaan.

Lisäksi aggrekaateissa on otettava huomioon, että ne käyttäytyvät, kuten hakulauseeseen lisätyt kentät. Esimerkiksi aggrekaatin saa lisätä, vaikkei olisi valittu yhtäkään kenttää. Hakuehto taas ei voi olla yksin hakulauseessa.

6.11.3 Käyttäjän muokkaama hakulause

Käyttäjä voi halutessaan muokata muodostunutta hakulausetta tai kirjoittaa sen alusta alkaen itse. Tässä tilanteessa virheen mahdollisuus on suuri. Jos lauseessa on virhe, ohjelma ei saa kaatua, vaan sen pitää ilmoittaa käyttäjälle virheesta.

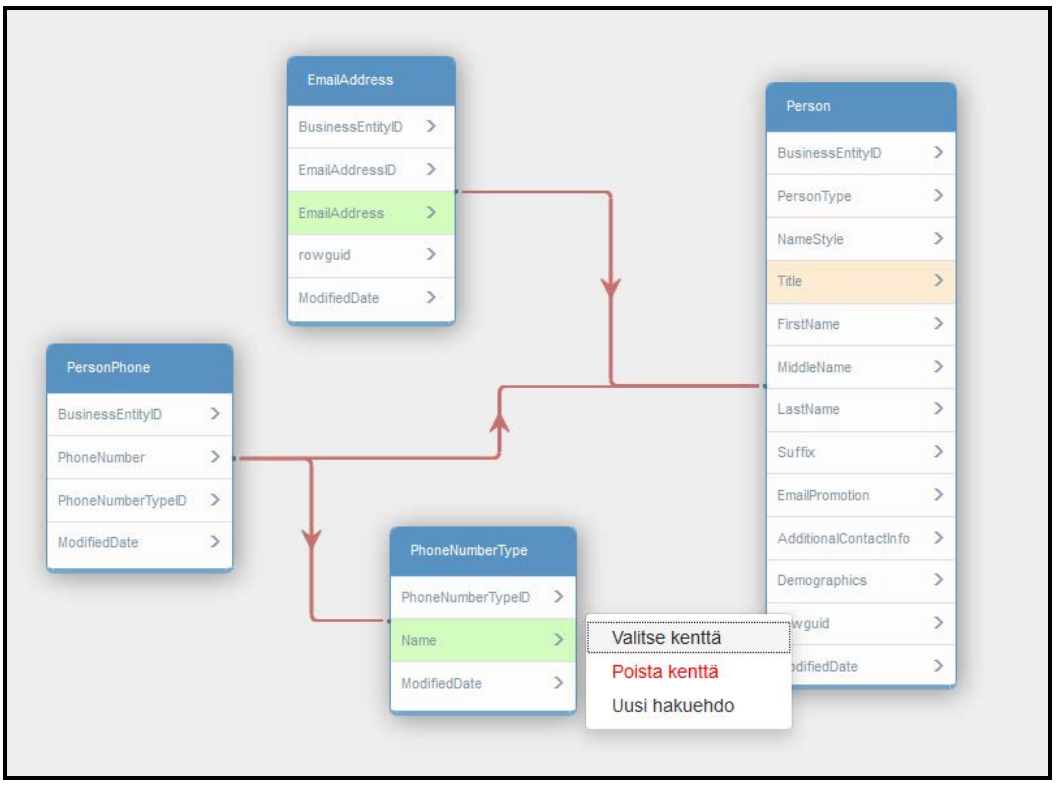
Koska työkalu on tarkoitettu pääasiassa käyttöliittymän kautta suoritettaviin hakulauseisiin, ei työkaluun ole lähdetty rakentamaan virheentunnistamisalgoritmia. Jos hakulauseessa on virhe, estetään ohjelman kaatuminen try{} catch-lausekkeen avulla ja ilmoitetaan, että lauseessa on virhe. Toinen virheilmoitus annetaan, jos hakulause ei tuota yhtäkään riviä. Käyttäjän tarkoitus on tuskin suorittaa lausetta, joka ei tuota tuloksia.

6.11.4 Valintojen poistaminen

Valintojen lisäämisen lisäksi on tärkeää, että käyttäjä voi myös poistaa valintoja. Poistamisen tekninen toteutus on aina haastavampaa kun lisääminen, koska siinä pitää ottaa huomioon enemmän asioita. Myös käyttöliittymä on päivitettävä vastaamaan hakulauseen todellista tilaa kaikissa tilanteissa.

Kuvassa 29 havainnollistetaan yksinkertaista poistotilannetta. Käyttäjä on poistamassa kenttää Name taulusta PhoneNumberType. Jos kenttä poistetaan, täytyy poistaa kaikki liitokset tauluun PersonPhone, koska se toimii vain välitauluna. Person taulun tulee jäädä, koska se sisältää hakuehdon.

Mikäli hakuehtoa ei olisi, poistettaisiin kaikki liitokset, koska EmailAddress jäisi ainoaksi tauluksi. Samoin täytyy määrittää poistaminen myös hakuehdoilla ja aggrekaattifunktioille. Taulua tai sen mahdollisia välitauluja ei tule säilyttää hakulauseessa, jos taulusta ei ole valittu muuta, eikä se toimi liitoksena jonnekin muualle.



KUVA 29 Liitokset on päivitettävä automaattisesti jos poistetaan viimeinen esiintymä taulusta

Käyttöliittymässä kenttien ja liitosviivojen värit on myös päivitettävä erikseen. Jos kentästä poistetaan jokin valinta, täytyy tarkastaa sisältääkö se vielä muitakin valintoja. Väri ei saa vaihtua kentän poiston yhteydessä valkoiseksi, jos se sisältää yhä toisen valinnan.

Ensimmäinen valittu taulu on erillinen olio, josta hakulauseen muodostus aloitetaan. Loput taulut tulevat mukaan listassa. Muut taulut liittyvät aina ensimmäistä taulua kohden. Jos taulu poistetaan, on määritettävä ensimmäinen taulu uudelleen ja mahdollisesti päivitettävä myös liitokset.

Kaikki käsittelijän funktiot palauttavat aina tiedon, kuinka käyttöliittymää tulee päivittää. Erilaisia tilanteita on paljon ja monet niistä tulevat ilmi vasta huolellisen testauksen myötä.

7 TIETOJEN ULOSVIENTI

7.1 Hakutuloksen tarkastelu

Kun valinnat on tehty, käyttäjä voi suorittaa hakulauseen ja hänelle esitetään ensimmäiset kymmenen tulosriviä. Hakutulos siirretään omaan rakenteeseensa, joka asettaa datan otsikkoriviin ja tulosriveiksi. Näin sen saa helposti taulukkoon esille ja lopulta tallennettavaksi. Mikäli hakutulos sisältää datatyypin bit, se muutetaan tuloksen esittämisen yhteydessä selkokieliseksi kuvan 30 mukaisesti.

```
var type = reader.GetDataTypeName(i);

if (type.Equals("bit"))
{
    var bit = reader.IsDBNull(i) ? (bool?)null : reader.GetBoolean(i);
    row.Data.Add( (bit == null ? "" : bit == true ? "Kyllä" : "Ei") );
}
else
{
    row.Data.Add(reader.GetValue(i).ToString());
}
```

KUVA 30 Bit muunnetaan selkokielelle tulokseen

Hakulause suoritetaan, tulokset avataan Ajax-kutsulla ja lopuksi esitetään dialogissa. Dialogin sulkemalla voi palata takaisin kenttien valintaan tarvittaessa.

7.2 Hakutuloksen tallentaminen

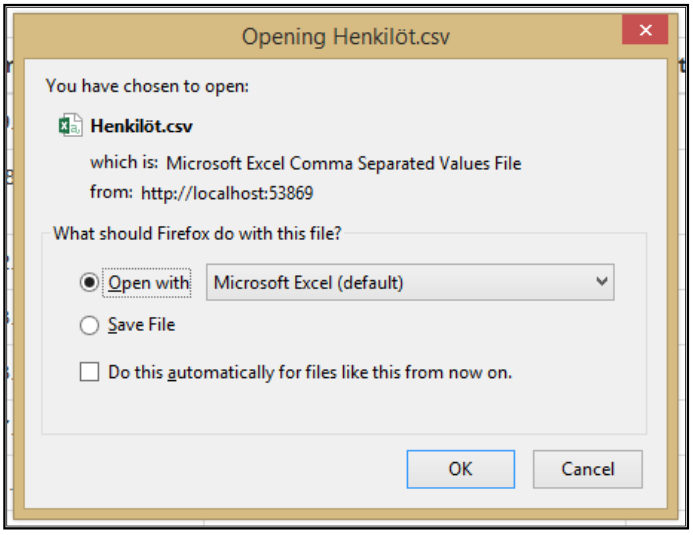
Koko hakutuloksen voi halutessaan tallentaa CSV- tai Excel-muodossa. Tulokselle voi halutessaan antaa oman tiedostonimen tai käyttää oletusnimeä. Excel-tallennuksessa käytetään hyväksi lisäosaa EPPlus. Se on helppokäyttöinen työkalu datan asetteluun Exceliä varten. Työkalun vapaa käyttäminen on sallittua GNU LGPL (Lesser General Public License) lisenssillä.

CSV-muotoiseen tallennukseen on myös oma .NET lisäosa. Se on nimeltään CSV-helper. CSV-helper on Josh Closen kehittämä vapaan lähdekoodin lisäosa. Se huolehtii automaattisesti esimerkiksi pilkkuja sisältävien arvojen asettamisesta lainausmerkkien sisään. Kuvassa 31 esitellään funktio, joka tallentaa hakutuloksen datan CSV-muodossa.

```
public void ExportToCSV( CSVModel csv, string filename = null )
{
    if ( string.IsNullOrEmpty(filename) )
    {
        filename = "csv_tulos";
    }

    System.Web.HttpResponse response = System.Web.HttpContext.Current.Response;
    response.ClearContent();
    response.Clear();
    response.ContentType = "text/csv";
    response.AddHeader( "Content-Disposition", "attachment; filename=" + filename + ".csv;" );

    if (csv != null && csv.Headers != null && csv.Rows != null)
    {
        using (var sw = new StreamWriter(response.OutputStream, Encoding.UTF8))
        {
            sw.AutoFlush = true;

            using (var csvWriter = new CsvWriter(sw))
            {
                csvWriter.Configuration.HasHeaderRecord = false;

                foreach (var h in csv.Headers[0].Data)
                {
                    csvWriter.WriteField(h);
                }
                csvWriter.NextRecord();

                for (int i = 0; i < csv.Rows.Count; i++)
                {
                    foreach (var r in csv.Rows[i].Data)
                    {
                        csvWriter.WriteField(r);
                    }
                    csvWriter.NextRecord();
                }

            }
            response.Flush();
            response.End();
        }
    }
}
```

KUVA 31 CSV-tuloksen tallentamisen koodi

Kuvassa 32 näkyy yllä olevan koodin tuottama tallennusikkuna käyttöliittymässä.



KUVA 32 CSV tallennus

8 YHTEENVETO JA POHDINTA

Työ oli erittäin mielenkiintoinen. Siinä riitti mukavasti haastetta ja opin tekniikoiden lisäksi valtavasti ihan perusrutiinia ohjelmointiin.

Työ pysyi aikataulussaan melko hyvin. Joihinkin toiminnallisuuksiin upposi enemmän aikaa, kuin osasin odottaa ja osa toiminallisuuksista muuttui ajan myötä. En kyennyt aikataulusuunnitelmassa huomioimaan ihan kaikkea, mitä työssä tulee tehdä. Nämä asiat tulivat sitten ilmi etenemisen myötä ja toivat lisää työtä, jota ei oltu huomioita suunnitelmassa. Esimerkiksi valintojen poistamista en huomioinut lainkaan tai aikaa, joka kuluu esimerkiksi suunnitteluun. Tarvittaessa tein vain pidempää päivää, että pysyisin ajallisesti aikataulussa.

Testaaminen vei myös yllättävän paljon aikaa. Kaikkea ei pysty ottamaan huomioon kerralla, joten testasin kaikki eri tilanteet yksitellen uuden ominaisuuden valmistumisen jälkeen ja katsoin, että asiat toimivat kuten pitääkin. Ajatusvirheitä löytyi lähinnä käyttöliittymän päivittämisessä. Useimmiten hakulause oli päivitetty täysin oikein, mutta käyttäjälle näytettiin asiat virheellisesti.

Pidin kirjaa työn edetessä kertyneistä tunneista sekä päivän mittaan tekemistäni asioista Microsoftin Excel-ohjelman avulla.

Pidimme yrityksen kanssa jatkuvasti yhteyttä sähköpostitse ja palaverein, joten työn edetessä oli selkeää, mihin suuntaan edetään. Lisäksi ongelmakohtiin pystyi puuttumaan nopeasti ja korjaamaan ne, jos tein jotakin väärin. Sain suunnitella vapaasti käyttöliittymää ja toteuttaa ominaisuudet haluamallani tavalla. Mielestäni oli todella mukavaa, että sain itse suunnitella työtä.

Yhteystyö yrityksen kanssa sujui mielestäni erinomaisesti. Sain rakentavaa palautetta aina kun esittelin uudet kehittämäni ominaisuudet. Lisäksi sain asiantuntevaa apua tarvittaessa ja hyviä esimerkkejä toteutuksesta. Olin myös tervetullut käymään yrityksessä milloin tahansa, jos tarvitsin lisäopastusta, mikä myös edisti ammatillista kehittymistäni työn edetessä.

Opin lisää SQL-lauseista. Työn kehityksessä piti tutkia, millaisia lauseita on edes mahdollista muodostaa. Lisäksi suunnitteleminen oli mielenkiintoista, koska sovelluksen piti olla helppokäyttöinen ja itsestäänselvä käyttää.

Tämä oli myös suurin projekti, mitä olen koskaan tehnyt. Projektin laajuuden vuoksi tuli vieläkin tärkeämmäksi pohtia myös koodin loogista jakamista luokkiin ja/tai luokkakirjastoihin. Jos kehittäisin sovelluksen oppimillani tiedoilla uudelleen, suunnittelisin projektin alusta alkaen tarkemmin, jolloin siitä tulisi selkeämpi kokonaisuus.

Haasteina työssä oli useimmiten yksinkertaisesti se, etten hallinnut vielä täysin MVC-ohjelmistokehitystä. Varsinkin työn alkuvaiheissa ongelmia ilmeni ihan perusasioissa. Mutta mitä pidemmälle työ eteni, sen enemmän opin ja asiat alkoivat sujua nopeammin.

Työhön voisi helposti kehittää lisää ominaisuuksia. Esimerkiksi käyttäjälle voisi antaa mahdollisuuden määrittää ryhmittelyssä kenttien järjestys. Myös rajausehtoihin voisi mahdollistaa funktioiden lisäämisen ja sulut. Lisäksi työhön voisi suunnitella sisäkkäisiä hakulauseita. Tallennettuihin hakulauseisiin voisi lisätä vielä muokkausmahdollisuuden.

Työtä on helppo kehittää tästä vieläkin monipuolisemmaksi työkaluksi. Uskon silti, että jo tässä vaiheessa siitä voi olla paljon hyötyä perushakujen muodostamisessa.

9 LÄHTEET

ENTITYFRAMEWORKTUTORIAL.NET 2015. What is Entity Framework? [Viitattu 2015-8-2]
Saatavissa: http://www.entityframeworktutorial.net/what-is-entityframework.aspx

MICROSOFT 2015. Using self-joins [Viitattu 2015-9-4]
Saatavissa: https://technet.microsoft.com/fi-fi/library/ms177490%28v=sql.105%29.aspx

MOGHADAMPOUR, G. 2009. C# ohjelmointi. Jyväskylä: Docendo Oy.

MOZILLA CONTRIBUTORS 2005–2015a. HTML (HyperText Markup Language) [Viitattu 2015-9-15]
Saatavissa: https://developer.mozilla.org/en-US/docs/Web/HTML

MOZILLA CONTRIBUTORS 2005–2015b. What is JavaScript, really? [Viitattu 2015-9-15]
Saatavissa: https://developer.mozilla.org/en-US/Learn/Getting_started_with_the_web/JavaScript_basics

REFSNES DATA 1999–2015. ASP.NET MVC Tutorial [Viitattu 2015-8-2]
Saatavissa: http://www.w3schools.com/aspnet/mvc_intro.asp

TECHTARGET 2000-2015a. Active Directory definition [Viitattu 2015-9-16]
Saatavissa: http://searchwindowsserver.techtarget.com/definition/Active-Directory

TECHTARGET 2000-2015b. Microsoft Active Directory Domain Services (AD DS) definition [Viitattu 2015-9-16]
Saatavissa: http://searchwindowsserver.techtarget.com/definition/Microsoft-Active-Directory-Domain-Services-AD-DS

THE JQUERY FOUNDATION 2015. What is jQuery [Viitattu 2015-8-2]
Saatavissa: https://jquery.com/