Saimaan ammattikorkeakoulu
Tekniikka Lappeenranta
Tietotekniikka
Organisaation IT-palvelut

Aleksi Stenholm

Kemikaalitietokanta

Opinnäytetyö 2015

Tiivistelmä

Aleksi Stenholm
Kemikaalitietokanta, 24 sivua
Saimaan ammattikorkeakoulu
Tekniikka Lappeenranta
Tietotekniikka
Organisaation IT-palvelut
Opinnäytetyö 2015
Ohjaajat: lehtori Mikko Huhtanen, Saimaan ammattikorkeakoulu,
Laboratorioinsinööri Eero Kaipainen, Lappeenrannan teknillinen yliopisto,
Nuorempi tutkija Esko Lahdenperä, Lappeenrannan teknillinen yliopisto

Opinnäytetyön tuloksena syntyi kemikaalitietokanta sekä tietokantaa käyttävä verkkosovellus Lappeenrannan teknillisen yliopiston kemiantekniikan osastolle. Tietokantaan tallennetaan kemiantekniikan osastolla olevat kemikaaliastiat ja niiden käyttömäärät.

Kemikaalitietokanta luotiin aikaisemmin käytössä olleen Microsoft Access -tietokannan pohjalta. Tietokanta saatettiin kolmanteen normaalimuotoon, jotta tietokanta pysyy eheänä ja tiedon toistuvuus saadaan poistettua.

Verkkosovelluksen toiminnallisuus rajattiin kemikaalien hallintaan, astioiden hallintaa, astioiden käyttöön sekä raportointiin. Kemikaalien hallinnassa käyttäjä ylläpitää listaa kemiantekniikan osastolla käytettävissä olevista kemikaaleista. Astioiden hallinnassa ylläpidetään listaa kemiantekniikan osaston varastoissa olevista kemikaaliastioista. Kun kemikaalia käytetään, käyttö kirjataan sovelluksessa. Sovelluksesta saadaan neljää erilaista raporttia: varastohuoneen inventaarioraportti, kemikaalin käyttöraportti, kemikaalien määrä vaaralausekkeiden mukaan sekä laboratoriolle kuuluvat kemikaalit.

Verkkosovelluksen ohjelmointiympäristönä käytettiin Visual Studio 2013:a. Sovelluksen ohjelmointiin käytettiin ASP.NET MVC 4-viitekehystä, eli sovelluksen verkkosivut ohjelmoitiin HTML:ää käyttäen ja taustalla oleva koodi C#-ohjelmointikieltä käyttäen. Tietokanta luotiin käyttäen Microsoft SQL Serveriä.

Asiasanat: ASP.NET, MVC, Tietokanta, SQL Server, C#

Abstract

Aleksi Stenholm
Chemical database, 24 pages
Saimaa University of Applied Sciences
Technology Lappeenranta
Degree Programme in Information Technology
IT-services in Organization
Bachelor´s Thesis 2015
Instructors: Mr. Mikko Huhtanen, Lecturer in Saimaa University of Applied Sciences, Mr. Eero Kaipainen, Laboratory Manager in Lappeenranta University of Technology, Mr. Esko Lahdenperä, Doctoral Student in Lappeenranta University of Technology

The result of this thesis was a chemical database and a web application that uses the database, made for the chemistry department of Lappeenranta University of Technology. The amount that department has of each chemical and their usage are stored in the database.

The database was created according to chemical department's old Microsoft Access database. It was normalized in to 3rd normal form in order to prevent redundancy.

The functionality of the web application was delimited to chemical management, container management, container usage, and reporting. In the chemical management users maintain a list of chemicals that chemistry department has in their use. In container management users maintain a list of containers that chemistry department has in their storage rooms. When chemical is used, the usage will be recorded in the application. There are four different reports that can be taken from the application: Inventory report, usage report, report according to hazard statements, and report of laboratories chemicals.

The web application was programmed using Visual Studio 2013 integrated development environment. The application was programmed using ASP.NET MVC4 framework, meaning that views were done using HTML and code running behind was done using C# programming language. The database was done using Microsoft SQL Server.

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

Sisällys

Termit ja käsitteet

.NET	Microsoftin luoma ohjelmistokomponenttikirjasto
Access	Microsoftin Office-pakettiin kuuluva tietokantaohjelmisto
C#	Microsoftin kehittämä ohjelmointikieli
CAS	Kemikaalien tunnistenumerojärjestelmä
CSS	Cascading Style Sheets. Verkkosivujen tyylejä määrittävä kieli
HTML	Hypertext Markup Language. Verkkosivujen luomiseen suunniteltu kuvauskieli
HTTP	Hypertext Transfer Protocol. Palvelinten ja selainten käyttämä tiedonsiirto protokolla
JavaScript	Verkko-ohjelmointi kieli
jQuery	JavaScript kirjasto
SDK	Software Development Kit
SQL	Structured Query Language. IBM:n kehittämä kyselykieli
VB.NET	Visual Basic. Ohjelmointikieli

1 Johdanto

Aihe opinnäytetyöhön tuli Lappeenrannan teknillisen yliopiston kemiantekniikan osastolta. He halusivat päivittää kemikaalien varastointi- ja käyttötietokantansa toimivampaan järjestelmään. Tietokanta oli toteutettu Microsoft Accessille ja sen käyttö oli haastavaa osalle käyttäjistä. Vanha tietokanta pitää sisällään noin 1700 kemikaalia, joihin on liitetty noin 4000 astiaa. Uudeksi tietokannaksi valikoitui Microsoft SQL Server 2012, sekä ohjelmistokehykseksi valkoitui ASP.NET MVC 4. Sovelluksen kehityksessä käytettiin myös teknologioita kuten JavaScript, jQuery sekä PetaPoco. Opinnäytetyö keskittyy pääasiallisesti kyseisiin teknologioihin sekä uuden tietokannan tekniseen toteutukseen.

1.1 Tavoitteet

Opinnäytetyön tavoitteena on päivittää kemiantekniikan osastolla käytössä oleva kemikaalitietokanta Microsoft SQL Server 2012-relaatiotietokantaohjelmalle sekä kehittää verkkosovellus tietokannan käyttöä varten käyttäen ASP.NET MVC 4-ohjelmistokehystä.

1.2 Lappeenrannan teknillinen yliopisto

Lappeenrannan teknillinen yliopisto (LUT) on yksi Suomen viidestätoista yliopistosta ja se on toiminut vuodesta 1969 tekniikan ja talouden yhdistävänä tiedeyliopistona (1).

Kemiantekniikan osasto kuuluu School of Engineering Science -schooliin, jonka tarkoitus on kehittää turvallisempia, ekologisempia ja energiatehokkaampia toimintatapoja, sekä luoda ratkaisuja teollisuuteen ja sen eri tuotantovaiheisiin. Kemikaalitietokanta-sovellus tulee palvelemaan erotus, puhdistus ja prosessit osaamisalueilla. (2.)

Erotus osaamisalueen tavoiteena on kehittää entistä ympäristöystävällisempiä, vähemmän jätettä tuottavia, energiatehokkaampia ja kustannustehokkaampia erotusprosesseja (3.)

Puhdistus osaamisalue tutkii ja kehittää erilaisia menetelmiä jätevesien, raakavesien sekä prosessivirtojen käsittelemiseksi. Keskeisimipiä

tutkimusaiheita ovat erityisesti haastavat jäteveden käsittelyprosessit kemian, sellu- ja paperiteollisuuden, kaivannaisteollisuuden sekä elintarvike- ja lääketeollisuuden sovellutuksissa. (4.)

Prosessit osaamisalueen painopisteenä on kehittää uusia valmistusmenetelmiä ja –laitteistoja prosessiteollisuudelle. Keskeisiä aihepiirejä ovat prosessien intensifionti, mallinnus ja simulointi, suunnittelun metodiikka, prosessien turvallisuus sekä uusien prosessilaitteiden kehitys. (5.)

Opinnäytetyössä yhteyshenkilöinä kemiantekniikan osastolta toimivat nuorempi tutkija Esko Lahdenperä sekä laboratorioinsinööri Eero Kaipainen. Kemikaalitietokannan käyttäjiä kemiantekniikan osastolla tulee olemaan noin 100, joista 6 on järjestelmän ylläpitäjiä.

2 Käytetyt teknologiat

Tässä luvussa käydään läpi opinnäytetyössä käytetyt teknologiat.

2.1 ASP.NET

ASP.NET (Active Server Pages .NET) on Microsoftin kehittämä dynaamisten verkkosivujen ja –sovellusten kehittämiseen tarkoitettu arkkitehtuuri. ASP.NET-arkkitehtuurissa käyteään hyväksi HTML:ää, CSS:ää, JavaScriptiä sekä palvelinskriptejä. ASP.NET on kehitetty vanhan ASP-teknologian (Active Sever Pages) pohjalta, joka on palvelimella tulkittava ohjelmointikieli. Näin ollen myös ASP.NET-koodi tulkitaan palvelimella. (6.)

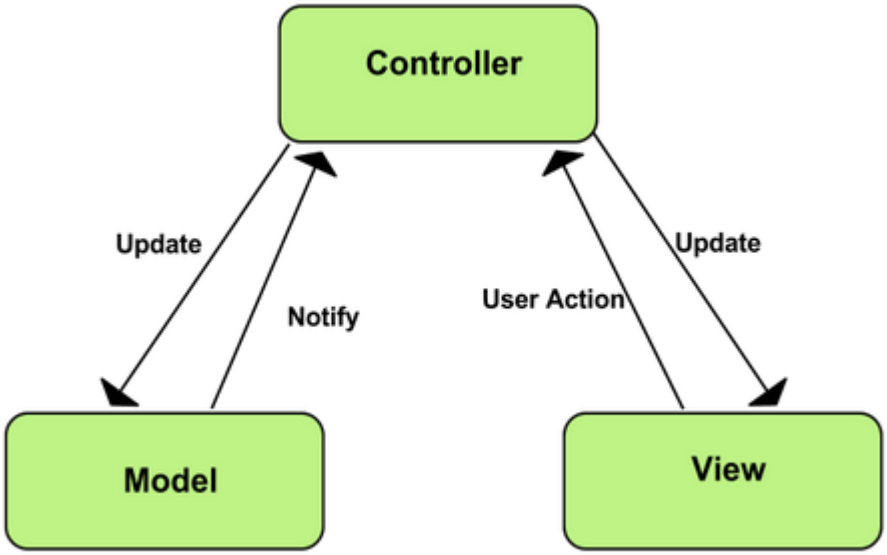
ASP.NET on osa .NET-arkkitehtuuria joten ohjelmistokehittäjän on mahdollista käyttää hyödykseen kaikkia .NET-arkkitehtuurin ominaisuuksia, joita ovat esimerkiksi kattavat luokkakirjastot. .NET-arkkitehtuuri mahdollistaa myös palvelinskriptien luomisen jollain .NET-arkkitehtuuriin kuuluvalla ohjelmointikielellä (C#, VB.NET, jne.). (7.)

ASP.NET-arkkitehtuuri jakautuu kolmeen eri ohjelmistoarkkitehtuurityyliin: Web Forms, Web Pages ja MVC. Web Pages-tyyli on hyvin samanlainen kuin PHP ja vanha ASP, eli verkkosivu koostuu yhdestä tiedostosta johon koodi on kirjattu

käyttäen Razor nimistä palvelimella tulkittavaa merkintäkieltä. Web Forms-tyylissä verkkosivut koostuvat kahdesta tiedostosta. .aspx-päätteinen tiedosto pitää sisällään käyttäjälle näkyvät sovelluksen osat. Toinen tiedosto on päätteeltään .aspx.cs (C#) tai .aspx.vb (VB.NET). Tähän tiedostoon on ohjelmoitu sovelluksen logiikka. Tässä opinnäytetössä käytettiin MVC-tyyliä ja sen toiminta selvitettään seuraavassa kappaleessa tarkemmin. (8.)

2.2 MVC

Model View Controller eli MVC on yksi ASP.NET-arkkitehtuurin ohjelmistoarkkitehtuurityyleistä. MVC:ssa sovellus on jaettu kolmeen osaan jotka kommunikoivat keskenään. Osat ovat Model, View ja Controller. Kuviossa 1 kuvataan arkkitehtuurin kolmen osan keskenäistä kommunikaatiota. (9.)



Kuvio 1: MVC-arkkitehtuuri (10)

2.2.1 Model

MVC-arkkitehtuurisa on kolmenlaisia modeleita: Data, business ja view model. Data modelit ovat luokkia jotka kommunikoivat tietokannan kanssa. Business modelit ovat luokkia jotka prosessoivat dataa sekä kommunikoivat Data modelien kanssa lukeakseen ja tallettaakseen dataa tietokantaan. View modelit ovat luokkia jotka sisältävät datan joka lähetetään käyttäjän ruudulle. View modelit

eivät prosessoi dataa millääänlailla vaan ne ovat puhtaasti vain datan esittämiseen tarkoittettuja luokkia. (11.)

2.2.2 View

Viewt ovat käyttäjälle selaimessa näkyviä HTML-sivuja. Näiltä sivuilta käyttäjien tekemät komennot lähetetään Controller luokille HTTP-protokollan avulla. Viewt sisältävät myös palvelimella tulkittavaa merkintäkieltä, jotka View moottori tulkitsee HTML:äksi. MVC-arkkitehtuurissa on mahdollista valita kahdesta eri View moottorista joilla kummallakin on eri merkintäkieli. View moottorit antavat ohjelmoijalle mahdollisuuden yhdistää C#- tai VB.NET-koodia HTML:än kanssa viewejä tehdessään. Ensimmäinen view moottori on ASPX. ASPX on ollut kuulunut ASP.NET MVC-arkkitehtuuriin sen alusta alkaen ja sen syntaksi muistuttaa hyvin paljon Web Forms-syntaksia. Kuviossa 2 on esimerkki ASPX-syntaksista. (12.)

```
<%{
    List<string> names = new List<string>();
    names.Add("Jack");
    names.Add("John");
    names.Add("Lily");
    names.Add("Mary");

    foreach(string name in names)
    {
        %><li><%=name%></li><%
    }
}%>
```

Kuvio 2: Esimerkki ASPX-syntaksista

Toinen view moottori on Razor. Razor view moottori lisättiin ASP.NET MVC-arkkitehtuuriin MVC3 julkaisussa. Kuviossa 3 on esimerkki Razor-syntaksista.

```
@{
    List<string> names = new List<string>();
    names.Add("Jack");
    names.Add("John");
    names.Add("Lily");
    names.Add("Mary");

    foreach(string name in names)
    {
        <li>@name</li>
    }
}
```

Kuvio 3: Esimerkki Razor-syntaksista

2.2.3 Controller

Controller luokat prosessoivat HTTP-pyyntöjä joita View sivut lähettävät käyttäjän tehdessä toimintoja sivulla. Controller toimii viewn ja modelin välissä ohjaten käyttäjän tekemät toiminnot oikeaan model luokkaan sekä välittävän modelin tietokannasta hakemat tiedot takaisin käyttäjälle. (13.)

2.3 PetaPoco

PetaPoco on pieni yhdestä C# tiedostosta koostuva ORM (Object-Relational Mapping) .NET-viitekehykseen. Sen tehtävänä on jäljitellä relaatiotietokannan rakennetta ja kommunikoida tietokannan sekä sovelluksen välissä. ORM luo tietokannassa olevat taulut objekteiksi joita voidaan käyttää tietokantaa käyttävässä sovelluksessa. (14.)

Ohjelmoija voi myös luoda oman ORM:n. Tässä opinnäytetyössä päädyttiin käyttämään PetaPoco ORM:ää sen sijaan, että oltaisi päädytty luomaan oma ORM alusta alkaen tai käytetty .NET-viitekehyksessä jo valmiina olevaa Entity Frameworkiä. Päätös tehtiin aikaisempien kokemuksieni pohjalta. Olin aikaisemmin työskennellyt PetaPocon kanssa ja huomannut sen toimivat Entity Frameworkiä nopeammin.

2.4 Microsoft SQL Server 2012

Microsoft SQL Server on tietokanta-ohjelmisto jolla voidaan luoda relaatiotietokantoja. Microsoft SQL Server on tarkoitettu luomaan tietokantoja

Windows pohjaisten sovellusten käyttöön. Ohjelmiston mukanaan kuuluu myös tietokannan hallintajärjestelmä, mutta tätä käytetään harvoin suurissa tietokantakokonasuuksissa. Pääasiallisesti käyttäjät kommunikoivat tietokannan kanssa jonkin muun sovelluksen kautta. Käyttäjän ja tietokannan välinen kommunikaatio tapahtuu SQL kielen avulla. (15.)

Microsoft SQL Server soveltuu monenlaisten tietokantojen ylläpitoon. Pienissä tietokantatoteutuksissa tietokanta voidaan asentaa paikallisesti tietokoneelle, jolla sitä käytetään. Näin ollen tietokannan hallinnointi tapahtuu paikallisesti.

Suurissa tietokantatoteutuksissa tietokanta asennetaan palvelimelle. Näin ollen käyttäjät käyttävät sekä hallinnoivat tietokantaa internetin yli. Microsoft SQL Serverin toimiessa palvelimella järjestelmässä tapahtuvat työt jaetaan asiakas- ja palvelintietokoneiden välille. Palvelintietokoneet hallinnoivat sekä ylläpitävät tietokantaa. Asiakastietokoneet vastaavat tietokannan sisältämästä tiedosta, eli tallentavat ja muokkaavat tietokannassa olevaa tietoa. (16.)

3 Työvaiheet

Tässä luvussa käydään läpi työvaiheet, jotka opinnäytetyössä käytiin läpi.

3.1 Tietokannan suunnittelu

Uuden tietokannan pohjana on käytetty kemiantekniikan osaston vanhaa Microsoft Access-pohjaista kemikaalitietokantaa. Kyseinen tietokanta ei täyttänyt hyvin suunnitellun tietokannan kriteerejä. Tieto ei ollut eheää ja tiedossa oli paljon toistoa. Asiakas halusi saada tietokannan uudistettua uudelle alustalle sekä muokata tietokannan rakennetta niin, että hyvän tietokannan kriteerit täyttyvät.

Uusi tietokanta luotiin vanhan tietokannan pohjalta. Tietokannan suunnittelu aloitettiin saattamalla vanha tietokanta parhaaseen mahdolliseen normaalimuotoon. Tässä tapauksessa kyseinen tietokanta saatiin kolmanteen normaalimuotoon ja näin tietokannasta saatiin poistettua tiedon turha toisto. Vanhan tietokannan tauluja jaettiin useampiin tauluihin sekä monen suhde moneen relaatioita luotiin huomattavasti lisää.

LUT:n IT-osasto ilmoitti, mitä tietokanta-alustoja heillä on ylläpidettävänä ja mille niistä on mahdollista luoda uusi kemikaalitietokanta. Näistä alustoista valituksi tuli Microsoft SQL Server.

Asiakas toivoi uuden kemikaalitietokantasovelluksen olevan verkkosovellus, jotta mitään ylimääräisiä asennuksia käyttäjien tietokoneille ei tarvita. Koska SQL Server tukee Windows-pohjaisia sovelluksia valittiin LUT:n IT-osaston kanssa käytettäväksi ASP.NET-ohjelmistoarkkitehtuuria. MVC-arkkitehtuurimalli valikoitui käytettäväksi, sillä se soveltuu hyvin kemikaalitietokanta tyylisiin sovelluksiin sekä minulla oli jo ennestään hieman kokemusta kyseisestä mallista.

Vanhaa tietokantaa käytettiin kemiantekniikan osastolla tietokannan päälle tehdyllä VBA-sovelluksella. Tätä sovellusta käytettiin uuden verkkosovelluksen suunnittelun pohjana. Asiakkaan kanssa kävimme läpi kaikki sovelluksen toiminnot ja niihin liittyvät ongelmat ja näiden keskustelujen pohjalta suunnittelin uuden sovelluksen toiminnallisuuden.

3.2 Tietokannan normalisointi

Tietokannan normalisoinnilla pyritään ehkäisemään tiedon toistuvuus tietokannassa sekä säilyttämään tietokannan eheys. Normaalimuotoja on monia, ja tietokantaa suunnitelessa vähimmäisvaatimuksena on hyvä pitää kolmatta normaalimuotoa.

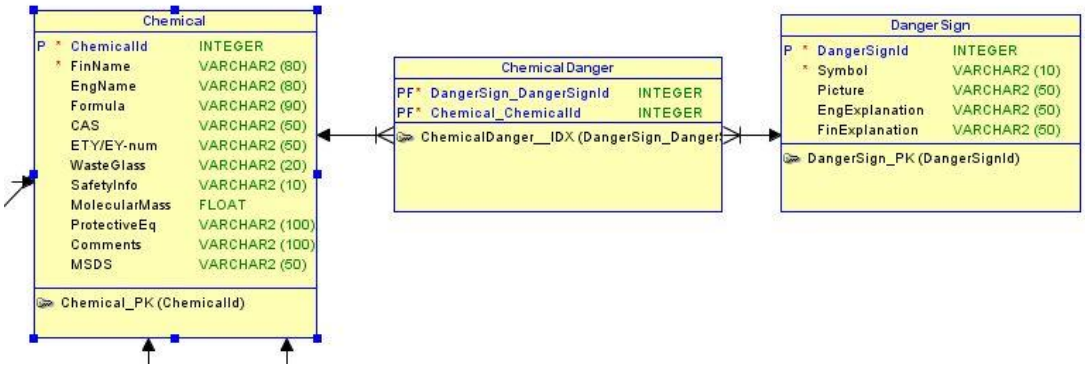
Kolmas normaalimuoto tarkoittaa, että tietokanta täyttää ensimmäisen ja toisen normaalimuodon vaatimukset, sekä jokaisen tietokantataulun sarakkeen on oltava riippuvainen taulun pääavaimesta (17).

Ensimmäinen normaalimuoto tarkoittaa, että tietokantatauluissa ei ole toistuvaa tietoa. Jos taulussa on toistuvia tietoja, siirretään ne omiksi tauluikseen. (17.)

Toisessa normaalimuodossa tietokantataulu on ensimmäisessä normaalimuodossa ja taulujen sarakkeet, jotka eivät ole avaimia, ovat riippuvaisia ainoastaan taulujensa avaimista. Tämä tarkoittaa siis sitä, että jos tiedetään taulun avain, tiedetään myös muut arvot. (17.)

3.3 Tietokannan toteutus

Tietokannan suunnittelun jälkeen tietokannasta laadittiin relaatiomalli jossa on kuvattuna tietokannan taulut, taulujen perusavaimet, kenttien tietotyypit sekä taulujen väliset suhteet. Relaatiomalli luotiin Oraclen Datamodeler-ohjelmaa käyttäen. Kyseiseen ohjelmaan syötetään taulujen tiedot ja se luo relaatiomallin näiden tietojen pohjalta. Kuviossa 4 on osa kemikaalitietokannan relaatiomallista.



Kuvio 4: Esimerkki relaatiomallista

Tietokannan luomiseen tarkoitettu SQL-skripti on rakennettu relaatiomallin pohjalta. Oraclen Datamodelerissa skriptin luominen onnistuu helposti, sillä relaatiomallista pystyy suoraan luomaan SQL-skriptin tietokannan luomiseen Oraclen, Microsoftin sekä IBM:n DB2-tietokannoille. Tässä opinnäytetyössä Oraclen Datamodelerista saatua skriptiä jouduttiin hieman muokkaamaan. Kuviossa 5 on esimerkki Kemikaalitietokannan luomiseen tarkoitetusta SQL-skriptistä. Skriptissä luodaan Chemical-taulu tietokantaan.

```
CREATE
  TABLE Chemical
  (
    ChemicalId   INTEGER NOT NULL IDENTITY(1,1) ,
    FinName      VARCHAR (80) NOT NULL ,
    EngName      VARCHAR (80) ,
    Formula      VARCHAR (90) ,
    CAS          VARCHAR (50) ,
    "ETY/EY-num" VARCHAR (50) ,
    WasteGlass   VARCHAR (20) ,
    SafetyInfo   VARCHAR (10) ,
    MolecularMass FLOAT ,
    ProtectiveEq VARCHAR (100) ,
    Comments     VARCHAR (100) ,
    MSDS         VARCHAR (50) ,
    CONSTRAINT Chemical_PK PRIMARY KEY CLUSTERED (ChemicalId)
);
```

Kuvio 5: SQL-skripti Chemical-taulun luomiseen

Verkkosovelluksen toteutusvaiheessa tietokantaa käytettiin lokaalisti, eli tietokanta oli luotu siihen tietokoneeseen, jolta sovellusta ohjelmoitiin. Testausvaiheeseen siirryttäessä tietokanta siirrettiin LUT:n IT-hallinnon tietokantapalvelimelle, jotta tietokantaa pystyttiin alkaa käyttämään normaalissa käyttöympäristössään.

3.4 Verkkosovelluksen toteutus

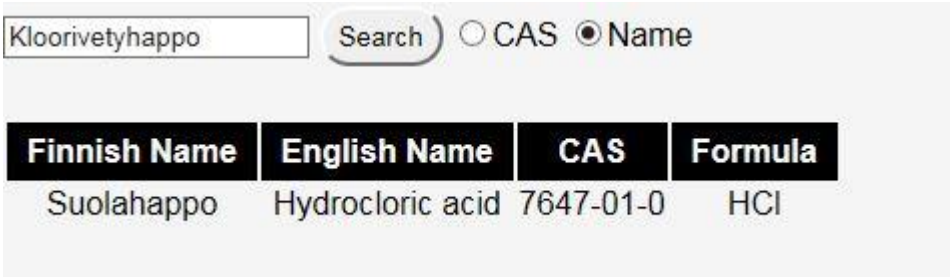
Verkkosovelluksen pohjana käytettiin vanhaa Access-sovellusta. Uuteen sovellukseen haluttiin tuoda vanhan sovelluksen toiminnot, mutta niitä muokattiin helppokäyttöisemmiksi. Pidimme asiakkaan kanssa palavereja kahden viikon välein ja näissä palavereissa kävimme läpi toimintoja joita sovellukseen tarvitaan. Tarvittaviksi toiminnoiksi rajasimme kemikaalien hallinnoinnin, astioiden hallinnoinnin, astioiden käyttön, astioiden haun, kemikaalien hakemisen synonyymeillä sekä raportoinnin.

3.4.1 Kemikaalien hallinnointi

Kemikaalien hallinnoinnilla tarkoitetaan kemikaalien syöttämistä tietokantaan, niiden poistamista sekä muokkaamista. Kun kemikaali lisätään tietokantaan, ainoa pakollinen kenttä on suomenkielinen nimi. Kemikaaleja saatetaan lisätä

tietokantaan vajanaisin tiedoin, joita täydennetään myöhemmin, mutta suomenkielinen nimi on pakollinen. Kemikaalin poistamiseen tietokannasta on luotu T-SQL-transaktio, sillä kemikaali on viiteavain monessa taulussa. Kun kemikaali poistetaan, siihen liittyvät muut tietueet on poistettava myös. Transaktio yrittää poistaa kaikki kemikaaliin liittyvät tietueen ja jos ongelmia ei ilmene, poisto tapahtuu. Ongelmien ilmetessä poistaminen kumotaan ja käyttäjälle näytetään virheilmoitus, jossa selostetaan ongelma. Transaktiolla saadaan tietokannan eheys pidettyä paremmin hallinnassa, sillä poisto ei voi tapahtua vain puoliksi käyttäen transaktiota.

Kemikaaleille syötetään myös synonyymejä, koska samalla kemikaalilla voi olla monta eri kaupallista nimeä. Hakutoiminto mahdollistaa kemikaalin löytymisen tietokannasta, vaikka kemikaali olisi syötetty toisella nimellä tietokantaan. Tällaisen toiminnon tarpeellisuus nousi usein esille palavereissa. Synonyymien käyttö on toteuttettu seuraavalla tavalla. Tietokantaan luotiin Synonym-niminen taulu, johon kaikki tietokantaan syötettävät synonyymit lisätään. Chemical- ja Synonym-taulujen väliin luotiin M:N-relaatio, eli käytännössä taulu, jonka tietueet koostuvat Chemical-taulun pääavaimesta sekä Synonym-taulun pääavaimesta. Näin ollen kemikaalille on saatu linkitettyä synonyymejä. Hakutoiminto hakee ensin Chemical-taulusta kemikaaleja nimen mukaan, ja jos haku ei palauta tuloksia, se katsoo, onko hakusana synonyymi. Jos hakusana on synonyymi, haku tarkistaa, mihin kemikaaliin kyseinen synonyymi liittyy ja näyttää kemikaalin käyttäjälle. Hakutoiminnolla voidaan kemikaaleja hakea tietokannasta myös uniikkia CAS-numeroa käyttäen. Kuviossa 6 on esimerkki hakutoiminnosta ja siitä, kuinka se on palauttanut kemikaalin, vaikka haku on tehty synonyymillä.



Kuvio 6: Hakutoiminto

3.4.2 Astioiden hallinnointi

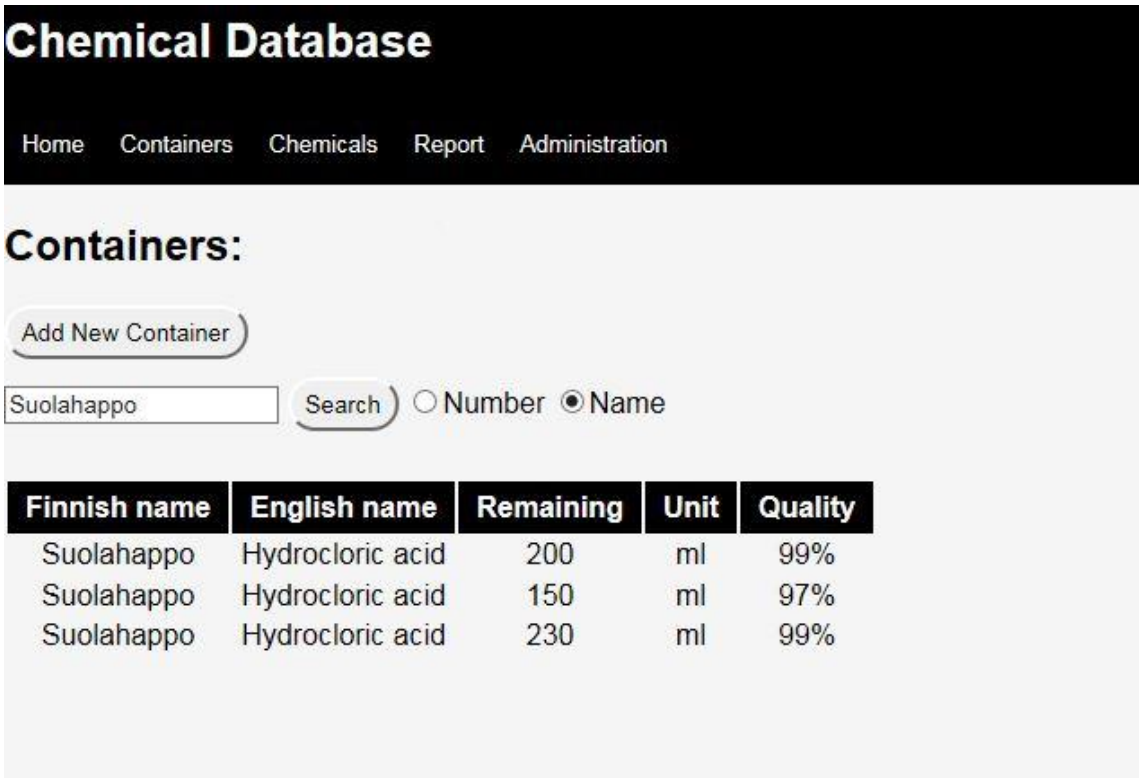
Astioiden hallinnointiin kuuluu astioiden lisääminen, poistaminen ja muokkaaminen. Uutta astiaa lisätessä käyttäjä valitsee, mitä kemikaalia astia sisältää sekä mikä laboratorio omistaa kyseisen astian. Käyttäjä myös syöttää tiedot astiasta. Pakollisia tietoja ovat astian numero sekä varastohuone, jossa astiaa säilytetään. Kemikaali ja laboratorion valintaa varten tehtiin jQueryä käyttäen tekstikenttä, joka hakee tietokannan taulusta koko ajan käyttäjän syöttämää merkkijonoa vastaavia kemikaaleja, eli siis ennakoiva tekstinsyöttö. Kemikaaleja voi olla tuhansia tietokannassa, joten normaalia pudotuvalikkoa ei voitu käyttää. Kuviossa 7 on esimerkki ennakoivan tekstinsyötön toimivuudesta.

Kuvio 7: Astian lisäämiseen tehty kemikaalinvalinta

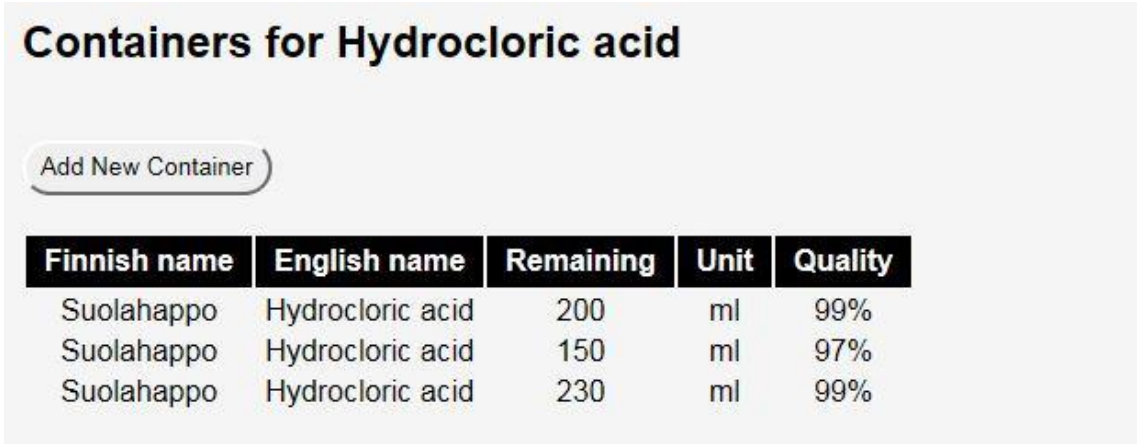
Astian poistaminen tietokannasta on järjestelmänvalvojan tehtävä. Normaalikäyttäjän poistaessa astia verkkosovelluksessa muuttuu Container taulussa IsRemoved-kentän arvo, mutta kemikaali ei poistu tietokannasta. Tämän jälkeen astia ei tule esille hauissa eikä listauksissa. Tällainen toiminnallisuus tehtiin, koska asiakas haluaa säilyttää poistettujen astioiden tietoja sekä käyttötietoja pidempään kuin astia on käytössä. Koska käyttötietoihin tallentuu astian id-numero viiteavaimena, on käyttötietoja mahdotonta säilyttää, jos kemikaali poistetaan tietokannasta kokonaan. Näin ollen järjestelmänvalvoja on ainoa, joka voi poistaa kemikaalit kokonaan tietokannasta.

Tietokannassa olevia astioita voidaan sovelluksessa hakea kahdesta paikasta. Astioiden hakemiseen on luotu oma sivu, jolla voidaan hakea astioita joko

käyttäen astian numeroa tai kemikaalin nimeä. Kuviossa 8 on esitetty tämä sivu. Kemikaalin nimen mukaan hakiessa haku toimii samalla tavalla kuin aikaisemmin selitetty kemikaalihaku. Kemikaaliin liittyät astiat on myös listattu kemikaalin tarkemmissa tiedoissa, kuten kuviosta 9 voi nähdä.



Kuvio 8: Astiahaku



Kuvio 9: Astia lista kemikaalin tiedoissa

3.4.3 Astioiden käyttö

Astioiden käytön taltiointi on kemikaalitietokantasovelluksen yksi päätehtävistä. Käyttäjä kirjaa sovelluksessa ylös, kuinka paljon hän käytti kemikaalia, milloin hän käytti ja mihin projektiin. Nämä tiedot tallentuvat tietokantaan. Sovellus päivittää automaattisesti astiassa jäljellä olevan määrän syötetyn käytön perusteella.

3.4.4 Käyttäjien todennus

Jokainen käyttäjä kemiantekniikan osastolla omistaa LUT:n toimialueeseen liitetyn käyttäjätunnuksen jolla käyttäjä kirjautuu tietokoneelleen. Kemikaalitietokanta-sovellus todentaa käyttäjät käyttäen tätä samaa käyttäjätunnusta. Kun käyttäjä avaa sovelluksen, sovellus tarkistaa, onko tietokannan käyttäjätaulussa saman nimistä käyttäjää samalta toimialueelta. Jos käyttäjää ei löydy, evätään häneltä pääsy sovellukseen. Jos käyttäjä löytyy, pääsee hän käsiksi sovellukseen, mutta laajuus riippuu rooleista, jotka käyttäjälle on asetettu. Sovelluksessa on kaksi roolia: NormalUser sekä Administrator. NormalUser pääsee hakemaan, lisäämään, poistamaan ja muokkaamaan astioita sekä kemikaaleja. Administrator pääsee näiden lisäksi vielä suorittamaan lopullisia poistoja, hallinnoimaan käyttäjiä sekä hakemaan raportteja.

3.4.5 Raportit

Asiakkaan kanssa pidetyissä palavereissa tulimme siihen tulokseen, että järjestelmästä on saatava ulos neljää eri tyyppistä raporttia: vaaralausekkeiden mukaan tehtävä raportti, varastohuoneen inventaarioraportti, kemikaalien käyttöraportti sekä raportti laboratorioiden omistuksessa olevista kemikaaleista.

Vaaralausekkeiden mukaan tehtävässä raportissa käyttäjä valitsee vaaralausekkeet, joihin liitetyt astiat hän haluaa mukaan raporttiin. Järjestelmä hakee kaikki kyseisiin vaaralausekkeisiin yhdistetyt astiat ja niiden kemikaalimäärät. Suomen laissa on määritetty tietyt rajat, milloin kemikaaleista täytyy tehdä ilmoitus tai anoa lupaa kemikaalien säilyttämiseen (18). Kyseisellä raportilla saadaan helposti selvitettyä, onko rajoissa pysytty.

Varastohuoneen inventaarioraporttiin käyttäjä syöttää huoneen numeron ja järestelmä palauttaa tiedot kaikista astioista, jotka ovat säilötty kyseiseen huoneeseen.

Kemikaalien käyttöraportiin käyttäjä syöttää aikavälin, jolta raportti halutaan saada ja järjestelmä näyttää käyttäjälle kaikkien kemikaalien käyttömäärät tältä aikaväliltä.

Laboratorion omistamat kemikaalit näytetään käyttäjälle hänen valitseman laboratorion mukaan.

Raportit luodaan Microsoftin OpenXML SDK:ta hyväksikäyttäen. Järjestelmä kopioi verkkolevylle tallennetun Word-pohjan, avaa sen ja kirjoittaa SQL-haun palauttamat rivit dokumenttiin. Kuviossa 10 on esimerkki metodista joka luo inventaarioraportin

```
[HttpPost]
0 references
public ActionResult InventoryReport(string storageRoom, string unitList)
{
    List<ContainerModel> modelList = ContainerModel.GetInventoryReport(storageRoom);
    double amount = 0;
    string filePath = FilePathModel.GetFilePath();
    System.IO.File.Copy(filePath + "Inventaario.docx", "C:/Temp/Inventaario.docx", true);

    using (WordprocessingDocument doc =
        WordprocessingDocument.Open(@"C:/Temp/Inventaario.docx", true))
    {
        var body = doc.MainDocumentPart.Document.Body;
        var paras = body.Elements<Paragraph>();

        foreach (var para in paras)
        {
            foreach (var run in para.Elements<Run>())
            {
                foreach (var text in run.Elements<Text>())
                {
                    if (text.Text.Contains("hakuperuste"))
                    {
                        text.Text = text.Text.Replace("hakuperuste", storageRoom);
                    }
                    if (text.Text.Contains("hakutulos"))
                    {
                        text.Text = "";
                        Run runn = new Run();
                        Run runFooter = new Run();
                        runFooter.AppendChild(new RunProperties(new Bold()));
                        para.InsertAfter(runn, run);
                        para.InsertAfter(runFooter, runn);
                        RunProperties runProperties = run.AppendChild(new RunProperties(new Bold()));
                        run.AppendChild(new Text("Amount"));
                        run.AppendChild(new TabChar());
                        run.AppendChild(new Text("Chemical"));
                        run.AppendChild(new TabChar());
                        run.AppendChild(new Text("Container Number"));
                        run.AppendChild(new TabChar());
                        run.AppendChild(new Text("Storage Cabin"));
```

Kuvio 10: Metodi inventaarioraportin luomiseen

3.5 Sovelluksen laajuus

Vanha Microsoft Accessilla toteutettu tietokanta piti sisällään 20 taulua. Tietokannan kolmanteen normaalimuotoon saattamisen jälkeen, sekä turhien taulujen poistamisen jälkeen uuteen tietokantaan tuli 19 taulua. Koko tietokannan rakenne löytyy liitteestä 1.

Verkkosovellus koostuu 12 controller-luokasta, 18 model-luokasta sekä 45 view-sivusta. Contoller ja model-luokat sisältävät yhteensä 194 metodia ja koko sovellus koostuu noin 5000 rivistä koodia. Tarkemmat yksityiskohdat löytyvät sovelluksesta. Verkkosovelluksen toiminnallisuudesta tehtiin yksinkertaiset käyttötapauskaaviot, mutta niiden tarkempaa dokumentointia ei katsottu tarpeelliseksi, sillä sovelluksen tekijä ja suunnittelija on sama henkilö.

4 Testaus ja jatkokehitys

Tässä luvussa käydään läpi kemikaalitietokanta-sovelluksen testausprosessi sekä kemiantekniikan osastolta saadut jatkokehitysideat

4.1 Testaus

Kemikaalitietokanta-sovelluksen testaus suoritettiin kemiantekniikan osaston yhteyshenkilöiden toimesta. Sovelluksen ollessa vielä keskeneräinen, se laitettiin LUT:n palvelimille, jotta sovelluksessa valmiina olleita toiminnallisuuksia päästiin testaamaan niiden oikeassa käyttöympäristössä. Kemiantekniikan osaston yhteyshenkilöt toimittivat testaustensa perusteella listan muutettavista asioista, joiden perusteella sovellusta lähdettiin muuttamaan. Kun muutokset saatiin suoritettua, ajettiin uusi versio palvelimille ja asiakas toimitti uuden listan muutoksista.

4.2 Jatkokehitys

Asiakkaan kanssa pidetyissä kokouksissa nousi esille monia ehdotuksia järjestelmän parantamiseksi, mutta osa niistä jouduttiin jättämään opinnäytetyöstä pois aikasyistä.

Viivakoodien lukeminen astioista oli yksi ehdotus, joka nousi esille. Jokaiseen astiaan on merkitty viivakoodi, joka pitää sisällään astian tunnuksen. Tätä voisi käyttää hyväksi esimerkiksi astian käyttöä kirjatessa.

Toinen ajatus, joka nousi esille kokouksissa, oli, että sovellusta voisi lähteä kehittämään enemmän kohti varastokirjanpitoa tekemällä sovellukseen hälytysrajat. Kyseiset rajat tarkkailisivat kemikaalien määriä varastohuoneissa, ja kun määrä alittaa asetetun rajat, ilmoittaa sovellus siitä käyttäjälle. Näin kemikaalia olisi aina saatavilla, kun sitä tarvitaan.

Raportointiin tullaan tulevaisuudessa todennäköisesti myös tarvitesemaan lisää toimintoja. Tähän opinnäytetyöhön sisällytettiin vain kaikkein oleellisimmat raportit, mutta on hyvin todennäköistä että käytön edetessä huomataan uusia tarpeita. Myös raportteihin käytetyt valmiit Microsot Word-pohjat haluttaisiin tulevaisuudessa kääntää Microsoft Exceliin, jotta saaduissa raporteissa voitaisi mahdollisesti suorittaa laskutoimituksia.

Tällä hetkellä sovelluksessa ei ole käyttäjää ohjaavia toimintoja. Tällaisen tekeminen helpottaisi sovelluksen käyttöä sellaisille henkilöille jotka eivät ole aikaisemmin kyseistä sovellusta tai vanhaa sovellusta käyttäneet.

Kemikaali voi olla monessa eri muodossa, esimerkiksi nesteenä tai rakeena. Tällä hetkellä tietokannan kemikaalitauluun on syötettävä kumpikin tapaus erikseen. Tulevaisuudessa tietokantaa ja sovellusta voisi muokata niin, että kemikaali lisättäisiin vain kerran ja eri olomuodot löytyisivät kemikaalin tarkemmista tiedoista.

5 Yhteenveto

Opinnäytetyön tavoitteena oli luoda LUT:n kemiantekniikan osastolle kemikaalitietokanta sekä verkkosovellus sen käyttämistä varten. Verkkosovelluksen toivottiin olevan helppokäyttöinen. Verkkosovellus ja tietokanta on siirretty yliopiston IT-hallinnon palvelimille testikäyttöön. Kun sovelluksen viimeinen versio saadaan valmiiksi, siirtyy ylläpito vastuu LUT:n IT-osastolle sekä muutamalle admin-käyttäjälle kemiantekniikan osastolta. Kaikki

sovelluksen tulevat käyttäjät eivät ole vielä sovellusta päässeet testaamaan, joten helppokäyttöisyydestä ei tässä vaiheessa voida vielä puhua. Palaute opinnäytetyössä mukana olleilta henkilöiltä on ollut positiivista. Verkkosovelluksen kehittäminen jatkuu vielä alkukesälle 2015.

Omat tavoitteeni opinnäytetyössä oli luoda vaatimusten pohjalta toimiva ratkaisu päästä syventymään tietokantoihin, niiden luontiin, hallinnointiin sekä tietokantaohjelmointiin. Vaikka olin aikaisemmin työskennellyt samanlaisten tekniikoiden parissa, opin paljon uutta ja hyödyllistä tietokantojen käytöstä ja niiden käyttämisestä sovelluksen osana.

Vaikenta opinnäytetössä oli aikamääreiden asettaminen. Yliopiston kemiantekniikan osastolla ei ollut kiire saada tietokantaa ja sovellusta käyttöön, joten heiltä ei asetettu opinnäytetyölle määräaikaa. Näin ollen jouduin itse asettamaan aikataulun ja ajan edetessä huomasin sen olevan haastavampaa kuin luulin. Eri sovellusten osien valmistukseen kuluvaa aikaa arvoidessa arvio saattioi olla liian vähän, mutta useammin se oli liian paljon.

Verkkosovelluksen vieminen yliopiston IT-hallinnon palvelimelle aiheutti eniten harmia opinnäytetyön läpi viemisen kannalta. Toiminnallisuudet jotka olivat valmiina ja toimivat hyvin kehitysympäristössä, eivät välttämättä toimineet palvelimella.

Tämä opinnäytetyö tuotti LUT:n kemiantekniikan osastolle kemikaalitietokannan sekä sitä käyttävän verkkosovelluksen. Niiden toivotaan tulevan käyttöön kesän 2015 kuluessa.

Kuviot

Lähteet

1. Lappeenrannan teknillinen yliopisto. Esittely. http://www.lut.fi/tutustu-meihin/yliopiston-esittely. Luettu 22.2.2015

2. LUT School of Engineering Science. http://www.lut.fi/school-of-engineering-science/tutkimus. Luettu 1.6.2015

3. LUT Erotus. http://www.lut.fi/school-of-engineering-science/tutkimus/erotus. Luettu 1.6.2015

4. LUT Puhdistus. http://www.lut.fi/school-of-engineering-science/tutkimus/puhdistus. Luettu 1.6.2015

5. LUT Prosessit. http://www.lut.fi/school-of-engineering-science/tutkimus/prosessit. Luettu 1.6.2015

6. ASP.NET. http://www.w3schools.com/aspnet/aspnet.asp. Luettu 26.2.2015

7. What is ASP.NET? http://www.javascriptkit.com/howto/aspnet.shtml. Luettu 26.2.2015

8. Paz, J. 2013. Beginning ASP.NET MVC 4, 1

9. Paz, J. 2013. Beginning ASP.NET MVC 4, 7-10

10. Google. MVC Architecture. https://developer.chrome.com/apps/app_frameworks. Luettu 10.3.2015

11. Paz, J. 2013. Beginning ASP.NET MVC 4, 69

12. Paz, J. 2013. Beginning ASP.NET MVC 4, 47

13. Paz, J. 2013. Beginning ASP.NET MVC 4, 39

14. Topten Software. PetaPoco http://www.toptensoftware.com/petapoco/. Luettu 4.5.2015

15. FunctionX. Introduction to Microsoft SQL Server. http://www.functionx.com/sqlserver/Lesson01.htm. Luettu 25.3.2015

16. Granroth, K. Mitä SQL-kieli on? http://edu.phkk.fi/opiskelu/sqlkieli/SQL%20kieli.doc. Luettu 25.3.2015

17. Microsoft. Tietokannan normalisoinnin perusteiden kuvaus. https://support.microsoft.com/en-us/kb/283878/fi. Luettu 17.4.2015

18. Valtioneuvoston asetus vaarallisten kemikaalien käsittelyn ja varastoinnin valvonnasta. http://www.finlex.fi/fi/laki/alkup/2012/20120855.

Liitteet

Tietokannan rakenne

```
-- Generated by Oracle SQL Developer Data Modeler 3.3.0.747
--   at:        2015-01-14 14:47:13 EET
--   site:      SQL Server 2008
--   type:      SQL Server 2008

Create type idList as Table (
Id integer
)
go

create procedure StatementReport
 @list as dbo.idList readonly
 as
 begin
    select distinct Container.ContainerId, Container.Remaining, Con-
tainer.ContainerNumber, Container.ChemicalId, Container.Unit from Con-
tainer
    left join Chemical on Chemical.ChemicalId = Container.ChemicalId
    left join ChemicalStatement on ChemicalStatement.Chemical_Chemi-
calId = Chemical.ChemicalId
    where ChemicalStatement.Statement_StatementId in (select Id from
@list) AND Container.IsRemoved=0
 end
go

create procedure DeleteChemical(@Id int)
as
begin transaction
    begin try
        delete from ChemicalStatement where Chemical_ChemicalId = @Id
        delete from ChemicalDanger where Chemical_ChemicalId = @Id
        delete from Container where ChemicalId = @Id
        delete from ChemicalGHS where Chemical_ChemicalId = @Id
        delete from ChemSynonym where Chemical_ChemicalId = @Id
        delete from Chemical where ChemicalId = @Id
        commit
    end try
    begin catch
        rollback
    end catch
go

create procedure DeleteProject(@Id int)
as
begin transaction
    begin try
        delete from Usage where Usage.ProjectId = @Id
        delete from Project where Project.ProjectId = @Id
        commit
    end try
    begin catch
        rollback
    end catch
go
```

```
create procedure InsertSynonym(@Id int, @Name varchar(50))
as
begin transaction
    declare @SynId int
    begin try
        insert into SYNONYM values (@Name)
        set @SynId = (select SYNONYM.SynonymId from SYNONYM where SYN-
ONYM.Name=@Name)
        insert into ChemSynonym values (@Id, @SynId)
        commit
    end try
    begin catch
        rollback
    end catch
go

CREATE
  TABLE ChemSynonym
  (
    Chemical_ChemicalId INTEGER NOT NULL ,
    Synonym_SynonymId   INTEGER NOT NULL ,
    CONSTRAINT ChemSynonym__IDX PRIMARY KEY CLUSTERED (Chemical_Chemi-
calId,
    Synonym_SynonymId)
);
CREATE
  TABLE Chemical
  (
    ChemicalId   INTEGER NOT NULL IDENTITY(1,1) ,
    FinName      VARCHAR (80) NOT NULL ,
    EngName      VARCHAR (80) ,
    Formula      VARCHAR (90) ,
    CAS          VARCHAR (50) ,
    "ETY/EY-num" VARCHAR (50) ,
    WasteGlass   VARCHAR (20) ,
    MolecularMass FLOAT ,
    ProtectiveEq VARCHAR (500) ,
    Comments     VARCHAR (100) ,
    MSDS         VARCHAR (100) ,
    CONSTRAINT Chemical_PK PRIMARY KEY CLUSTERED (ChemicalId)
);

CREATE
  TABLE ChemicalDanger
  (
    DangerSign_DangerSignId INTEGER NOT NULL ,
    Chemical_ChemicalId     INTEGER NOT NULL ,
    CONSTRAINT ChemicalDanger__IDX PRIMARY KEY CLUSTERED (
    DangerSign_DangerSignId, Chemical_ChemicalId)
);

CREATE
  TABLE ChemicalGHS
  (
    Chemical_ChemicalId INTEGER NOT NULL ,
    GHSSign_GHSSignId   INTEGER NOT NULL ,
    CONSTRAINT ChemicalGHS__IDX PRIMARY KEY CLUSTERED (Chemical_Chemi-
calId,
    GHSSign_GHSSignId)
);
```

```
CREATE
  TABLE ChemicalStatement
  (
    Chemical_ChemicalId   INTEGER NOT NULL ,
    Statement_StatementId INTEGER NOT NULL ,
    CONSTRAINT ChemicalStatement__IDX PRIMARY KEY CLUSTERED (
    Chemical_ChemicalId, Statement_StatementId)
);

CREATE
  TABLE DangerSign
  (
    DangerSignId   INTEGER NOT NULL IDENTITY(1,1),
    Symbol         VARCHAR (10) NOT NULL ,
    Picture        VARCHAR (50) ,
    EngExplanation VARCHAR (50) ,
    FinExplanation VARCHAR (50) ,
    CONSTRAINT DangerSign_PK PRIMARY KEY CLUSTERED (DangerSignId)
);

CREATE
  TABLE FilePath
  (
    [Path] VARCHAR(200) NOT NULL
);


CREATE
  TABLE GHSSign
  (
    GHSSignId      INTEGER NOT NULL IDENTITY(1,1) ,
    Symbol         VARCHAR (10) NOT NULL,
    Picture        VARCHAR (50) NOT NULL ,
    EngExplanation VARCHAR (50) NOT NULL ,
    FinExplanation VARCHAR (50) NOT NULL ,
    SweExplanation VARCHAR (50) NOT NULL ,
    CONSTRAINT GHSSign_PK PRIMARY KEY CLUSTERED (GHSSignId)
);

CREATE
  TABLE Laboratory
  (
    LaboratoryId INTEGER NOT NULL IDENTITY(1,1) ,
    Name         VARCHAR (100) NOT NULL ,
    CONSTRAINT Laboratory_PK PRIMARY KEY CLUSTERED (LaboratoryId)
);

CREATE
  TABLE Producer
  (
    ProducerId    INTEGER NOT NULL IDENTITY(1,1),
    Name          VARCHAR (50) NOT NULL ,
    Email         VARCHAR (50) ,
    WebPage       VARCHAR (50) ,
    Phone         VARCHAR (15) ,
    ContactPerson VARCHAR (40) ,
    CONSTRAINT Producer_PK PRIMARY KEY CLUSTERED (ProducerId)
);

CREATE
  TABLE Project
```

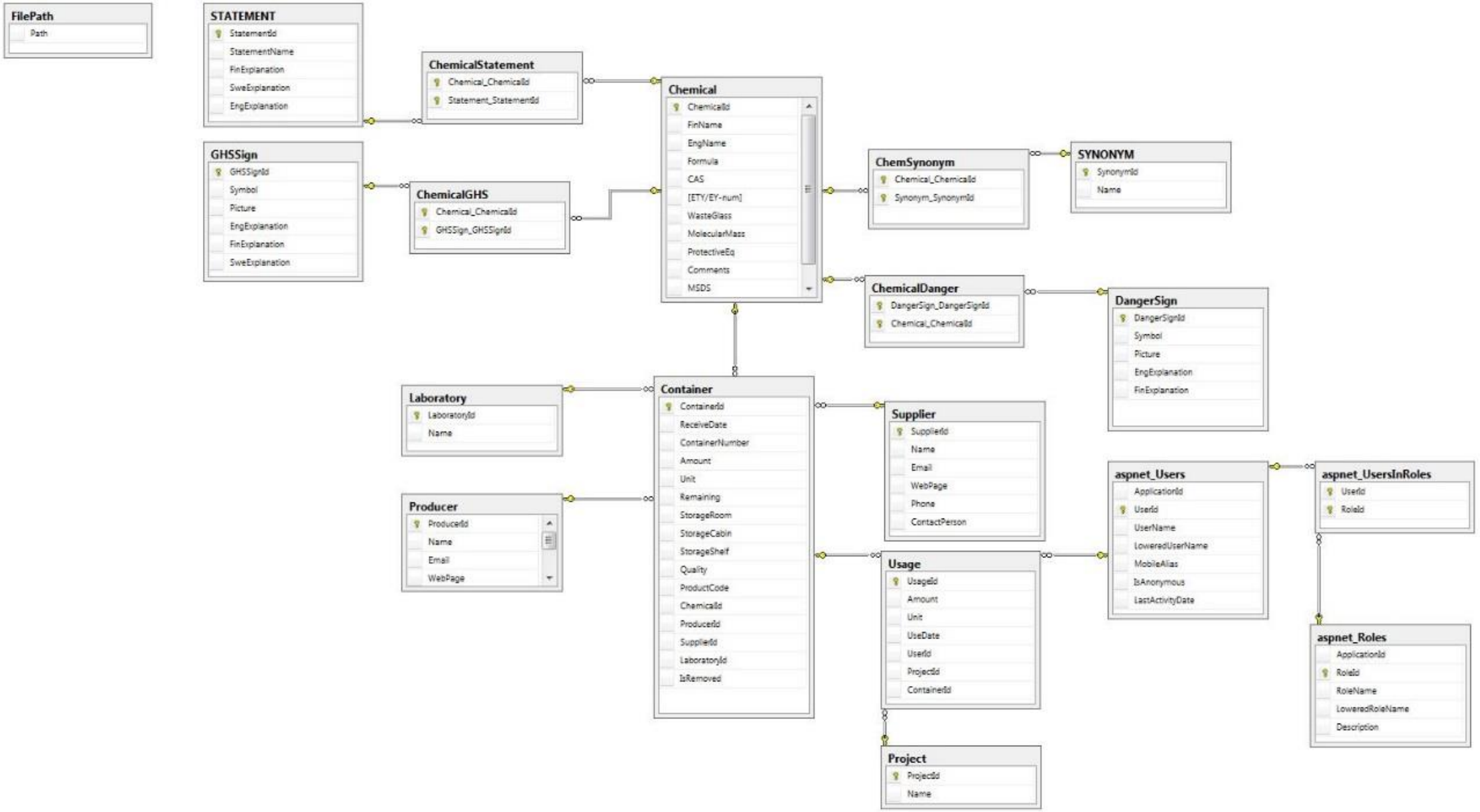
```
  (
    ProjectId INTEGER NOT NULL IDENTITY(1,1),
    Name      VARCHAR (100) NOT NULL ,
    CONSTRAINT Project_PK PRIMARY KEY CLUSTERED (ProjectId)
);

CREATE
  TABLE STATEMENT
  (
    StatementId    INTEGER NOT NULL IDENTITY(1,1),
    StatementName VARCHAR (15) NOT NULL,
    FinExplanation VARCHAR (250) NOT NULL ,
    SweExplanation VARCHAR (250) NOT NULL ,
    EngExplanation VARCHAR (250) NOT NULL ,
    CONSTRAINT Statement_PK PRIMARY KEY CLUSTERED (StatementId)
);

CREATE
  TABLE Supplier
  (
    SupplierId    INTEGER NOT NULL IDENTITY(1,1),
    Name          VARCHAR (50) NOT NULL ,
    Email         VARCHAR (50) ,
    WebPage       VARCHAR (50) ,
    Phone         VARCHAR (15) ,
    ContactPerson VARCHAR (40) ,
    CONSTRAINT Supplier_PK PRIMARY KEY CLUSTERED (SupplierId)
);

CREATE
  TABLE SYNONYM
  (
    SynonymId INTEGER NOT NULL IDENTITY(1,1),
    Name      VARCHAR (50) NOT NULL ,
    CONSTRAINT Synonym_PK PRIMARY KEY CLUSTERED (SynonymId)
);

CREATE
  TABLE Usage
  (
    UsageId INTEGER NOT NULL IDENTITY(1,1),
    Amount Float NOT NULL ,
    Unit      VARCHAR (2) ,
    UseDate   DATE NOT NULL ,
    UserId    uniqueidentifier ,
    ProjectId INTEGER ,
    ContainerId  INTEGER ,
    CONSTRAINT Usage_PK PRIMARY KEY CLUSTERED (UsageId)
);

CREATE
  TABLE Container
  (
    ContainerId  INTEGER NOT NULL IDENTITY(1,1),
    ReceiveDate DATE NOT NULL,
    ContainerNumber VARCHAR (10) NOT NULL,
    Amount FLOAT NOT NULL ,
    Unit VARCHAR (2) NOT NULL ,
    Remaining FLOAT NOT NULL ,
    StorageRoom  VARCHAR (20) ,
    StorageCabin VARCHAR (20) ,
```

```
    StorageShelf VARCHAR (20) ,
    Quality      VARCHAR (100) ,
    ProductCode  VARCHAR (20) ,
    ChemicalId   INTEGER ,
    ProducerId   INTEGER ,
    SupplierId   INTEGER ,
    LaboratoryId INTEGER ,
    IsRemoved BIT NOT NULL,
    CONSTRAINT Container_PK PRIMARY KEY CLUSTERED (ContainerId)
);

ALTER TABLE ChemSynonym
ADD CONSTRAINT FK_ASS_2 FOREIGN KEY
(
Chemical_ChemicalId
)
REFERENCES Chemical
(
ChemicalId
)

ALTER TABLE ChemSynonym
ADD CONSTRAINT FK_ASS_3 FOREIGN KEY
(
Synonym_SynonymId
)
REFERENCES SYNONYM
(
SynonymId
)

ALTER TABLE ChemicalDanger
ADD CONSTRAINT FK_ASS_4 FOREIGN KEY
(
DangerSign_DangerSignId
)
REFERENCES DangerSign
(
DangerSignId
)

ALTER TABLE ChemicalDanger
ADD CONSTRAINT FK_ASS_5 FOREIGN KEY
(
Chemical_ChemicalId
)
REFERENCES Chemical
(
ChemicalId
)

ALTER TABLE ChemicalGHS
ADD CONSTRAINT FK_ASS_6 FOREIGN KEY
(
Chemical_ChemicalId
)
REFERENCES Chemical
(
ChemicalId
)
```

```
ALTER TABLE ChemicalGHS
ADD CONSTRAINT FK_ASS_7 FOREIGN KEY
(
GHSSign_GHSSignId
)
REFERENCES GHSSign
(
GHSSignId
)

ALTER TABLE ChemicalStatement
ADD CONSTRAINT FK_ASS_8 FOREIGN KEY
(
Chemical_ChemicalId
)
REFERENCES Chemical
(
ChemicalId
)

ALTER TABLE ChemicalStatement
ADD CONSTRAINT FK_ASS_9 FOREIGN KEY
(
Statement_StatementId
)
REFERENCES STATEMENT
(
StatementId
)

ALTER TABLE Usage
ADD CONSTRAINT Usage_Project_FK FOREIGN KEY
(
ProjectId
)
REFERENCES Project
(
ProjectId
)

ALTER TABLE Usage
ADD CONSTRAINT Usage_User_FK FOREIGN KEY
(
UserId
)
REFERENCES aspnet_Users
(
UserId
)

ALTER TABLE Usage
ADD CONSTRAINT Usage_Container_FK FOREIGN KEY
(
ContainerId
)
REFERENCES Container
(
ContainerId
)

ALTER TABLE Container
```

```
ADD CONSTRAINT Container_Chemical_FK FOREIGN KEY
(
ChemicalId
)
REFERENCES Chemical
(
ChemicalId
)

ALTER TABLE Container
ADD CONSTRAINT Container_Laboratory_FK FOREIGN KEY
(
LaboratoryId
)
REFERENCES Laboratory
(
LaboratoryId
)

ALTER TABLE Container
ADD CONSTRAINT Container_Producer_FK FOREIGN KEY
(
ProducerId
)
REFERENCES Producer
(
ProducerId
)

ALTER TABLE Container
ADD CONSTRAINT Container_Supplier_FK FOREIGN KEY
(
SupplierId
)
REFERENCES Supplier
(
SupplierId
)
go

create trigger synDelete on ChemSynonym
after delete
as
    declare cursorId cursor for select Synonym_SynonymId from deleted
    declare @Id int
    open cursorId
    fetch next from cursorId into @Id
    while @@FETCH_STATUS=0
    begin
        delete from SYNONYM where SynonymId=@Id
    fetch next from cursorId into @Id
    end
    close cursorId
    deallocate cursorId
go

-- Oracle SQL Developer Data Modeler Summary Report:
--
-- CREATE TABLE                            17
-- CREATE INDEX                             0
-- ALTER TABLE                             20
```

```
-- CREATE VIEW                          0
-- CREATE PACKAGE                       0
-- CREATE PACKAGE BODY                  0
-- CREATE PROCEDURE                     0
-- CREATE FUNCTION                      0
-- CREATE TRIGGER                       0
-- ALTER TRIGGER                        0
-- CREATE DATABASE                      0
-- CREATE DEFAULT                       0
-- CREATE INDEX ON VIEW                 0
-- CREATE ROLLBACK SEGMENT              0
-- CREATE ROLE                          0
-- CREATE RULE                          0
-- CREATE PARTITION FUNCTION            0
-- CREATE PARTITION SCHEME              0
--
-- DROP DATABASE                        0
--
-- ERRORS                               0
-- WARNINGS                             0
```


FilePath
Path
STATEMENT
StatementId
StatementName
FinExplanation
SweExplanation
EngExplanation
ChemicalStatement
Chemical_ChemicalId
Statement_StatementId
Chemical
ChemicalId
FinName
EngName
Formula
CAS
[ETY/EY-num]
WasteGlass
MolecularMass
ProtectiveEq
Comments
MSDS
GHSSign
GHSSignId
Symbol
Picture
EngExplanation
FinExplanation
SweExplanation
ChemicalGHS
Chemical_ChemicalId
GHSSign_GHSSignId
ChemSynonym
Chemical_ChemicalId
Synonym_SynonymId
SYNONYM
SynonymId
Name
ChemicalDanger
DangerSign_DangerSignId
Chemical_ChemicalId
DangerSign
DangerSignId
Symbol
Picture
EngExplanation
FinExplanation
Laboratory
LaboratoryId
Name
Container
ContainerId
ReceiveDate
ContainerNumber
Amount
Unit
Remaining
StorageRoom
StorageCabin
StorageShelf
Quality
ProductCode
ChemicalId
ProducerId
SupplierId
LaboratoryId
IsRemoved
Supplier
SupplierId
Name
Email
WebPage
Phone
ContactPerson
Producer
ProducerId
Name
Email
WebPage
Usage
UsageId
Amount
Unit
UseDate
UserId
ProjectId
ContainerId
Project
ProjectId
Name
aspnet_Users
ApplicationId
UserId
UserName
LoweredUserName
MobileAlias
IsAnonymous
LastActivityDate
aspnet_UsersInRoles
UserId
RoleId
aspnet_Roles
ApplicationId
RoleId
RoleName
LoweredRoleName
Description