Peter Virtanen

Kemikaalitietokannan hallintaohjelman suunnittelu

Tietotekniikan koulutusohjelma
Ohjelmistotekniikan suuntautumisvaihtoehto
2009

KEMIKAALITIETOKANNAN HALLINTAOHJELMAN SUUNNITTELU

Virtanen, Peter
Satakunnan ammattikorkeakoulu
Tekniikan koulutusohjelma
Syyskuu 2009
Kivi, Karri
Sivumäärä:33

Asiasanat: PHP, Mysql

Tämän opinnäytetyön aiheena oli kemikaalitietokannan hallintaohjelman toteutus Porin ammattiopiston prosessi-, kemian ja materiaalitekniikan osastolle. Opinnäytetyö jaettiin kahteen osaan, suunnittelu ja toteutus. Toisen osan eli toteutus-osan toteutti Yrjö Hyyti. Tämä työ käsittelee Porin ammattiopistolle käyttöön tulleen järjestelmän tietokannan ja järjestelmän suunnittelua. Järjestelmän suurin käyttäjäkunta tulee olemaan n.15-19 -vuotiaat opiskelijat. Suunnittelussa erityinen huomio kiinnitettiin helppokäyttöisyyteen, ei ulkoasuun. Opinnäytetyössä käsitellään myös ensimmäisen oikean ohjelmistoprojektin vaikeuksista ja siitä, mitä projekti opetti tulevaisuuden projekteja silmällä pitäen.

DESIGNING WWW-BASED MAINTENANCE TOOL FOR CHEMICAL DATABASE

Virtanen, Peter
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Information Technology
September 2009
Kivi, Karri
Number of pages:33

Key words: PHP, Mysql

The theme of this thesis was to create a chemical database management program to Chemical and Material Engineering departments of Pori College. This thesis was divided into two parts, designing and execution. The other part, ie execution part, was made by Yrjö Hyyti. This thesis concentrates on designing the system and database which were introduced in Pori College. The main users of the system will be mainly 15-19 year old students. In designing the system the special attention was paid to the user friendliness of the system, less to its appearance. In addition this thesis tells about difficulties that the first real programming project brought and what was learned from it regarding future projects.

SISÄLLYS

SYMBOLI- JA TERMILUETTELO

HTML(*Hypertext Markup Language*) on avoimesti standardoitu kuvauskieli, jolla voidaan kuvata hyperlinkkejä sisältävää tekstiä eli hypertekstiä. HTML tunnetaan erityisesti kielenä, josta Internet-sivut rakentuvat.

JavaScript on Netscape Communications Corporationin kehittämä pääasiassa Web-ympäristössä käytettävä komentosarjakieli. JavaScriptin tärkein ominaisuus on mahdollisuus lisätä Web-sivuille dynaamista toiminnallisuutta.

MySQL on suosittu ja tehokas SQL-tietokantojen hallintajärjestelmä.

PHP (*Hypertext Preprocessor*) on Perlin kaltainen ohjelmointikieli, jota käytetään erityisesti Web-palvelinympäristöissä dynaamisten web-sivujen luonnissa. PHP on komentosarjakieli, jossa ohjelmakoodi tulkitaan vasta ohjelman suoritusvaiheessa. PHP:tä voidaan käyttää useilla eri alustoilla ja käyttöjärjestelmillä.

XAMPP on helposti asennettava ohjelmisto-paketti, joka sisältää Apache-palvelimen päällä toimivat MySQL:n, PHP:n ja Perl:in.

LDAP (Lightweight Directory Access Protocol) on hakemistopalvelujen käyttöön tarkoitettu verkkoprotokolla. LDAP:in käyttötarkoitus on pääasiassa käyttäjäntunnistus. Sitä tukevat useimmat UNIX-järjestelmät ja Microsoftin Active Directory käyttää LDAP:ia pohjanaan Kerberoksen ohella. LDAP soveltuu myös käyttöoikeuden tarkistamiseen.

1. JOHDANTO

Lähtökohtana tälle työlle oli se, että ammattikoulun kemianopiskelijoilla kului suuri aika eri kemikaalipakkausten hakemiseen. Kemikaaleja haettiin A3-kokoon tulostetun Exel-taulukon avulla. Taulukoissa oli monta saraketta ja A3-lomakkeita oli todella monta. Kemikaalien etsiminen ei siis ollut helppoa.

Ammattikoululta kysyttiin SAMK:sta tietoteknistä ratkaisua heidän ongelmaansa.

Exel-taulukossa oli jonkun verran virheitä ja siinä oli turhia välilehtiä. Kemikaaleja taulukkoon oli kuitenkin kirjattu yli tuhat, joten koko taulukon uusiksi kirjaaminen olisi vaatinut suuren ajan.

Alkukäsitys ammattikoululla oli, ettei olemassa olevaa Exel-taulukkoa olisi mahdollista yhdistää uuteen järjestelmään ja että se vaatisi suuren työmäärän.

Kokouksien jälkeen päädyimme siihen, että asiakkaan toiveista huolimatta jo olemassa olevaa Exel-taulukkoa tultaisiin hyödyntämään uudessa järjestelmässä.

On selvää, että valmis järjestelmä tulee nopeuttamaan tehtävässään töitä huimasti. Järjestelmän tehokas käyttö vaatiikin järjestelmän hyvän tuntemuksen. Haasteena suunnittelussa onkin se, miten saadaan mahdollisimman helppokäyttöinen ja mahdollisimman nopea järjestelmä. Suunnittelussa pitää myös muistaa se, että järjestelmä pitää olla helposti uudistettavissa. Järjestelmän uudistaminen pitää onnistua yhtä helposti järjestelmän alkuperäisten tekijöiden tai täysin ulkopuolisten käsien kautta.

Laajojen järjestelmien toteuttaminen oli täysin uusi asia, kun lähdimme toteuttamaan työtä. Osasimme kuitenkin aikatauluttaa työn hyvin, ja se olikin suurin syy siihen, miten saimme työn nopeasti valmiiksi.

1.1 Hallintajärjestelmä

”Tietokantajärjestelmä säilyttää tietoja. Ihmisen tehtävä on määrittää tietojen muotoa, käyttökelpoisuutta ja luotettavuutta rajaavat säännöt. Järjestelmä tarjoaa ihmisille, tietojen käyttäjille, mahdollisuudet määrittää, lisätä, hakea, poistaa ja muuttaa tietoja joillakin ilmaisuilla, jonkin täsmällisen kielen komennoilla. Tietokantajärjestelmän pitää noudattaa sille annettuja sääntöjä ja valvoa, että myös käyttäjät noudattavat omia sääntöjään.” /1, 75/

Tietokantojen toiminta, niin kuin monen muunkin järjestelmän toiminta on riippuvainen eheästä sisällöstään. Eheä sisältö ei kuitenkaan ole taattua, sillä usein asiakasprojekteissa tietokantajärjestelmät tulevat suuren käyttäjäjoukon käyttöön. Suurien käyttäjäkuntien takia järjestelmän kehityksessä pitää keskittyä helppokäyttöisyyteen. Helppokäyttöisyyden ei tarvitse silti tarkoittaa sitä, että järjestelmän sisällön pitäisi olla yksinkertainen.

Suurille käyttäjäkunnille ja pienemmissäkin projekteissa on tärkeää, että toiminnallisuus taataan sopimalla yhtenäiset säännöt tietokannan täyttämiseen. Kun järjestelmää luodaan ja kantaan syötetään samaan sarakkeeseen monta eri arvoa, on tärkeää että asiakkaalle on opastettu tarkoin kuinka kukin kohta täytetään. Paras tilanne olisi se, että järjestelmään ei tarvitsisi tehdä mitään sääntöjä käyttäjälle muistettavaksi.

Esim. eräällä kemikaalilla on viisi eri pakkauskokoa ja ne ilmoitetaan samalla rivillä samassa sarakkeessa. Eri pakkauskoot pitää erottaa jollakin erityisellä merkillä esim. pilkulla. Kun eri kemikaalit erotetaan pilkulla, voidaan koodauskielestä riippumatta saada järjestelmä ymmärtämään se, että pilkusta pilkkuun oleva arvo on yksi oma arvonsa. Järjestelmään lisätään syöttölomakkeeseen asetus, joka tarkistaa, että syötettävä arvo pitää olla tiettyä muotoa (kuten tässä esim. 100ml, 200ml, 500ml). Tällä varmistetaan se, ettei vääriä erotusmerkkejä kantaan pääse syntymään.

Kun järjestelmän käyttö opastetaan ensimmäistä kertaa, asiakas ei tietenkään opi kaikkea kerralla. Käyttöohjeetkin yleensä pölyyntyvät kirjahyllyssä. Tietokantojen eheys voidaan

kuitenkin säilyttää hyvin ohjelmoiduilla syöttölomakkeilla. Järjestelmä ei yksinkertaisesti vain anna käyttäjän täyttää lomakkeita väärin. Väärin täytettäessä järjestelmä voi vielä antaa ohjeen kohdan oikeaoppiseen täyttämiseen, näin virheen sattuessa käyttäjän ei tarvitse edes vilkaista ohjekirjaa.

2. JÄRJESTELMÄLLE ASETETTUJA VAATIMUKSIA

”Vaatimusten analysointivaiheessa tarkastellaan mallinnettavaa yritystä ja haastatellaan käyttäjiä ja johtoa nykyisen järjestelmän ja tulevien tarpeiden analysoimiseksi sekä selvitetään liikeyritysten yleiset tietotarpeet.” /2, 28/

Kaikille järjestelmille asetetaan tietyt alkukriteerit. Tässäkin työssä asetettiin paljon kriteeterejä, jotka valmiin työn pitää täyttää. Työn alkuvaiheessa meillä ei ollut montaakaan kriteeriä, vaan niitä alkoi tippua pikkuhiljaa työn edetessä. Myös suunnittelutyö tapahtui osin pätkissä. Tarvittava suunnittelutyö tehtiin aina tarpeen vaatiessa. Huomasin, että tulevissa projekteissa pitää vaatia enemmän määrityksiä tilaajalta. Työ oli totetutettava määrätyssä ajassa, mutta ilman määrityksiä on mahdotonta täysin taata, että valmis järjestelmä vastaa asiakkaan tarpeita.

Järjestelmän vaatimusten asettamista vaikeutti sekin, että kemikaalien varoitusmerkit olivat muuttumassa uusiin ja erilaisten vaaraa ilmoittavien lauseiden muodosta ja määrästäkään ei ollut varmuutta. Koska ei ollut varmaa, mitä uudet vaatimukset sisälsivät, lähdimme toteuttamaan järjestelmää jossa olisi mahdollisimman hyvät muokkausmahdollisuudet uusille vaatimuksille.

Alkumäärityksissä oli selvää myös se, että järjestelmään tulee kaksi käyttäjätasoa: opettaja ja oppilas.

Asiakas toivoi, että heidän jo olemassa olevat tunnukset olisivat käytössä tässäkin järjestelmässä. Ammattikoulun järjestelmissä on LDAP-kirjautuminen, jossa jokaisella

oppilaalla ja opettajalla on oma tunnuksensa. Tällä kirjautumisominaisuudella oli se idea, että nähtäisiin kuka mahdollisesti oli tehnyt pahojaan järjestelmälle.

2.1 Opettajien vaatimukset

Järjestelmän tärkeimmät ominaisuudet oli helppo määritellä.

Opettajien tuli kyetä muokkaamaan kemikaalin kaikkia tietoja ja lisäämään uusia kemikaaleja. Opettajien tuli myös pystyä lisäämään ja muokkaamaan järjestelmässä tarvittavia vaaraa ja turvallisuustoimenpiteitä merkitseviä lauseita.

Yleinen idea oli se, että opettaja olisi järjestelmän ylläpitäjä, joka pystyisi päivittämään järjestelmää helposti ja nopeasti. Ylläpitoon tarvittiin vain työkalu, jolla kemikaaleihin pääsisi käsiksi kaikilla koulun tietokoneilla, ja josta kaikkia tarvittavia toimenpiteitä voitaisiin suorittaa.

Opettajien vaatimuksiin kuului myös raporttien tekeminen. Opettajien piti pystyä tekemään raportti mistä vain kemikaalista. Raportissa tulisi näkyä kaikki aineen ominaisuudet ja uudet ja vanhat varoitusmerkit. LDAP-kirjautuminen olisi ollut opettajille tarpeellinen. Kuitenkin ajan vähyyden vuoksi kokouksissa päätettiin, ettei LDAP-kirjautumista toteutettaisi.

Opettajat tarvitsevat kuitenkin kirjautumismahdollisuuden, sillä jos kannassa ei olisi mitään kirjautumista, olisi se jo alkuvaiheessa altis ilkivallalle. Päätettiin siis, että kirjautuminen toteutetaan istunto-tekniikalla. Alkujaan ideana oli, että kaikille opettajille tehtäisiin omat tunnukset. Tästä ideasta kuitenkin luovuttiin, sillä kannan sisällä ei ole kenenkään henkilökohtaisia asioita. Päätettiin, että kannan muokkaukseen tullaan käyttämään vain yhtä salasanaa, jonka kaikki opettajat tietävät. Vaaranahan on, että oppilaat saavat tietää salasanan ja menevät tekemän pahojaan kannalle. Tässä ensimmäisessä versiossa kuitenkin sallittiin tämänkaltainen ratkaisu, josta jo alkutekijöissään tiedetään, ettei se oli kovin pitkäikäinen. Toisessa versiossa tunnukset saataisiin istuntoihin LDAP-tekniikalla.

Istunnossa LDAP-palvelimelta kysytään, ovatko tunnus ja salasana oikein ja mikäli ne ovat, järjestelmään kirjaudutaan. Ensimmäisessä järjestelmäversiossa kirjautuminen tapahtuu yhdellä salasanalla, joka on kirjoitettu PHP-koodiin. Mikäli salasana, joka

kirjoitettiin on oikein, järjestelmään kirjaudutaan. Tämä vastaa teoriassa LDAP-kirjautumista, mutta ei ole yhtä hienostunut.
Ensimmäisissä kokouksissa luovuttiin siitä ideasta, että järjestelmää varten tehtäisiin omat tunnukset. Ensinnäkin tunnukset toisivat ylläpidolle lisää työtä ja uusien oppilaiden tarvitsisi tehdä kahdet tunnukset, sekä jo olevaan järjestelmään että tähän uuteen järjestelmään. Olisi turhaa luoda järjestelmälle omat tunnukset josta on suuri vaiva kun samalla työmäärällä saataisiin tehtyä vanhoilla LDAP-tunnuksin toimiva kirjautuminen. Yhden salasanan järjestelmä kuulostaa ja on niin kätevä, että kaikista muista kirjautumistoteutuksista päätettiin ensimäisessä versiossa luopua.

2.2 Oppilaiden vaatimukset

Oppilaiden vaatimuksissa oli kemikaalien haku, tietyn kemikaalin määrän muokkaaminen ja raporttien tekeminen. Raportilla tarkoitetaan tietystä kemikaalista tehtyä ominaisuuslistaa varoitusmerkkeineen. Oppilaille ja opettajille ei toteuteta omaa hakua. Opettajat käyttävät tätä käyttäjätasoa silloin kun mitään muokkauksia ei tarvitse tehdä, lukuun ottamatta kappalemäärän muokkausta, joka ei vaadi kirjautumista.

2.3 Viranomaisvaatimukset

Näihin vaatimuksiin kuului lähinnä REACH-asetuksen antamat määritteet.
”REACH-asetus on Euroopan parlamentin ja -neuvoston asetus N:o 1907/2006 kemikaalien rekisteröinnistä, arvioinnista, lupamenettelyistä ja rajoituksista, joka tuli voimaan kesäkuun 1. päivänä 2007. Lyhenne REACH tulee sanoista Registration, Evaluation, Authorisation and Restriction of CHemicals. REACH-asetuksella korvataan noin 40 eri säädöstä ja se on suoraan jäsenmaita sitovaa lainsäädäntöä.

Tärkeimpänä tavoitteena asetukselle on varmistaa terveyden- ja ympäristönsuojelun korkea taso, tehostaa EU:n kemianteollisuuden kilpailukykyä, edistää vaihtoehtoisten menetelmien kehittämistä aineiden vaarojen arvioimiseksi sekä taata tavaroiden vapaa liikkuvuus Euroopan unionin sisämarkkinoilla.

Keinoina tavoitteiden toteuttamisessa ovat aineiden rekisteröinti, tiettyjen aineiden arviointi, vaarallisimpien aineiden lupamenettely sekä ns. suojaverkkona kemikaalien rajoitukset. Järjestelmä perustuu aineiden ja kemikaaliseosten sisältämien aineiden sekä tietyissä tapauksissa myös esineissä olevien aineiden riskinhallintaan.

REACH asettaa entistä enemmän vastuuta teollisuudelle, kun on kyse riskeistä, joita kemikaalit saattavat aiheuttaa terveydelle ja ympäristölle.

Käyttöturvallisuustiedote on asiakirja, jolla välitetään tietoa aineen tai valmisteen ominaisuuksista, riskeistä sekä turvallisesta käytöstä teollisuus- tai ammattikäyttöön.” /3/

2.4 Omat vaatimukset

Omat vaatimukset tulevat olemaan lähinnä laadullisia. Laatu pitää vastata sellaista tasoa, että järjestelmää voi ylpeänä esitellä vaikkapa ansiona CV:ssä. Järjestelmässä kannattaa myös pitää sellainen laatu yllä, että jatkokehityksellä siitä voi saada jopa kaupallisen. Täysin vastaavaa järjestelmää ei markkinoilla ole. Työ ei tule kuitenkaan olemaan pioneerityötä, vaan tämän järjestelmän kaltaisia PHP-sovelluksia on paljon. Järjestelmän luominen vain on ainutlaatuista, vaikka samankaltaisuuksia löytyisikin. Internetistä ei löytynytkään yhtään suoraa ratkaisua mahdollisiin ongelmiimme.
Järjestelmä tulee kemikaaleille, mutta valmistuttuaan järjestelmän kemikaalit voidaan muuttaa mahdollisesti minkä vain yrityksen tuotteiksi. Järjestelmä onkin tältä osin alallaan erityinen. Lähin vastaava järjestelmä on Kemiarvi. Kemiarvi MS Access-pohjainen järjestelmä kemikaaleille. Järjestelmä olikin ammattikoulun valintalistoilla, mutta sen hinta ei miellyttänyt. Access-pohjaiset ratkaisut tarkoittavat samalla myös sitä että ohjelma pitää asentaa monelle koneelle erikseen. Järjestelmää ei voi etäkäyttää samoin kuin ammattikoululle tulevaa järjestelmää. Tietysti Kemiarvissa olisi paljon hyviä puolia käytännössä, mutta jos ajatellaan järjestelmää, jossa ei tarvitse olla mitään hienouksia, on tekemämme järjestelmä parempi vaihtoehto. Järjestelmämme on myös

ilmainen ammattikoululle, eikä mitään lisenssejä tarvita, kuten vastaavissa kaupallisissa tarvitaan.
Omat vaatimukset voi tiivistää niin, että teemme järjestelmän, josta voi olla ylpeä.

3. JÄRJESTELMÄN TARVITSEMIEN TYÖKALUJEN VALINTA JA SUUNNITTELU

”Tietokantasovelluksella (eli tietokantajärjestelmällä) tarkoitetaan tietokannassa olevaa tietoa sekä ohjelmistoa, jonka avulla tietoja voidaan hakea, lisätä, päivittää ja poistaa. Ohjelmisto koostuu sovellusohjelmistosta sekä systeemiohjelmistosta (esim. tietokannan hallintajärjestelmä, väliohjelmisto, selainohjelmisto, tietoliikenneohjelmisto).” /4/

Järjestelmä on helpoin toteuttaa Mysql-tietokantaan. Toinen tarvittava osa on PHP. PHP:llä ja Mysql-tietokannalla yhdessä on helppo toteuttaa tietokanta ja sen hallintaohjelma. Tietokantaa on helppo käyttää koulun sisäverkossa PHP:llä tehdyllä hallintaohjelmalla.
Aivan aluksi mietittiin, mitä ongelmia valitut osat tuottavat työssä. Mysql ei välttämättä tue skandinaavisia merkkejä ja koulussa on opetettukin välttämään niiden käyttöä. Koska jo ennalta tiedettiin, että skandinaavisten aakkosten käyttäminen kannassa olisi epävarmaa, niiden käyttämisestä kannan tärkeimmissä osissa luovuttiin. Tärkeimpiin osiin kuuluu erityisesti pääavaimet ja kaikkien sarakkeiden nimet. Ammattiopiston toivomus kuitenkin oli, että tietokantaohjelmisto olisi mahdollisimman suomenkielinen. PHP:llä voidaan ohjelmoida tietokannasta otetut nimet näkymään erinäköisenä. Esim. kemikaalin ensimmäinen ominaisuus on kem_id, se oli opettajien mielestä huono nimi, sillä se ei ole oikea sana. Myös alaviiva nimessä näyttää omituiselta. Kuten sanottu PHP:n vahvuus on siinä, että se on ohjelmointikieli. PHP:llä saisimme kemikaalien ominaisuudet näkymään eri nimillä kuin ne olisi nimetty kannassa. Mukaanlukien kem_id nimettiin ”Avaimeksi”, joka oli helpompi käsittää.

Yhtenä PHP:n vahvuuksina pitää vielä mainita se, että sillä luodut sovellukset toimivat mukisematta kaikilla selaimilla. Tämä johtuu siitä, että PHP on palvelinpään ominaisuus. Oli toteutus mikä tahansa, PHP toimii yhtä varmasti kaikilla selaimilla. Mysql ei tuota mitään ristiriitaisuuksia selaimissa, sillä selain ei ”näe” sitä. Ainoastaan PHP on se osa, joka kyselee tarvittavat tiedot Mysql-kannasta ja tulostaa ne pyynnöstä ruudulle. Myöskään PHP:n päivittäminen ei ole käyttäjän niskoilla, koska PHP on palvelinpään sovellus. Esimerksi Flash vaatii käyttäjältä yhtä mittaa päivityksiä. Flash olisikin ollut tämän järjestelmän toteutukseen oiva kieli, mutta se on hyvin raskas ja kuten edellä mainittiin, se vaatii päivittämistä käyttäjän toimesta. Flash:hän ei ole ohjelmointikieli, vaan käyttää ActionScript-kieltä toimiakseen. Flash mahdollistaa näyttäviä esityksiä ruudulle ja on myös oiva työkalu erilaisten taulukkojen näyttämiseen. Ongelmina on vain sen suuri konetehojen tarve ja se, että sillä tehdyt sovellukset latautuvat kauan. Tälle järjestelmälle on täysin yhdentekevää, onko se näyttävä vai ei. Ainoastaan vaadittu informaatio pitää näkyä ilman suurempia hienouksia. Tässä syitä siihen miksi Flash jätettiin valitsematta järjestelmän toteutustyökaluksi. Valintaan vaikuttivat myös se, että järjestelmä ei saanut maksaa juuri mitään. Mysql on ilmainen ja suosittu tietokantajärjestelmä, joka oli helppo valinta tämän takia.

4. KEMIKAALINOMINAISUUDET

Yksi kemikaali sisältää monta ominaisuutta.

-tuotenimi suomeksi ja englanniksi
-CAS-numeron
-kemikaalin kaavan
-varaston
-paikan varastossa
-hyllyn
-kappalemäärän

-pakkauskoon

-aineosat

-R-, H-, S- ja P-lauseet

-käyttöturvatiedotteen eli KTT:n päivämäärän

-käyttötarkoituksen ja paikan

-suurimman käyttömäärän tiedot

-suurimman varastoidun määrän tiedot

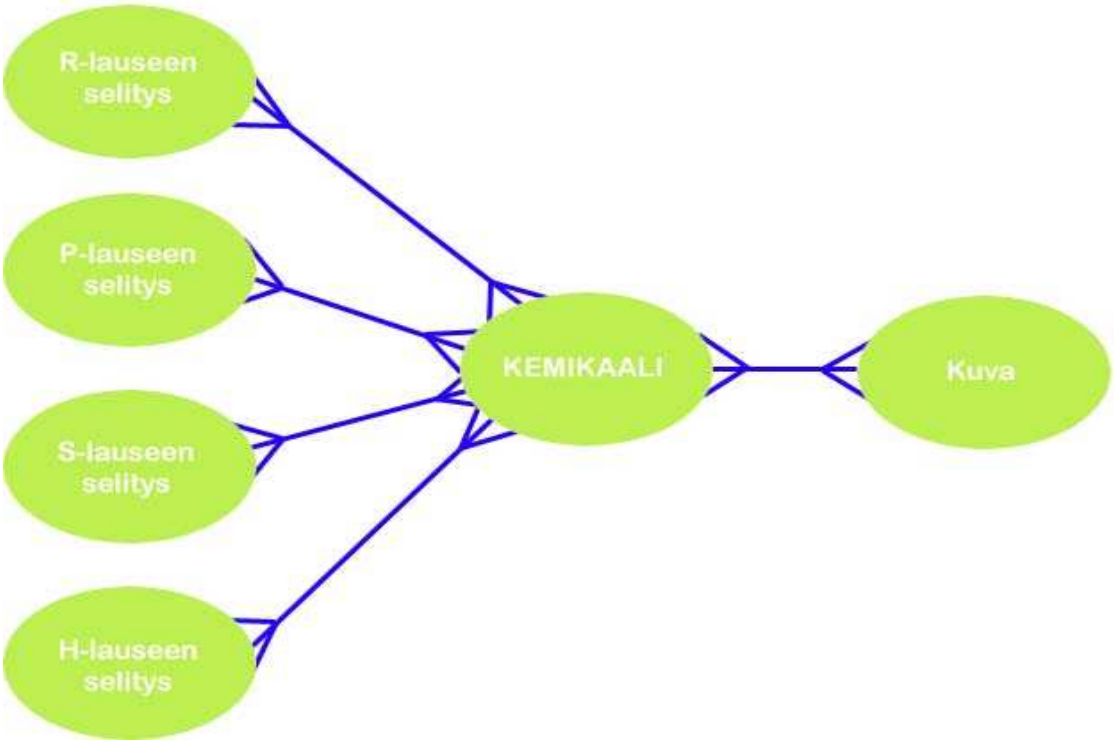
-toimittajan tiedot

-varoitussymbolia vastaava kirjain

-kaksi saraketta lisätiedoille

-Lisäksi kemikaalilla on maksimissaan 10 uutta ja vanhaa varoitusmerkkiä ja pdf-muodossa oleva käyttöturvatiedote(KTT).

4.1 Kemikaali-taulun ja muiden tarvittavien taulujen yhteys



Kuva 1. Yhteydet

Kaikki yhteydet ovat monen suhde moneen. Periaate on: yhdellä kemikaalilla on monta (varoitus)kuvaa ja yksi kuva voi kuulua monelle kemikaalille.
Kuvassa 1. on kuvattu yksi kemikaali. Kyseisellä kemikaalilla on monta varoituskuvaa ja kemikaali sisältää monta R-, H-, S-, ja P-lausetta. Esim. yhdellä R-lauseelle voi olla vain yksi selitys, mutta yhdellä kemikaalilla voi olla monta R-lausetta. Tästä johtuu monen suhde moneen yhteys R-, H-, S-, ja P-lauseisiin.

Tietokantaan muodostetaan kuusi taulua, jotka nimetään yllä olevien yksilöehdokkaiden mukaan. Kuusi taulua tarvitaan yhden sijaan kannan toiminnallisuuksien selkeyttämiseen. Kaikki tieto voitaisiin sisällyttää yhteen tauluun, mutta silloin taulusta tulisi liian sekava. Kaikki tieto saadaan helpommin toisesta taulusta tarvittaessa. Esim. kuvia on turha tallettaa yhteen tauluun, sillä Kuvat-taulun pääavainta vastaavalla kirjaimella kuva saadaan haettua Kuvat-taulusta helpommin. Taulujen keskinäisten yhteyksien hyötykäyttäminen selviää tarkemmin kuvasta 3.

4.2 Kemikaalin ominaisuuksien toteutuksen suunnittelu

Exel-taulukon kemikaaleja tutkittaessa huomattiin, että muutamat kemikaalit saattoivat olla taulukossa moneen kertaan. Tämä ei silti ollut virhe, vaikka niitäkin taulukossa oli monia. Kemikaalit olivat samalla nimellä taulukossa, koska niiden pakkauskokoja oli monenlaisia. Esim. jotain tiettyä happoa saattoi olla monessa 50ml, 100ml ja 500ml pakkauksissa.
Alun perin suunnitelma oli, että pääavaimena käytettäisiin kemikaalin kaavaa, sillä sehän on kuitenkin täysin omansa eri kemikaaleilla. Kuitenkin huomattiin, että sama kaava saattaa esiintyä taulukossa moneen kertaan. Tämä johtuu tietenkin siitä, että eri pakkauskoolla on sama kaava. Jokaiselle, omalla rivillään olevalle kemikaalille annetaan täysin erityinen avain.

”Jokaisella relaatiolla (perustaululla) on oltava (rivin yksilöivä) perusavain, jonka on oltava yksikäsitteinen ja minimaalinen. Pakollisuusvaatimuksen mukaan avaimen

jokaisen osan arvon pitää olla jokaisella hetkellä tunnettu (siis mikään osa ei saa olla NULL). Jos perusavain on yhdistetty, ei sen mikään osa saa olla yksikäsitteinen. Muuten perusavain ei olisi minimaalinen
Relaatiotietokanta-järjestelmän on syytä jollakin määrityksiin perustuvalla menettelyllä estää perusavainten null-arvot.
Avaimen ja indeksin välinen ero on se, että perusavain on rivin yksilöivä tekijä. Indeksi taas on tehokkaan saantitekniikan aikaansaamiseen tarkoitettu hakemisto. Hakutekijänä käytetyt ja tehokasta saantipolkua vaativat sarakkeet eivät välttämättä ole perusavaimia.
Tietokannan hallintajärjestelmän pitää pystyä jollakin, käyttäjän kannalta määrityksiin perustuvalla, tavalla estämään identtisen rivien tallentuminen perustauluun”/1, 87/

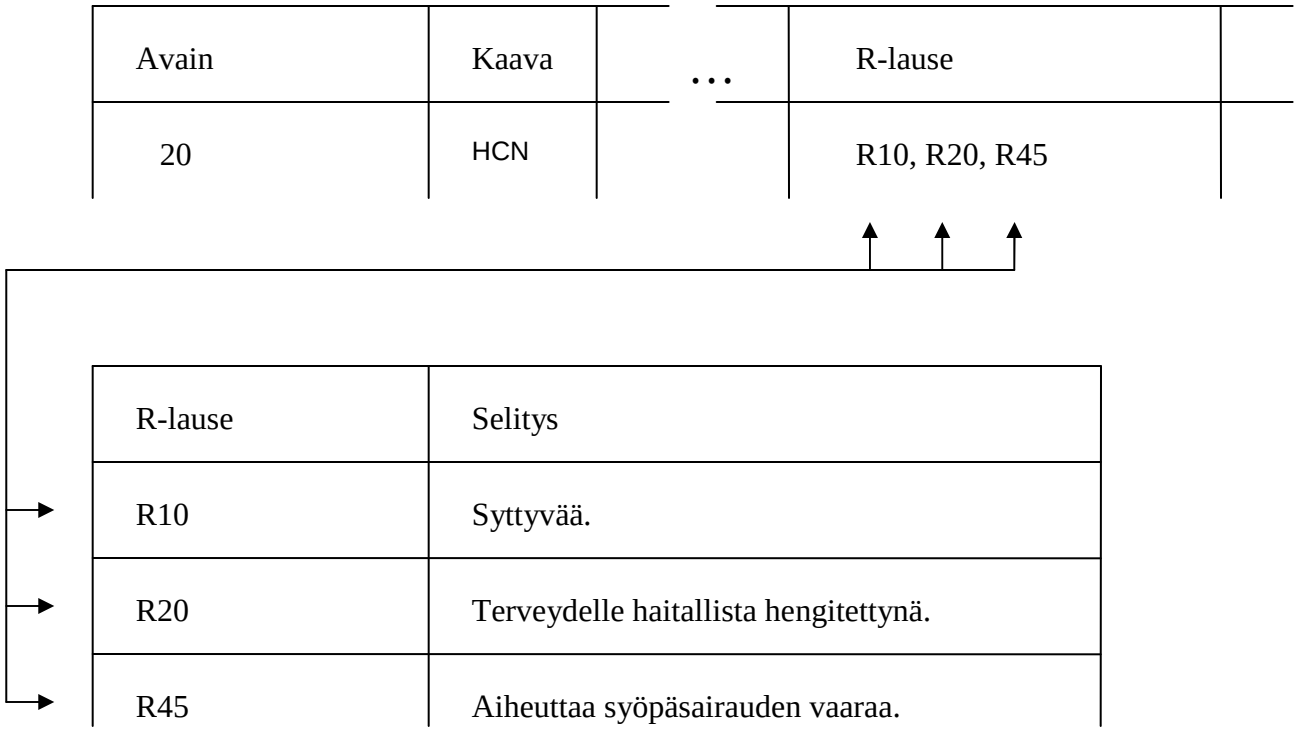
Kuten edellä mainittiin, pääavain (perusavain) on taulun tärkein sarake. Taulukossa ei ollut yhtään sellaista saraketta, joka olisi kelvannut pääavaimeksi. Jokaiselle kemikaalille tarvittaisiin pääavain joka olisi esim. väliltä 1- 2000. Päätettiin, että teemme avaimen, joka ei ole mitenkään liittyväinen itse kemikaaliin. Avain tulisi olemaan juokseva luku, joka vain tekisi jokaisesta rivistä oman uniikin kemikaalin. Kemikaali näkyy käyttäjälle Avain, mutta tulisi olemaan kannassa kem_id.
Uniikkius ei sinänsä kemikaalin hakuun vaikuta, sillä hakuehtoihin tulisi kuulumaan myös kaavalla haku, jolloin saadaan kaikki purkkikoot tietystä kemikaalista. Avain on lähinnä vain järjestelmän toiminnan tarvitsema ominaisuus.

Kauppanimet suomeksi ja englanniksi tultaisiin kannassa nimeämään Tuotenimi_fi ja Tuotenimi_eng. Molemmat kauppanimet tulisivat omille sarakkeilleen. Sarakkeet tulisivat sijaitsemaan kannan päätaulussa nimeltä kemikaalit, kuten seuraavat muutkin ominaisuudet.

Cas-numero on tunnistenumero joka on kehitetty helpottamaan kemikaalien tunnistamista ilman pitkiä ja monimutkaisia nimiä. Cas-numero nimettiin kantaan CAS_numero. Cas-numero koostuu kirjaimista ja numeroista, joten sille ei voitu asettaa suoraan kantaan määritettä integer. Integer on määrite kannassa, joka määrää sarakkeen sisältämään vain numeroita. Jos tiedon muoto haluttaan tietyssä muodossa, se voidaan määritellä PHP:ssä.

Kemikaalin kaava tulisi näkymään kannassa nimellä Kaava. Kaava onkin niitä vähiä nimiä, jotka ovat kannassa sillä nimellä, joka voidaan näyttää samana hakutuloksissa. Kemikaalin kaava tulee sisältämään ala- ja yläindeksejä. Mysql ei kuitenkaan tunne mitään ominaisuuksia, joilla ala- ja yläindeksit saataisiin hakutuloksissa näkymään suoraan oikein. Nämä ominaisuudet saadaan toteutettua html-koodilla, joka voidaan kirjoittaa html-koodina kantaan, ja ne näkyvät kannasta otettaessa alaindekseinä.
Tämä onkin ainoita ominaisuuksia, joissa käyttäjän/muokkaajan pitää muistaa jotain, mikä ei kuulu aivan normaaleihin rutiineihin. Kokouksissa päätettiin, että ala- ja yläindeksien tekeminen html:llä on niin helppoa, ettei sen toisenlaiseen toteuttamiseen tarvitse kuluttaa enempää aikaa. Muokkaaville opettajille tullaan opettamaan _,, ^{ja} -tagien käyttö. Sub-tagien sisällä oleva teksti muuttuu alaindeksiksi ja sup-tagien sisällä oleva teksti yläindeksiksi.

R-, H-, S- ja P-lauseet ovat kannanosa, missä kemikaalin R-, H-, S- ja P-lauseet yhdistyy toiseen tauluun. Jokainen, näistä neljästä lauseesta oman sarakkeensa kemikaalitaulussa jonne lauseet tulevat. Lauseet ovat muotoa R20, H1, S3/5, P24 jne. Nämä asiat tulevat näkymään, joka kemikaalissa omissa sarakkeissaan peräkkäin kuten kuva2. esittää. Yhdellä kemikaalilla voi myös olla monta kutakin eri lausetta. Itse lauseet eivät sano mitään, tarvitaan myös selitykset lauseille. Lauseet ovat yli 50 merkkiä pitkiä. On selvää, ettei selityksiä kannata kirjoittaa kemikaalien perään, sillä ruudun luettavuus kärsii siitä.
Joka lauseen selitykset pistetään omaan tauluunsa. Taulun pääavaimena toimii lause joka on esim. R20. R20 vastaa r_lauseet taulussa jotakin lausetta esim. Terveydelle haitallista hengitettynä. Näin kaksi kannan taulua ikään kuin keskustelevat keskenään.



Kuva 2. R-lause-taulun ja Kemikaalit-taulun yhteys. (Yllä tietty esimerkki kemikaali kemikaalit-taulusta ja alla on esimerkki r_lauseet taulun sisällöstä joka vastaa kemikaaliin kuuluvaa selitystä.)

Vaikka kuvassa2. on esimerkki vain R-lauseesta, myös H-,S- ja P-lauseet saavat pääavaimensa samoin kuin R-lauseet.

REACHin vaatima KTT:n päivämäärä tulisi näkymään kannassa KTT_pvm. Käyttäjälle se tulisi näkymään KTT:n päivämäärä tai KTT. KTT-sarakkeelle ei tule muuta määritystä kuin se, että se pitää olla numeroina. Sarakkeelle voisi antaa PHP:ssä muoto määrityksen esim. 12.3.2005, mutta ominaisuus ei sinänsä ole tärkeä, sillä päivämäärän muoto ei vaikuta toiminnallisuuteen. Opettajat saavat siis itse muokata KTT:n päivämäärän siihen muotoon kuin haluavat. Mahdollisuutena on, että eri opettajat tekevät päivämäärän eri muodossa, mutta sillä ei sinänsä ole mitään väliä. KTT:n päivämäärä ei tule koskaan olemaan hakuehto. Kuten edellä mainittiin, KTT:n päivämäärä ei tule vaikuttamaan toiminnallisuuksiin, tämän takia sen muodolla ei ole väliä. Mikäli saraketta käytettäisiin

hakuehtona, olisi muotomääritys pakollinen. Jos muodolle ei olisi määritystä ja opettajat muokkaisivat päivämäärät milloin missäkin muodossa, olisi mahdollista, että haut eivät toimisi odotetulla tavalla.
Sama pätee muihinkin kemikaalin ominaisuuksiin. Kaikissa sarakkeissa pitää olla tietty sama muoto, jos kemikaalin ominaisuutta tullaan käyttämään hakuehtona.

”Käyttötarkoitus ja –paikka” tulee olemaan kannassa nimellä Ktkp. Sarakkeen otsikkona tulee käyttäjälle näkymään käyttötarkoitus ja –paikka. Sarakkeessa ei tule olemaan mitään erikoisominaisuuksia.

Samoin ”Suurin käyttömäärä” ja ”Suurin varastoitu määrä” ei tule pitämään mitään hienouksia sisällään. Mysql:n sarakkeiden ominaisuuksiin voisi määritellä, että sisältö pitää olla numeerinen. Grammat ja kilogrammat merkataan lukujen perään, siispä sarakkeen sisällön määritys pitää olla kuitenkin tekstiä. Nämä kaksi tulevat kannassa näkymään Skm ja Svm-muodoissa. Käyttäjä tulee näkemään ne alussa mainitussa muodossa.

Toimittaja-sarakkeessa ei myöskään ole mitään uutta. Se tulee kannassa näkymään T_nimi. Käyttäjä tulee näkemään sen sarakkeen otsikossa ”Toimittaja”.

Varoitussymbolia vastaava kirjainsarake tulee kannassa näkymään muodossa varoitus_link. Käyttäjälle sarakkeen otsikkona lukee ”Varoitussymboli”.
Tämän sarakkeen varoitussymbolit tullaan jakamaan toisistaan pilkulla. Näin ollen PHP:ssä saadaan määriteltyä syntaksi, jolla eri kirjainsymbolit saadaan määriteltyä omakseen. Tämän sarakkeen oikeintäyttö on tärkeää koko kannalle. Varoitussymboli ei tule olemaan hakuehtona, mutta symbolilla tulee olemaan suurempi merkitys.
Symbolia tulee aina vastaamaan uusi ja vanha varoitusmerkki. Varoitusmerkit tullaan kuvina sijoittamaan eri tauluun kannassa. Kantaan ei kuitenkaan saa talletettua kuvia jpeg- tai gif-muodossa, joten kuvat pitää tallettaa tauluun linkkimuodossa. Linkit ohjaavat PHP:n kannan linkkien kautta selaimen serverillä sijaitsevaan kansioon, josta kuvat löytyvät. Kuvat tulevat ”Varoitusmerkki”-sarakkeessa näkymään samassa muodossa kuin

ne palvelimen kuvakansiossa näkyvät. Kuva 3 selittää varoitussymbolin ja varoituskuvan yhteyden.

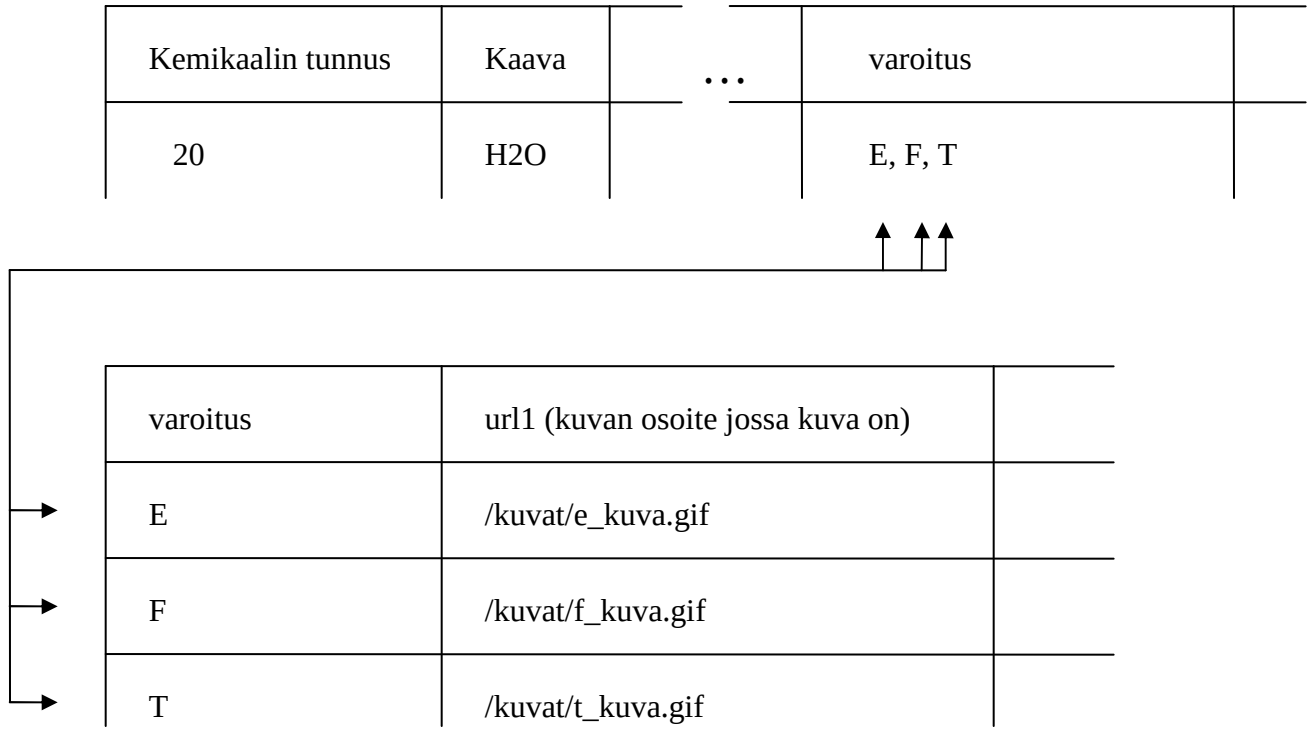
Varoitusmerkit tulevat siis olemaan eri taulussa kuin kemikaalien ominaisuudet. Taulun nimeksi tulee ”Kuvat”. Rivejä taulussa tulee olemaan niin monta kuin varoitusmerkkejäkin on. Pääavain vie yhden sarakkeen. Uusi ja vanha varoituskuva tai sen linkki tarvitsee yhden sarakkeen kumpikin. Lisäksi varmuuden vuoksi olisi suosoiteltavaa, että taulussa olisi muutama varasarake. Sarakkeita Kuvat-taulussa tulee siis olemaan vähintään kolme. Taulun pääavaimena tulee toimimaan varoitussymboli (esim.X), ja muina sarakkeina toimii symbolia vastaava varoitusmerkki kuvalinkkeinä. Jotta kuvat ovat tulosteissa käyttäjälle samankokoiset, määritetään linkkien perään html-koodina koko määritykset, leveys 77px ja korkeus 77px.

Kuvat-kansioon opettajalla tulee olemaan oikeudet. Opettaja voi muuttaa kansion sisällä olevat kuvat haluamikseen. Opettajan ei tarvitse pitää huolta muusta kuin, että nimet pysyvät samana kuin vanhat nimet olivat. Tämä ominaisuus on tärkeä, sillä uudet varoitusmerkit eivät välttämättä ole lopulliset, ja niitä saattaa joutua vaihtamaan. Vielä ei ollut varmaa, tarvitaanko vanhoja varoitusmerkkejä ollenkaan. Vanhat varoitusmerkit kuitenkin päätettiin ottaa mukaan järjestelmään, sillä uudet eivät ole vielä käytössä. Vanhat varoitusmerkit saadaan poistettua järjestelmästä ilman ohjelmointia seuraavalla tavalla:

Muutetaan kantaan vanhojen kuvien kooksi 1x1px. Muutetaan linkki osoittamaan kuvakansiosta kuvaa nimeltä blank.jpg. Lopuksi lisätään kuvakansioon kuva joka on yhden pikselin kokoinen ja valkoinen. Nimetään kuva blank.jpg:ksi. Kuvan koolla ei sinänsä ole väliä, koska kokomääritys on jo annettu kannassa. Tärkeintä on, että kuva on valkoinen, jotta se ei huomiota herättävästi erotu pohjasta.

Tällä tavalla järjestelmä voidaan halutessa lopettaa näyttämästä vanhoja varoitusmerkkejä. Ainoa kauneusvirhe tulee olemaan se että varoitusmerkki sarakkeen alussa tulee olemaan maksimissaan kaksitoista pikseliä tyhjää tilaa, minimissään yksi. Tämä johtuu siis siitä, että kuvat on ohjelmoitu näkymään, ja ilman ohjelmointia niitä ei saa pois sarakkeista.

Kauneusvirhe on niin pieni, ettei se tule tuottamaan ongelmaa.



Kuva 3. Kuvat-taulun ja Kemikaalit-taulun yhteys

Kuvasta 3. selviää miten tietylle kemikaalille saadaan Kuvat-taulusta varoitussymbolia vastaava varoituskuva. Varoitussymbolinkirjaimet voidaan korvata näkymään kuvina varoitussarakkeen kohdalla tai varoituskuvat voidaan näyttää toisessa kohdassa samalla rivillä.

Viimeisenä suunnitelma siitä, millä yksittäisen kemikaalin pdf-tiedostot saadaan näkyviin. Kemikaalihan tarvitsi KTT:n pdf-muodossa. KTT-tiedostot tullaan sijoittamaan ”Käyttöturvallisuustiedote”-kansioon palvelimella. Tiedostot ovat jo olemassa, joten ne pitää nimetä uudelleen. Nimeäminen tehdään kem_id:n eli Avaimen mukaan.
Esimerkkitapauksena otetaan alumiini. Alumiinin avain on 81. Käyttöturvallisuustiedote-kansiossa alumiinia vastaava KTT:n pdf-tiedosto uudelleen nimetään 81.pdf.
Tulosteessa haun jälkeen alumiinin avaimen kohdalla on 81 alleviivattuna linkin merkiksi. Painettaessa numeroa avautuu juuri uudelleen nimetty pdf-tiedosto ruudulle.

4.3 Sarakkeiden otsikoiden hienosäätö

Sarakkeiden otsikot ovat hyvin pitkiä lopullisessa muodossaan. Kuten edellä on jo mainittu, leveys on sivusuunnassa hyvin suuri. Sarakkeita on yli kaksikymmentä. Tähän kaikkeen kun vielä lisätään pitkät sarakeotsikot, on leveyssuuntainen pituus täydellä haulla hankalaa luettavaa.

Sarakeotsikoissa ei kaikkiin otsikoihin voi käyttää vakioleveyttä (esim. pisimmän mukaan). Sarakkeen otsikko vaikuttaa alla oleviin riveihin. Rivi-informaatio saattaa olla muutaman merkin kuten esim. ”Avain”-sarakkeessa oleva numero. Mikäli sarakkeet jätettäisiin vakio leveiksi, rivi olisi typerän näköinen. Tyhjää tilaa olisi turhaan liikaa.

Tyyli-tiedostoon pitää määrittää jokaiselle sarakkeen otsikolle erikseen oma leveysmääritys. Toinen tapa on tehdä <p> -tagilla rivi manuaalisesti siihen kohtaan, mihin se halutaan. Tagit pitäisi kirjoittaa html-koodiin. Tagit olisivat PHP-koodissa siinä kohdassa, jossa on määritelty sarakkeiden otsikot näkymään eri nimellä, kuin ne saadaan suoraan kannasta. Mahdollinen rivityskohta olisi tietenkin sopivan monen tavun jälkeen. Käytännössä ratkaisu olisi työläs ja epäkäytännöllinen, mutta mahdollinen. Pitää kuitenkin ottaa huomioon erilaiset näytöt, joissa on hyvät resoluutiot. Tarpeeksi hyvällä resoluutiollahan koko sivusuuntainen informaatio näkyisi ongelmitta. Tässä tapauksessa koko rivityksen miettiminen olisi turhaa, mutta moiset resoluutiot ovat harvinaisia ja lähinnä graafikkojen käytössä. Sellaisia näyttöjä joissa resoluutio lähentelee leveydessään 2000 pikseliä, tuskin tulee ammattiopiston tietokoneluokissa näkymään.

Tyylitiedoston etu on siinä, että selain tekee itse rivityksen. Sarakkeen leveydeksi voidaan yksilöllisesti joka sarakkeelle määritellä hyvä koko tai sitten kaikille sarakkeille yksi ”liian pieni” koko. Liian pieni koko tarkoittaa sitä, että määritellään kaikki sarakeotsikot näkymään esim. Avainsarakeotsikon levyisenä. Selain siis rivittää otsikot itse tämän jälkeen.

Yksi vaihtoehto on myös käyttää eri fonttia otsikoissa. Tässä ei kuitenkaan suurempia tilan säästöjä tule näkyviin.

Tietojen tarkastelu ruudulta ei ole pääasia, vaan niin sanotut raportit. Raportit siis tehdään suoraan tulosteesta, joita haut antavat ja jotka näkyvät ruudulla. Raportit tulostetaan suurimmalta osin A3-kokoiselle paperille. Testit osoittivat, että karsimalla vain muutama

sarake, mahtuu tuloste kokonaan paperille luettavassa muodossa. Karsimalla lisää sarakkeita tietenkin raporttien kirjasinkoko suurenee ja on tulostettavissa vielä A4-kokoisellekin arkille.
Mitään täydellistä ratkaisua ongelmalle ei ole. Leveyssuuntaisia sarakkeita tarvitaan niin monta kuin niitä on. Tulostukseen voidaan valita sarakkeet, jotka halutaan tulostaa, ja tämä parantaa tilannetta hieman. Ainoa tapa ratkaista ongelma täysin olisi, vain karsia turhat sarakkeet, kuten lisätietoa2-sarake pois toteutuksesta. Se on täyttä spekulointia, mutta näin ongelma saisi ratkaisun, sillä turhia sarakkeita tulee olemaan liikaa.

5. TYÖNJAKO

Jotta ohjelma saataisiin tehokkaasti tehtyä, oli tehtävä suunnitelma tehtävänjaosta. Kun toinen hoitaa samalla toista osa aluetta ja toinen toista, säästyy aikaa. Otetaan esimerkiksi oppilaiden hakukone ja jaetaan työt osiin. Osien toteuttamiseen menevä aika kannattaa miettiä. Näin saadaan toinen henkilö toteuttamaan monta osaa pieniä ohjelmia, sillä välin, kun toinen tekee suurempaa ohjelmaa.

Koko ohjelman/järjestelmän osat ovat:
1. Ylä- ja alatunnisteiden toteutus on yksi osa. Tunnisteet tulevat olemaan joka sivulla samat, joten ne molemmat vaativat vain yhden koodinpätkän. Kutsun seuraavissa kohdissa koodinpätkiä ohjelmiksi.
2. Sarakkeiden otsikoita hakeva ohjelma on osa, joka hakee tietokannasta kannan sarakkeiden todelliset nimet.
3. Sarakkeiden otsikot pitää muuntaa luettavampaan muotoon. Sarakkeiden muutosohjelma muuntaa alkuperäiset tietokannasta saadut otsikot luettavampaan muotoon.
4. Ruudulle tulostuvat rivit haetaan rivien-hakuohjelmalla tietokannasta.
5. Tulostettujen rivien määrän voi laskea niitä laskevalla ohjelmalla.
6. Pdf-tiedostoja hakeva ohjelma hakee tiedostot kansiosta ja näyttää ne ruudulla.

7. Varoitussymbolia vastaavat varoitusmerkkikuva-tiedostot haetaan tietokannasta. Varoitusmerkkikuva-tiedostoille tehdään tulostusohjelma, joka näyttää merkin ruudulla.
8. R-, H-, S- ja P-lauseille tehdään nappi tai linkki, joka ohjaa seuraavalle sivulle. Sivulta voidaan lukea kyseisen kemikaalin varoituslauseet.

Työ toteutetaan kahden hengen voimalla ja kutsun toteuttajia nimellä Herra X ja Herra Y. Herra X toteuttaa ensimmäisen ja neljännen kohdan. Herra Y toteuttaa toisen ja kolmannen kohdan. Toinen ja kolmas kohta muistuttavat niin paljon toisiaan, että saman henkilön kannattaa hoitaa molemmat osat. Kun edelliset kohdat on tehty Herra X ottaa toteuttaakseen kohdan kahdeksan ja Herra Y kohdan numero seitsemän. Lopuksi molemmille jaetaan loput eli kohdat 5 ja 6.
Työt on jaettu vaatimustasonsa mukaan. Koska toiset työt ovat vaativampia, vievät ne enemmän aikaa. On siis tärkeää välttää töiden jakamista summittaisesti. Tämän järjestelmän osat jakautuvat tasaisesti. Osat 1 ja 4 ovat yhdessä suunnilleen yhtä vaativat, kuin kohdat 2 ja 3. Osat 7 ja 8 ovat haastavia tehtäviä molemmat, on siis helppo jakaa ne kahdelle ihmiselle puoliksi. Jäljelle jäänet osat eli 5 ja 6 toteuttaa se, joka suoriutuu tehtävistään nopeammin. Kohta viisi on mukava lisä järjestelmässä ja ei ole korkealla prioriteetilla toteutuslistalla. Kohta kuusi on tärkeä ominaisuus, mutta se ei todellisuudessa ole ohjelma vaan linkki. Linkki vain viittaa suoraan kansioon, missä pdf-tiedostot sijaitsevat.

5.1 Tehtävät käytännössä

Jako on hyvin pelkistetty. Järjestystä toteutukselle kannattaa miettiä siten, että ne saa testattua toisen tarvittavan osan kanssa. Esimerkiksi kuvien toteutusta ei voi testata, jos rivien sisältöä näyttävää toteutusta ei ole vielä toteutettu. Osat kannattaa toteuttaa omina funktioinaan jos mahdollista.
Käytännössä jako on vain suuntaa antava, ja tehtävät toteutetaan yhteisvoimin. Toteutuksessa kannattaa muistaa, että tietyn ohjelman muuttujanimet nimennyt henkilö tekisi koko ohjelmanosan. On helpompaa tehdä ohjelma kokonaan itse, sillä muuttujien

”opettaminen” toiselle on vaikeaa. Ilman nimeämiskaavaa koodi on vaikealukuista. Toisella toteuttajalla kuluisi täysin turhaa aikaa koodin tutkimiseen. Muuttujanimet ovatkin niitä asioita, jotka sekoittavat yleensä ulkopuolista koodinlukijaa. On mahdotonta tietää toisen nimeämiä muuttujia seuraamatta kehitystyötä kokoajan, ellei nimeämiseen ole käytetty jotain keskenään sovittua kaavaa. Yleensä kaavalla tehdyt nimet ovat kirjaimia, joihin lisätään tarvittaessa järjestysnumero.

Mikäli toinen ei keksi ongelmaan ratkaisua, kannattaa tietenkin turvautua toiseen koodaajaan. Koska koodi on muuttujien takia yleensä vaikealukuista, on parempi antaa toisen lukea koodia hetken. Koodin tutkiskelun ja koodaajalta itse koodiin liittyvien asioiden, kuten muuttujiin liittyvien kysymyksien jälkeen yleensä toinen koodaaja saa käsityksen siitä mitä kohdassa on yritetty. Yhteistyö onkin tärkeää osittamisesta huolimatta, sillä kommunikaatio ja toisen avustaminen tekevät toteutuksesta paljon mukavampaa.

5.2 Funktiot

Koodissa kannattaa ottaa huomioon, että käytetään mahdollisimman paljon funktioita. Funktiot kannattaa luoda silloin, kun ohjelmassa on selkeästi sellainen osa, joka toimii kuin oma ohjelmansa. Funktioilla saadaan myös helposti luotua ohjelmien sisään toiminnallisuuksia vain yhdellä rivillä koodia. Hyvänä esimerkkinä toimii funktio, joka kannattaa tehdä tietokanta-sovelluksissa. Tietokannan ja PHP:n väliin kannattaa tehdä yhdistä–funktio, jota voi käyttää kätevästi joka kerta, kun kantaa pitää käyttää sovelluksessa. Kerran tehtyä yhdistä-funktiota on helppo käyttää aina tarvittaessa yhteyttä kannan ja sovelluksen välillä. Myös mahdollinen tietokannan siirto uudelle palvelimelle hyötyy tästä. Ilman funktion tekemistä uuden palvelimen osoite ja salasanat pitää vaihtaa moneen kohtaan ohjelmassa. Funktion avulla riittää, kun yhdistä-funktiossa vaihtaa kertaalleen osoitteen ja salasanan.

Funktioilla saadaan muissakin osioissa käytettyä samoja muuttuja nimiä kuin muissa ohjelman osissa. Ilman funktiota muuttujat pitäisi nimetä erinimisiksi, koska ilman funktioita ei duplikaatteja sallita. Funktioissa on myös se etu, että melkein kaikkia ohjelman osia voidaan muokata ilman, että koko ohjelma lakkaa testatessa toimimasta.

Toisaalta yhtä yhdistä-funktiota käytettäessä yksikään kannan resursseja vaativa osio ei enää toimi. Virheen havaitseminen on kuitenkin huomattavasti helpompaa yhdestä funktiosta kuin monesta ei-funktiolla toteutetusta osasta. Niinpä funktioilla saadaan koodista myös selkeämpää.

6. SUUNNITTELUN VAIKEUDET

Työ itsessään ei ollut kovinkaan vaikea, mutta tämä ei kuitenkaan tarkoita sitä, ettei työ olisi ollut haastava. Suunnitteluun vaikeuksia loi se, ettei työlle annettu missään vaiheessa kunnon määrityksiä. Monien kokouksien ansiosta saimme kuitenkin jotain määrityksiä selville. Projektilta vei paljon aikaa myös se, että projektin alussa pidettiin asiakkaan toimesta kokousten väleillä parinkin viikon välejä. Alkukokouksissa emme saaneet minkäänlaisia määrityksiä. Ilman määrityksiä on pakko yrittää improvisoida. Piti vain keskittyä työn muille alueille, kuten eri toiminnallisuuksien toteutuksen suunnitteluun. Työssä oli kuitenkin alkusuunnitelma, jonka kaikkien osioiden toteuttamisesta ei vielä ollut selvää kuvaa. Väliajat käytettiin järjestelmän toteuttamisen ”opiskeluun”.

Tulevaisuuden tulevia projekteja ajatellen osaan vaatia asiakkaalta kunnon määritykset työn toteutukselle, kokousaikataulun ja esityslistaa kokouksiin.

Koska työ oli ensimmäinen oikea projektimme, emme oikein tienneet, mitä asiakkaalta tulisi ja saisi vaatia. Asiakkaalla pitäisi olla hyvä käsitys siitä, mitä hän haluaa. On projektin toteuttajan tehtävä selvittää, kuka asioista päättää ja selittää asiakkaalle mahdollisuuksien rajat. On myös tärkeää, että kokouksissa keskitytään olennaisiin asioihin. Kokouksen aihe alueen rajaaminen sopivan pieneksi kerrallaan auttaa pysymään aikataulussa. Kun määrityksiä tehdään, on tärkeää, että kysytään määritteet oikeilta ihmisiltä. Pitää ottaa huomioon kaikki projektin osakkaat ja heidän tarpeensa. Väärät määritteet johtavat väistämättä projektin epäonnistumiseen.

Määritteet pitää antaa heti alussa. Tässä työssä määritteitä saatiin enemmän projektin loppu puolella. Tällöin työhön oli jo kulunut lähes kaikki sille varatut viikot, mukaan lukien raportin kirjoittamiselle varattu aika. Oli mahdotonta lähteä enää toteuttamaan

kaikkia asiakkaan toiveita, joita projektin loppupuolella alkoi sadella todella paljon. Järjestelmän toteuttamisen jälkeen jäi kuitenkin harmittamaan se, että järjestelmästä olisi voitu suuremmalla ajalla saada paljon enemmän irti kuin nyt saimme.
On kuitenkin selvää, ettei opinnäytetyön puitteissa saada niin paljon aikaiseksi kuin oikeiden rahallisten projektien kohdalla. Järjestelmä on laadukkaasti toteutettu, opinnäytetyönä. Järjestelmässä on kuitenkin vielä paljon parantamisen varaa.

6.1 Suunnittelu ilman asiakkaan vaatimuksia

Hyvän toteutuksen pohjana oli hyvä suunnitelma siitä, mitä oppilaat ja opettajat haluavat. Aina toteutettaessa suunniteltiin asiat niin, että ne olisivat mahdollisimman paljon käyttäjän mieleen.
Suunnittelussa auttoi paljon kokemus nykyajan internetsivustoista, kuten esim. Facebook, Youtube ja Mikseri.net. Edellä mainittujen sivujen käyttö on jouhevaa, sillä niiden toteuttamisessa on kulutettu aikaa suunnitteluun. Suunnittelussa on huomioitu selvästi se, että sivustoja pääsee nopeasti ja helposti heti käyttämään ilman mitään suurempaa aivojumppaa.
Huonoja sivuja on myös sattunut kohdalle paljon. Esimerkkinä eräs käyttämäni sivu on erittäin yksinkertaisen näköinen. Sivujen ei siis tarvitse olla näyttävät täyttääkseen tehtävänsä. Kuitenkin sivujen sudenkuoppa ei ole sen yksinkertainen ulkoasu vaan se, että hakujen muotoja ei ole opastettu sivuilla. Sinänsä opastus olisi tarpeeton, jos hauissa voisi hakea niin kuin Googlessa. Haut eivät kuitenkaan toimi periaatteella ”hae millä vain sanalla tai sanan osalla”. Haut ovat sivujen toiminnallisuutta ajatellen tärkeimpiä asioita. Sivustolla haetaan asioita monimutkaisilla koodeilla. Helppokäyttöisyyden vuoksi sivuille olisi voinut tehdä koodia vastaavan sanan tai lauseen. On lähes mahdotonta hakea tiettyä asiaa sivulta, sillä haettavan asian koodia ei voi arvata ja koodi on saattanut muuttua.
Hyvien ja huonojen esimerkkien avulla saimme kartoitettua teorian siitä, miten huono ja hyvä sivu toteutetaan. Hyvän sivun tulee olla erittäin käyttäjäystävällinen.

6.2 Suunnittelua testeillä

Suunnittelussa piti tehdä paljon testejä, jotta valmiin toteutuksen virheet voitaisiin minimoida. Aihealue Mysql ja PHP olivat hyvin tuttuja. Jotkin ratkaisut tiedettiin toimiviksi, ja niitä ei tarvinnut testailla. Uusia asioita ja ihan tuntemattomiakin ratkaisuja löytyi internetistä paljon. Moni muukin oli kohdannut samoja ongelmia. Ratkaisuja oli todella paljon. Lähes yhtä monta ei-toimivaa ratkaisua oli julkaistu sivuilla useaan otteeseen. Välillä sivustoilta löytyi paljonkin apua. Apu oli kuitenkin usein käyttökelvotonta ja toimi yleensä avun sijaan haittana. Kuitenkin oli sivuja, joista löytyi todella hyviä ohjeita ja koodin pätkiä, joita suunnitellessa järjestelmää voitiin hyöty käyttää. Ongelmana avustavia koodipätkiä etsiessä oli se, että moni pätkä oli kirjoitettu Javalla tai muulla ohjelmointikielellä. Ratkaisuja löytyi jopa enemmän Javalla toteutettuna, mutta PHP oli tutumpi ohjelmointikieli, joten yritimme pitäytyä siinä. Kaikki ratkaisut eivät ole kuitenkaan täysin PHP:llä toteutettavissa. Työssä pitääkin käyttää osin Javaa tiettyihin ominaisuuksiin. Javan käyttö testeissä osoitti kuitenkin sen, että se ei ole kovin selainystävällinen. Testasimme tiettyjä ominaisuuksia Javalla toteutettuna eri selaimilla. Selaimet, joita käytimme, olivat Internet Explorer, Mozilla ja Opera. Selain, jota ammattikoululla suurimmaksi osaksi käytetään, on Internet Explorer. Internet Explorer osoittautuikin testatuista selaimista virheherkimmäksi. Opera oli sietokykyisin ja näytti lähes kaikki testit oikein mukisematta. Virheet liittyivät lähinnä Javaan Internet Explorerin kohdalla. Niinpä Javan käyttöä järjestelmässä vältettiin virheettömyyden takaamiseksi.

Testityökaluna toimi XAMP. Xampilla luotiin virtuaalinen palvelin joka käsitti PHP:n. Myös Mysql-palvelin löytyi paketista vakiona. Asennus oli todella helppoa, ja serveri oli muutamissa minuuteissa käynnissä. Suunnittelutesteissä hyödynnettiin Exel-taulukkoa, josta saatiin hyvä pohja kannan testaamiselle. Tiesimme jo entuudestaan, että palvelin johon järjestelmä tulisi, käyttäisi Windows Server 2008-käyttöjärjestelmää ja siihen asennettua web-palvelinohjelmisto IIS7:ää. Suunnittelussa onkin luotettu siihen, että mikä toimii XAMP:issa, toimii myös IIS7-ympäristössä. Ainoat ratkaisevat erot järjestelmissä on, että toiseen pitää asentaa erikseen jokainen komponentti ja sen tietoturva tulee olemaan eri luokkaa kuin toisen. Tällöin puhutaan Windows Server 2008-

käyttöjärjestelmästä, jonka tietoturva on parempi kuin kotikoneen Windows Xp-käyttöjärjestelmässä. XAMP, joka testeissä toimi Windows Xp:n päällä, oli toimintakunnossa kaikkine asetuksineen PHP.ini-tiedostoja myöten hetkessä. Windows Server 2008 vaatiikin järjestelmää varten paljon pitemmän asennustyön kuin XAMP vaati testejä varten.

6.3 Testien merkitys suunnittelussa

Kun suunniteltava työ on ennestään tuntematon, ja vastaavaa työtä ei henkilökohtaisesti ole koskaan toteuttanut, on testaaminen todella tärkeää. Työtä suunnitellessa ei voi vain olettaa, että halutut ominaisuudet toimivat valmiissa järjestelmässä. Myös asiakkaalta määrityksiä ja vaatimuksia koottaessa on suunnittelijan ja toteuttajan hyvä tietää, mihin järjestelmä kykenee ja mihin ei. PHP:n mahdollisuudet ovat kuitenkin rajatut, vaikka se erinomainen ohjelmointikieli onkin. PHP:hen voi upottaa mukaan muitakin kieliä, mutta kuten edellä mainittiin, yhteensopivuusongelmat hankaloittavat ominaisuuden käyttöä. Teoriassahan kaikki PHP-toteutukset ovat mahdollisia, mutta se vaatii kaverikseen muita ohjelmointikieliä.

Vaaditut sovellukset eivät olleet monimutkaisia. Kuitenkin moni asia, kuten kuvat sivulla, pitää saada näkymään. Pdf-tiedostoihin pitää päästä sivuilta käsin, ja tietyistä merkeistä pitää saada näkymään niitä vastaavia lauseita. Myöskään kuvien pistäminen sivulle ei html-kielellä ole vaikeaa. Html-kielessä kuva saadaan näkymään -tagin avulla. Järjestelmä on kuitenkin interaktiivinen. Järjestelmä ei tule olemaan sivusto, jossa näytetään tiettyjä ”vakiosivuja”. Sivujen sisältö muuttuu sen mukaan, mitä käyttäjä syöttää hakutuloksiin. Se tekee järjestelmän ilman testauksia ennalta arvaamattomaksi. Esimerkkinä voidaan edelleen pitää kuvia, joita sivulla pitää näkyä. Kuvienhan pitää kuulua niille kemikaaleille, joita käyttäjä on hakenut. Hakua ei koskaan voi tietää ennalta, joten sivut eivät voi olla staattisia ja muuttumattomia, joihin kuvat on vain liimattu. Kuville pitää keksiä täysin uusi tapa näyttää ne.

Teorioita toteutuksesta tuli heti muutamia. Jos niistä kirjoitettaisiin suunnitelmaan, ja ei olisi varmuutta, tuleeko se toimimaan, olisi toteutusvaiheessa mahdollisesti edessä uuden

ratkaisun keksiminen. Hyvä suunnitelma, ja ennen kaikkea kevyesti testattu suunnitelma, säästää aikaa toteutuksessa. Myös asiakkaalle voi varmoin mielin kertoa, mihin järjestelmä pystyy ja mihin ei.

Asiakkaalle tuli jo suunnitteluvaiheen jälkeen kova into antaa uusia määritteitä järjestelmälle. Yksi näistä oli etiketinteko-ominaisuus järjestelmään. Etiketin olisi pitänyt tulostaa ruudulle kemikaalin nimi, kuva ja varoituslause. Etiketti ohjelman tulosteet oli määrä tulostaa ja kiinnittää pakkauksien kylkeen.

Ylimääräisen ominaisuuden tekeminen järjestelmään on aina riskialtista. Pienessä ajassa on todella vaikea tehdä varmasti toimivaa ohjelmaa. Ilman kunnon suunnittelua ja testaamista ”etiketti-kone” olisi saattanut toimia hetken. Testaamattomalla ohjelmalla on kuitenkin yleensä tapana tehdä jotain odottamatonta. Jos ei ole varmuutta, että ohjelma toimii, kannattaa siihen menevä aika mieluummin käyttää jo olemassa olevien ohjelmien hiomiseen. Toisin sanoen, rajataan aihe alue ja keskitytän tärkeimpään. Suunnitelma toteutuksesta tehtiin nopeasti. Kemikaalin nimi, kuvat, varoituslauseet tai merkit otettaisiin suoraan kannasta kuten muutkin ominaisuudet. Suurimman osan järjestelmästä toteutettuamme tiesimme, että vaikka asia olisi teoriassa selvä ja helppo toteuttaa, voisimme olettaa, että tielle tulee odottamattomia yllätyksiä. Asia päätettiin jättää suunnitteluasteelle, sillä sitä ei ehdittäisi testata käytännössä, ja täten päätettin, että sitä ei oteta lopulliseen toteutukseen. Myös asiakas unohti asian myöhemmin ja keskittyi järjestelmän tarkasteluun jo sellaisenaan. Tarkasteltuaan järjestelmää asiakas huomasi suuren puutteen järjestelmässä, joka pakotti uudelleen suunnittelemaan järjestelmän tiettyjä toimintoja. Aihe alueen rajaaminen pakottaa keskittymään oikeaan asiaan, ja tärkeitä huomioita ja puutteita ehtii löytyä ennen kuin tuote julkaistaan. Muutoksen suunnittelu oli onneksi helppoa, sillä järjestelmässä oli osioita, jotka olivat lähes samanlaisia toteutettavan puuttuvan osan kanssa. Tuntuukin, että suunnitteleminen jälkikäteen on huomattavasti helpompaa kuin etukäteen.

7. LOPULLISEN OHJELMAN ANALYSOINTI

Alallamme on kova kilpailu työpaikoista. Tämä opinnäytetyö voi hyvin tehtynä olla niitä meriittejä, joita työelämässä saatetaan arvostaa. Työn pohjalta ja sen kautta tulleen kokemuksen myötä voi oman ohjelmistofirman perustaminen olla mahdollisempaa. Työn tulos tulee kuitenkin olemaan oikea ohjelmisto, jota tullaan käyttämään, ja se tulee tarpeeseen. On siis selvää, että työpanos, inspiraatio ja intohimo pitää olla mukana järjestelmää toteutettaessa.

Koulussamme opetetut tietokannan hallintaohjelmat ovat helpoille tauluille. Tauluja on yleensä maksimissaan kolme, ja sisältö on suhteellisen helppoa.

Oli suuri harppaus siirtyä muutamasta taulusta, rivistä ja sarakkeesta tuhansien kemikaalien kantaan, jossa sarakkeita on kymmeniä ja taulujakin kahdeksan. Tauluista käyttöön tuli tässä vaiheessa työtä vain kuusi. Missään vaiheessa suunnittelua ei kuitenkaan tullut tunnetta, että joku asia olisi liian vaikea. Järjestelmän suunnittelun ja toteutuksen aikana jouduimmekin vain muutaman kerran turvautumaan ohjaajamme apuun. Tarvitsimme apua työhön kuitenkin vähemmän, mitä alussa odotimme. Alussa ei ollut varmaa, voiko halutunlaisen järjestelmän toteuttaa kaksi opiskelijaa. Hyvin opetellut ja opetetut asiat olivat kuitenkin tärkeää apu tätä työtä toteutettaessa. Työ oli haastava, ja uusia asioita oli paljon. Työssä oli kuitenkin suuri osa juuri sitä, mitä juuri koulussa oli opetettu. Kuten jo edellä mainittu, internet oli suuri apu työssä. Internetistä ei kuitenkaan löytynyt suoria ratkaisuja, vaan omalla päättelykyvyllä internetistä löydetyt asiat avustivat työtä. Koodausta oppii vain tekemällä, se todella pitää paikkansa. Mitä enemmän työtä teki, sitä paremmin osasi seuraavan osan ohjelmasta tehdä. On mahtava tunne todella osata asioita, joiden toteutuksen tekemistä alussa pelkäsi. Varmuus seuraavia haasteita varten kasvaa ja tuntee, että kuuluu niihin harvoihin, jotka osaavat koodata muutakin kuin html:ää.

7.1 Pohdintaa

Kun ajattelee valmista ohjelmaa, oli tärkeää, että konvertoimme Exel-taulukon suoraan kantaan. Kanta on monimutkainen ja sisältää paljon erikoismerkkejä. Jopa merkkejä, joita kuuluu Mysql:n syntaksiin. Nämä erikoismerkit olivat jo tiedossa, ja testasimme paljon kantaa ja sen toimivuutta Exel-taulukon sisällön avulla.
Olen varma, että jos kantaa ei olisi käännetty, olisi tyhjä kanta tuonut mukanaan kaikenlaisia odottamattomia virheitä, varsinkin kannan muokkauksessa.

Ohjelmaa toteutettaessa oli käynyt selväksi, että ohjelma pitää olla mahdollisimman maalaisjärjellä ymmärrettävä. IT-alan ihmisenä ei ensimmäisenä tullut mieleen, että suoraan kannasta otetut nimet eivät välttämättä kelpaisi opettajille niiden informatiivisuuden puutteen vuoksi. Nimet ovat suuntaa-antavia, ja olivathan ne melkein samoja kuin suomen kielessä, mutta se tosiasia, että alaviivat ja lyhenteet sekoittavat normaalia ihmistä, ei loppujen lopuksi tullut yllätyksenä. Tälläisten pikkuasioiden toteuttamiseen kului hyvä tovi ohjelmiston toteuttamiseen yhteensä kulutetusta ajasta. On kuitenkin erittäin tärkeää, että käyttäjä näkee kaikki asiat ruudulla heti ymmärrettävässä muodossa. Tämä on ehdottomasti tärkeimpiä ominaisuuksia tietokannanhallintaohjelmiston tehokasta käyttöä ajatellen. Myös se, että asioita ruudulla ei tarvitse ihmetellä ja pohtia, tekee varmuuden ohjelman käytölle, joka on korvaamatonta siihen verrattuna, että käyttäjä ei olisi varma täyttämistään lokeroista. Ruudulla näkyvä informaatio on haluttu hyvin simppeliksi. Mahdolliset, ulkonäköseikatkin on halutessa päivitettävissä css-tiedostoilla.

LÄHTEET

/1/ Jaakko R, Arto S, Martti L, Eero R, Kari S., 1993, Relaatiotietokannat, 3. painos, Helsinki, Valtion painatuskeskus

/2/ Michael J. H., 2000, Tietokannat: suunnittelu ja toteutus, Helsinki, IT Press

/3/ Reachin asettamat asetukset. [Verkkodokumentti] [Viitattu 1.9.2009]

Saatavissa: www.reachneuvonta.fi

/4/ Tietokantasovelluksen suunnittelu ja toteutus. Mauri L. [Verkkodokumentti] [Viitattu 15.8.2009] Saatavissa:http://users.jyu.fi/~mauri/tjtst12/