Dokumentaatio

1. Johdanto

- Järjestelmästä:

Järjestelmän tarkoituksena on luoda käyttäjälle mahdollisuus hallita, ylläpitää ja tarkastella tietokantaansa haluamallaan tavalla. Tietokannan hallinta- ja ylläpitotehtävissä käytettävät erilaiset lomakkeet ja tarkastelussa käytettävät asetelmat ovat täysin käyttäjän itsensä muokattavissa. Järjestelmä tarjoaa myös tyokalut tietokannan sisällön rakenteen muovaamiseen ja muokkaamiseen.

- Tarkoituksesta:

Tietokannat taipuvat moneen eri tehtävään, ja tämän takia ei ole aina välttämätöntä rajoittaa käyttäjän ja tietokannan toimintaa keinotekoisesti. Toki tietokantaa voidaan aivan hyvin käyttää pelkän SQL-kielen sekä komentorivin kautta, mutta tietyissä tilanteissa on tarkoituksenmukaista upottaa tietokantaan talletettu tieto esimerkiksi jonkinlaiseen arkistomalliin. Tällöin säilötyn informaation asiayhteyskin tulee todennäköisesti paremmin esille.

Tämän järjestelmän tarkoituksena on ratkaista tietynlainen muokattavuuden ongelma, joka liittyy todennäköisesti aika moneen tietokannanhallintajärjestelmään. Toisaalta jotkut järjestelmät antavat, ja ehkä ihan asianmukaisestikin, pelkän komentorivin sekä SQL-tulkin käyttäjän käyttöön, mutta toki sellaisiakin järjestelmiä löytyy, jotka tarjoavat käyttäjille lomakkeita, joiden avulla tietokantaa voidaan muokata. Ongelmana näissä kuitenkin on se, että lomakkeet ja asetelmat ovat vain harvoin ns. loppukäyttäjän muokattavissa, ja tämä ns. loppukäyttäjä pakotetaan käyttämään sellaista lomaketta ja/tai asetelmaa, joka ei välttämättä ole millään tavalla sopiva juuri hänen tehtäväänsä ajatellen. Tämän nimenomaisen järjestelmän tarkoituksena on antaa tälle loppukäyttäjälle mahdollisuus muokata näitä lomakkeita haluamallaan tavalla XML-metakielen avulla.

Esimerkiksi seuraavalla XML-koodilla(C-kielellä tekemäni version esimerkin mukaisesti) voitaisiin luoda eräänlainen arkistomalli, tai asetelma, joka toimisi rakenteena, johon tietokannan Asiakas-taulusta haettu data upotettaisiin aina rivikohtaisesti:

```
<?xml version="1.0" ?>
<layout>
<name>Asiakkaat</name>
<sqlstatement>SELECT * FROM Asiakas</sqlstatement>
<visuals>
<div x="10" y="10" width="300" height="400" r="62000" g="62000" b="62000">
<text x="10" y="10" font="Monospace 14"><string>Asiakas</string></text>
<text x="10" y="40" font="Monospace 12"><string>Etunimi:</string><data column="Etunimi"/></text>
<text x="10" y="70" font="Monospace 12"><string>Sukunimi:</string><data column="Sukunimi"/></text>
<text x="10" y="100" font="Monospace 12"><string>Yritys:</string><data column="Yritys"/></text>
<text x="10" y="130" font="Monospace 12"><string>Titteli:</string><data column="Titteli"/></text>
</div>
```

</visuals>
</layout>

Havainnoitavana tuloksena olisi jotakuinkin seuraavanlainen malli:

```
Asiakas
Etunimi:[column:Etunimi]
Sukunimi:[column:Sukunimi]
Yritys:[column:Yritys]
Titteli:[column:Titteli]
```

Ja kun tietokantaa ruvettaisiin selaamaan, täyttyisi tämä malli seuraavanlaisella tavalla:

```
Asiakas
Etunimi:Matti
Sukunimi:Meikäläinen
Yritys:Yritys Oy
Titteli:Toimitusjohtaja
```

Jos siis oletetaan, että tietokannan taulussa Asiakas on ylläolevan kaltainen rivi.
Ja sama pätee myös ylläpitoon käytettäviin lomakkeisiin, joiden avulla voidaan poistaa, lisätä ja muokata taulussa jo olevia rivejä.

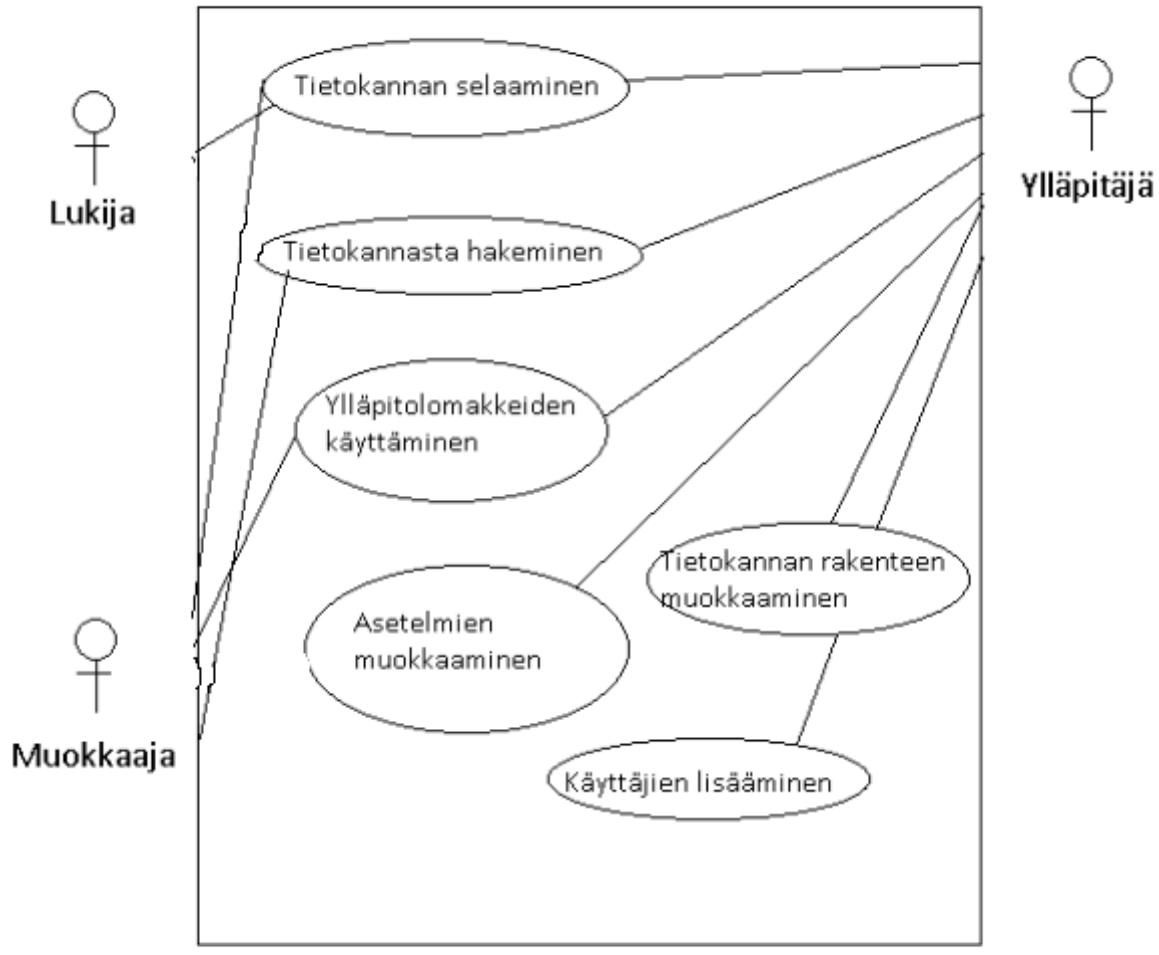
- Toteutus-/toimintaympäristöstä:

Järjestelmä käyttää MySQL:ää sekä PHP:tä palvelinpuolen toiminnallisuuden mahdollistavana skriptikielenä. Käyttäjäpuolella järjestelmä nojautuu vahvasti Bootstrapiin sekä Javascriptiin(ja edelleen jQueryyn tietyissä tilanteissa). Tässä tapauksessa järjestelmä toteutetaan laitoksen users-palvelimella, mutta tämä tuskin on sinänsä mikään rajoittava tekijä.

---

2. Yleiskuva järjestelmästä

- Käyttötapauskaavio:



3. Käyttötapaukset

Lukijan käyttötapaukset:

Tietokannan selaaminen: Lukija voi selata niitä asetelmia, jotka ovat hänelle avoimia.

Muokkaajan käyttötapaukset:

Tietokannan selaaminen: Muokaaja voi selata niitä asetelmia, jotka ovat hänelle avoimia.

Tietokannasta hakeminen: Muokkaaja voi käyttää järjestelmän hakupalveluja ko. käyttäjän yleisten asetelmarajoitusten mukaisesti.

Ylläpitolomakkeiden käyttäminen: Muokkaaja voi yleisten oikeuksiensa rajoissa käyttää ylläpitolomakkeita.

Ylläpitäjän käyttötapaukset:

Tietokannan selaaminen: Ylläpitäjä voi selata tietokantaa vapaasti.

Tietokannasta hakeminen: Ylläpitäjä voi käyttää järjestelmän hakupalveluja vapaasti.

Asetelmien muokkaaminen: Ylläpitäjä voi muokata asetelmien XML-rakennetta vapaasti.

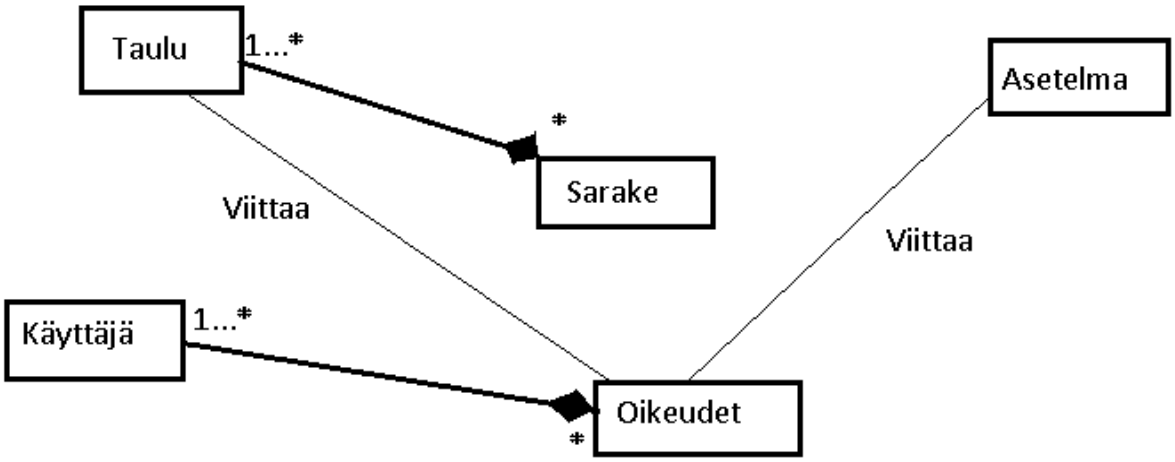
Ylläpitolomakkeiden käyttäminen: Ylläpitäjä voi vapaasti käyttää ylläpitolomakkeita.

Tietokannan rakenteen muokkaaminen: Ylläpitäjä voi muokata tietokannan rakennetta(ts. lisätä/poistaa/muokata tauluja, asetelmia, sarakkeita jne.)

Käyttäjien lisääminen: Ylläpitäjä voi lisätä käyttäjiä ja asettaa käyttörajoituksia ja -oikeuksia.

4. Järjestelmän tietosisältö

- Käsitekaavio:



Tietokohde: Taulu
Taulun id, kokonaisluku, avain
Nimi, merkkijono, taulun nimi

Tietokohde: Sarake
Sarakkeen id, kokonaisluku, avain
Taulun id, kokonaisluku, mihin tauluun sarake liittyy?
Sarakkeen nimi, merkkijono, sarakkeen nimi
Sarakkeen tyyppi, merkkijono, sarakkeen tyyppi

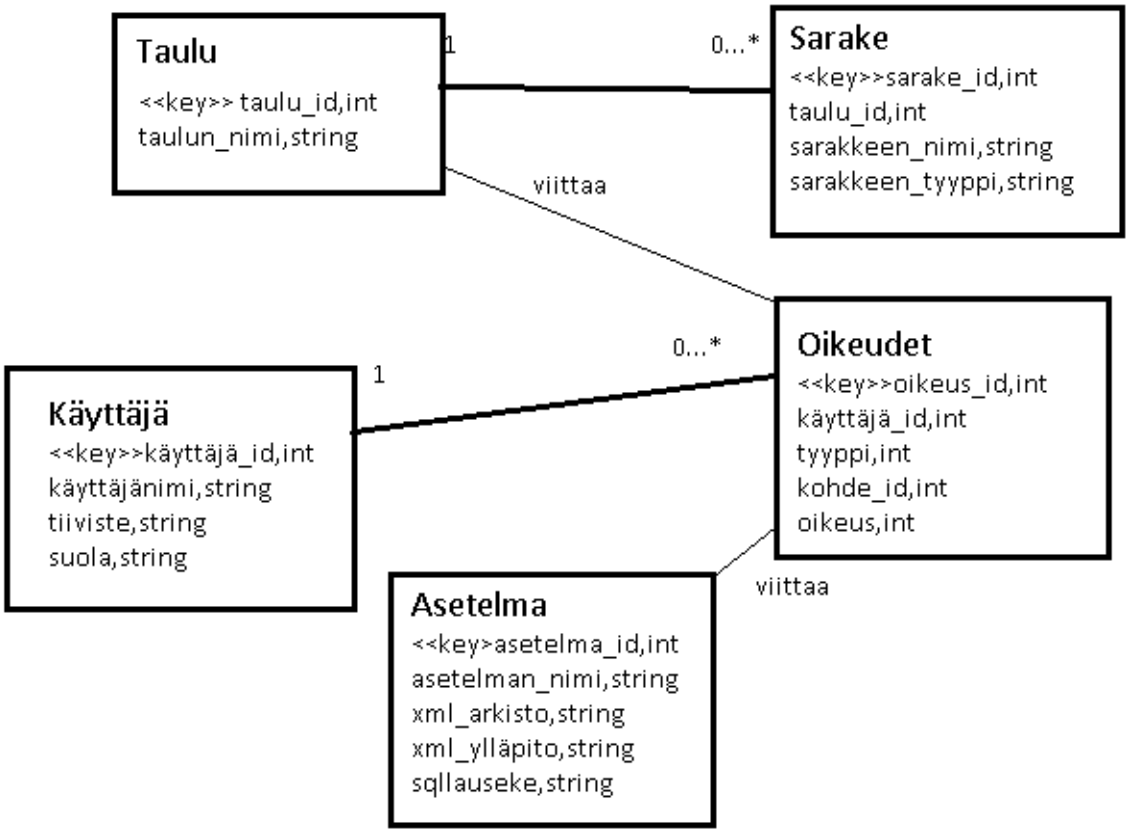
Tietokohde: Asetelma
Asetelman id, kokonaisluku, avain
Asetelman nimi, merkkijono, asetelman nimi
Arkisto-XML, merkkijono, arkiston rakennetta kuvaava XML-informaatio
Ylläpito-XML, merkkijono, ylläpitolomakkeen rakennetta kuvaava XML-informaatio
SQL-lauseke, merkkijono, minkälaisesta joukosta asetelma hakee informaationsa?

Tietokohde: Käyttäjä
Käyttäjän id, kokonaisluku, avain
Käyttäjän nimi, merkkijono, käyttäjän nimi
Salasanatiiviste, merkkijono, tunnistautumisessa käytettävä verrokki
Salasanasuola, merkkijono, tunnistautumisessa käytettävä suola

Tietokohde: Oikeudet
Oikeuden id, kokonaisluku, avain
Käyttäjän id, kokonaisluku, mihin käyttäjään oikeudet liittyvät?
Tietokohde, kokonaisluku, viittaa joko tauluun tai asetelmaan
Tietokohteen id, kokonaisluku, mihin tietokohteeseen oikeus viittaa?
Oikeus, kokonaisluku, 0=lukuoikeus, 1=ylläpito-oikeus, 2=ylläpitäjä

---

5. Relaatiotietokantakaavio



6. Järjestelmän yleisrakenne

Järjestelmä voidaan jakaa ensin kahteen eri osioon; selaimella suoritettavaan koodiin sekä palvelimella suoritettavaan koodiin. Selaimella suoritettava koodi löytyy hakemistosta js. Selaimen ja palvelimen välisessä kommunikaatiossa käytetään ajax-tekniikkaa ja informaatio palautetaan palvelimelta selaimelle json-muodossa.

Palvelinpuolen PHP-koodi löytyy hakemistosta php. PHP-koodi jakaantuu kolmeen tärkeinpään osa-alueeseen; tietokantakoodiin, käskyjä selaimelta vastaanottavaan koodiin, sekä kontrolleriin, joka mm. palauttaa, tietokantakoodia ja xml-koodia hyväksikäyttäen, käskyjen mukaista json-dataa selaimelle. Kontrolleri myös muokkaa tietokantaa ja tarkistaa käyttöoikeudet.

---

7. Järjestelmän komponentit

JavaScript

skripti.js

-Selainpuolen toiminnallisuus. Sisältää erityisiin ylläpitotoimiin vaadittavia metodeja.

skripti_luku.js

-Selainpuolen toiminnallisuus. Sisältää vain luku-, kirjoitus ja hakutoimiin vaadittavia metodeja.

PHP

column.php

- Luokkamainen tietue, joka sisältää sarakkeen rakenteen.

controller.php

- Kontrolleri, joka ohjaa järjestelmää. Käyttää muiden tiedostojen(erityisesti db.php:n ja xml.php:n) tuottamia palveluja hyväkseen.

db.php

- Sisältää erityisiä tietokantametodeja. Kaikki tietokantaa syötettävä tai sieltä haettava data kulkee näiden metodien kautta.

handle_post.php

- Ottaa vastaan selainpuolelta lähetetyn datan, ja ohjaa sen kontrollerille.

hash.php

- Tämän tiedoston metodeja käytetään käyttäjän salasanasta laskettavan tiivisteen tuottamiseen.

layout.php

- Luokkamainen tietue, joka sisältää asetelman rakenteen.

login.php

- Tämän tiedoston metodeja käytetään autentikoinnissa.

permission.php

- Tämän tiedoston metodeja käytetään silloin, kun halutaan tarkastaa, minkälaisen oikeudet kullakin käyttäjällä on.

project.php

- Luo käyttöliittymän ”dynaamisen” osan.

row.php

- Luokkamainen tietue, joka sisältää rivin rakenteen.

search_:results.php

- Luokkamainen tietue, joka sisältää hakutulosten rakenteen.

table.php

- Luokkamainen tietue, joka siältää taulun rakenteen.

user.php

- Luokkamainen tietue, joka sisältää käyttäjän rakenteen.

xml.php

- Tämän tiedoston metodit parsivat xml-datan, ja tuottavat siitä selainpuolelle lähetettävää html-dataa.

settings_db.php

- Sisältää tietokantayhteyden muodostamisessa käytettävät parametrit.

HTML-PHP

index.php

- Pääsivu.

login_auth.php

- Kirjautumissivu.

---

9. Asennustiedot

Järjestelmä asennetaan käyttökuntoon kopioimalla dist-hakemiston koko sisältö haluttuun hakemistoon, ja suorittamalla sql/create-tables.sql-tiedoston sisältö tietokantapalvelimella. Tämän lisäksi oikeanlaisen tietokantayhteyden muodostamiseksi myös settings_db.php:n sisältöä tulee muuttaa asianmukaisella tavalla.

---

10. Käynnistys- / käyttöohje

Järjestelmä sijaitsee tällä hetkellä osoitteessa http://mattipul.users.cs.helsinki.fi/viz/.

Käytettävästä XML-kielestä

Järjestelmä käyttää XML-metakieltä kahdessa paikassa, arkistopuolella sekä ylläpitopuolella. Näistä sitten luodaan asianmukaiset HTML-merkkijonot palvelinpuolella ja lähetetään takaisin selaimelle.

ARKISTO XML DTD-MÄÄRITTELY:

```
<!DOCTYPE LAYOUT [

<!ELEMENT VISUALS(SECTION*)>
<!ELEMENT SECTION(TEXT*,IMAGE*,TABLE*)>
<!ELEMENT TEXT(STRING*,DATA*)>
<!ELEMENT IMAGE(#CDATA )>
<!ELEMENT TABLE(ELEMENT*)>
<!ELEMENT STRING(#CDATA)>
<!ELEMENT DATA(#CDATA)>
<!ELEMENT ELEMENT(#CDATA)>

<!ATTLIST SECTION STYLE CDATA #IMPLIED>
<!ATTLIST TEXT STYLE CDATA #IMPLIED>
<!ATTLIST IMAGE STYLE CDATA #IMPLIED>
<!ATTLIST TABLE STYLE CDATA #IMPLIED>
<!ATTLIST TEXT STYLE CDATA #IMPLIED>
<!ATTLIST AREA STYLE CDATA #IMPLIED>
<!ATTLIST TABLE COL CDATA #IMPLIED>
```

<!ATTLIST TABLE ROW CDATA #IMPLIED>

<!ATTLIST IMAGE COLUMN CDATA #IMPLIED>

<!ATTLIST DATA COLUMN CDATA #IMPLIED>

]>

YLLÄPITO XML DTD-MÄÄRITTELY

<!DOCTYPE MANAGELAYOUT [

<!ELEMENT TABLE(SECTION*, SEARCHSECTION*, SEARCHRESULT*)>

<!ELEMENT SECTION(TEXT*,IMAGE*,OK*,ENTRY*,DELETE*,AREA*)>
<!ELEMENT TEXT(STRING*,DATA*)>

<!ELEMENT AREA EMPTY>

<!ELEMENT IMAGE (#CDATA)>

<!ELEMENT STRING(#CDATA)>

<!ELEMENT DATA(#CDATA)>

<!ELEMENT SEARCHSECTION (SEARCHENTRY*,SEARCHOK*)>

<!ELEMENT SEARCHRESULT (CHANGE*)>

<!ELEMENT CHANGE EMPTY >

<!ELEMENT ENTRY EMPTY >

<!ELEMENT OK (#CDATA)>

<!ELEMENT DELETE(#CDATA)>

<!ELEMENT SEARCHENTRY EMPTY >

<!ELEMENT SEARCHOK (#CDATA)>

<!ATTLIST TABLE NAME CDATA #REQUIRED>

<!ATTLIST TABLE TO CDATA #REQUIRED>

<!ATTLIST SEARCHSECTION SQLSTATEMENT CDATA #REQUIRED>

<!ATTLIST SEARCHSECTION IDENTIFIER CDATA #REQUIRED>

<!ATTLIST SEARCHRESULT IDENTIFIER CDATA #REQUIRED>

<!ATTLIST CHANGE FROM CDATA #REQUIRED>

<!ATTLIST CHANGE TO CDATA #REQUIRED>

<!ATTLIST ENTRY COLUMN CDATA #REQUIRED>

<!ATTLIST SEARCHENTRY COLUMN CDATA #REQUIRED>

<!ATTLIST SECTION STYLE CDATA #IMPLIED>

<!ATTLIST TEXT STYLE CDATA #IMPLIED>

<!ATTLIST IMAGE STYLE CDATA #IMPLIED>

<!ATTLIST TABLE STYLE CDATA #IMPLIED>

<!ATTLIST TEXT STYLE CDATA #IMPLIED>

<!ATTLIST IMAGE COLUMN CDATA #IMPLIED>

<!ATTLIST DATA COLUMN CDATA #IMPLIED>

<!ATTLIST ENRTY STYLE CDATA #IMPLIED>

<!ATTLIST AREA STYLE CDATA #IMPLIED>

<!ATTLIST SEARCHENTRY STYLE CDATA #IMPLIED>

<!ATTLIST OK STYLE CDATA #IMPLIED>

<!ATTLIST DELETE STYLE CDATA #IMPLIED>

<!ATTLIST SEARCHOK STYLE CDATA #IMPLIED>
<!ATTLIST SEARCHSECTION STYLE CDATA #IMPLIED>

<!ATTLIST SEARCHRESULT STYLE CDATA #IMPLIED>

]>

ELEMENTEISTÄ

TEXT

- Lisää tekstiä asetelmaan. Voi sisältää STRING-elementtejä, jotka sisältävät tavallista tekstiä, ja DATA-elementtejä, jonka sisältö tulkitaan sarakkeeksi, ja tällöin dataa otetaan kunkin rivin määritellystä sarakkeesta.

IMAGE

- Lisää kuvan asetelmaan. Kuvan osoite lisätään joko elementin sisälle, tai sitten attribuuttiin COLUMN, jolloin sarakkeen sisältö tulkitaan kuvan osoitteeksi.

TABLE(Arkisto)

- Lisää arkistoasetelmaan taulun. Voi sisältää ELEMENT-elementtejä, jotka ovat soluja.

STRING

- Tavallinen merkkijonoelementti.

DATA

- Poimii rivistä dataa.

ELEMENT

- Taulun solu.

SEARCHSECTION

- Luo hakupalikan. Voi sisältää SEARCHENTRY- ja SEARCHOK-elementtejä. Tulostaa tuloksensa SEARCHRESULT-elementtiin. Molempien IDENTIFIER-attribuutit määrittelevät olemassaolevan SEARCHRESULT/SEARCHSECTION-parin.

SEARCHRESULT

- Luo tulospalikan hakupalikalle. Voi sisältää CHANGE-elementtejä, jolla voidaan vaihtaa tuloksen sarakkeen nimeä.

CHANGE

- FROM-attribuutti määrittelee lähdesarakkeen ja TO=attribuutti sen, miksi sarakkeen nimi vaihtuu tuloksen osalta.

ENTRY

- Luo tekstikentän, jolla voidaan syöttää dataa tietokantaan.

AREA

- Luo tekstikentän, jolla voidaan syöttää dataa tietokantaan.

OK

- Luo nappula, joka lähettää datan palvelimelle.

DELETE

- Poistaa ko. Rivin.

SEARCHENTRY

- Luo tekstikentän, jolla voidaan hakea tietokannasta.

SEARCHOK

- Luo palikan, joka lähettää hakupyynnön palvelimelle.

TABLE(Ylläpito)

- Välilehti, joka assiosioituu johonkin tiettyyn tietokantatauluun.

---

11. Testaus, tunnetut bugit ja puutteet & jatkokehitysideat

Järjestelmä on vielä tietyssä mielessä hiukan raakile, koska todellisuudessa ylläpitopuolella käytettävät SQL-lausekkeet tulisi muodostaa tietynlaisen tietokantakaavion kautta käyttöliittymän puolella liittämällä tauluja yhteen sarakkeiden perusteella pelkästään hiiren klikkausten avulla. XML-kielenkin oikea tarkoitus on vain välittää dataa selaimen ja palvelimen välillä oikeanlaisessa muodossa; todellisuudessa asetelmien graafinen ulkoasu tulisi myös muodostaa hiiren avulla erityisessä muokkaustilassa. Ajanpuutteen vuoksi päätin vetää tässä tapauksessa mutkat hiukan suoremmiksi.

Bugeja varmasti löytyy vielä, niin kuin yleensä, mutta nykyisessä muodossaan järjestelmä ottaa varsin hyvin huomioon poikkeustilanteita. Erityistä huomiota tulee kuitenkin kiinnittää siihen, että järjestelmä pitää ylläpitäjää oikeasti ylläpitäjänä, joka tietää mitä hän tekee, sillä järjestelmän käyttämien dynaamisten(taulun/sarakkeen nimi riippuu käyttäjän toimista) SQL-lausekkeiden injektiosuojaamista ei voi toteuttaa ns. absoluuttisesti. Prepare-metodin kun tulee käsittääkseni tietää, mihin tauluun/sarakkeeseen viitataan. Toki, muutama viikko sitten tietyt asiat olisi ehkä pitänyt tehdä näiden dynaamisten lausekkeiden osalta toisin. Päätinkin sitten, että koska ylläpitäjällä on tässä tilanteessa muutenkin lähes täysi muokkausvalta tietokannan rakenteen suhteen, on syytä olettaa, että ylläpitäjä myös toimii valtansa mukaisesti. Täydellistä injektiosuojausta ei siis ole, mutta taulujen ja sarakkeiden olemassaolo tarkastetaan silti. Sen sijaan kaikki muut SQL-lausekkeet ovat asianmukaisesti parametrimuodossa. Injektiosuojauksen sijasta käyttöoikeushallintaa tulisikin harjoittaa, tällaisen projektin ollessa kyseessä, nimenomaan myös tietokannan puolella.

```
$ret_object=$this->db_select( "SELECT COUNT(*) FROM ".$table_name);
```

Ja, edelleen, on myös hyvä huomauttaa, että vaikka tietyissä tilanteissa, kuten ylläolevassa esimerkissä, SQL-injektion olemassaolo saattaa näyttää selvältä, niin tavallinen käyttäjä ei kuitenkaan koskaan pääse määrittelemään tätä $table_name muuttujaa, vaan ylläpitäjä määrittelee sen syöttämässään XML-datassa. Eli ylläpitäjä voi siis halutessaan tehdä

kaikenlaista jäynää, mutta ilmeisesti useimmissa järjestelmissä se on tarkoituskin.

$this->db_exec("CREATE TABLE ".$table_name." (".$columns_sql.");");

Ja esimerkiksi myös ylläoleva pätkä on selvästi injektiohaavoittuvainen, mutta vain ylläpitäjä pääsee suorittamaan tällaisia koodirivejä.

Periaatteessa olisi siis järkevä ensin autentikoida minimaalisilla oikeuksilla, ja sitten avata uusi tietokantayhteys käyttäjätyypin mukaan, jos haluttaisiin ehkäistä maksimaalisella tavalla SQL-injektioiden tuomat riskit, mutta toisaalta, jos autentikointimetodit ovat rikki, kaikki ovat sen jälkeen automaattisesti rikki. Ja jos autentikointi taas pelaa oikean mallin mukaisesti, tällaisia järjestelyjä ei teoriassa edes tarvittaisi.

Tällä hetkellä järjestelmä poistaa taulujen, sarakkeiden, ja asetelmien nimistä sekä käyttäjänimistä ja salasanoista kaikki erikoismerkit ja tyhjiä merkkijonoja ei näiden osalta hyväksytä. Olen näitä asioita testannut useisiin kertoihin, mutta toki silti saattaa jostakin löytyä jokin erikoismerkki tms. joka silti menee läpi. Kehittäjänsilmillä kun on hankalanpaa tällaisia huomata.

---

12. Omat kokemukset