Tampereen ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Digimedia
Ville Nissilä

Opinnäytetyö

Tilastojen hallintajärjestelmän toteutus PHP-tekniikalla

Työn ohjaaja Lehtori Petri Heliniemi
Työn tilaaja Alma Media Interactive Oy,
ohjaajana Etuovi.com tuotantopäällikko Tero Mesiranta
Tampere 1/2010

Tekijä	Ville Nissilä
Työ	Tilastojen hallintajärjestelmän toteutus PHP-tekniikalla
Sivumäärä	44
Valmistumisaika	tammikuu 2010
Työn ohjaaja	Lehtori Petri Heliniemi
Työn tilaaja	Alma Media Interactive Oy

TIIVISTELMÄ

Yritysten sisällä ei aina ole aikaa suunnitella ja toteuttaa selkeitä ja toimivia toimintamalleja usein toistuvien tehtävien suorittamiseksi. Tehtävät suoritetaan alun alkaen kiireellisesti ajanpuutteen vuoksi. Kun tällaisten tehtävien määrä kasvaa yrityksen kasvaessa ja ajan kuluessa, on toistuvien tehtävien määrä kasvanut jo niin suureksi, että tehtävien suorittamiseen kuluu jo huomattava aika.

Opinnäytetyö perustuu Alma Media Interactive Oy:n toimeksiantoon, jonka tavoitteena oli toteuttaa Etuovi.comin tuotantoryhmälle työkalu asiakkaille toimitettavien tilastojen muodostamiseen, niiden lähettämiseen ja hallinnoimiseen. Tilastot oli alunperin toimitettu käsityönä ajamalla SQL-kyselyitä, kopioimalla tulokset Excel-taulukoihin, ja lähettämällä tilastot sähköpostilla asiakkaille. Tämä vei suhteellisen paljon aikaa ja koettiin epämiellyttäväksi tehtäväksi.

Opinnäytetyössä käydään läpi toimeksiannon toteutuksessa käytetyt tekniikat, suunnitteluprosessi ja sovelluksen toteutus.

Avainsanat: PHP, tietokanta, Oracle, PL/SQL

Writer	Ville Nissilä
Thesis	Developing statistics management system with PHP
Pages	44
Graduation time	January 2010
Thesis supervisor	lecturer Petri Heliniemi
Co-operating company	Alma Media Interactive Oy

ABSTRACT

Inside companies' own patterns, there isn't always enough time and effort to create explicit and functional patterns for performing frequent tasks. Very often these frequent tasks are performed quickly because of lack of time. When the amount of these tasks keeps increasing the total time spent can increase considerably.

This thesis is based on commission by Alma Media Interactive Oy. The main purpose of this thesis is to implement a system for Etuovi.com production team for generating, sending and managing statistics that are delivered for customers frequently. Formerly these statistics were compiled manually by executing SQL-queries and copying the results to Excel-sheets. After this procedure the statistics were sent to customers via email. This task was considered unpleasant and it took quite a lot of time to perform.

This thesis describes the techniques used in the commission and describes the designing process of the management system. It also comes up with the implementation and describes the main features of the system.

Keywords: PHP, oracle, database, PL/SQL

Sisällysluettelo

1 Johdanto

Suoritin työharjoitteluni Alma Media Interactive Oy:llä Etuovi.comin tuotantoryhmässä kesinä 2008 ja 2009. Työtehtäviini kuului tuolloin Etuovi.com-tuoteperheen ylläpidolliset tehtävät. Ensimmäisen harjoittelujakson jälkeen olen työskennellyt samoissa työtehtävissä osa-aikaisesti. Etuovi.com on Suomen suosituin asunto- ja kiinteistökaupan internetpalvelu, jonka asiakkaita ovat pääasiassa kiinteistönvälitysyritykset.

Toimeksianto ja tavoite

Etuovi.com toimittaa asiakkailleen ja yhteistyökumppaneilleen tilastoja heidän halutessaan. Tilastot ovat tietokannasta haettavia tietoja esimerkiksi myynnin määrästä, kävijämääristä tai asiakastiedoista, joita asiakkaat käyttävät liiketoimintansa seuraamiseen.

Aikaisemmin tilastot toimitettiin asiakkaille suorittamalla tarvittavat tietokantakyselyt käsin ja kokoamalla kyselyiden tulokset Excel-asiakirjoihin. Tämän toimenpiteen jälkeen valmiit tilastot lähetettiin asiakkaille sähköpostilla. Tilastojen toimittaminen koettiin työyhteisössäni varsin aikaa vieväksi ja epämiellyttäväksi tehtäväksi.

Opinnäytetyöni tavoitteena on suunnitella ja toteuttaa toimeksiantajalleni WWW-pohjainen järjestelmä tilastojen muodostamiseen, lähettämiseen ja hallintaan. Järjestelmän avulla tilastojen tuottamiseen kuluvaa aikaa saadaan vähennettyä ja tilastojen ulkonäköä yhtenäistettyä.. Idea opinnäytetyöni aiheesta tuli itseltäni, sillä asiakkaille toimitettavien tilastojen toimittaminen on ollut vastuullani. Järjestelmän avulla tilastoinnit pystyy toteuttamaan sellainenkin henkilö, joka ei näitä ole aikaisemmin toteuttanut. Omana tavoitteenani on kerrata ja hyödyntää aikaisemmin oppimiani olio-ohjelmoinnin ja PHP-ohjelmointikielen taitojani ja toteuttaa ensimmäinen WWW-sovellus, joka käyttää Oraclen tietokantaa tietovarastona.

2 Käytetyt tekniikat

Tässä luvussa kerrotaan tekniikat, joita käytin toimeksiannon sovelluksen toteutuksessa. Luvun tarkoituksena ei ole käydä tekniikoita perusteellisesti läpi, vaan antaa lukijalle yleiskuva tekniikoista, joita olen käyttänyt.

2.1 HTML/XHTML

HTML ja XHTML ovat SGML-kieleen pohjautuvia merkkauskieliä. Ne määrittävät syntaksin ja ohjeet selaimelle siitä, miten dokumentin sisältö esitetään. Dokumentin sisältönä voi olla mm. tekstiä, kuvia ja hyperlinkkejä. HTML- ja XHTML-merkkauskielissä dokumentin sisältö sijoitetaan elementtien sisään. Elementin alkutagi (esim. <h1>) merkitsee, mistä elementin alue alkaa, ja lopputagi (tässä tapauksesa </h1>), mihin elementin vaikutusalue loppuu. Näiden tagien väliin jäävä sisältö näytetään selaimella. HTML:ssä kaikilla aloitustageilla ei ole vastaavaa lopetustagia, mikä on merkittävin ero HTML:n ja XHTML:n välillä. (Musciano & Kennedy 2006, 8,18) Esimerkkikoodi 1 havainnollistaa h1-tagin oikeaoppisen käytön.

```
<h1>Otsikko</h1>
```

Esimerkkikoodi 1. Ensimmäisen tason otsikko (X)HTML-merkkauskielellä

XHTML noudattaa XML:n tiukempia syntaksimääritteitä. Elementtien nimet ovat HTML:ssä ja XHTML:ssä samoja, mutta niiden käyttö on hieman erilaista XHTML:ssä.

Tärkeimmät XHTML:n erot HTML:ään verrattuna:

- kaikki elementit ja attribuutit tulee kirjoittaa pienellä
- aloitustageilla on niitä vastaavat lopetustagit
- sisällöttömät tagit tulee myös lopettaa, esim.
- attribuuttien arvot tulee sijoittaa lainausmerkkeihin
- erikoismerkit <, >, &, ', ja ” tulee aina esittää entiteetteinä, koska selain voi tulkita ne tagin osiksi.

XHTML:n määrittely sisältää tämän lisäksi vielä joukon muita tarkempia sääntöjä syntaksista. (Niederst Robbins 2006, 2-3)

2.2 CSS

CSS-tyylisäännöstö (eng. *Cascading Style Sheets*), on kokoelma sääntöjä, jotka kuvaavat HTML-dokumentin ulkoasun joitakin piirteitä, esimerkiksi fontin värin, kirjaisinkoon tai taustavärin. Yleisimmin tyylisäännöstö kirjoitetaan omaan tiedostoonsa, johon viitataan HTML-dokumentissa.

CSS:än avulla on mahdollista määrittää koko sivustoa koskevia tyylejä yhdessä tiedostossa, eikä tyylejä tarvitse erikseen määrittää HTML-sivulla. Tämä vähentää koodin toistoa ja helpottaa yhtenäisen ulkoasun määrittämisen koko sivustolle. Lisäksi koko sivuston ulkoasua voidaan muuttaa myöhemmin koskematta itse HTML-kuvaukseen. CSS-tyylimääreillä voidaan määrittää esimerkiksi tyylit HTML-tageille (Esimerkkikoodi 2).

```
p{
    background-color: white;
    font-size: 12px;
    color: black;
}
```

Esimerkkikoodi 2. Esimerkki p-tagin tyylimäärittelystä css-tiedostossa. Kappaleen taustaväri on valkoinen, fontin koko 12 pikseliä ja väri musta.

2.3 PHP

PHP (*PHP: Hypertext Preprocessor*) on vapaan lähdekoodin scriptikieli, joka on pääasiassa suunniteltu dynaamisen sisällön tuottamiseen WWW-sivuille. Tähän tarkoitukseen PHP tarvitsee toimiakseen PHP-tulkin ja WWW-palvelimen dokumenttien lähettämiseen. PHP mahdollistaa myös scriptien suorittamisen komentoriviltä, kuten Perl, awk ja Unixin shell. (Lerdorf & Tatroe 2002, 1)

PHP syntyi vuonna 1995 Rasmus Lerdorfin toimesta. Alunperin PHP oli Lerdorfin kokoelma Perl/CGI-skriptejä, jotka hän oli tehnyt helpottamaan WWW-sivujen ylläpitoa ja tuotantoa. Vuonna 1997 PHP:n kehitys kulminoitui, kun PHP 2.0 julkaistiin. Siitä tuli varsin suosittu ohjelmoijien keskuudessa. PHP:n vahvuutena oli jo tuolloin sen helppo yhdistäminen HTML-koodin sekaan. Kun vuonna 1998 PHP 3.0 julkaistiin, käytti PHP:ta jo yli 50 000 käyttäjää. Vuonna 2000 eli noin 18 kuukautta PHP:n ensimmäisestä virallisesta julkaisusta, julkaistiin PHP 4.0. Tästä tuli PHP:n läpimurto yritystasolla, jota seurasi kasvu käyttäjien määrässä. Tähän aikaan PHP oli asennettu yli 3,6 miljoonalle domainille. (Gilmore 2006, 1-2)

PHP4 sisälsi joukon uusia ominaisuuksia:

- **Paranneltu resurssien hallinta:** yksi versioiden 3.x heikkouksista oli skaalautuvuus. Tämä pääosin siksi, että suunnittelijat aliarvioivat sen, kuinka paljon kieltä tultaisiin käyttämään suurten sovellusten tekemiseen. Kieltä ei oltu suunniteltu yritystason WWW-sivujen mittakaavoihin. Tuloksena version 4 resurssien hallintaa parannettiin huomattavasti.
- **Tuki olio-ohjelmoinnille: v**ersio 4 toi jonkinasteisen tuen olio-orientoituneelle ohjelmoinnille, vaikkakin sitä ei pidetty kovinkaan kehittyneenä. Tällä kuitenkin oli suuri merkitys uusien olio-ohjelmointia käyttävien käyttäjien houkuttelemisessa. Perinteiset luokka- ja olio-menetelmät olivat saatavilla, kuten myös olioiden kuormittaminen ja ajonaikainen luokkien tieto.
- **Natiivi sessioiden hallinta:** HTTP-sessioden hallinta, joka oli saatavilla jo versioon 3.0 kolmennenosapuolen laajennoksen PHPLIB avulla, tuotiin versioon 4 sisäänrakennettuna.
- **Salaus:** MCrypt-kirjasto tuotiin jakeluversioon saataville. Tämä mahdollisti salausalgoritmien kuten Blowfish, MD5, SHA1 ja TripleDES käytön.
- **ISAPI-tuki:** mahdollisti Microsoftin IIS WWW-palvelimen käytön PHP:n kanssa ISAPI-moduulina. Tämä paransi suorituskykyä ja turvallisuutta huomattavasti.
- **Natiivi COM/DCOM-tuki:** mahdollisti COM- ja DCOM-objektien käytön PHP:lla.
- **Natiivi tuki Java-ohjelmointikielelle:** mahdollisti Java-objektien käytön PHP-sovelluksesta.

- **Perl-yhteensopiva säännöllisten lausekkeiden tuki:** perl-ohjelmointikielen tehokas merkkijonojen käsittely tuotiin PHP:hen version 4 myötä.

Näiden ominaisuuksien lisäksi PHP 4 toi suuren määrän uusia funktioita, tehostaen kielen mahdollisuuksia WWW-sovellusen rakentamisessa. (Gilmore 2006, 2-3)

PHP versio 5:llä oli myös suuri merkitys PHP:n kehityksessä, vaikka edellisissä versioissa olikin enemmän suuren mittakaavan uudistuksia. Versio 5 sisältää parannuksia vanhoihin toiminnallisuuksiin ja uusia ominaisuuksia, jotka on aikaisemmin yhdistetty vain ns. kehittyneempiin ohjelmointikieliin: (Gilmore 2006, 3-4)

- **Parannellut olio-ohjelmoinnin ominaisuudet:** näkyvin muutos version 5:ssä on olio-ohjelmoinnin paranneltu arkkitehtuuri. Versio 5 sisältää useita uudistuksia, kuten erillisen muodostimet (eng. constructor) ja hajottimen (eng. destructor), olioden kloonaamisen, abstraktit luokat, muuttujien vaikutusalueen ja rajapinnat.
- **Try/catch -tyylisen poikkeustenhallinnan:** PHP5 sisälsi muista ohjelmointikielistä tutun poikkeustenhallinnan. Try-lohkon sisällä voidaan ottaa kiinni mahdolliset poikkeukset, ja käsitellä ne toivotulla tavalla.
- **Paranneltu merkkijonojen käsittely:** merkkijonojen ajonaikaista käsittelyä optimoitiin versioon 5.
- **Paranneltu XML:n ja Web-palveluiden tuki:** XML-tuki perustuu nykyään libxml2-kirjastoon, ja XML:n käsittelyyn on tehokas lisäosa nimeltä SimpleXML. Lisäksi PHP 5:een on saatavilla joukko muita www-palvelujen lisäosia, kuten SOAP-lisäosa.
- **Natiivi tuki SQLite-tietokannoille:** PHP 5 sisältää tuen kompakteille ja tehokkaille SQLite-tietokannoille.

2.4 SQL

SQL tulee sanoista *Structured Query Language*, eli strukturoitu kyselykieli. SQL ei ole pelkästään kyselykieli, vaan kattaa myös muita alueita; SQL mahdollistaa tietokannan rakenteen määrittelyn ja muuttamisen (CREATE**,** ALTER), päivitykset, lisäykset ja poistot (UPDATE**,** INSERT**,** DELETE). Kuten myös tapahtumakäsittelyjen ohjauksen

(ROLLBACK**,** COMMIT), valtuuksien ja turvallisuuksien määrittelyn (GRANT**,** REVOKE). SQL:n avulla on myös mahdollista käyttää ja hallita kohdistimia (DECLARE**,** CURSOR**,** FETCH). SQL sisältää myös API-rajapinnat eri ohjelmointikieliin, jotka mahdollistavat tietokantojen ja ohjelmien välisen kommunikoinnin. *(Hovi 2004, 14)*

2.5 Oracle tietokanta

Oracle on tunnettu relaatiotietokanta, kuten myös kyseistä tietokantaa valmistava Larry Ellisonin johtama yhtiö. Yhtiö myy tietokannan hallintaohjelmistoa, mutta samaa nimeä käytetään sekä tietokannasta että ohjelmistosta.

Oraclen tietokanta tallentaa datan loogisesti taulualueisiin ja fyysisesti tiedostoihin. Se hallinnoi kaikkea talletettua tietoa sen avulla, joka on tallennettu systeemitaulualueelle. Oracle-tietokantaan on mahdollista tallentaa myös esimerkiksi funktioita, proseduureja sekä tehtäviä, jotka kirjoitetaan SQL-, PL/SQL- tai Java-ohjelmointikielellä.

2.5.1 PL/SQL

PL/SQL tulee sanoista *Procedural Language extensions to the Structured Query Language*. PL/SQL mahdollistaa monimutkaistenkin tietokantaoperaatioiden ohjelmoinnin ja hallinnan Oraclen tietokannoissa. Se sisältää ohjelmointikielille tyypilliset kontrolli- ja silmukkarakenteet sekä mm. poikkeustenkäsittelyn.

Vuonna 1991 Oracle julkaisi Oracle Version 6.0:n. Yksi tärkeimmistä sen ominaisuuksista oli niin kutsuttu ”proseduraalinen optio” eli PL/SQL. Suunnilleen samoihin aikoihin Oracle julkaisi myös päivityksen SQL*Forms Version 2.3:een. SQL* Forms V3.0 toi ensimmäisenä PL/SQL:n ohjelmistokehittäjien saatavaksi, joka mahdollisti proseduraalisen logiikan sisällyttämisen SQL-kielen ympärille. PL/SQL:n ensimmäinen julkaisu ei mahdollistanut PL/SQL-ohjelmien, kuten proseduurien tai funktioiden tallentamista tietokantaan, mutta tästä huolimatta PL/SQL otettiin lämpimästi vastaan kehittäjien keskuudessa. (Feuerstein & Pribyl 2002, 3, 8-9)

PL/SQL on korkeasti strukturoitu, luettava ja helposti omaksuttavissa oleva ohjelmointikieli. Sen helppo ja tehokas syntaksi tekee siitä helposti lähestyttävän ja opittavan. PL/SQL on käytettävissä kaikissa Oraclen tietokannoissa ja käyttöönotto ei vaadi erityisiä asennustoimia. PL/SQL-ohjelmia voidaan suorittaa ajamalla tietokantaan tallennettuja PL/SQL-ohjelmia, tai suoraan scriptinä (Esimerkkikoodi 3).

```
DECLARE
  v_count NUMBER;
BEGIN
  SELECT count(*)
    INTO v_count
    FROM table1
   WHERE rowcount > 5;

  IF v_count > 10 THEN
    DBMS_OUTPUT.PUT_LINE('Kysely tuotti yli 10 riviä tuloksia');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Kysely tuotti 10 tai vähemmän riviä tuloksia');
  END IF;
END;
```

Esimerkkikoodi 3. Esimerkki yksinkertaisesta PL/SQL-ohjelmasta

2.5.2 Herätteet

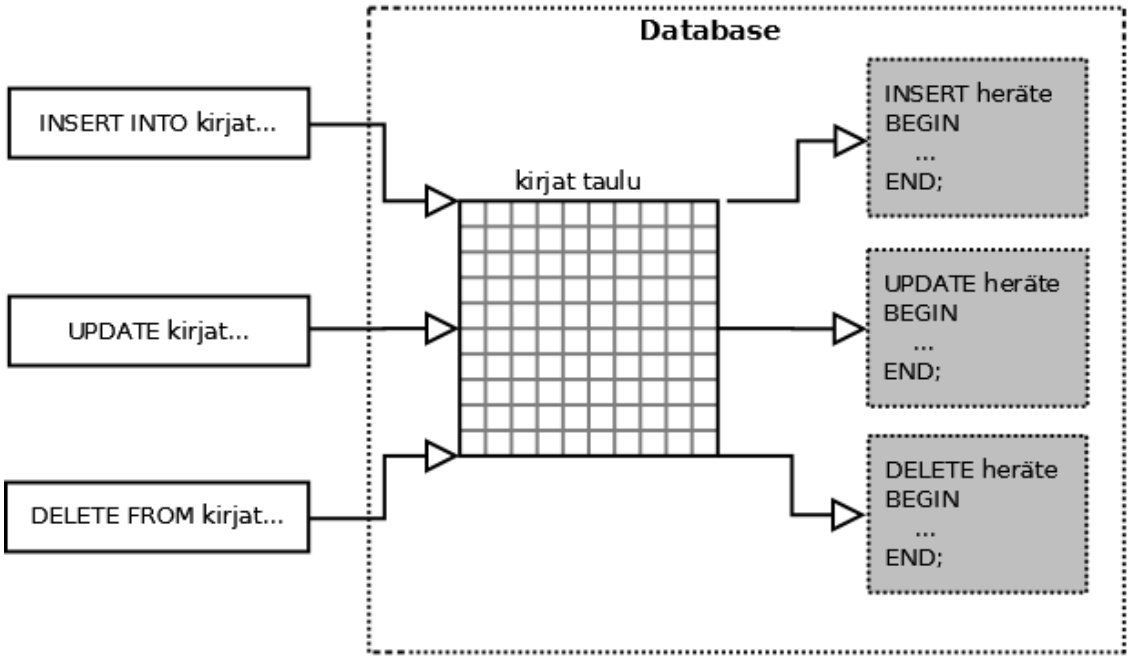
Herätteet (eng. triggers) ovat tietokantatapahtumia, jotka suoritetaan jonkin tietyn tietokantaoperaation yhteydessä. Herätteillä voidaan toteuttaa monimutkaisiakin tarkistuksia ja päivitystoimintoja, sekä tietokannan seurantaa ja hallintaa. Heräte laukeaa, kun herätteessä määritelty operaatio suoritetaan tarkkailtavaan tietokantaobjektiin. Lauetessaan heräte suorittaa sille määritellyn PL/SQL-ohjelmakoodin. Yleisimmin herättimiä käytetään seuraavanlaisiin käyttötarkoituksiin:

- validoimaan tietokantataulujen muutoksia

  Koska tietokantaobjektiin kohdistuva toiminto kohdistuu suoraan itse objektiin, voidaan herättimen avulla tehokkaasti suorittaa toiminnon validointi ja tehdä tarvittavat muutokset myös muihin haluttuihin tauluihin.
- automatisoimaan tietokannan ylläpitoa

Alkaen Oracle8i:stä on ollut mahdollista tehdä käynnistys- ja sulkemisherättimiä, suorittamaan tarvittavia tietokannan ylläpitotoimia.

Herätteet laukeavat, kun herätteen tarkkailemaan tauluun kohdistetaan DML-lause (INSERT, UPDATE tai DELETE) (*Kuvio1*). DML-herätteitä voidaan käyttää monella tapaa. Niitä voi asettaa laukeamaan ennen DML-lauseiden suoritusta tai niiden jälkeen. Laukeamisen voi myös asettaa tapahtumaan jokaisen rivin kohdalla johon DML-lause vaikuttaa. (Feuerstein & Pribyl 2002, 653-654) Toimeksiannossa käytetään herätteitä tarkastamaan tietokantatauluihin tehtäviä INSERT-lauseita ja kasvatetaan sekvenssien avulla taulujen perusavainsarakkeiden numerosarjoja. Sekvensseistä lisää seuraavassa kappaleessa 2.5.3.



Kuvio1. Herätteet laukeavat vastaavasta DML-lauseesta ja herätteen PL/SQL-ohjelmakoodi suoritetaan. (Feuerstein & Pribyl 2002, 654)

2.5.3 Sekvenssit

Sekvenssit ovat tietokantaobjekteja, joiden avulla käyttäjät voivat generoida uniikkeja kokonaislukuja. Yleisimmin sekvenssejä käytetään luomaan perusavainsarakkeiden uniikkeja kokonaislukuja. (Oracle Corporation 2003. 170)

Kun sekvenssinumero luodaan (Esimerkkikoodi 4), kasvatetaan sekvenssiä tämän jälkeen yhdellä, näin saadaan aina taulukohtaisesti haettua sekvenssistä seuraavana vuorossa oleva perusavainsarakkeen uniikki kokonaisluku. Kun sekvenssi on luotu, päästään sen arvoihin CURRVAL- ja NEXTVAL-komennoilla käsiksi. CURRVAL palauttaa sekvenssin nykyisen arvon, ja NEXTVAL seuraavan luvun, joka myös kasvattaa sekvenssin arvoa yhdellä. Esimerkkikoodi 5 kuvaa, kuinka nextval-komentoa käytetään.

```
CREATE SEQUENCE supplier_seq
    MINVALUE 1
    START WITH 1
    INCREMENT BY 1
    CACHE 20;
```

Esimerkkikoodi 4. Esimerkki uuden sekvenssin luomisesta.

```
INSERT INTO suppliers
(supplier_id, supplier_name)
VALUES
(supplier_seq.nextval, 'Name of supplier');
```

Esimerkkikoodi 5. Esimerkki sekvenssin käytöstä

2.5.4 Schema

Schema on kokoelma käyttäjän loogisia datakokonaisuuksia (schemaobjekteja). Scheman omistaa tietokannan käyttäjä, ja sillä on sama nimi kuin käyttäjällä. Jokainen käyttäjä voi olla vain yhden scheman omistajana. Schemaobjekteja voidaan luoda ja hallita SQL:n avulla ja schemat voivat sisältää mm. seuraavia objekteja: klustereita, tietokantalinkkejä, herätteitä, proseduurikirjastoja, indeksejä, Java-luokkia, materialisoituja näkymiä, sekvenssejä, funktioita, proseduureja, paketteja, tauluja ja näkymiä.

Ainoastaan scheman omistajalla on oikeus schemaobjektien lukuun ja muokkaukseen. Schemojen avulla voidaan järjestää yhteen kuuluvia schemaobjekteja saman käyttäjän alle. Schemat eivät kuitenkaan ole täysin suojattuja muilta käyttäjiltä. Tauluihin kohdistuviin SELECT-, UPDATE- ja INSERT-lauseisiin voidaan myöntää muillekin tietokannan käyttäjille oikeuksia. Näin schemojen sisällä voidaan tehdä SQL-kyselyjä, poistoja ja lisäyksiä.

2.6 Cron

Cron on yksi Unix-pohjaisissa käyttöjärjestelmissä pyörivistä perusprosesseista. Kaikissa Unix-järjestelmissä on jokin Cronin versio osana järjestelmän ajastusta. Cron mahdollistaa toistuvien tehtävien suorittamisen käyttäjän Crontab-nimisessä tiedostossa määrittämänä ajankohtana. Cron tarkistaa minuutin välein Crontab-tiedoston, ja jos kyseiselle kellonajalle on ajastettu tehtävä, ajaa Cron sen. (Gagné 2003, 234) Jokaisen yhden rivin pituisen merkinnän muoto on seuraava:

minuutit tunnit kuukaudenpäivä kuukausi viikonpäivä komento

Kenttien erottimina ovat välilyönnit. Viimeinen kenttä, *komento,* voi kuitenkin sisältää myös välilyöntejä. Viisi ensimmäistä kenttää määrittävät ajankohdat, jolloin komento tulee käynnistää. Merkitykset on esitetty kuviossa 2.

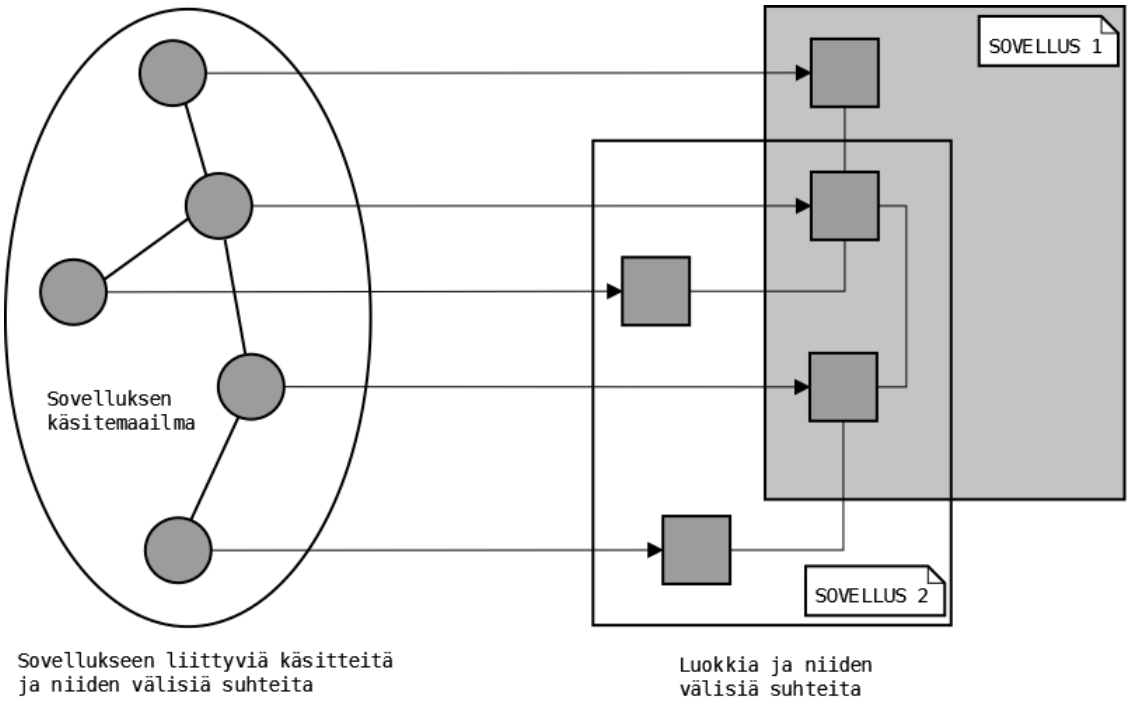
Kenttä	Alue
Minuutit	0-59
Tunnit	0-23 (0 = keskiyö)
Kuukaudenpäivä	1-31
Kuukausi	1-12
Viikonpäivä	0-6 (0 = sunnuntai)

Kuvio 2. Crontab-merkinnän aikakentät (Frisch 1997, 382)

3 Olioperustainen ohjelmistokehitys

Aikanaan proseduraaliset ohjelmointikielet oli suunniteltu pienten sovellusten toteuttamiseen. Ajatus sovelluksista, jotka sisältäisivät miljoonia rivejä ohjelmakoodia tuntuivat absurdeilta. Proseduraalinen ohjelmointitapa toimi hyvin pienten sovellusten totetuttamisessa, mutta sovellusten koon kasvaessa tuli sovellusten toteuttamisesta proseduraalisesti vaikeaa, koska ohjelmakoodin määrä kasvoi. (Barney, McLaughling 2008, 46)

Olioperustaisen ohjelmistokehityksen idea on pelkistetysti se, että sovellusohjelmisto simuloi kyseiseen sovellukseen liittyvän maailman konkreettisia tai abstrakteja tapahtumia. Ohjelmistot jaetaan osiin, jotka edustavat sovelluksen maailman olioita; nämä ovat yksiköitä, joilla on tiettyjä ominaisuuksia ja tietynlainen käyttäytyminen. Simuloinnissa oliot ovat vuorovaikutuksessa toistensa kanssa, käyttäen hyväksi toistensa tarjoamia palveluja. Olioperustaisen ohjelmiston keskeinen piirre on, että se on samanrakenteinen sovelluksen käsitemaailman kanssa (kuvio 3). (Koskimies 2000, 24)



Kuvio 3. Olioperustainen ohjelmistokehitys (Koskimies 2000, 25)

3.1 Olioperustaisuuden edut

Olioperustaisessa ohjelmistokehityksessä järjestelmiin tehtävät muutokset eivät usein aiheuta suuria muutoksia ohjelmistorakenteessa, koska nämä eivät oleellisesti muuta sovelluksen käsitemaailmaa ja muutoskohdat on helppo löytää. Sovelluksen ollessa samanrakenteinen käsitteistön kanssa, ovat ohjelmistomuutokset suuruudeltaan suhteessa niitä aiheuttaneisiin sovelluksen vaatimusten muutoksiin. Ohjelmiston rakentamisesta tulee hallittavissa oleva prosessi, jossa sovelluksen käsiteanalyysista johdetaan ajettavissa oleva koodi. Ohjelmiston osien uudelleenkäyttö on paremmin mahdollista, koska näillä on sovellusalueen peruskäsitteistöön liittyvä merkitys. Olioperustaisuus antaa mahdollisuudet systemaattiseen ohjelmistokehitykseen ja kustannusten vähentämiseen sekä ohjelmiston rakentamisen, että ylläpidon osalta. (Koskimies 2000, 25)

3.2 Olion määritelmä

Olio voidaan määritellä ympäristöstään erottuvana kokonaisuutena: sillä on oma identiteettinsä, sisäinen rakenteensa sekä suhteet tiettyyn ympäristöön. Teknisessä mielessä olio on ohjelman strukturoinnin perusyksikkö, joka – toisin kuin perinteisessä proseduraalisessa ohjelmoinnissa – ei ole puhtaasti toiminnallinen (esim. aliohjelma) tai puhtaasti tietoa säilyttävä (esim. tietue), vaan se sisältää nämä molemmat aspektit. Tällaista tiedon ja siihen liittyvien toimintojen pakkaamista yhdeksi suojatuksi kokonaisuudeksi kutsutaan ohjelmistotekniikassa kapseloinniksi (eng. Encapsulation).

Oliolla on seuraavat keskeiset ominaisuudet Koskimiehen (2003, 30) mukaan:

- Olio pystyy tarvittaessa suorittamaan sille tietyt määritellyt ominaiset toiminnot. Näitä toimintoja kutsutaan operaatioiksi.
- Olio pystyy tallentamaan tietoa olion attribuutteihin. Operaation suoritus voi muuttaa attribuuttien arvoja, eli olion tilaa.
- Jokaisella oliolla on sen identifioiva tunniste, eli viite. Kahdella oliolla on eri viite vaikka ne olisivatkin toistensa kopioita.
- Olio on suojattu kokonaisuus, jonka käyttö on rajattu tiettyihin muotoihin. Yleensä vain tietyt piirteet ovat käytettävissä olion ulkopuolelta.

3.3 Luokka

Olioden piirteet määritellään tavallisesti olion luokassa. Tällöin jokaisella oliolla on yksikäsitteinen luokka, jonka mukaan olio on luotu. Olio on tämän luokan ilmentymä. Luokka kertoo, mitä attribuutteja ja operaatioita oliolla on: jokaisella tietyn luokan oliolla on samat attribuutit ja operaatiot, vain attribuuttien arvot vaihtelevat. Luokka on eräänlainen muotti, josta olio luodaan (Esimerkkikoodi 6) (Koskimies 2000, 37) .

```php
<?php

public class Person
{
    private $name;
    private $age:

    public function __construct($name,$age) {
        $this->name = $name;
        $this->age = $age;
    }

    public function sayHello() {
        print "Hi, my name is $this->name and I am $this->age years old.";
    }

    public function changeName($newName) {
        $this->name = $newName;
    }
}
?>

<?php
    $person = new Person("Ville",25);
    $person->changeName("Esko");
    $person->sayHello();
?>
```

Esimerkkikoodi 6. Esimerkki yksinkertaisesta Person-luokasta, olion luomisesta ja operaation käytöstä.

4 PHPExcel-luokkakirjasto

Työssäni päätin käyttää PHPExcel-nimistä luokkakirjastoa Excel-asiakirjojen kirjoittamiseen. PHPExcel tarjoaa joukon PHP-luokkia, joiden avulla voi kirjoittaa ja lukea lukuisia eri tiedostoformaatteja. Työssä käytin Excel 5.0-tiedostoformaattia sen vanhuudesta huolimatta. Excel 5.0 sisälsi kaikki tarvitsemani ominaisuudet ja excel-tiedostoformaattien ollessa alaspäin yhteensopivia, on se myös varmasti yhteensopiva asiakkaiden käyttöjärjestelmissä.

PHPExcel kirjoittaa seuraaviin tiedostoformaatteihin:

- Excel 2007, BIFF5 (Excel 5.0), BIFF8 (Excel 97 ja ylöspäin), PHPExcel Serialized Spreadsheet
- CSV
- HTML
- PDF.

PHPExcel lukee seuraavia tiedostoformaatteja:

- Excel 2007, BIFF5 (Excel 5.0), BIFF8 (Excel 97 ja ylöspäin), PHPExcel Serialized Spreadsheet, Excel 2003 XML
- SYLK
- CSV.

4.1 Arkkitehtuuri

PHPExcel-luokkakirjaston arkkitehtuuri on rakennettu niin, sitä voidaan käyttää tallentamaan ja lukemaan luotuja asiakirjoja suoraan järjestelmän muistiin tai muistista. Tämä mahdollistaa esimerkiksi sen, että luotaisiin www-pohjainen näkymä excel-asiakirjasta, joka kommunikoi PHPExcel-olion kanssa. Kuten tavalliset taulukkolaskenta-ohjelmat, myös PHPExcel mahdollistaa useiden taulukkojen sisältämisen yhteen asiakirjaan, jotka sisältävät soluja, kaavoja, kuvia jne.

4.2 Taulukoiden luominen

Excel-taulukon luonti aloitetaan luomalla PHPExcel-luokasta olio. Luotaessa olio PHPExcel-luokasta, luodaan oletuksena uusi välilehti, ja se asetetaan aktiiviseksi. Myöhemmin uusia välilehtiä voidaan luoda PHPExcel-luokan createSheet-metodilla. Taulukon soluihin kirjoitetaan tekstiä setCellValue-metodilla viittaamalla solun sijaintiin sarakkeen ja rivin avulla. Esimerkkikoodi 7 esittää uuden taulukon luomisen ja sen tallentamisen haluttuun sijaintiin.

```
// Luodaan uusi PHPExcel -luokan olio
$objPHPExcel = new PHPExcel();
// Kirjoitetaan tekstiä soluun A1
$objPHPExcel->getActiveSheet()->setCellValue('A1','Hello world!');
// Luodaan uusi välilehti
$objPHPExcel->createSheet();
// Asetetaan juuri luotu välilehti aktiiviseksi
$objPHPExcel->setActiveSheet(1);
// Kirjoitetaan tekstiä soluun A1
$objPHPExcel->getActiveSheet()->setCellValue('A1','Hello world!');

// Tallennetaan luotu taulukko
// Luodaan uusi olio PHPExcel_Writer_Excel5 -luokan olio, joka tallentaa Excel5-formaatissa
$objWriter = new PHPExcel_Writer_Excel5($objPHPExcel);
// Tallennus
$objWriter->save("statistics/1/Esimerkki.xls");
```

Esimerkkikoodi 7. Excel taulukon luominen ja tallentaminen

5 Määrittely ja suunnittelu

Järjestelmän määrittelyssä ja suunnittelussa sain toimeksiantajaltani vapaat kädet. Päämääräksi asetin toteuttaa järjestelmä, jonka avulla voisi mahdollisimman helposti toimittaa asiakkaille heidän haluamansa tilastot. Järjestelmän avulla voisi myös ainakin osittain hallinnoida tilastoja. Tilastot oli toimitettu asiakkaille samalla tavalla jo useita vuosia, ja mielestäni tilastojen toimittamiseen kului aikaa varsin paljon. Tilastojen muodostaminen toimittaminen ei ollut vaativa prosessi, mutta tämä prosessi tehtiin kokonaan käsityönä, joka vei paljon aikaa.

Aloitin suunnittelutyön etsimällä vaihtoehtoja sille, miten PHP:llä voisi luoda Excel-taulukoita. Valinta oli suhteellisen helppo, sillä tarjolla oli ainoastaan PHPExcel-niminen luokkakirjasto, jolla voi kirjoittaa ja lukea Excel-taulukoita. Muitakin vaihtoehtoja toki oli, mutta PHPExcel mahdollistaa Excel-taulukoiden tallennuksen binäärimuotoisissa Excel-tiedostoformaateissa. Muut vaihtoehdot, joita tutkin, eivät tähän pystyneet, vaan tallensivat asiakirjat esim. XML-formaatissa tai pilkulla erotettuna datana. Tällöin esimerkiksi asiakirjan muotoilu on rajoitettua, tai mahdotonta. PHPExcel näytti siis sopivan tarpeisiin erittäin mainiosti.

5.1 Toiminnalliset vaatimukset

Järjestelmän vaatimuksiksi asetettiin seuraavat toiminnot, joihin järjestelmän pitäisi pystyä:

- tilaston vastaanottajien lisäys
- tilaston vastaanottajien poisto
- tilaston yleistietojen muokkaus
- tilaston poistaminen
- tilastojen muodostaminen excel-taulukoiksi
- usean tilaston muodostaminen kerralla
- muodostettujen tilastojen lähettäminen asiakkaille
- tulevien tilastojen muistutus.

Tilastoja pitää pystyä muokkaamaan, Esimerkiksi tilaston vastaanottajia pitää pystyä poistamaan tai lisäämään, sillä joskus tilastojen vastaanottajat muuttuvat. Tilastoja pitää pystyä myös tarvittaessa poistamaan järjestelmästä. Poistettua tilastoa ei poisteta kannasta fyysisesti, vaan sen status asetetaan ei-voimassaolevaksi. Tällöin tilasto pystytään myöhemmin palauttamaan helposti. Tärkein vaatimus, johon järjestelmän pitää pystyä, on Excel-asiakirjojen muodostus PHPExcel-luokkakirjaston avulla. Järjestelmän avulla pitää voida myös lähettää tilastot asiakkaille sähköpostin liitetiedostoina. Järjestelmän tulee myös muistuttaa tulevista tilastoista. Tämä totetutetaan Unixin Cron-palvelun avulla, johon ajastetaan PHP-skripti, joka tarkastaa, onko kyseiselle päivälle lähetettäviä tilastoja. Jos kyseiselle päivälle löytyy lähetettäviä tilastoja, lähettää scripti sähköpostia tuotantoryhmälle, joka töihin tullessaan muodostaa ja lähettää tilastot vastaanottajille järjestelmän avulla.

Kun järjestelmä vastaisi näihin edella asetettuihin toiminnallisiin vaatimuksiin, saataisiin tehtyä järjestelmä, jonka avulla tilastoinnin kokonaisprosessi saataisiin huomattavasti helpommaksi. Tilastointien suorittamiseen kuluva työtuntimäärä saataisiin pienemmäksi ja tilastoinnin voisi suorittaa järjestelmän avulla myös henkilö, jolla ei ole aikaisempaa kokemusta tilastojen toimittamisesta.

Suurin osa tilastoista muodostettiin ja toimitettiin samalla tavalla asiakkaille:

- tilasto muodostettiin vähintään kerran kuukaudessa
- joka kuukausi tehtiin uusi Excel-taulukko tai taulukkoon luotiin uusi välilehti uudelle tilastoinnille
- tilasto tallennettiin palvelimelle
- tilasto lähetettiin sähköpostiliitteenä asiakkaalle.

Edellä kuvatusta toimintatavasta muodostin toimintamallin, jota käytin apuna tietokannan ja sovelluksen suunnittelussa. Esimerkiksi sm_run_day-tietokantataulu sisältää kaikki tilastojen muodostamispäivät, sm_statistic-taulun new_excel_flag ilmaisee boolean-arvolla, että muodostetaanko ajettaessa uusi Excel-tiedosto, vai lisätäänkö olemassaolevaan Excel-taulukkoon uusi välilehti.

5.2 Tietokannan suunnittelu käsiteanalyysin avulla

Käsiteanalyysi on ensimmäinen vaihe tietokannan suunnittelussa. Käsiteanalyysin avulla määritetään ja kuvataan havainnollisella kaaviolla tietokantaan talletettavia tietoja, jotta lopulta voidaan perustaa tarpeita palveleva tietokanta. Käsiteanalyysissä kuvataan sitä reaalimaailmasta rajattua osaa eli kohdealuetta, jota on tarkoitus kuvata tietokannassa. Lopputuloksena syntyy käsitemalli, joka kuvaa toisaalta kohdealuetta ja toisaalta se määrittelee pohjan tietokannan fyysiselle rakenteelle. (Hovi, Huotari & Lahdenmäki. 2005, 32)

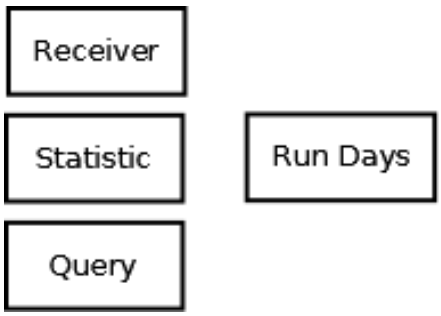
Käsiteanalyysissä kuvataan kolmenlaisia asioita:

- käsitteitä
- tietoja
- yhteyksiä.

5.2.1 Käsitteet

Käsite kuvaa asiaa, esinettä, henkilöä, paikkaa tai tapahtumaa, josta halutaan säilyttää tietoa tietokannassa. Käsite voi olla konkreettinen asia, kuten henkilö, tai abstrakti, kuten budjettikuukausi. Käsitteet kuvaavat asioita, joista halutaan tallettaa tietoa tietokantaan tulevaa käyttöä varten. Tätä varten on tärkeä muodostaa käsitteet asioista, joita todella tarvitaan. (Hovi ym. 2005, 35)

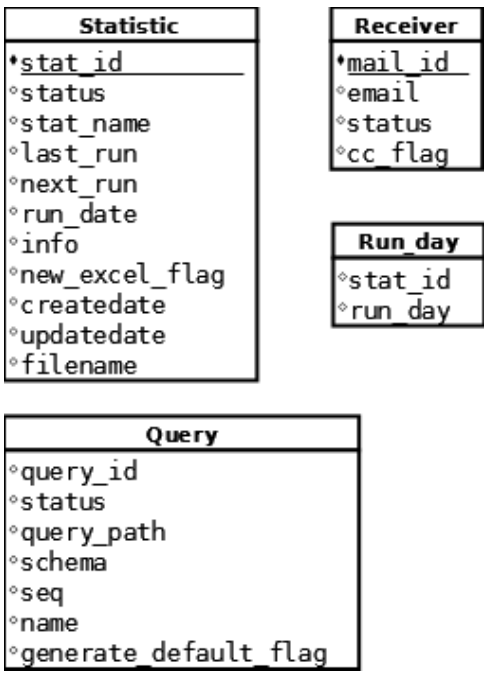
Käsiteanalyysin tekemisen aloitin miettimällä tilastoinnin tämänhetkistä kokonaisprosessia, siihen kuuluvia toimenpiteitä ja toimenpiteiden suorittamiseen tarvittavia työkaluja. Keskeisiä käsitteitä löytyi alunperin kolme kappaletta: statistic, query ja receiver. Myöhemmin etsiessäni tietoja käsitteille (Kappale 5.2.2), huomasin run_date-käsitteen olevan moniarvoinen, sillä yksittäinen tilasto voidaan toimittaa useammin kuin kerran kuukaudessa vastaanottajalle. Tuossa vaiheessa päätin muodostaa tästä oman myös käsitteensä. Kuvio 4 kuvaa käsiteanalyysissä löytyneet käsitteet.



Kuvio 4. Käsiteanalyysissä löytyneet käsitteet

5.2.2 Tiedot

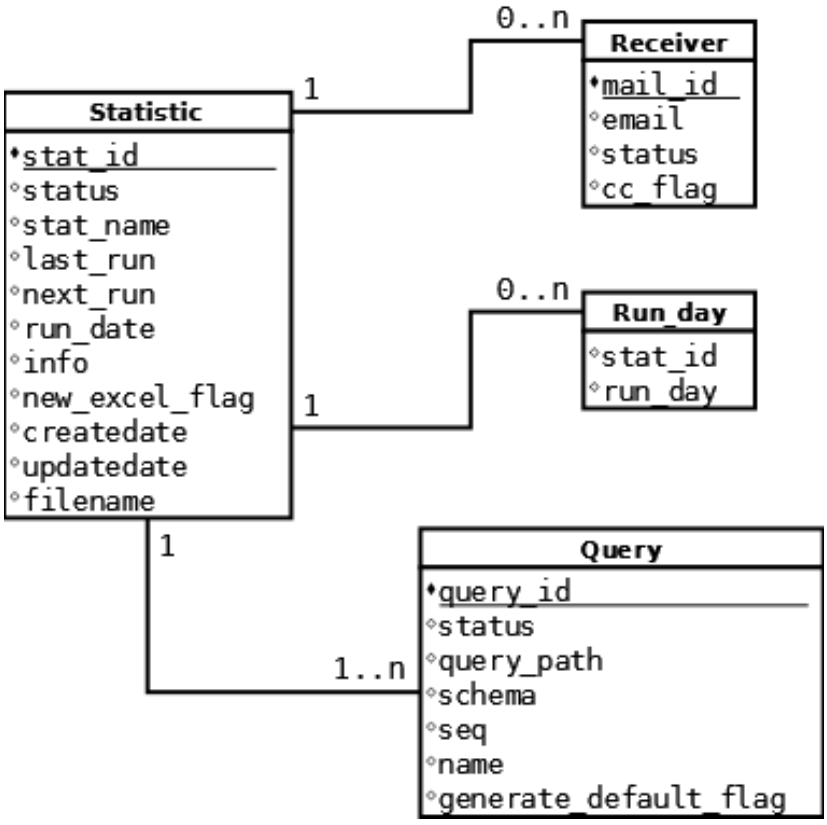
Tieto eli attribuutti kuvaa käsitettä. Käsitteeseen liittyy joukko tietoja, ja kukin tieto on käsitteen ominaisuus. Tieto on yleisimmin teksti-, numero- tai päivämäärä-tyyppinen. Kullakin käsitteellä on oltava yksilöivä perusavain, eli esimerkiksi tilastoa ei voi tallentaa tietokantaan ilman perusavainta (stat_id). (Hovi ym. 2005, 36) Kuviossa 4 on esitetty jokaisen käsitteen tiedot. Perusavaimet on merkitty alleviivattuina.



Kuvio 5. Käsitteiden attribuutit

5.2.3 Yhteydet

Käsitteiden välillä on yhteyksiä eli suhteita. Yksi-moneen-suhteet ovat yleisimpiä ja tärkein käsitemalleissa käytettävä tyyppi. Yksi-moneen-yhteyttä kutsutaan myös isä-lapsi-yhteydeksi. Isällä on monta lasta, lapsella yksi isä. Yksi-yhteen-yhteys on harvinaisempi, ja on yleensä merkki huonosta suunnittelusta. Yksi-yhteen-suhteiden käsitteet voidaan usein yhdistää samaksi käsitteeksi ja yksinkertaistaa käsiteanalyysia. (Hovi ym. 2005, 37) Käsitteiden väliset yhteydet on kuvattu kuviossa 6.



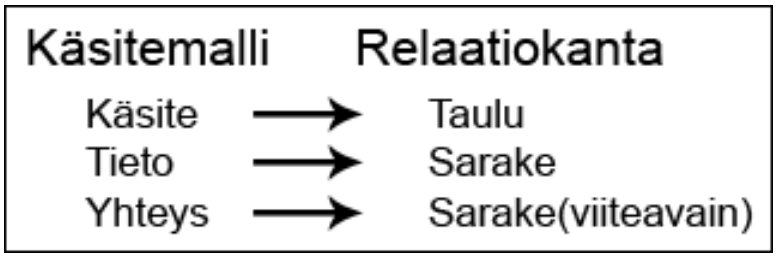
Kuvio 6. Käsitteiden väliset yhteydet

Kuvion 6 esityksestä käy ilmi, että yhdellä tilastolla on vähintään yksi tietokantakysely. Tilastolla ei välttämättä ole ainuttakaan vastaanottajaa, mutta tarvittaessa niitä voi olla kuitenkin useampia. Tilastolla ei tarvitse olla sähköpostivastaanottajaa, koska joitan tiedostoja tallennetaan palvelimille, josta asiakkaat hakevat tilastot FTP:llä.

5.2.4 Taulurakenteiden muodostaminen

Käsitemalli edustaa tietomallia, jossa on kolmenlaisia objekteja: käsitteitä, tietoja ja yhteyksiä. Relaatiotietokantojen tietomalli on relaatiomalli, jossa on vain kahdenlaisia objekteja: tauluja ja attribuutteja. Käsitemallin yhteyksille ei siis ole suoraa, omaa vastinettaan relaatiomallissa – niistäkin tulee sarakkeita. Käsitemallin avulla muodostetaan relaatiotietokantakaavio.

Kuten kuviossa 7 käy ilmi, jokaisesta käsitemallin käsitteestä tulee relaatiotietokannan taulu ja tiedoista eli attribuuteista tulee taulujen sarakkeita. Perusavaimet on jo käsitemalliin kuvattu tässä vaiheessa, joten niitä ei tarvitse enää miettiä. Yksi-moneen-suhteista syntyy viite-avaimia. Lapsi-tauluihin tulee viiteavaimeksi isä-taulun perusavain. Tässä vaiheessa tulee myös miettiä sarakkeiden tietotyypit ja määrittää pakollisuudet (NOT NULL) ja mahdollisest oletusarvot (DEFAULT). Viiteavaimille määritetään NOT NULL -pakollisuudet, jos yhteys on pakollinen. Liite 1 kuvaa relaatiotietokannan taulurakenteen.



Kuvio 7. Käsitemallista relaatiokantaan

6 Toteutus

Määrittelyn ja suunnittelun jälkeen aloitin järjestelmän teknisen toteutuksen. Tekniseen toteutukseen kuului sovelluksen tarvitsemien tietokantaobjektien luominen ja sovellusalueen luokkien toteutus. Kummassakin työvaiheessa käsiteanalyysistä ja relaatiokantamallista oli merkittävästi hyötyä. Toteutusvaiheessa ei tarvinut suunnittelutyöhön enää kuluttaa aikaa.

6.1 Tietokannan toteutus

Järjestelmää toteuttaessa aloitin luomalla tietokannan, jonka päälle järjestelmän toiminta perustuu. Omassa kehitysympäristössäni loin tietokantaan uuden käyttäjän STAT_MANAGER, ja tälle käyttäjälle määritin oman scheman, jonka alle järjestelmän tietokantaobjektit luotiin. Järjestelmän varsinainen tuotantoversio tultaisiin luomaan jo olemassaolevaan schemaan, kun järjestelmän kehitystyö olisi valmis.

6.1.1 Taulujen luonti

Tietokannan taulut luotiin käsiteanalyysin perusteella. Taulujen muodostamislauseiden luonti oli varsin helppo prosessi, sillä sarakkeiden tietotyypit ja niiden pituudet oli jo valmiiksi mietitty vastaamaan järjestelmän vaatimuksia (Liite 1). Taulujen eteen liitettiin vielä SM_-etuliite, jotta järjestelmän taulut eroaisivat selkeästi scheman muista tauluista. Liite 2 sisältää kaikkien järjestelmän taulujen luontilauseet.

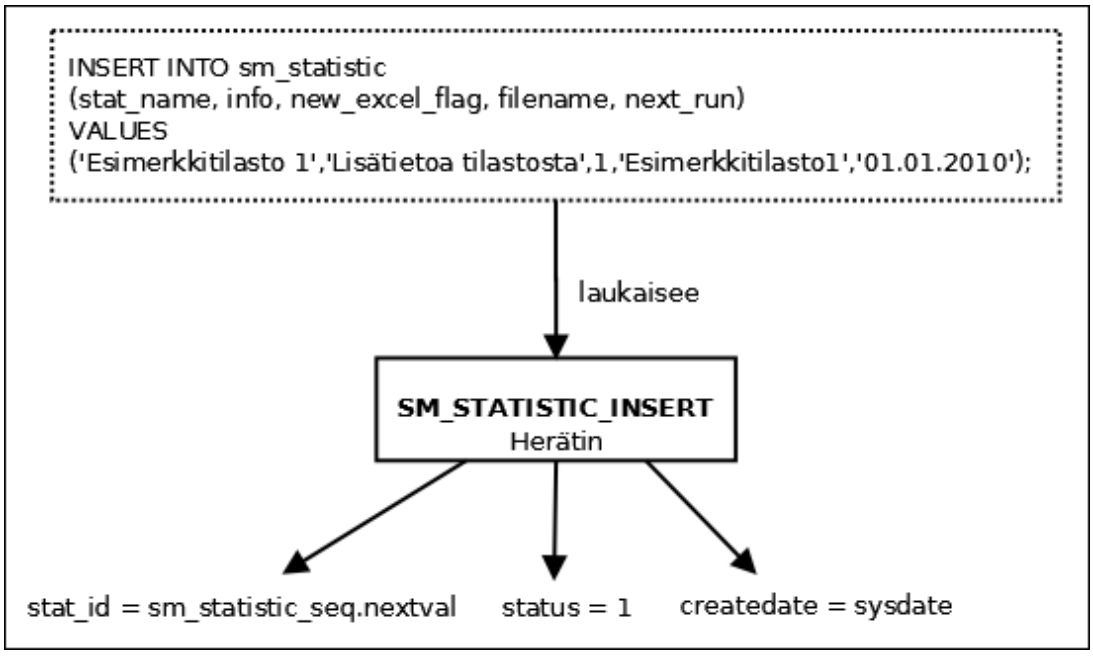
6.1.2 Sekvenssien ja herättimien toteutus

Oraclen tietokannat eivät sisällä MySQL-tietokantojen AUTO_INCREMENT -attribuuttia, jolla voitaisiin luoda uniikki tunniste tietokantataulun riville. Oraclen tietokannoissa taulujen perusavainten luonti hoidetaan sekvenssien avulla.

Kun tietokannan taulut ja rajoitteet oli luotu, tehtiin kaikille paitsi SM_RUN_DAY -tau-

lulle oman taulukohtaisen sekvenssin. SM_RUN_DAY-taulu ei tarvitse sekvenssiä, koska sillä ei ole perusavainta. Sen yksilöivänä id:nä toimii STAT_ID, joka on taulun viiteavain.

Sekvenssien alkuarvoiksi asetettiin 1, joka on jokaisen taulun ensimmäisen rivin tunniste. Tietokantaan tallennettu sekvenssi kasvattaa tätä arvoa automaattisesti joka kerta, kun sekvenssi on palauttanut uuden arvon. Jokaiselle sekvenssille luotiin myös sitä vastaava herätin, joka laukeaa, kun sen tarkkailemaan tauluun kohdistuu INSERT-lause. Lauetessaan herätin hakee sekvenssin avulla NEXTVAL-arvon ja asettaa tämän lisättävän rivin perusavaimen arvoksi. Samalla herätin myös asettaa rivin STATUS-sarakkeen arvoksi 1, ja CREATEDATE-sarakkeen arvoksi sen hetkisen aikaleiman. Tällä tavoin toimittaessa näiden sarakkeiden arvot voidaan jättää määrittämättä, kun lisätään tauluun uutta riviä. Kun tietokantatauluun kohdistuu INSERT-lause, laukeaa herätin, joka asettaa riville seuraavan perusavaimen arvon, statuksen aktiiviseksi ja luontipäivän sen hetkiseksi ajaksi (Kuvio 8).



Kuvio 8. Herättimen ja sekvenssin vaikutus INSERT-lauseessa

6.2 Sovellusalueen luokat

Järjestelmän pienestä koosta johtuen tulin siihen tulokseen, että sovellusalueen luokat on järkevintä ja yksinkertaisinta muodostaa käsiteanalyysin käsitteistä (Statistic, Receiver ja Query). Vaikka Hovi ym (2005, 127). kuvaavat luokkakaavion ja käsitemallin luomisen siinä järjestyksessä, että luokkakaavio muunnetaan käsitemalliksi, koin sovellusalueen luokkien muodostamisen helpommaksi päinvastaisessa järjestyksessä. Hovi ym. mainitsevat, että käsitemallin voi myös muuntaa luokkakaavioksi soveltaen. Koska luokat muodostuivat käsitemallin käsitteistä, saivat luokat myös samat attribuutit, joita käsitteilläkin oli.

6.2.1 Perusluokat

Käsiteanalyysin kolme käsitettä (Statistic, Query ja Receiver) muodostivat sovellusalueen luokat (Statistic, StatisticQuery ja EmailReceiver), joita kutsun tässä työssä perusluokiksi. En sisällyttänyt kaikkia järjestelmän toimintoja näihin kolmeen perusluokkaan, sillä luokista olisi tullut erittäin suuria ja vaikeasti hallittavia. Pyrin pitämään rakenteen modulaarisena luomalla jokaiselle toiminnolle oman luokkansa tai PHP-tiedostonsa. Näin tarvittaessa uusien toimintojen lisääminen sovellukseen on helpompaa ja sovelluksen selkeä rakenne säilyy.

Luokkien attribuutit suojattiin private-määreillä, koska attribuutit ovat aina oliokohtaisia. Näin attribuuttien arvoihin pääsee käsiksi ainoastaan luokan olioiden ja näiden metodien avulla. Kaikista perusluokkien metodeista tehtiin julkisia public-määreillä, jotta niiden ominaisuuksiin päästään käsiksi luokan ulkopuolellakin.

Perusluokat sisältävät ainoastaan luokkien muodostimet sekä set- ja get-metodit. Get-metodit palauttavat attribuutin arvon. Get-metodeilla päästään siis käsiksi suoraan luokan suojattuihin attribuutteihin. Set-metodeilla taas voidaan asettaa luokan attribuuteille uusia arvoja. Tein set-metodien yhteyteen ominaisuuden, joka tallentaa set-metodia kutsuttaessa luokan taulukkotyyppiseen muuttujaan avain-arvo-pariksi tietokannan vastaavan sarakkeen nimen ja attribuutin uuden arvon (Esimerkkikoodi 8). Näin jokaisella oliolla on oma taulukkonsa, jossa on kaikki sen muuttuneet attribuutit. Database-luokan

storeItem-metodin avulla voidaan helposti tallentaa olion muuttuneet attribuutit tietokantaan. StoreItem saa parametriksi taulun nimen, muuttuneiden attribuuttien taulukon, taulun perusavainsarakkeen nimen ja perusavainsarakkeen uniikin tunnisteen (Esimerkkikoodi 9).

```
public function setInfo($a)
{
    $this->info = $a;
    $this->changedValues['info'] = $this->info;
}
```

Esimerkkikoodi 8. Statistic-luokan setInfo-asetinmetodi

```
$db->storeItem("statistic",$statistic->getChangedValues(),"statistic_id",$statistic->getStatId());
```

Esimerkkikoodi 9. Esimerkki storeItem-metodin käytöstä

6.2.2 Apuluokat

Tarkastelemalla järjestelmälle asetettuja toiminnallisia vaatimuksia (kappaleen 3 alku), tein luokkia, joita tässä työssä kutsun apuluokiksi. Apuluokkien avulla järjestelmä selviää kaikista sille määritellyistä toiminnallisista vaatimuksista. Toimintoja olisi tietysti voinut sisällyttää perusluokkiin, mutta koin selkeämmäksi tehdä apuluokkia, joita käytän toimintojen suorittamiseen. Tämä selkeyttää sovelluksen rakennetta, eikä tee perusluokista tarpeettoman suuria. Sovelluksen apuluokkia ovat Mail, Database, Page ja LogWriter. Tämän kappaleen koodiesimerkit eivät ole otteita toimeksiannon sovelluksen ohjelmakoodista, vaan havainnollistavat yleisellä tasolla, kuinka apuluokkia käytetään.

Database-luokka vastaa nimensä mukaisesti tietokantayhteyden muodostamisesta ja sisältää funktioita tietokantakyselyiden tulosten käsittelyyn. Koska tilastojen SQL-kyselyt suoritetaan useisiin eri schemoihin, täytyi Database-luokan toiminta muodostaa niin, että yhteys voidaan muodostaa schema-kohtaisesti. Database-luokan olio luodaan antamalla olion luonnissa muodostimen parametreiksi käyttäjänimi, salasana ja nk. connection string, joka tässä tapauksessa on sama kuin käyttäjänimi. Itse kantayhteyden luominen tapahtuu connect-funktiolla, joka käyttää PHP:n oci_connect-funktiota yhteyden

muodostamiseen. Database-luokan execute-funktion avulla voidaan suorittaa SELECT-, INSERT- ja UPDATE- tyyppisiä SQL-kyselyjä. Funktio saa parametrikseen SQL-kyselyn ja kyselyn tyypin. Kun kysely on suoritettu, voidaan kyselyn tulokset siirtää moniulotteiseen taulukkoon Database-luokan staattisen funktion fetchToArray avulla. Omasta mielestäni kyselyn tulosten sijoittaminen moniulotteiseen taulukkoon, helpottaa tulosjoukon käsittelyä. FetchToArray järjestää taulukon halutun avaimen perusteella nousevasti. Tämän ominaisuuden avulla viittaaminen tietyn rivin tiettyyn tietokantasarakkeeseen on helppoa. Esimerkkikoodi 10 esittää tietokantayhteyden muodostamisen, SQL-kyselyn suorittamisen, tuloksen asettamisen moniulotteiseen taulukkoon ja tietyn avaimen arvon tulostamisen taulukosta (Koodiesimerkki 10).

```
$connection = new Database("STAT_MANAGER","password","STAT_MANAGER");
$connection->connect();
$result = $connection->execute("SELECT * FROM statistics","SELECT");
$resultArray = Database::fetchToArray($result,"statistic_id");

print $resultArray[0]["STATISTIC_ID"];
```

Esimerkkikoodi 10. Esimerkki Database-luokan käytöstä

Mail-apuluokka vastaa sähköpostin lähetyksestä. Sähköpostiviestit lähetetään tekstimuotoisena, ja niihin on mahdollista liittää liitetiedostoja. Sähköpostiviestit voidaan lähettää usealle vastaanottajalla, ja vastaanottajan voi määrittää vastaanottajaksi kopiona. Mail-luokan olion luonnissa muodostimen parametreiksi annetaan viestin aihe, viestin tekstisisältö ja lähettäjän sähköpostiosoite. CreateRecipientList-funktiolla voidaan antaa oliolle lista vastaanottajista ja createCcList-funktiolla vastaanottajat, jotka saavat viestin kopiona. CreateMsg-funktio koostaa viestin siihen muotoon, että viesti voidaan lähettää PHP:n sisäänrakennetun mail-funktion avulla. Se asettaa viestin header-kentät ja asettaa viestille merkistökoodaukseksi UTF-8:n. AddAttachment-funktiolla viestiin voidaan liittää liitetiedosto. Koodiesimerkki 11 sisältää createMsg-funktion. Esimerkkikoodi 11 näyttää kuinka Mail-luokan olio luodaan, viesti muodostetaan, siihen lisätään liitetiedosto ja viesti lähetetään vastaanottajille.

```
$receivers = array("address1@nowhere.com","address2@nowhere.com");
$ccReceivers = array("address3@nowhere.com","address4@nowhere.com");

$mail = new Mail("Subject","This is a test message","ville.nissila@pe.tamk.fi");
$mail->createRecipientList($receivers)
     ->createCcList($ccReceivers)
     ->createMsg()
     ->addAttachment("Statistic.xls","/statistics/1/")
     ->send();
```

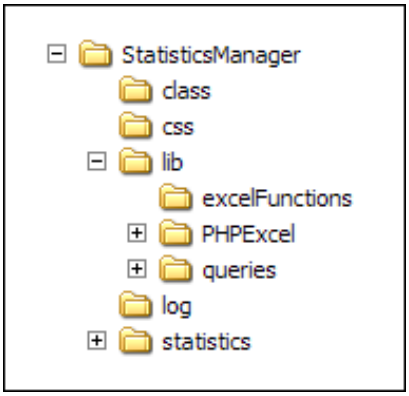
Esimerkkikoodi 11. Esimerkki Mail-luokan käytöstä

Page-apuluokka sisältää yleiskäyttöisiä funktioita, joilla voidaan suorittaa sovelluksen usein toistuvia toimintoja, kuten navigaatiopalkin tulostaminen. Page-luokan funktiot vähentävät koodin toistoa ja tekevät ohjelmakoodista luettavampaa. Page-luokan funktiot ovat kaikki staattisia, joten Page-luokasta ei tarvitse luoda oliota.

Kun ajetaan useita tilastoja yhdellä kertaa, voi virhetilanteen sattuessa olla vaikeaa tai jopa mahdotonta määrittää, minkä tilaston ajo on epäonnistunut, ja mistä syystä virhetilanne on aiheutunut. Tätä varten tein LogWriter-nimisen apuluokan. LogWriter on varsin yksinkertainen luokka, joka kirjoittaa tekstitiedostoon yksittäisen tilaston ajon alkamisajankohdan ja ajon päättymisajankohdan. Lisäksi virhetilanteen sattuessa kirjoittaa LogWriter-luokka lokitiedostoon tiedon virheen aiheuttajasta.

6.3 Sivuston rakenne

Sovelluksen sisäiset toiminnot jaoteltiin omiksi PHP-tiedostoiksi. Rakenne pyrittiin tekemään mahdollisimman modulaarisesti, jotta tulevaaisuudessa mahdollisien uusien toimintojen lisääminen sovellukseen olisi helppoa. Sivuston ja sovelluksen rakenteesta tulikin varsin selkeä (Kuvio 8).



Kuvio 8. Sivuston rakenne

StatisticManager-kansio

Kansio sisältää sovelluksen ja sivuston osat. Kansion juuressa ovat PHP-tiedostoissa HTML-tiedostot.

class-kansio

Kansio sisältää kaikki sovellusalueen PHP-luokat. Luokkien tiedostonimet on kirjoitettu luokille tutusti isoilla alkukirjaimilla. Database-luokka vastaa tietokantayhteyden muodostamisesta ja sisältää tarpeellisia tietokantafunktioita. Statistic.php, EmailReceiver.php ja StatisticQuery.php (perusluokat) sisältävät käsiteanalyysissä löytyneiden käsitteiden (statistic, receiver ja query) tiedot ja ominaisuudet, sekä asetin- ja palautinmetodit.

Database.php vastaa tietokantayhteyden muodostamisesta ja sisältää tarpeellisia funktioita kyselyjen käsittelemiseen. Mail.php vastaa sähköpostin lähetystoiminnosta. Page.php sisältää useasti järjestelmän toiminnassa käytettyjä funktioita, kuten footer- ja header-tietojen tulostuksen. LogWriter kirjoittaa Excel-taulukoita muodostettaessa lokitiedostoon tietoa Excel-taulukon muodostamisen etenemisestä.

css-kansio

Kansio sisältää sivustolla käytety css-tyylimääreet. Kansiossa on ainoastaan yksi tiedosto styles.css.

lib-kansio

Kansio sisältää PHPExcel-luokkakirjaston. Tätä luokkakirjastoa käytetään Excel-taulukoiden muodostamiseen PHP:lla. ExcelFunctions-alikansio sisältää phpExcelFuncion.php.tiedoston, joka sisältää apufunktioita Excel-tiedostojen muodostamiseksi. Queries-alikansiossa säilytetään tilastojen sql-kyselyt, joita käytetään tilastodatan keräämiseen.

log-kansio

Kansio sisältää lokitiedoston sm_run.log, johon tallennetaan Excel-taulukoiden muodostamisesta lokitietoja. Tiedot voivat olla hyödyllisiä, jos esimerkiksi tilaston muodostaminen epäonnistuu jostain syystä. Lokitiedostoon tulostetaan kaikki viimeaikaiset virheilmoitukset.

statistics-kansio

Kansio sisältää tilastokohtaiset alikansiot, jotka on nimetty tilaston uniikin id:n mukaan (stat_id). Jokaisessa tilastokohtaisessa kansiossa on kaikki tilastoista muodostetut Excel-taulukot, jotka on nimetty tilaston nimen ja muodostuspäivämäärän mukaan.

6.4 Sivuston toiminta

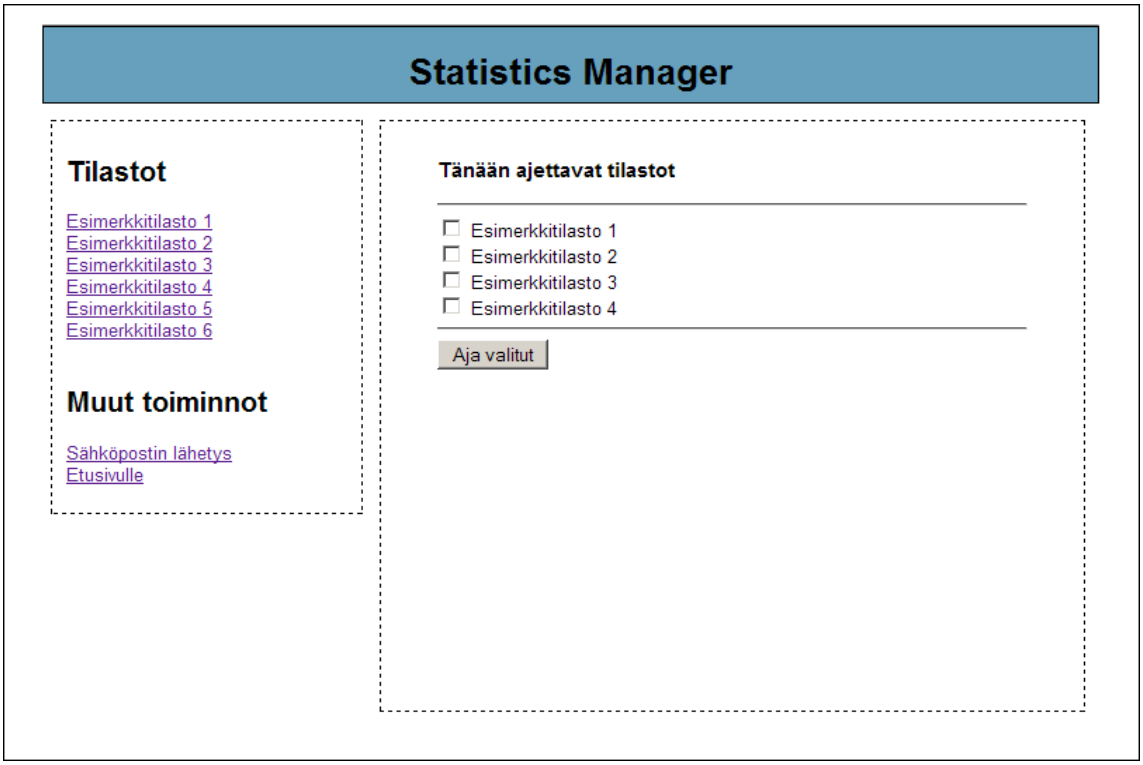
Järjestelmän päätarkoitus on Excel-tiedostojen muodostaminen ja tilastojen lähettäminen sähköpostinlähetystoiminnon avulla asiakkaille. Järjestelmän avulla on voidaan myös osittain hallinnoida tilastojen tietoja. Sivuston rakenne koostuu kolmesta osasta, joiden sisältö tulostetaan div-elementtien sisälle; yläpalkista, johon tulostetaan järjestelmän nimi, vasemman puoleisesta navigaatiopalkista, josta hoidetaan suurin osa navigoinnista ja oikeanpuoleisesta osiosta, johon dynaaminen sisältö tulostetaan.

Käyttäjän tullessa etusivulle, näytetään käyttäjälle tieto siitä, mitä tilastoja on kyseiselle päivälle toimitettavana. Näin saadaan ilman ylimääräistä sivulla navigointia tieto siitä, mitkä tilastot tulee toimittaa kyseisenä päivänä. Index.php-sivulle on sisällytetty in-

fo.php, joka muodostaa tietokantayhteyden ja hakee SQL-kyselyllä tilastot, joiden next_run on pienempi tai sama kuin kuluva päivä, ja joiden last_run on erisuuri kuin kuluva päivä (Esimerkkikoodi 12). Tilastot on mahdollista ajaa samasta näkymästä valitsemalla valintalaatikoista muodostettavat tilastot ja klikkaamalla ”aja valitut” -painike. Näin tilastojen muodostaminen on mahdollista suorittaa mahdollisimman nopeasti. Kuvio 9, on kuvankaappaus etusivulta.

```php
<?php
$query = "
SELECT *
FROM sm_statistic s
WHERE s.next_run <= to_date(sysdate,'DD.MM.YYYY')
AND s.last_run != to_date(sysdate,'DD.MM.YYYY')";

$db = new Database("STAT_MANAGER","password","STAT_MANAGER");
$db->connect();
$stats = Database::fetchToArray($db->execute($query,"SELECT"));
$db->close();

if (count($stats) > 0) {
    Page::formStart("GET","run.php");
        Page::heading("Tänään ajettavat tilastot","h3");
        Page::hr();
        Page::createInputs($stats,"checkbox","STAT_ID","STAT_NAME");
        Page::hr();
        Page::input("submit",null,"Aja valitut",null,true,null);
    Page::formEnd();
} else {
    Page::p("Ei tilastoja tälle päivälle");
}
?>
```

Esimerkkikoodi 12. info.php



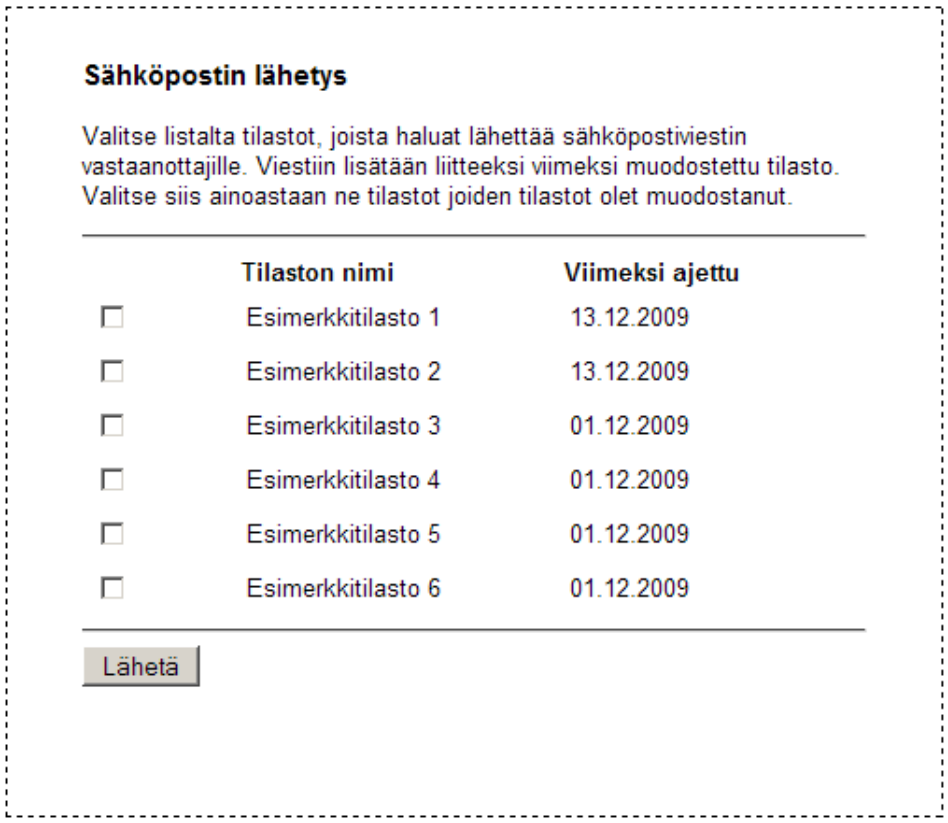
Kuvio 9.Järjestelmän etusivu

Etusivun tilastolinkeistä aukeaa view.php-sivu, jossa voi tarkastella yksittäisen tilaston tietoja. Muokkaa-linkkiä painamalla avautuu tilaston muokkausnäkymä (edit.php), josta tilaston tietoja voidaan muokata. Painettaessa tallenna painiketta editSubmit.php-lomakkeenkäsittelijä tarkastaa syötteiden oikeellisuuden ja vertaa syötteiden arvoja olion tämänhetkisiin arvoihin. Jos arvo on muuttunut, asetetaan olion asetinmetodilla olion uudeksi arvoksi syötteenä saatu arvo. Tämän jälkeen Database-luokan storeItem-metodilla voidaan tallentaa muuttunut Statistic-luokan olio tietokantaan sm_statistic-tauluun. Esimerkkikoodi 13 esittää syötteiden tarkastuksen, syötteiden asettamisen olion tiedoiksi ja olion tietojen tallentamisen tietokantaan.

```php
// Filtering values
if ((!filter_var($status,FILTER_VALIDATE_INT)) or ($status-1 > 0)) {
    header("Location: edit.php?id=$id&msg=Tarkista status");
}

if (strlen($info) > 2000) {
    header("Location: edit.php?id=$id&msg=Info ylittää maksimipituuden (2000 merkkiä).");
}

if (Page::filterDate($lastRun) {
    header("Location: edit.php?id=$id&msg=Viimeksi ajettu -päivämäärä ei ole kelvollinen.");
}

if (Page::filterDate($nextRun) {
    header("Location: edit.php?id=$id&msg=Seuraava ajo -päivämäärä ei ole kelvollinen.");
}

if ($info != $stat->getInfo()) {
    $stat->setInfo($info);
}
if ($status != $stat->getStatus()) {
    $stat->setStatus($status);
}
if ($lastRun != $stat->getLastRun()) {
    $stat->setLastRun($lastRun);
}
if ($nextRun != $stat->getNextRun()) {
    $stat->setNextRun($nextRun);
}
```

Esimerkkikoodi 13. Syötteiden tarkistus ja olion tietojen päivittäminen

Kun halutut tilastot on muodostettu, voidaan sähköpostinlähetystoiminnon avulla lähettää juuri muodostetut tilastot vastaanottajille. Tämä tapahtuu klikkaamalla navigointipalkista ”Sähköpostin lähetys” -linkkiä. Linkki aukaisee mail.php-sivun. Sivulla on lista kaikista tilastoista ja niiden perässä päivämäärä, milloin ne on viimeksi muodostettu. Tästä listasta voidaan valita tilastot, joista lähetetään sähköposti-ilmoitus asiakkaalle (Kuvio 10).



Kuvio 10. Tilastojen sähköpostinlähetys

7 Yhteenveto

Toimeksiannon tavoitteena oli suunnitella ja toteuttaa toimeksiantajalle järjestelmä toistuvien tilastojen toimittamiseksi asiakkaille ja tilastojen hallitsemiseksi. Toteutustekniikaksi valitsin PHP-ohjelmointikielen ja tietovarastona toimi Etuovi.com-tuoteperheen käyttämä Oraclen tietokanta. PHP-ohjelmoinnista minulla oli aikaisempaa kokemusta suoritettujen opintojen muodossa. Oraclen tietokanta oli tullut erittäin tutuksi työtehtävissäni Etuovi.comin tuotantoryhmässä, kuten myös PL/SQL-ohjelmointi.

Suurimmat haasteet järjestelmän toteutuksessa olivat järjestelmän suunnittelussa, sillä itsenäisen järjestelmän suunnittelusta minulla ei ollut aikaisempaa kokemusta. Sovelluksen suunnittelussa tuli ottaa huomioon, että järjestelmä kykenee suoriutumaan kaikista tämänhetkisistä tilastoista ja uusia tilastoja voitaisiin lisätä järjestelmään tarvittaessa. Järjestelmä pitää tuotantoryhmää myös ajantasalla tilastojen muodostamisesta, ajamalla Cron-ajastetun PHP-scriptin päivittäin samaan kellonaikaan. Scripti lähettää tarvittaessa tiedon ajettavista tilastoista tuotantoryhmälle sähköpostitse.
Ominaisuuden ansiosta kenenkään henkilön ei erikseen tarvitse muistaa tilastojen ajamisesta, vaan järjestelmä tekee sen automaattisesti.

Omasta mielestäni toteuttamani järjestelmä vastaa hyvin sille asetettuja vaatimuksia, ja helpottaa tuotantoryhmän työskentelyä toimitettavien tilastojen osalta. Niiden toimittamiseen kuluu huomattavasti vähemmän aikaa, ja kuten jo mainittu, tilastojen muodostamisen muistamisesta ei kenenkään tarvitse erikseen vastata, vaan se on jätetty järjestelmän hoidettavaksi.

Henkilökohtaisesti opinnäytetyöprojektini oli erittäin vaativa, mutta samalla erittäin hyödyllinen. Perehdyin paljon olio-ohjelmointiin PHP-ohjelmointikielellä ja kokonaisen projektin suunnittelu alusta alkaen tuli tutuksi. Opinnäytetyössäni huomasin, kuinka tärkeää ohjelmiston huolellinen suunnittelu on projektin onnistumiseksi.

7.1 Kehitettävää

Jatkokehityksenä tilastojen muodostamisen ja niiden lähetyksen asiakkaille voisi automatisoida. Tällä hetkellä tämä ei kuitenkaan vielä ole järkevää, sillä järjestelmää ei ole vielä otettu käyttöön. Järjestelmän virheetön toimivuus tulee vielä testata testiympäristössä ja järjestelmän toimintaa tulee tarkkailla käyttöönoton jälkeen tarkasti, jotta mahdolliset virheet saadaan eliminoitua. Kun järjestelmän toimivuus on todettu tuotantoympäristössä, voidaan Cron-ajastetun tarkastusscriptin toimintaa laajentaa automaattisella tilastojen muodostuksella ja sähköpostinlähetyksellä.

Järjestelmä ei tällä hetkellä mahdollista uusien tilastojen lisäämistä käyttöliittymän avulla. Uudet tilastot lisätään sijoittamalla uuden tilaston SQL-kyselyt järjestelmään, ja lisäämällä tilaston tiedot suoraan tietokantaan. Järjestelmä mahdollistaisi tällaisen ominaisuuden lisäämisen melko helposti, mutta tämä ominaisuus rajattiin toistaiseksi järjestelmän ulkopuolelle. Uusia tilastoja tulee kuitenkin enää melko harvoin, joten tilaston lisäämisominaisuuden toteuttaminen saattaisi vaatia liika työtunteja siihen nähden, kuinka vähän aikaa kuluu uuden tilaston lisäämiseen suoraan tietokantaan.

Lähdeluettelo

Painetut

Barney, Lee & McLaughling, Michael 2008. Oracle Database AJAX & PHP Web Application Development. The McGraw-Hill Companies, Inc.

Gagné, Marcel 2003. Linux System Administration – A User's Guide. Addison-wesley Professional

Feuerstein, Steven & Pribyl, Bill 2002. Oracle PL/SQL Programming. O'Reilly & Associates, Inc.

Frisch, Aeleen 1997. Essential Systen Administration, Second Edition. O'Reilly & Associates, Inc.

Gilmore, W. J. 2006. Beginning PHP and MySQL 5: from novice to professional, Second edition. Apress.

Hovi, Ari, Huotari, Jouni & Lahdenmäki, Tapio 2005. Tietokantojen suunnittelu & indeksointi. Jyväskylä: Docendo Finland Oy.

Hovi, Ari 2004. SQL-opas. Jyväskylä: Docendo Finland Oy

Koskimies, Kai 2000. Oliokirja. Jyväskylä: Gummerus Kirjapaino Oy.

Lerdorf, Rasmus & Tatroe, Kevin 2002. Programming PHP. O'Reilly & Associates, Inc.

Musciano, Chuck & Kennedy, Bill 2006. HTML & XHTML: The Definitive Guide, 6th Edition. O'Reilly Media, Inc.

Niederst Robbins, Jennifer 2006. HTML and XHTML pocket reference, third edition. O'Reilly Media, Inc

Online

Oracle Corporation 2003. Oracle Database SQL Reference 10g Release 1 (10.1). [online] [viitattu 07.01.2010] http://download.oracle.com/docs/cd/B14117_01/server.101/b10759.pdf

Liitteet

Liite 1 Relaatiotietokannan taulurakenne

Statistic

Column Name	Data Type	Nullable	Data Default	COLUMN ID	Primary Key	COMMENTS
STAT_ID	NUMBER(10,0)	No	(null)	1	1	(null)
STAT_NAME	VARCHAR2(500 BYTE)	No	(null)	2	(null)	(null)
CREATEDATE	DATE	No	(null)	3	(null)	(null)
UPDATEDATE	DATE	Yes	(null)	4	(null)	(null)
STATUS	NUMBER(1,0)	No	(null)	5	(null)	(null)
LAST_RUN	DATE	Yes	(null)	6	(null)	(null)
NEXT_RUN	DATE	Yes	(null)	7	(null)	(null)
INFO	VARCHAR2(2000 BY...	Yes	(null)	8	(null)	(null)
NEW_EXCEL_FLAG	NUMBER(1,0)	No	(null)	9	(null)	(null)
FILENAME	VARCHAR2(500 BYTE)	Yes	(null)	10	(null)	(null)

Query

Column Name	Data Type	Nullable	Data Default	COLUMN ID	Primary Key	COMMENTS
QUERY_ID	NUMBER(10,0)	No	(null)	1	1	(null)
STAT_ID	NUMBER(10,0)	No	(null)	2	(null)	(null)
CREATEDATE	DATE	No	(null)	3	(null)	(null)
UPDATEDATE	DATE	Yes	(null)	4	(null)	(null)
STATUS	NUMBER(1,0)	No	(null)	5	(null)	(null)
QUERY_PATH	VARCHAR2(100 BYTE)	No	(null)	6	(null)	(null)
QUERY_FILENAME	VARCHAR2(100 BYTE)	No	(null)	7	(null)	(null)
QUERY_SCHEMA	VARCHAR2(50 BYTE)	No	(null)	8	(null)	(null)
NAME	VARCHAR2(100 BYTE)	Yes	(null)	9	(null)	(null)
SEQ	NUMBER(5,0)	No	(null)	10	(null)	(null)
GENERATE_DEFAULT_EXCEL_FLAG	NUMBER(1,0)	No	(null)	11	(null)	(null)

Receiver

Column Name	Data Type	Nullable	Data Default	COLUMN ID	Primary Key	COMMENTS
MAIL_ID	NUMBER(10,0)	No	(null)	1	1	(null)
STAT_ID	NUMBER(10,0)	No	(null)	2	(null)	(null)
CREATEDATE	DATE	No	(null)	3	(null)	(null)
UPDATEDATE	DATE	Yes	(null)	4	(null)	(null)
STATUS	NUMBER(1,0)	No	(null)	5	(null)	(null)
EMAIL	VARCHAR2(100 BYTE)	No	(null)	6	(null)	(null)
CC_FLAG	NUMBER(1,0)	No	(null)	7	(null)	(null)

Run day

Column Name	Data Type	Nullable	Data Default	COLUMN ID	Primary Key	COMMENTS
STAT_ID	NUMBER(10,0)	No	(null)	1	(null)	(null)
RUN_DAY	NUMBER(2,0)	No	(null)	2	(null)	(null)

Liite 2 Tietokannan taulujen luontilauseet

```
CREATE TABLE sm_statistic (
   stat_id        NUMBER(10)       NOT NULL,
   stat_name      VARCHAR2(500)    NOT NULL,
   createdate     DATE             NOT NULL,
   updatedate     DATE,
   status         NUMBER(1)        NOT NULL,
   last_run       DATE,
   next_run       DATE,
   info           VARCHAR2(2000),
   new_excel_flag NUMBER(1)        NOT NULL,
   filename       VARCHAR2(500),
   CONSTRAINT sm_statistic_pk PRIMARY KEY (stat_id)
);
CREATE TABLE sm_receiver (
   mail_id        NUMBER(10)    NOT NULL,
   stat_id        NUMBER(10)    NOT NULL,
   createdate     DATE          NOT NULL,
   updatedate     DATE,
   status         NUMBER(1)     NOT NULL,
   email          VARCHAR(100) NOT NULL,
   cc_flag        NUMBER(1)     NOT NULL,
   CONSTRAINT sm_mail_pk
      PRIMARY KEY (mail_id),
   CONSTRAINT sm_mail_fk
      FOREIGN KEY (stat_id)
      REFERENCES sm_statistic(stat_id)
);
CREATE TABLE sm_query (
   query_id       NUMBER(10)     NOT NULL,
   stat_id        NUMBER(10)     NOT NULL,
   createdate     DATE           NOT NULL,
   updatedate     DATE,
   status         NUMBER(1)      NOT NULL,
   query_path     VARCHAR2(100) NOT NULL,
   query_filename VARCHAR2(100) NOT NULL,
   query_schema   VARCHAR2(50)  NOT NULL,
   name           VARCHAR2(100),
   seq            NUMBER(5)      NOT NULL,
   generate_default_excel_flag NUMBER(1) NOT NULL,
   CONSTRAINT sm_query_pk
      PRIMARY KEY (query_id),
   CONSTRAINT sm_query_fk
      FOREIGN KEY (stat_id)
      REFERENCES sm_statistic(stat_id)
);
CREATE TABLE sm_run_day (
   stat_id NUMBER(10) NOT NULL,
   run_day NUMBER(2)  NOT NULL,
   CONSTRAINT sm_run_day_fk
      FOREIGN KEY (stat_id)
      REFERENCES sm_statistic(stat_id)
);
```