VIRTUAALIVANHUS-TIETOKANNAN SUUNNITTELU JA TOTEUTUS

Opinnäytetyö

Annika Immonen

Tietotekniikan koulutusohjelma

Ohjelmistotekniikka

SAVONIA-AMMATTIKORKEAKOULU TEKNIIKKA KUOPIO

Koulutusohjelma

Tietotekniikan koulutusohjelma, ohjelmistotekniikka

Tekijä

Annika Immonen

Työn nimi

Virtuaalivanhus-opetuspelin tietokannan suunnittelu ja toteutus

Työn laji	Päiväys	Sivumäärä
Opinnäytetyö	7.6.2011	45+31

Työn valvoja	Yrityksen yhdyshenkilö
Lehtori Sami Lahti	Projekti-insinööri Hannu Karppinen

Yritys

Savonia-ammattikorkeakoulu, Digitaalisen median kehityskeskus

Tiivistelmä

Opinnäytetyö tehtiin osana Virtuaalivanhus-projektia Savonia-ammattikorkeakoulun Digitaalisen median kehitysyksikölle. Virtuaalivanhus on terveydenhoito-oppilaitosten opiskelijoille suunnattu opetuspeli. Projektiin osallistuu neljä koulua ja projektin tavoite on saada peli julkaistua vuonna 2012. Pelin tavoitteena on lisätä vanhustyön mielekkyyttä ja tehostaa opiskelua. Opinnäytetyön tavoitteena oli tietokannan ensimmäisen version suunnittelu ja toteutus. Tietokannan lisäksi opinnäytetyöhön kuului käyttöliittymätekniikoiden esitutkimus.

Tietokannan suunnittelu tehtiin syksyn 2010 määrittelyiden pohjalta. Sisällöntuottajien tarinat ja testidata tukivat määrittelyä. Tietokanta suunniteltiin relaatiomallilla ja tietokannan rakenne tehtiin monipuoliseksi ja helposti jatkokehitettäväksi. Tietokannan vaatimuksena oli myös videoiden ja kuvien tallentaminen. Tietokanta suunniteltiin Microsoft Visio-ohjelmalla. Pelin pisteet määräytyvät pelin hahmon neljästä eri hyvinvointialueen pistemäärästä. Hyvinvointialueen objektien käsittely suunniteltiin tietokannassa käyttäen EAV-mallia. Tietokannan toteutus tehtiin SQL Server 2008 R2 tietokantapalvelimelle käyttäen Management Studio-työkalua. Tietokannan tauluja ja yhteyksiä testattiin testidatalla.

Opinnäytetyön tavoite saavutettiin. Tietokannasta tuli laaja ja paljon tietoa sisältäessään voi tietokannan koko kasvaa suureksi, mutta kattaa määrityksissä suunnitellut vaatimukset.

Avainsanat

tietokannat, relaatiomalli, Virtuaalivanhus-projekti, SQL

Luottamuksellisuus

julkinen

SAVONIA UNIVERSITY OF APPLIED SCIENCES

Degree Programme

Information Technology

Author

Annika Immonen

Title of Project

Database Design and Implementation of Virtuaalivanhus Teaching Game

Type of Project	Date	Pages
Final Project	7June 2011	45+31

Academic Supervisor	Company Supervisor
Mr. Sami Lahti, Lecturer	Mr. Hannu Karppinen, Project Engineer

Company

Savonia University of Applied Sciences

Abstract

This final project was made as part of the Virtuaalivanhus project in Savonia University of Applied Sciences Digital Media Development Unit. Virtuaalivanhus is a teaching game aimed for health education students. The project involves four schools and its goal is to release the game in 2012. The aim of the game is to increase the sense of geriatric nursing and enhance learning. The aim of the thesis was the first version of the database design and implementation. In addition, the final project consisted of research for user interface technologies.

The design of the database was based on the definitions of August 2010. The content producers' stories and test data supported the definition. The database was designed to a relational model and the structure of the database was made versatile and easy to develop in the future. One requirement of the database was also to save videos and images. The database was designed with Microsoft Visio program. The game score was determined by the characters' four different well-being points. The processing of the data of the wellbeing objects in the database the EAV model was used. The implementation of the database was made for SQL Server 2008 R2 database server using Management Studio tool. The database tables and references were tested with the test data.

As a result, the main goal was achieved. The database became large and while containing a lot of data the size of the database can grow large. However, it covers the specified requirements.

Keywords

databases, relational model, Virtuaalivanhus project, SQL

Confidentiality

public

SISÄLTÖ

1 JOHDANTO 6
2 TIETOKANNAT 8
2.1 Historia 8
2.2 Relaatiotietokannat 9
2.3 Tietueet 9
2.4 Kentät 9
2.5 Avaimet 10
3 SQL 11
3.1 Tietokannan luominen 11
3.2 Lisääminen, poistaminen, muokkaaminen 12
3.3 Kyselyt 13
3.4 SQL Server 2008 R2 14
3.5 SQL Server Management Studio 14
4 VIVA-TIETOKANNAN SUUNNITTELU 15
4.1 ER-malli 15
4.1.1 Kohdetyypit 16
4.1.2 Suhteet 17
4.1.3 Ominaisuudet 18
4.1.4 Kardinaalisuus 18
4.2 Relaatiokaavio 20
4.3 Normalisointi 27
4.4 EAV-malli 27
5 VIVA-TIETOKANNAN TOTEUTUS 32
6 TOTEUTUSTEKNIIKOIDEN ESITUTKIMUS VIRTUAALIVANHUS-PELIN KÄYTTÖLIITTYMÄÄN 39
6.1 Flash 39
6.2 Silverlight 40
6.3 HTML5 40
7 TYÖN TULOKSET 42
LÄHTEET 44

LIITTEET

Liite 1 Projektisuunnitelma

Liite 2 Määrittelydokumentti

1 JOHDANTO

Tämä opinnäytetyö tehtiin osana Virtuaalivanhus-nimellä käynnistettyä projektia. Projektin tarkoitus on julkaista opetuskäyttöön tarkoitettu peli terveydenhoitoalan opiskelijoille. Projektiryhmässä on mukana neljä oppilaitosta: Savonia-ammattikorkeakoulu, Pohjois-Karjalan ammattikorkeakoulu, Savon ammatti- ja aikuisopisto ja Pohjois-Karjalan koulutuskuntayhtymä. Projekti alkoi elokuussa 2010 ja arvioitu julkistamisaika on 2012 vuoden lopussa. Opetuspeli on suunnattu toisen ja kolmannen asteen opiskelijoille ja sen sisältö on rajattu vastaamaan oppilaitosten tiettyjä kursseja.

Pelin ideana on virtuaalivanhuksen hoitaminen. Opiskelija menee pelissä hoitajan roolissa kotikäynnille ja suorittaa tunneilla oppimiaan asioitaan. Opiskelija saa toiminnoistaan pisteitä, jotka näytetään pelin lopussa analyysi-osiossa opiskelijalle. Pelin kautta opiskelija oppii myös havainnoimaan esimerkiksi muistihäiriöisen ikäihmisen turvallisuutta, kotona selviytymistä parantavia tekijöitä ja iäkkään ihmisen arkipäivän elämää.

Projektin tavoitteena on lisätä vanhustyön mielekkyyttä ja saada enemmän opiskelijoita valitsemaan vanhustyö suuntautumisekseen. Vanhustyö pääsuuntautumisena on ollut vähäistä viime vuosien aikana, vaikka alalla on nyt ja tulevaisuudessa tekijöistä pula. Lisäksi projektin avulla pystyttäisiin kouluttamaan enemmän motivoituneita vanhustyön ammattilaisia ja näin edistettäisiin ikääntyneiden kotona selviytymistä. Projektiin kuuluu Virtuaalivanhus-pelin lisäksi Second Lifessa sijaitseva ”Ideaalikoti”, johon on kytketty yrityksiä, jotka tarjoavat ikääntyville tukivälineitä kotiin. Projektin avulla uudistetaan vanhustyön opetusta ja tehdään oppimisprosessista asiakaslähtöinen.

Projekti on saanut rahoitusta Euroopan sosiaalirahastolta ja Pohjois-Savon elinkeino-, liikenne- ja ympäristökeskukselta. Savonia-ammattikorkeakoulun Digitaalisen Median Kehityskeskus vastaa projektin teknisestä puolesta, eli ohjelman toteutuksesta. Digitaalisen Median Kehityskeskuksesta projektiryhmään kuuluvat Hannu Karppinen, joka vastaa tuotannon teknisestä koordinoinnista, Hannu Hoffren, jonka vastuualueena on pelin toteutus, Mikko Pääkkönen, joka vastaa tietokannan rakentamisesta ja Helena Meriläinen, jonka vastuualueena on käyttöliit-

tymäsuunnittelu ja käytettävyystestaus. Kokonaisuudessaan projektiryhmään kuuluu parikymmentä henkilöä ja projektipäällikkönä toimii Savonia-ammattikorkeakoulun terveysalan puolelta Päivi Tiilikainen.

Opinnäytetyö ja osuuteni projektissa oli suunnitella ja toteuttaa syksyllä 2010 tehtyjen määrittelyiden perusteella Virtuaalivanhus-pelin tietokanta. Tietokanta on ensimmäinen versio ja muuttunee projektin edetessä. Tavoitteena on suunnitella tietokanta, jota on helppo kehittää ja ylläpitää. Opinnäytetyöhöni kuului myös tutkimus kolmesta käyttöliittymätekniikasta, joista yhden projektiryhmä mahdollisesti valitsisi Virtuaalivanhus-pelin toteutukseen.

Suoritin työharjoittelun Savonia-ammattikorkeakoulun Digitaalisen median kehitysyksikössä osittain samaan aikaan opinnäytetyön tekemisen kanssa, joten projektin jäsenet olivat minulle tuttuja entuudestaan. Työharjoittelussa tutustuin Management Studio-työkaluun, joten erillistä koulutusta opinnäytetyötä varten ei tarvittu.

Tietokannan suunnittelussa käytin aluksi paperia ja kynää ja sen jälkeen Microsoft Office Visiota. Projektin alussa sain paljon aineistoa projektista, projektin tavoitteista ja määrittelypalavereiden kautta pelin vaatimuksista ja ideasta. Opinnäytetyön alussa kirjoitin myös oman määrittelydokumentin. Aineiston lähteisiin hankin kirjoina kirjastoista ja verkkodokumentteina Internetistä.

Virtuaalivanhus-tietokannasta tuli suuri, mutta tietokanta pystyy varastoimaan tiedot, mitkä määriteltiin.

Opinnäytetyöni kertoo yleisesti tietokannoista ja niihin liittyvästä SQL-kielestä, Virtuaalivanhus-tietokannan suunnittelusta, valituista suunnittelumenetelmistä ja tietokannan toteutuksesta. Opinnäytetyön lopussa tutkin kolmea käyttöliittymätekniikkaa.

Opinnäytetyön projektisuunnitelmassa on kerrottu opinnäytetyön vaiheistuksesta, aikataulusta, riskeistä ja projektin tavoitteista. [Liite 1]

2 TIETOKANNAT

Tietokanta (Database, DB) on varasto, jossa on tietoa. Tietokannat ovat hyvin yleisiä ohjelmistoissa ja myös jokaisen ihmisen arkielämässä. Tietokantana voidaan pitää esimerkiksi kalenteria, luetteloa dvd-levyistä tai auton huoltokirjaa. Tietokannat helpottavat tietojen ylläpitoa ja etsimistä. Melkein jokainen ohjelmisto sisältää tietokannan ja suurimmissa ohjelmistoissa voi olla käytössä jopa kymmeniä tietokantoja.

Tietokanta rakentuu tiedoista, joilla on merkitys. Tietokantaan tallennetaan ne tiedot, joita käyttäjä haluaa myöhemmin käsitellä. Tietokanta voi olla pieni, vain yhden taulun kokoinen, tai suuri ja monimutkainen.

Yhteen tietokantaan tallennetaan yleensä yhden ohjelmiston tiedot. Tietokannan käyttäminen vaatii tietokantaohjelmiston käyttöä, joita on saatavilla maksullisista aina ilmaisiin saakka. Tietokannat voidaan jakaa kahteen kategoriaan: tiedosto- ja palvelinpohjaisiin ratkaisuihin. Tiedostopohjainen tietokanta sijaitsee tiedostona tietokoneella, jota asiakasohjelmat käyttävät. Palvelinpohjainen tietokanta taaskin on erillisellä tietokantapalvelimella ja tietokantaa käyttävät ohjelmat ottavat yhteyden siihen. Palvelinpohjainen ratkaisu on yleisimmin käytössä suuremmissa ohjelmistoissa, koska siinä on parempi suorituskyky, turvallisuus ja luotettavuus. [1]

2.1 Historia

Tietokantojen historia alkaa 1960-luvulta. Ensimmäiset tietokannat olivat hierarkkisia tietokantoja, joissa tiedot järjestettiin hierarkkiseen muotoon. Muoto muistuttaa puu-mallia, jossa ylimpänä on juuri, jolla on lapsitasoja ja niillä on lapsitasoja jne. Tietokantojen rakenne kuitenkin oli jäykkä eikä tietojen turhaa toistoa pystytty välttämään.

Verkkotietokannat korjasivat hierarkkisten tietokantojen ongelmat. Tietojen väliset suhteet korvattiin joukkoina. Erona hierarkkisiin tietokantoihin oli se, että lapsitasolla pystyi olemaan useampia äititasoja.

Vuonna 1970 Edgar F. Codd esitteli relaatiomallin, joka oli siihen mennessä yksinkertaisin ja joustavin malli. Luvussa 2.2 esitellään lisää relaatiomallia. [2]

2.2 Relaatiotietokannat

Relaatiomalli on maailman suosituin ja käytetyin tietokantamalli. Relaatiotietokantaan tiedot tallennetaan tauluihin. Yksi taulu muodostuu tietueista ja tietue muodostuu kentistä. Tietueiden ja kenttien järjestyksellä ei ole väliä, vaan tietueen tunnistaa kentän avulla, jossa on muista poikkeava arvo, uniikki arvo. Tämä ominaisuus mahdollistaa sen, ettei tietojen olemassaolo riipu tavasta tallentaa tietokantaan eli käyttäjän ei tarvitse tietää relaatiomallissa tietokannan rakenteesta mitään tietoja tallentaessaan. Tämä eroaa hierarkkisessa ja verkkotietokantamalleista, joissa on tärkeää tuntea tietokannan rakenne tietoja haettaessa. Relaatiomallissa tauluille luodaan yhteyksiä, jotka voivat olla mistä taulusta tahansa mihin vain. [3]

2.3 Tietueet

Taulu sisältää yhden tai useampia tietueita. Tietue on yksi rivi, jossa on kaikki taulun kentät. Tietueessa on yksi tai useampia kenttiä. Esimerkiksi auton huoltokirjassa tietue vastaa yhtä huoltokertaa ja puhelinluettelossa yhtä ihmistä. Tietokantaan lisätään aina yksi tietue kerrallaan. Poistettaessa tietoa poistetaan koko tietue. Jokaisella tietueella pitää olla vähintään yksi perusavain, joka yksilöi tietueen. Perusavaimista lisää kohdassa 2.5. [2]

2.4 Kentät

Kentät määrittävät tauluun tallennettavaa tietoa. Niitä voi olla yksi tai useampia. Esimerkiksi auton huoltokirja-tietokannassa kenttinä voisivat olla *päivämäärä*, jolloin autoa huollettiin, ja tekstikenttä, joka oli *huollettava asia*. Tietueen yksilöivä kenttä olisi *id*, joka olisi automaattisesti tallentuva juokseva luku.

Kenttiä luotaessa on määritettävä kaksi pakollista ominaisuutta: *kentän nimi* ja *tietotyyppi*. Muita määriteltäviä ominaisuuksia ovat kentän *maksimipituus* eli kuinka pitkän arvon kenttään voi tallentaa, *null-määritys* eli onko kenttään pakko syöttää tieto, *tarkistukset*, jotka rajaavat syötettävää tietoa, *oletusarvo*, kentälle voi asettaa jonkin arvon valmiiksi, ja *syöttömaski*, joka kuvaa kenttään tallennettavat merkit.

Tietokannasta haetaan tietoa kenttien mukaan, ei tiedon järjestyksen perusteella. Auton huoltokirja-tietokannassa tietueet voitaisiin järjestellä päivämääräkentän mukaan aikajärjestykseen. [2]

2.5 Avaimet

Jokaisessa taulussa on siis vähintään yksi pääavain (primary key). Pääavain yksilöi tietueen, eli jokaisella pääavain-kentällä on oma arvo, joka ei voi olla missään muussa saman taulun pääavain-kentässä. Toisin sanoen samaa arvoa ei voi olla. Auton huoltokirjassa se olisi id, puhelinluettelossa puhelinnumero ja henkilötietorekisterissä sosiaaliturvatunnus. Kun taulussa on useampia pääavaimia, samaa pääavainyhdistelmää ei voi olla toista.

Taulussa voi olla myös vierasavaimia (foreigner key). Vierasavain muodostetaan kahden taulun välille. Vierasavain koostuu tiedosta, jota tarvitaan kummassakin taulussa. Esimerkiksi asiakasrekisterissä olisi asiakkaat-taulu, jossa olisi kentät asiakasid (pääavain), nimi, osoite ja puhelinnumero, sekä tilaukset-taulu, jossa olisi kentät tilausid (pääavain), asiakasid (vierasavain), tilauspvm, tuoteid ja määrä. Asiakkaat-taulun asiakasid-pääavain yksilöi kaikki asiakkaat ja tilaukset-taulun tilausid-pääavain yksilöi kaikki tilaukset. Tilaukset-taulun asiakasid-vierasavain on viittaus asiakkaaseen ja asiakasid:n mukaan tilauksen tehnyt asiakas on tunnistettavissa. [2]

3 SQL

SQL eli Structured Query Language on standardoitu kyselykieli, jonka kehitti IBM. SQL-kielen avulla saadaan esimerkiksi rakennettua tietokanta ja lisättyä tai poistettua tauluja ja tietoa. Koska kieli on standardoitu, pitäisi samojen kyselyiden toimia kaikissa tietokannoissa.

SQL-kielessä on joitakin vakiosanoja, jotka on varattu kielen omiin toimintoihin. Ne on tapana kirjoittaa kokonaan isoilla kirjaimilla. SQL-kielessä käytetään englantia, joten lauseiden ymmärtäminen ja muodostaminen on helppoa. Usein käskyihin voidaan yhdistellä useita tauluja, hakea vain tietyt kentät tietyn kentän mukaan ja vielä järjestellä tietueet, jolloin lauseista tulee pidempiä ja monimutkaisempia. [4]

3.1 Tietokannan luominen

Ensiksi tietokanta pitää luoda. Luomiseen tarvitaan määrittää tietokannalle nimi, tietokannan omistajan tunnus ja salasana. Tietokannalle voi lisätä myös käyttäjiä.

Tietokannan luonti:

```
CREATE DATABASE db_name;
```

Käyttäjän luonti ilman ja tietokantaan kirjautumisen yhteydessä:

```
CREATE USER user_name;

CREATE LOGIN user_name
   WITH PASSWORD = 'password’;

USE db_name;

CREATE USER user_name FOR LOGIN user_name;

GO
```

Tietokannan poistaminen:

```
DROP DATABASE db_name;
```

3.2 Lisääminen, poistaminen, muokkaaminen

SQL-kielen avulla pystyy hallitsemaan helposti tietokantaa. Sillä voi lisätä, muokata ja poistaa tauluja, kenttiä ja tietueita. Muokatessa ja poistaessa oikean tietueen löytää esimerkiksi yksilöivän kentän avulla.

Taulun luonti:

```
CREATE TABLE table_name
(
column_name1 data_type,
column_name2 data_type,
column_name3 data_type
)
```

Taulun poistaminen:

```
DROP TABLE table_name;
```

Tietueen lisääminen:

```
INSERT INTO table_name
VALUES (value1, value2, value3,...)
```

Tietueen päivitys:

```
UPDATE table_name
SET column1=value, column2=value2,...
WHERE some_column=some_value
```

Tietueen poisto:

```
DELETE FROM table_name
WHERE some_column=some_value
```

3.3 Kyselyt

SQL-kielessä eniten käytetyimmät ovat kyselyt. Kyselyiden avulla käyttäjä hakee tarvitsemansa tiedon tietokannasta. Tietoa voi hakea yhdestä tai useasta taulusta kerrallaan tai tietoa voi hakea pelkkien kenttien mukaan.

Haetaan kaikki tiedot taulusta:

```
SELECT * FROM table_name
```

Haetaan halutun kentän mukaan:

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator value
```

Haetaan kentät yhdistäen kaksi taulua:

```
SELECT column_name(s)
FROM table_name1
INNER JOIN table_name2
ON table_name1.column_name=table_name2.column_name
```

Haetaan kentät halutun kentän mukaan järjestyksessä:

```
SELECT column_name(s)
FROM table_name
ORDER BY column_name(s) ASC|DESC
```

[5] [6]

3.4 SQL Server 2008 R2

SQL Server on Microsoftin kehittämä Microsoft Windows-käyttöjärjestelmälle suunnattu tietokantapalvelin. SQL Server on julkaistu monilla eri kielillä ympäri maailmaa. Ensimmäinen SQL Server (1.0) julkaistiin vuonna 1989 ja viimeisin julkaistu on vuonna 2010 julkaistu SQL Server (2008 R2), joka on kolmastoista SQL Serverien listassa.

Palvelimella on omat hallintaohjelmat, joiden avulla tietokantojen ylläpito ja hallinta helpottuu. Palvelin on luotettava ja tehokas ja suunnattu suuriin ohjelmistoihin. Palvelin tukee myös esimerkiksi erilaisia työkaluja, jotka antavat raportteja palvelimesta tai jonka avulla pystyy tallentamaan tietokantaan tuhansia rivejä samaan aikaan. [7]

3.5 SQL Server Management Studio

SQL Server Management Studio on SQL Server-tietokantojen hallitsemiseen tehty työkalu. Se on yhteensopiva SQL Server 2005 ja uudempien versioiden kanssa. Työkalu yhdistää kolmen aiemman tietokannan hallintatyökalun ominaisuudet. Se on tarkoitettu sekä kehittäjien että pääkäyttäjien käyttöön. Työkalussa on scriptieditori, jolla voi kirjoittaa SQL-kielen komentoja, ja graafinen editori, jolloin työkalu kirjoittaa itse SQL-kielen komennot.

Graafista editoria käyttämällä tietokannan hallitseminen on nopeaa. Työkalu pystyy tietokannan luomisesta saakka aina tietueiden syöttämiseen asti. [6]

4 VIVA-TIETOKANNAN SUUNNITTELU

Niin kuin kaikki projektit tämäkin projekti jakaantuu määrittely-, suunnittelu- ja toteutusvaiheisiin. Tietokannan määrittelyvaihe lähti käyntiin useilla palavereilla muiden projektin oppilaitosten edustajien kanssa. Palavereissa määriteltiin pelin kulkua ja ominaisuuksia, joiden kautta tietokannalta vaadittavat ominaisuudet ja rakenne määräytyi.

Määrittelydokumentissa määriteltiin pelin toimintoja ja pelin kulkua, mikä auttoi tietokannan suunnittelua. Dokumentissa kuvataan myös pelin kulkua ja kerrotaan opettajan ja opiskelijan toiminnoista pelissä. [Liite 2]. Projektiryhmään nimettiin myös sisällöntuottajat, jotka hankkivat esitiedot ja sisältöä peliin eli tarinat, joiden perusteella saatiin tarkat määritykset tietokannalle.

Tietokannan vaatimuksena oli, että se olisi mahdollisimman joustava ja helposti päivitettävissä, koska projekti on pitkä ja tietokannan rakenne mahdollisesti muuttuu projektin edetessä. Tietokannan suunnittelumalleiksi valittiin ER-malli ja EAV-malli, koska tämä olisi joustava, kun tallennettavaa tietoa tulee paljon. EAV-mallista kerrotaan lisää kohdassa 4.4.

Suunnitteluvaiheeseen kuuluu itse tietokannan suunnittelu määrittelyiden perusteella. Mitä paremmin määrittely on tehty, sitä paremmin tietokannan pystyy suunnittelemaan. Suunnittelu aloitetaan tekemällä käsiteanalyysi, jossa suunnitellaan tietokantaa loogisella tasolla ja muodostetaan käsitemalli. Käsitemalli on karkea versio tietokannasta, mutta siinä hahmotellaan pelin tärkeimmät asiat. Käsitemalli pitää sisällään ER-mallin suunnittelun ja sen muuttamisen relaatiomalliksi. Tämän jälkeen relaatiomalliin voidaan syöttää testidataa eli esimerkkiarvoja ja normalisoidaan relaatiot. Normalisoinnilla pyritään tehostamaan tietojen tallennusta tietokantaan normalisoinnin sääntöjen avulla. [8]

4.1 ER-malli

Chenin esittelemä ER-malli (Entity-Relationship model) esiteltiin maailmalle 1976. ER-malli on tunnetuin ja käytetyin tietomalli. Se on helposti muokattavissa ja soveltuu useille tietokantatuotteille.

ER-malli kuvaa kohdealuetta, jota on tarkoitus kuvata tietokannassa ja sen tarkoitus on määrittää tietokannan fyysiselle rakenteelle pohja. Malli on karkea kuvaus tietokannasta ja toimiikin siten hyvänä kommunikaatiovälineenä IT-ammattilaisten ja liiketoiminnan edustajien välillä. ER-malli rakennetaan useasta eri vaiheesta lisäten aina jotain uutta edelliseen suunnitelmaan.

4.1.1 Kohdetyypit

ER-mallin suunnittelussa lähdetään liikkeelle hahmottamalla kohdetyypit eli vahvat ja heikot kohteet. Heikot kohteet ovat riippuvaisia jostakin vahvasta kohteesta, eli ilman vahvaa kohdetta ei olisi heikkoa kohdetta.

Viva-tietokannan kohdetyyppejä ovat esimerkiksi

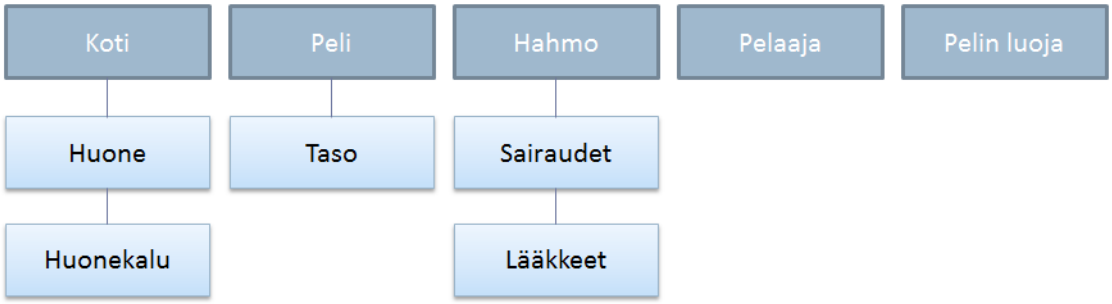
Vahvat kohteet:

- hahmo (Figure)
- pelaaja (Player)
- pelin luoja (GameCreator)
- peli (Game)
- koti (Home).

Heikot kohteet:

- huone, riippuu kodista (Room)
- huonekalu, riippuu huoneista (Furniture)
- sairaudet, riippuu hahmosta (Diseases)
- lääkkeet, riippuu sairauksista (Meds)
- tasot, riippuu pelistä (Levels).

Kuva 1 vahvat kohteet ovat ylimmällä rivillä ja heikot kohteet sen vahvan kohteen alapuolella, mistä ovat riippuvaisia. Koti on vahva kohde, ilman kotia ei olisi huoneita. Huone taas on vahva kohde huonekalulle.



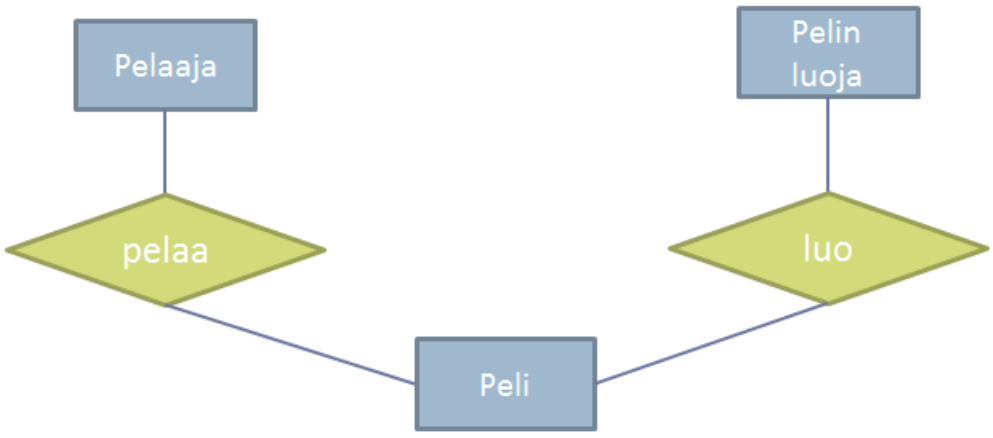
Kuva 1. Vahvat ja heikot kohteet.

4.1.2 Suhteet

Kohdetyyppien välille määritellään suhteet. Suhteet ovat yleensä verbejä, ne ilmaisevat miten kohde liittyy toiseen kohteeseen. Kuva 2 tarkastellaan seuraavia suhteita:

- Pelin luoja luo pelin.
- Pelaaja pelaa peliä.

Piirretään kuvaamalla kohteita suorakulmioilla ja suhteita timanttikuviolla. Timanttikuvioon kirjoitetaan suhteen nimi.

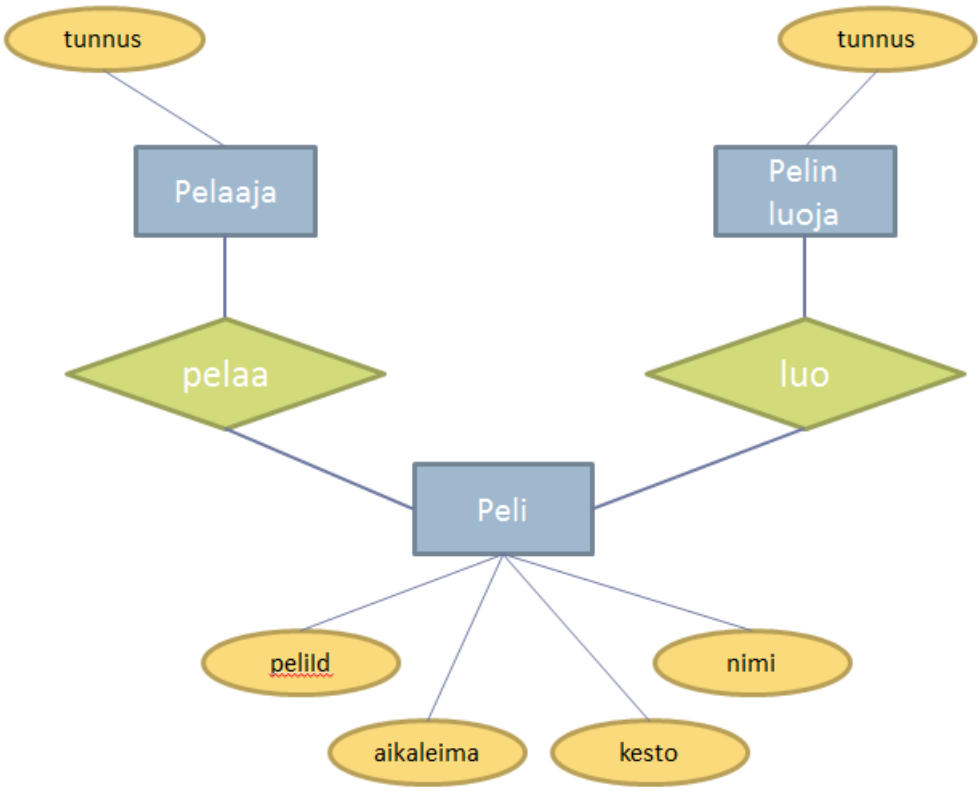


Kuva 2. Pelaajan ja pelinluojan suhteet.

4.1.3 Ominaisuudet

Jokaiselle kohteelle määritellään omat ominaisuutensa. Ominaisuudet ovat niitä tietoja, joita kohteista halutaan tallentaa tietokantaan. Ominaisuudet voivat olla myös tietoa muista tauluista; tällä tavalla taulut saadaan kytkettyä toisiinsa.

Kuva 3 kuvataan ellipsin muotoisilla kuvioilla kohteiden ominaisuuksia. Pelaaja kirjautuu peliin koulun tunnuksilla, joten pelaajasta ei tarvitse tallentaa muuta tietoa kuin käyttäjätunnus. Pelin luoja-taululle tehdään samoin. Pelin ominaisuuksina tallennetaan yksilöivä id, aikaleima, jolloin peli on luotu, pelin kesto ja pelin nimi.



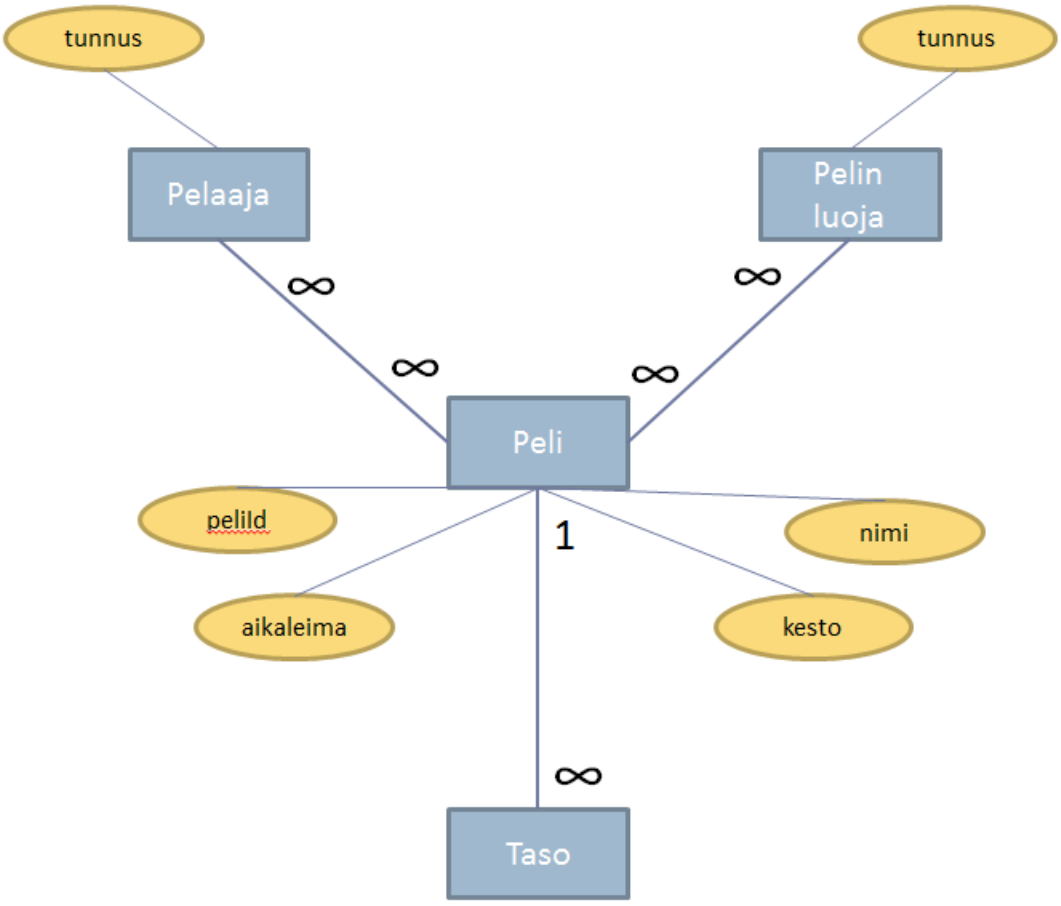
Kuva 3. Pelaaja-, pelinluoja- ja peli-taulujen ominaisuudet.

4.1.4 Kardinaalisuus

Kardinaalisuus kertoo kohteiden suhteiden määrän. Kardinaalisuuksia voi olla kolme erilaista:

- yhden suhde yhteen (one-to-one, 1 to 1)
- yhden suhde moneen (one-to-many, 1 to M) tai monen suhde yhteen (many-to-one, M to 1)
- monen suhde moneen (many-to-many, M to M).

Kuva 4 tarkastellaan suhteiden kardinaalisuuksia. Kardinaalisuus merkitään käyttämällä ykkösen ja äärettömän merkkejä. Ääretön tarkoittaa yhtä tai useampaa. Yksi pelinluoja voi luoda yhden tai useamman pelin ja yhdellä pelillä voi olla yksi tai useampi pelinluoja. Pelaaja voi pelata yhtä tai useampaa peliä. Pelillä voi olla yksi tai useampi pelaaja. Pelissä on yksi tai useampia tasoja ja yksi taso voi olla vain yhdessä pelissä.



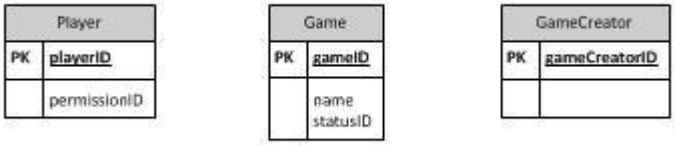
Kuva 4. Kardinaalisuudet.

4.2 Relaatiokaavio

Suunnittelun jälkimmäinen vaihe oli muuttaa tietomalli relaatiokaavioksi. Muuttamiseen pätee seuraavat perussäännöt:

- Jokainen kohde muutetaan relaatioksi.
- Jokainen Many-to-Many-suhde muutetaan relaatioksi.
- M-to-1-suhteet liitetään jo olemassa oleviin relaatioihin.
- Moniarvoisista ominaisuuksista luodaan uusi relaatio.
- Alityypeistä luodaan uudet relaatiot. [2]

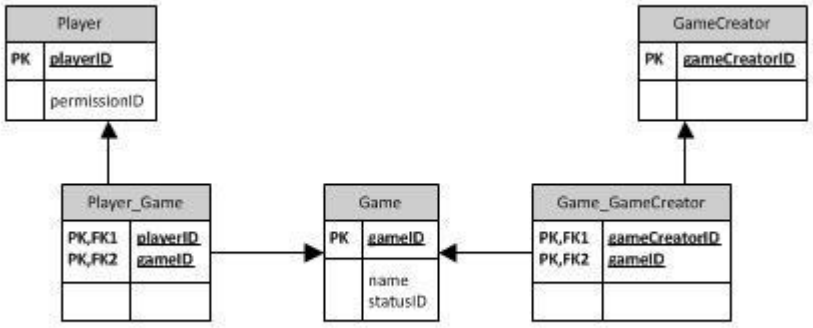
Edellisiä perussääntöjä noudatin myös Virtuaalivanhus-tietokannan suunnittelussa. Kaikki kohteet muutettiin relaatioiksi, esimerkiksi pelaaja (player), peli (game) ja pelinluoja (gameCreator). Jokaisella relaatiolla on uniikki kenttä. Esimerkin tauluissa kaikki uniikit kentät ovat laskurityyppisiä id-kenttiä, joka on juokseva kokonaisluku. Kuva 5. Kohteet muutetaan relaatioiksi. nähdään kolmesta vahvasta kohteesta muodostetut relaatiot.



Kuva 5. Kohteet muutetaan relaatioiksi.

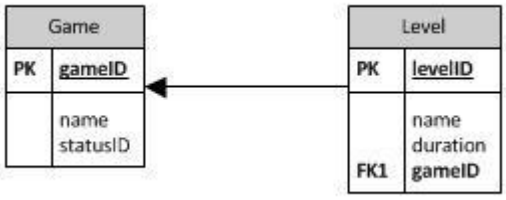
Jokainen Many-to-Many-suhde muutetaan omaksi relaatioksi. Esimerkiksi, pelaaja voi pelata yhtä tai useampaa peliä ja pelissä voi olla yksi tai useampi pelinluoja, jos kurssia opettaa vaikka kaksi opettajaa. Pelaajan ja pelin sekä pelinluojan ja pelin välille muodostetaan uudet relaatiot, jotta monesta moneen suhde toteutuisi. Relaatioiden välille luodaan yhteydet, jolloin uusiin Many-to-Many-suhteesta muodostuneisiin relaatioihin muodostuu vierasavaimet. Pelaaja_Peli-taulun (Player_Game) pelaajaID on Pelaaja-taulun uniikin arvon vierasavain eli viittaus toiseen tauluun. Pelaaja_Peli-taulu muodostuu siis kahdesta vierasavaimesta eli taulussa ei ole mitään uutta kenttää, vaan tieto viittauksesta, jonka avulla pysty-

tään hakemaan tietyn pelaajan kaikki pelit. Kuva 6 esitellään taulut ja niiden monesta moneen suhteista muodostuneet yhdistelmätaulut.



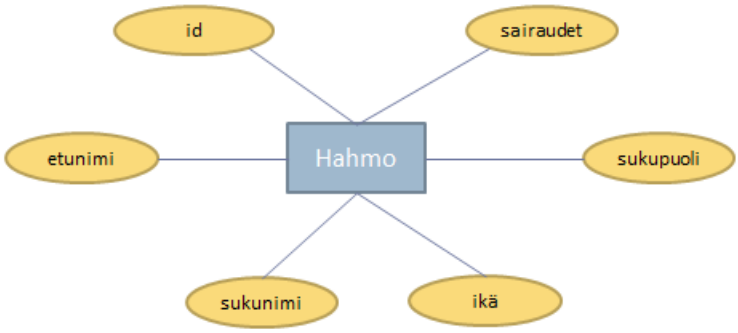
Kuva 6. Monesta moneen-suhteet muutetaan omiksi relaatioiksi.

Kaikki Many-to-One-suhteet liitetään jo olemassa olevaan relaatioon. Esimerkiksi Peli-taulussa (Game) voi olla yksi tai useampi taso (Level), mutta yksi taso voi olla vain yhdessä pelissä. Lisätään Taso-tauluun viittaus mihin peliin se kuuluu uniikin PeliID:n (GameID) avulla. Näin Taso-taulusta voidaan hakea kaikki tiettyyn peliin luodut tasot. Kuva 7 näytetään yhden suhde moneen-perussäännön ratkaisu.



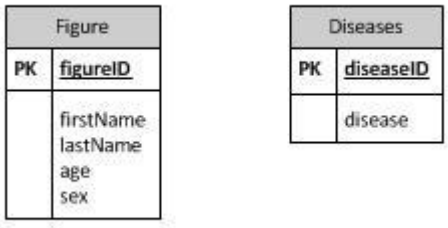
Kuva 7. Monen suhde yhteen-suhteessa viitataan jo olemassa olevaan relaatioon.

Relaatioiden moniarvoisista ominaisuuksista luodaan uusi relaatio, eli jos kenttään voisi tulla useampi arvo, luodaan kentälle oma relaatio. Kuva 8 Hahmolla (Figure) on ominaisuus sairaudet (Diseases), johon voi tulla useampia arvoja tai ei yhtään arvoa.



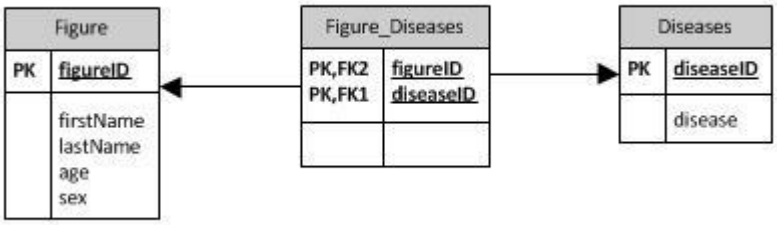
Kuva 8. Kohteella on moniarvoinen ominaisuus.

Kuva 9 Luodaan sairauksille oma taulu ja lisätään sairauksille uniikki id ja sairauden nimi-kenttä.



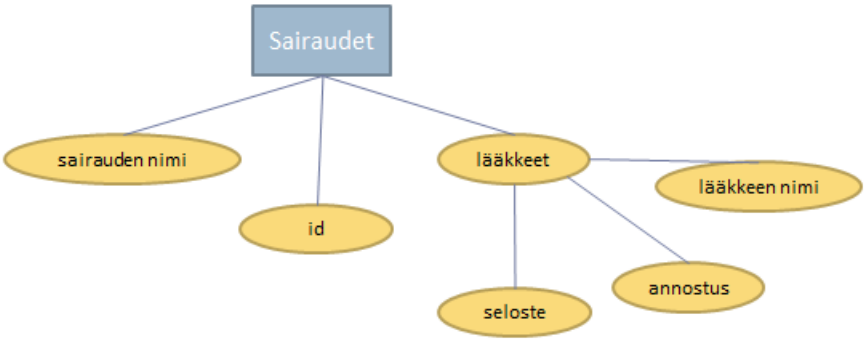
Kuva 9. Moniarvoiselle ominaisuudelle luodaan oma taulu.

Näille tauluille tulee taaskin monesta moneen-suhde. Hahmolla voi olla yksi tai useampia sairauksia ja yksi sairaus voi olla yhdellä tai useammalla hahmolla. Joten käytetään toisena esiteltyä perussääntöä ja luodaan monesta moneen-suhteelle oma taulu. Uuteen tauluun tulee Hahmo- ja Sairaudet-taulujen uniikit id:t. Tämä mahdollistaa hakemaan tietokannasta hahmot sairauden mukaan tai sairaudet, jotka hahmolla on. Kuva 10 näytetään hahmon ja sairauksien monesta moneen-suhteen luoma yhdistelmätaulu ja taulujen väliset suhteet.



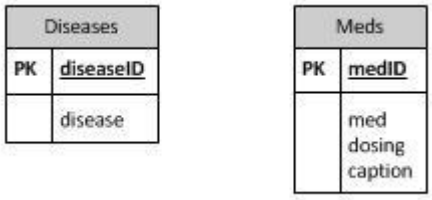
Kuva 10. Monesta moneen-suhteelle luodaan oma relaatio.

Viimeisenä perussääntönä on, että alityypeistä luodaan uudet relaatiot. Jos tauluun tulee kenttiä, jotka eivät kaikki liity suoraan toisiinsa on hyvä hajottaa taulu useammaksi tauluksi. Kuva 11 Sairaudet-kohteella on lääkkeet-ominaisuus.



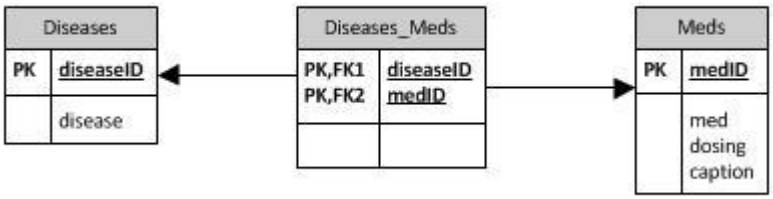
Kuva 11. Kohteella on ominaisuus, joka on alityyppi.

Lääkkeillä taaskin on omia ominaisuuksia, jotka eivät liity sairauksiin, joten lääkkeille muodostetaan oma taulu, johon tulee uniikki id ja ominaisuudet: lääkkeen nimi, annostus ja seloste. Kuva 12 esittää kohteen ja kohteen alityypin muuttamista kahdeksi relaatioksi.



Kuva 12. Alityypille luodaan oma relaatio.

Uudet relaatiot muodostavat myös monesta moneen-suhteen, joten suhteelle luodaan oma relaatio. Sairauteen voi olla yksi tai useampi lääke ja lääke voi kuulua yhteen tai useampaan sairauteen. Kuva 13 näytetään, kun uuteen relaatioon tulee kentiksi Sairaudet- ja Lääkkeet-taulujen uniikit id:t.



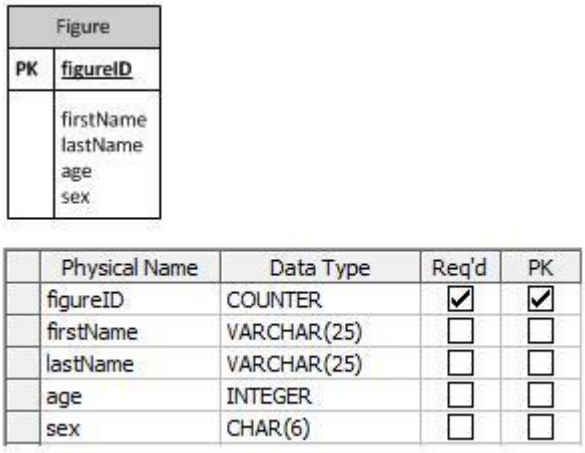
Kuva 13. Monesta moneen-suhteelle luodaan oma relaatio.

Tietokannan suunnitteluun kuuluu myös erilaisten ominaisuuksien määrittäminen relaation kentille. Jokaiselle kentälle pakollisia määrityksiä on kentän nimi ja tietotyyppi. Kentän nimi voi olla 64 merkkiä pitkä ja siinä ei saa käyttää seuraavia merkkejä: piste (.), huutomerkki (!), heittomerkki(’) ja hakasulut ([ ]). Tietotyyppejä on paljon erilaisia. Tekstille, numeroille, päivämäärälle ja esimerkiksi valuutalle on omat tietotyypit. Tekstille ja numeroille on jopa useampia vaihtoehtoja. Virtuaalivanhus-tietokannan suunnittelussa käytin seuraavia tietotyyppejä:

- Varchar (merkkijono)
- Char (merkki),
- Integer (kokonaisluku),
- Counter (laskurityyppinen tietotyyppi, alkaa numerosta yksi ja uusia tietueita luotaessa laskurin arvo on 1+ n, muodostuu automaattisesti),

- Blob (binary large object),
- Timestamp (Date- ja Time-tietotyyppien yhdistelmä eli sisältää tiedot: vuosi, kuukausi, päivä, tunti, minuutti ja sekunti) ja
- Numeric (desimaaliluku). [9] [10]

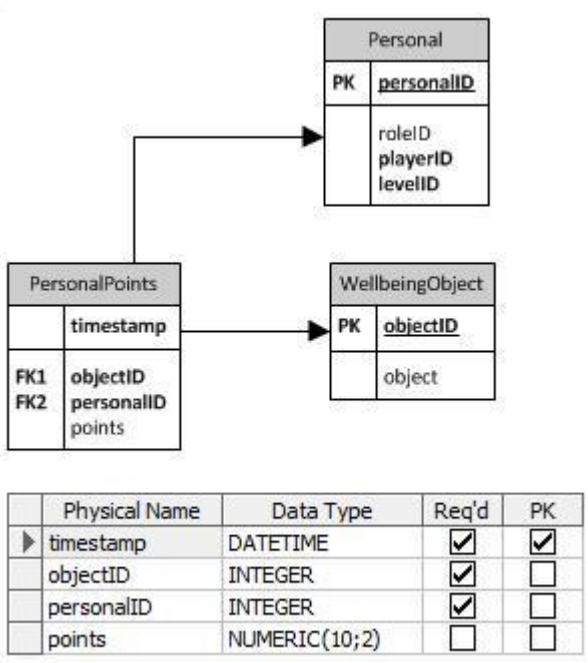
Esimerkiksi, tarkastellaan Hahmo-taulua, jossa on kentät hahmoID, etunimi, sukunimi, ikä ja sukupuoli. HahmoID on laskurityyppinen tieto eli juokseva luku. Etunimi ja sukunimi ovat tekstityyppistä tietoa, joissa sulkuihin on rajattu merkkien maksimimäärä. Ikä on kokonaisluku, jolloin käytetään Integeriä. Vaihtoehtoisesti olisi voinut käyttää myös Date-tietotyyppiä jolloin kentän arvoksi olisi asetettu hahmon syntymäaika. Sukupuoli-kenttään voi tulla joko mies tai nainen, jolloin rajataan tekstin koko kuuteen merkkiin. Hahmo-taulun ainut pakollinen kenttä on hahmoID ja se on myös relaation pääavain. Kuva 14 tarkastellaan Hahmo-relaation tietotyyppejä.



Kuva 14. Hahmo-relaation kentät ja niiden tietotyypit.

Toisessa esimerkissä tarkastellaan Henkilökohtaiset tulokset (PersonalPoints)-taulun tietotyyppejä. Uniikkina kenttänä on Timestamp-niminen kenttä, jonka tietotyyppi on timestamp, eli koostuu tietueen lisäyshetkestä vuodesta aina sekuntiin saakka. KohdeID- ja HenkilökohtainenID-kentät ovat vierasavaimia samannimisistä tauluista. Vierasavaimet ovat omissa tauluissaan tietotyyppiä laskuri ja

viitatussa taulussa Integer-tyyppisiä. Pisteet-kenttä on tietotyyppiä Numeric, koska pisteet voivat olla myös desimaalien tarkkuudella. Tietotyypin sulkuihin on määritelty desimaalien tarkkuus kahdella numerolla. Tässä taulussa pääavain on Timestamp ja vierasavaimina KohdeID- ja HenkilökohtainenID-kentät. Näihin kaikkiin kolmeen on myös pakko syöttää arvo. Kuva 15 nähdään Henkilökohtaiset tulokset-taulun viiteyhteydet ja tietotyypit.



	Physical Name	Data Type	Req'd	PK
▶	timestamp	DATETIME	☑	☑
	objectID	INTEGER	☑	☐
	personalID	INTEGER	☑	☐
	points	NUMERIC(10;2)	☐	☐

Kuva 15. Henkilökohtaiset pisteet-taulun kentät ja tietotyypit.

Blob-tietotyyppiä käytetään tallentamaan tietokantaan suuria määriä binääristä dataa, esimerkiksi kuvia tai videoita. Blobin sijaan kuvien ja videoiden tallentamiseen voi käyttää tekstityyppistä tietotyyppiä ja tallentaa kuvan tai videon urlin eli tiedoston lähdekansion ja tiedoston nimen.

Kentälle valinnaisia määrityksiä ovat *kuvaus*, joka selventää, mitä arvoja kenttään pitää syöttää, *maksimipituus*, joka määrittää, miten monta merkkiä arvoon saa tulla maksimissaan, *syöttörajoite*, jolla voidaan rajata käyttäjän syöttämiä arvoja, *pakollisuus*, joka määrittää sen, onko kentälle pakko syöttää arvo ja *oletusarvo*, joka ehdotetaan ensimmäisenä arvona kentälle. [10]

4.3 Normalisointi

Normalisointi on tärkeä osa tietokannan suunnittelua. Sillä pyritään tehostamaan ja järkeistämään tietokannan rakennetta. Normalisointiin kuuluu viisi normaalimuotoa, joiden avulla pyritään vähentämään tietojen toistamista, mikä aiheuttaa ongelmia tietojen hallinnoimisessa. Normaalimuodoista kolme ensimmäistä ovat tärkeimmät ja oleellisimmat.

Ensimmäinen normaalimuoto sallii jokaisen sarakkeen esittävän vain yhden arvon, eli listat ja tietotyypit sekaisin samassa kentässä pyritään poistamaan. Tietokanta muokataan toimimaan niin, ettei useampia arvoja pääse syntymään samaan kenttään.

Toisessa normaalimuodossa muokataan taulun sarakkeet niin, että taulussa ovat vain ne sarakkeet, jotka liittyvät taulun perusavaimeen. Tiedon toistuminen vältetään jakamalla taulu useammaksi tauluksi.

Kolmas normaalimuoto käsittelee riippuvuuksia. Riippuvuuksia poistamalla helpotetaan tiedon hallintaa. Jos kaksi kenttää ovat riippuvaisia toisistaan, muodostetaan näille oma taulu ja viitataan edellisestä taulusta riippuvuuksille muodostettuun tauluun. [2]

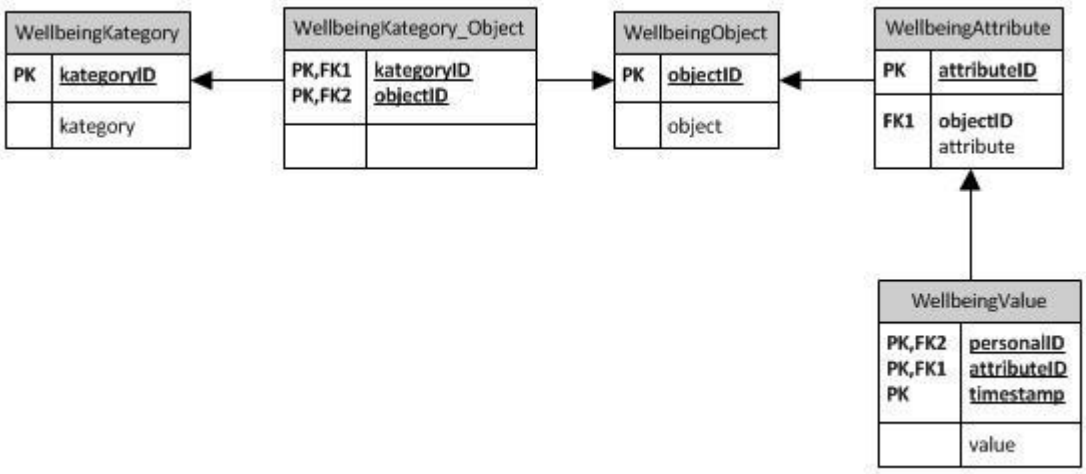
4.4 EAV-malli

EAV-malli (Entity-attribute-value model) sopii osaksi perinteistä relaatiomallia, kun tietokantaan pitää tallentaa paljon tietoa, jota ei ole kaikilla kohteilla. EAV-mallissa pyritään poistamaan mahdolliset *null*-arvot. EAV-mallissa taulut ovat ulkonäöltään pitkiä ja kapeita ja kun tietokannassa on paljon tietoa, tiedon etsintä hidastuu. Mallin ideana on se, että tietokantaan tallennetaan vain ja ainoastaan ne tiedot, joita kohteella on. EAV-mallin taulut ovat pidempään käytännöllisiä, koska taulujen rakenne pysyy samana, ainoastaan tallennettava tieto muuttuu.

EAV-malli on yleinen lääketieteen tietokannoissa, joissa tietoa on paljon, kovakoodattuja kenttiä pyritään välttämään ja attribuutteja pitää pystyä lisäämään tietokannan valmistumisenkin jälkeen ilman tietokannan rakenteen muuttelua.

EAV-mallissa tallennettavalla tiedolla on kolme osaa: kohde (entity), attribuutti (attribute) ja arvo (value). Kohde sisältää kuvauksen, johon attribuutti kuuluu. Attribuutti on se, joka sisältää tietyn arvon, muttei ole ominaisuutena välttämättä kaikilla. Esimerkiksi potilastietokannassa on kohde mittaustulokset, attribuutti on verensokeri ja arvo on 4,7. Mittaustuloksille tehdään EAV-mallin mukaan taulut, koska kaikilla potilailla ei ole tarvetta samoihin mittauksiin. Esimerkiksi verensokeria ei mitata jokaiselta potilaalta, jolloin omana kenttänä potilas-taulussa, kenttään tulisi näiden potilaiden kohdalle null-arvo. [11]

Virtuaalivanhus-tietokannassa pelin pisteet määräytyvät sen mukaan kuinka hyvinvoiva hahmo on. Jokaisella hahmolla on omat tärkeät hyvinvointialueensa ja painopisteensä alueille, joten käytin tässä kohtaa EAV-mallia tietokannan suunnittelussa. EAV-malliin lisättiin vielä Kategoria-taulun (Kategory), johon objektit kuuluvat. Kategoria- ja Objekti-tauluilla muodostui monesta moneen-suhde, joten suhteelle luotiin oma relaatio. Yksi attribuutti voi kuulua vaan yhteen objektiin, joten objektin uniikki id-kenttä liitettiin attribuutti-tauluun vierasavaimeksi. Attribuutille muodostuu arvoja, sen mukaan miten pelaajan toiminnot vaikuttavat attribuutteihin. Tätä varten muodostettiin oma taulu, jossa uniikkina kenttänä on timestamp-tyyppinen kenttä, attribuuttiID ja henkilökohtainenID, jonka kautta selviää pelaajaID ja tasoID. Kuva 16 nähdään EAV-malli piirroksena.



Kuva 16. EAV-malli.

Seuraavaksi tarkastellaan taulurakennetta esimerkkiarvoilla.

Kuva 17 Kategoria-taulussa on yksilöivä id, joka on juokseva kokonaisluku ja kategorian nimi. Hahmon arvot ja pelin pisteet koostuvat neljästä eri hyvinvoinnin osa-alueesta.

	kategoryID	kategory
	1	Psyykkinen
	2	Fyysinen
	3	Sosiaalinen
	4	Hengellinen

Kuva 17. Kategoria-taulu ja testidataa.

Objekti-taulussa on yksilöivä id ja objektin nimi. Kuva 18 on Objekti-taulu, johon on syötetty testidataa.

objectID	object
1	Ravitsemus
2	Lääkintä
3	Lepo
4	Liikunta
5	Kommunikaatio

Kuva 18. Objekti-taulu ja testidataa.

Kategoriaan voi kuulua monta objektia ja yksi objekti voi kuulua yhteen tai useampaan kategoriaan, joten näiden taulujen välille syntyy suhde monesta moneen, jolloin suhteelle tehdään oma relaatio. Kuva 19 uudesta relaatiosta nähdään, että lepo- ja liikunta-objektit kuuluvat psyykkiseen kategoriaan. Kun näille attribuuteille muodostuu pelissä arvoja eli pisteitä, vaikuttavat ne kaikkiin niihin kategorioihin, joihin ne on liitetty.

kategoryID	objectID
1	3
1	4
2	4
2	1
3	5

Kuva 19. Kategoria ja objekti-taulujen M2M-suhteesta muodostunut relaatio.

Kuva 20 Attribuutti-taulussa on yksilöivä id, attribuutin nimi ja vierasavaimena objektiID objektitaulun many-2-one-suhteesta. Attribuutti kuuluu vain yhteen objektiin.

attributeID	objectID	attribute
1	1	Nälkä
2	1	Ruuan tuoreus
3	2	Dosetin tila
4	3	Väsymystila
5	4	Kuntoilu
6	5	Apuvälineet

Kuva 20. Attribuutti-taulu ja testidataa.

Kuva 21 Arvo-taulussa ensimmäinen pelaaja on saanut toiminnoillaan hahmon Nälkä-attribuutin arvoksi 10, Dosetin tila -attribuutin arvoksi 60 ja Väsymystila-attribuutin arvoksi -8. Arvojen tallentuessa tallentuu tietueelle myös timestamp-tyyppinen kenttä, josta näkee, milloin arvo on tullut. Tämä mahdollistaa esimerkiksi sen, että arvoja voidaan verrata alku- ja lopputilanteeseen.

Pelaaja ei saa tietää pelissä objekti- tai attribuuttikohtaisia pisteitä, vaan pisteet näytetään kategoriakohtaisesti. Pisteet määräytyvät kategoriaan kaikkien kategoriaan liittyvien objektien ja niiden attribuuttien arvoista.

personalID	attributeID	timestamp	value
1	1	<Binary data>	10
1	3	<Binary data>	60
1	4	<Binary data>	-8
2	4	<Binary data>	75

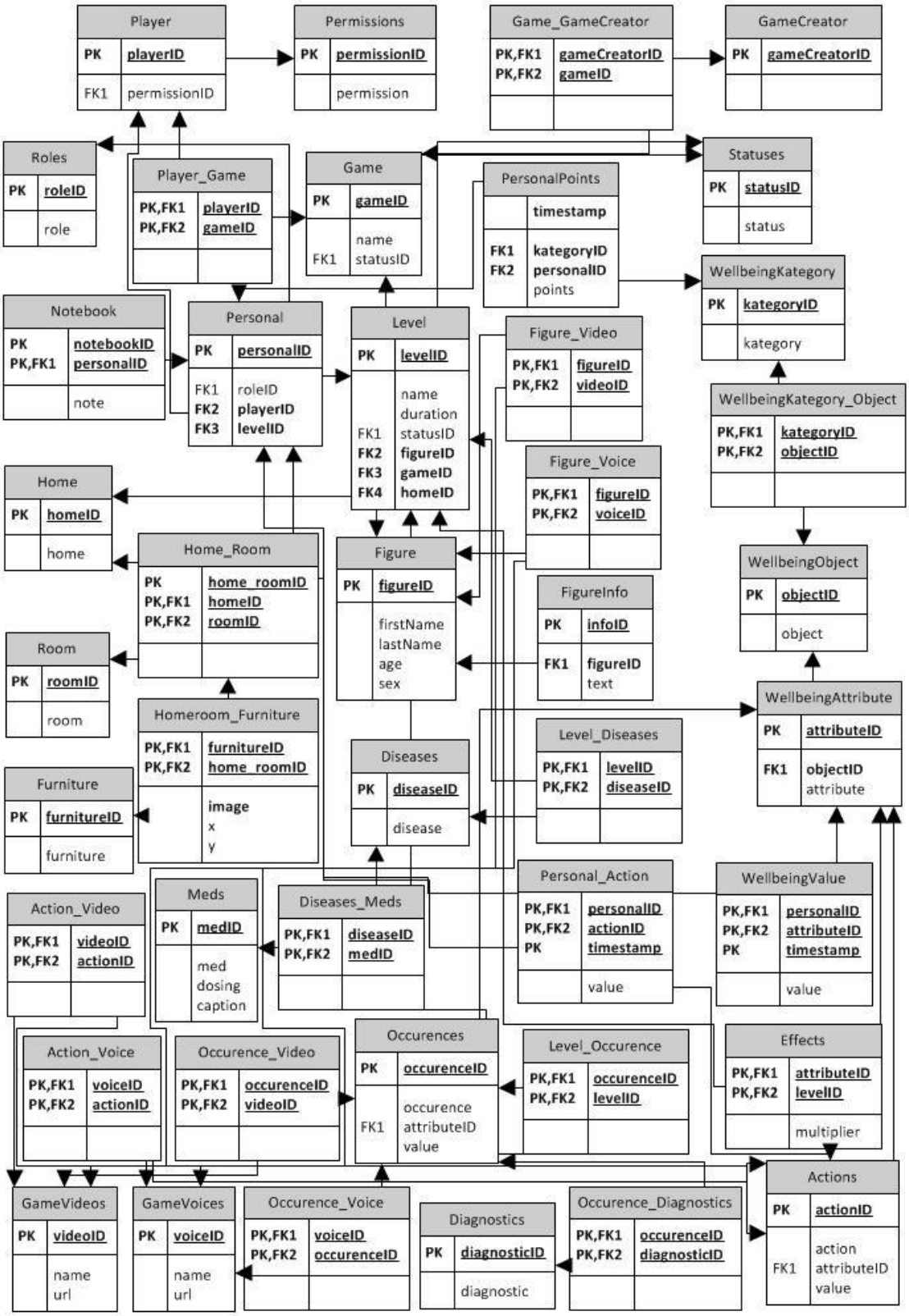
Kuva 21. Arvo-taulu ja testidataa.

5 VIVA-TIETOKANNAN TOTEUTUS

Kun tietokannan suunnittelu on saatu valmiiksi, voidaan toteuttaa tietokanta, eli luoda tietokanta. Mitä tarkemmin ja perusteellisemmin suunnittelu on tehty, sitä helpommin ja nopeammin toteutus sujuu. Tietokannan toteutuksen aikana on vielä viimeinen mahdollisuus ottaa suunnittelussa unohtuneet asiat huomioon, koska jos ohjelmiston toteutuksen edetessä tulee muutoksia tietokantaan, voivat ne vaikeuttaa ja aiheuttaa ohjelmakoodin kirjoittamiseen isojakin muutoksia. Mutta jos määrittely ja suunnittelu on tehty hyvin, ei muutosten määrä ja koko ole yleensä suuri.

Virtuaalivanhus-tietokanta luodaan Microsoft SQL Server 2008 R2-palvelimelle Management Studio-työkalua käyttäen. Kun tietokanta on palvelimella, on sen muokkaus ja kehitys useamman ihmisen tehtävissä. Palvelimesta otetaan myös varmuuskopio, jolloin palvelimen rikkoutuessa on tietokanta ja tiedot vielä tallella.

Tietokannasta tuli laaja, koska määrittelyt vaativat, että kaikki pelin tarvitsema tieto tallennetaan tietokantaan. Alla olevassa kuvassa (Kuva 22) esitellään Virtuaalivanhus-peliin suunniteltu ensimmäinen versio tietokannasta. Tietokantaan muodostui 43 taulua ja yli 50 viiteyhteyttä taulujen välillä.



Kuva 22. Virtuaalivanhus-pelin tietokanta.

Pelaaja-taulussa (Player) on pääavaimena laskurityyppinen pelaajaID (playerID) ja vierasavaimena integer-tyyppinen oikeudetID (permissionID). Taulusta lähtee viiteyhteydet Oikeudet-, Pelaaja_Peli- ja Henkilökohtainen-tauluihin.

Oikeudet-taulussa (Permissions) on pääavaimena laskuri-tyyppinen oikeudetID (permissionID) ja vierasavaimena tekstityyppinen oikeudet-kenttä (permission). Taulusta lähtee viiteyhteys Pelaaja-tauluun.

Pelinluoja-taulussa (GameCreator) on pääavaimena laskurityyppinen pelinluojaID (GameCreatorID). Taulusta lähtee viiteyhteys Peli_Pelinluoja-tauluun.

Peli-taulussa (Game) on pääavaimena laskurityyppinen peliID (GameID), teksti-tyyppinen nimi (name) ja vierasavaimena integer-tyyppinen statusID. Taulusta lähtee viiteyhteydet Pelaaja_Peli-, Taso- ja Status-tauluihin.

Pelaaja_Peli-taulu (Player_Game) on Pelaaja- ja Peli-taulujen yhdistelmätaulu, eli syntynyt uutena relaationa taulujen monesta moneen suhteesta. Taulussa on pääavaimina integer-tyyppiset pelaajaID ja peliID.

Peli_Pelinluoja-taulu (Game_GameCreator) on Peli- ja Pelinluoja -taulujen yhdis-telmätaulu, eli syntynyt uutena relaationa taulujen monesta moneen suhteesta. Taulussa on pääavaimina integer-tyyppiset peliID ja pelinluojaID.

Roolit-taulussa (Roles) on pääavaimena rooliID (RoleID) ja tekstityyppinen rooli (Role). Taulusta lähtee viiteyhteys Henkilökohtainen-tauluun.

Henkilökohtainen-taulussa (Personal) on pääavaimena laskurityyppinen henkilö-kohtainenID (personalID) ja vierasavaimina integer-tyyppiset rooliID (roleID), pe-laajaID (playerID) ja tasoID (levelID). Taulusta lähtee viiteyhteydet Pelaaja_Peli-, Roolit-, Taso-, Koti_Huone-, Henkilökohtainen_Toiminnot-, HyvinvointiArvo- ja Muistivihko-tauluihin.

Muistivihko-taulussa (Notebook) on pääavaimina muistivihkoID (notebookID) ja henkilökohtainenID (personalID) sekä tekstityyppinen muistivihkoID (notebookID). Taulusta lähtee viiteyhteys Henkilökohtainen-tauluun.

Status-taulussa (Status) on pääavaimena laskurityyppinen statusID ja tekstityyppinen status. Taulusta lähtee viiteyhteydet Peli- ja Taso-tauluihin.

HenkilökohtaisetPisteet-taulussa (PersonalPoints) on pääavaimena timestamp-tyyppinen timestamp-kenttä, vierasavaimina integer-tyyppiset kategoriaID (kategoryID) ja henkilökohtainenID (personalID) sekä numeric-tyyppinen pisteet (points). Taulusta lähtee viiteyhteydet Henkilökohtainen- ja HyvinvointiKategoria-tauluihin.

Taso-taulussa (Level) on pääavaimena laskurityyppinen tasoID (levelID), tekstityyppinen nimi (name), integer-tyyppinen kesto (duration) sekä vierasavaimina integer-tyyppiset statusID, hahmoID (figureID), peliID (gameID) ja kotiID (homeID). Taulusta lähtee viiteyhteydet Henkilökohtainen-, Peli-, Status-, Taso_Sairaudet-, Taso_Tapahtumat-, Koti-, Vaikutukset- ja Hahmo-tauluihin.

Koti-taulussa (Home) on pääavaimena laskurityyppinen kotiID (homeID) ja tekstityyppinen koti (home). Taulusta lähtee viiteyhteydet Taso- ja Koti_Huone-tauluihin.

Huone-taulussa (Room) on pääavaimena laskurityyppinen huoneID (roomID) ja tekstityyppinen huone (room). Taulusta lähtee viiteyhteys Koti_Huone-tauluun.

Koti_Huone-taulu (Home_Room) on Koti- ja Huone -taulujen yhdistelmätaulu, eli syntynyt uutena relaationa taulujen monesta moneen suhteesta. Taulussa on pääavaimina laskurityyppinen koti_huoneID sekä integer-tyyppiset kotiID ja huoneID.

Huonekalu-taulussa (Furniture) on pääavain laskurityyppinen huonekaluID (furnitureID) ja tekstityyppinen huonekalu (furniture). Taulusta lähtee viiteyhteys Koti-Huone_Huonekalu-tauluun.

KotiHuone_Huonekalu-taulu (HomeRoom_Furniture) on Huonekalu- ja KotiHuone-taulujen yhdistelmätaulu. Taulussa on pääavaimina integer-tyyppiset huonekaluID (furnitureID) ja kotihuoneID (homeroomID) sekä image-tyyppinen image-kenttä ja numeric-tyyppiset x ja y.

Hahmo-taulussa (Figure) on pääavain hahmoID (figureID), tekstityyppiset etunimi (firstname), sukunimi (lastname) ja sukupuoli (sex) sekä integer-tyyppinen ikä (age). Taulusta lähtee viiteyhteyksiä Taso-, Sairaudet-, Hahmo_Video-, Hahmo_Ääni- ja HahmonTiedot-tauluihin.

HahmonTiedot-taulussa pääavaimena on laskurityyppinen infoID, integer-tyyppinen hahmoID (figureID) ja tekstityyppinen teksti (text). Taulusta lähtee viiteyhteys Hahmo-tauluun.

Sairaudet-taulussa (Diseases) on pääavaimena sairausID (diseaseID) ja tekstityyppinen sairaus (disease). Taulusta lähtee viiteyhteyksiä Taso_Sairaudet- ja Sairaudet_Lääkkeet-tauluihin.

Taso_Sairaudet-taulu (Level_Diseases) on Taso- ja Sairaudet-taulujen yhdistelmätaulu. Taulussa on pääavaimina integer-tyyppiset tasoID (levelID) ja sairausID (diseaseID).

Lääkkeet-taulussa (Meds) on pääavaimena lääkeID (medID) ja tekstityyppiset lääke (med), annostus (dosing) ja seloste (caption). Taulusta lähtee viiteyhteys Sairaudet_Lääkkeet-tauluun.

Sairaudet_Lääkkeet-taulu (Diseases_Meds) on Sairaudet- ja Lääkkeet-taulujen yhdistelmätaulu. Taulussa on pääavaimina integer-tyyppiset sairausID (diseaseID) ja lääkeID (medID).

HyvinvointiKategoria-taulussa (WellbeingKategory) on pääavaimena laskurityyppinen kategoriaID (kategoryID) ja tekstityyppinen kategoria (kategory). Taulusta lähtee viiteyhteyksiä HenkilökohtaisetPisteet- ja HyvinvointiKategoria_Objekti-tauluihin.

HyvinvointiObjekti-taulussa (WellbeingObject) on pääavaimena laskurityyppinen objektiID (objectID) ja tekstityyppinen objekti. Taulusta lähtee viiteyhteyksiä HyvinvointiKategoria_Objekti- ja HyvinvointiAttribuutti-tauluihin.

HyvinvointiKategoria_Objekti-taulu (WellbeingKategory_Object) on Hyvinvointi-Kategoria- ja Objekti-taulujen yhdistelmätaulu. Taulussa on pääavaimina integer-tyyppiset kategoriaID (kategoryID) ja objektiID (objectID).

HyvinvointiAttribuutti-taulussa (WellbeingAttribute) on pääavaimena laskurityyppinen attribuuttiID (attributeID), vierasavaimena integer-tyyppinen objektiID ja tekstityyppinen attribuutti (attribute). Taulusta lähtee viiteyhteyksiä Tapahtumat-, HyvinvointiArvo-, Vaikutukset- ja Toiminnot-tauluihin.

HyvinvointiArvo-taulu (WellbeingValue) on Henkilökohtainen- ja HyvinvointiAttribuutti-taulujen yhdistelmätaulu. Taulussa on pääavaimina integer-tyyppiset henkilökohtainenID (personalID) ja attribuuttiID (attributeID) sekä timestamp-tyyppinen timestamp-kenttä. Taulussa on myös integer-tyyppinen arvo (value).

Vaikutukset-taulussa (Effects) on pääavaimina integer-tyyppiset attribuuttiID (attributeID) ja tasoID (levelID). Taulussa on myös integer-tyyppinen kerroin (multiplier).

Toiminnot-taulussa (Actions) on pääavaimena toimintoID (actionID), tekstityyppinen toiminto (action), vierasavaimena integer-tyyppinen attribuuttiID (AttributeID) ja integer-tyyppinen arvo (value). Taulusta lähtee viiteyhteyksiä Henkilökohtaiset_Toiminnot-, Toiminnot_Video- ja Toiminnot_Äänet-tauluihin.

Henkilökohtainen_Toiminnot-taulu (Personal_Action) on Henkilökohtainen- ja Toiminnot-taulujen yhdistelmätaulu. Taulussa pääavaimina on integer-tyyppiset henkilökohtainenID (personalID) ja toimintoID (actionID) sekä timestamp-tyyppinen timestamp-kenttä. Taulussa on myös integer-tyyppinen arvo (value).

Tapahtumat-taulussa (Occurences) on pääavaimena laskurityyppinen tapahtumaID (occurenceID), tekstityyppinen tapahtuma, vierasavaimena integer-tyyppinen attribuuttiID (attributeID) ja integer-tyyppinen arvo. Taulusta lähtee viiteyhteyksiä Tapahtumat_Video-, Tapahtumat_Äänet-, Taso_Tapahtumat- ja Tapahtumat_Oireet-tauluihin.

Taso_Tapahtumat-taulu (Level_Occurences) on Taso- ja Tapahtumat-taulujen yhdistelmätaulu. Taulussa on pääavaimina integer-tyyppiset tapahtumaID ja tasoID.

Oireet-taulussa (Diagnostics) on pääavaimena oireID (diagnosticsID) ja tekstityyppinen oire (diagnostic). Taulussa on viiteyhteys Tapahtumat_Oireet-tauluun.

Tapahtumat_Oireet-taulu (Occurence_Diagnostics) on Tapahtumat- ja Oireet-taulujen yhdistelmätaulu. Taulussa on pääavaimina integer-tyyppiset tapahtumaID (occurenceID) ja oireID (diagnosticID).

PeliVideot-taulussa (GameVideo) on pääavaimena laskurityyppinen videoID ja tekstityyppiset nimi ja url. PeliÄänet-taulussa (GameVoices) on pääavaimena laskurityyppinen ääniID (voiceID) ja tekstityyppiset nimi ja url.

Hahmo_Video-, Hahmo_Ääni-, Toiminnot_Video-, Toiminnot_Ääni-, Tapahtumat_Video- ja Tapahtumat_Ääni-taulut ovat samanrakenteisia. Kaikki ovat kahden taulun yhdistelmätauluja, toisena pääavaimena joko videoID tai ääniID. Tauluista on viiteyhteydet joko PeliVideo- tai PeliÄänet-tauluihin.

6 TOTEUTUSTEKNIIKOIDEN ESITUTKIMUS VIRTUAALIVANHUS-PELIN KÄYTTÖLIITTYMÄÄN

6.1 Flash

Flash julkaistiin ensimmäisen kerran vuonna 1996 Macromedian toimesta ja on tällä hetkellä Adoben omistuksessa ja kehitettävänä. Ensimmäinen käyttötarkoitus oli luoda käyttäjille helppo alusta tuottaa animaatioita, mutta uusimman version Flash CS5:n myötä tuki on 3D-pelikehitykseen saakka. Flash-sovellukset vaativat toimiakseen Adobe Flash Playerin.

Flash-sovellus rakennetaan luomalla scene ja lisäämällä timelinelle esimerkiksi piirtotyökalun avulla kuvio. Timelinen avulla kuvion voi suurentaa animaationa lisäämällä layerille Keyframen.

Flashissa ääntä voi lisätä soimaan taustalla jatkuvasti, timelinelle tai vaikka painikkeeseen. Tuetut äänimuodot Windowsille ovat: ASND, WAV, mp3, AIFF, Sound Only QuickTime Movies ja Sun AU.

Flash on käytetyin tekniikka pyörittämään videoleikettä www-sivulla. Video Import Wizard mahdollistaa videon tuomisen Flashiin. Flash tukee FLV- ja F4V-videomuotoja. Jos video ei ole oikeassa muodossa, se tarjoaa työkalun oikeaan muotoon koodaamiseen.

CS5 tarjoaa uudistetun väripaletin ja muotoiluvälineet Deco-työkalun avulla. Deco-työkalun avulla animaatioiden tekeminen on nopeaa ja helppoa. Code snippet panel tarjoaa yli 20 valmista koodin pätkää sovellusten kehittämiseen. ActionScript-kieli tarjoaa suuren luokkakirjaston kehittäjälle. Viimeisin versio ActionScript 3.0 on suunniteltu toimimaan ja kattamaan olio-ohjelmoinnin tarpeita.

Flashin uusina ominaisuuksina on integroituminen Adoben muiden ohjelmien kanssa. Se on yhteensopiva esimerkiksi sivuntaittoon tarkoitetun inDesign- ohjelman ja Photoshopkuvankäsittelyohjelman kanssa. Flash-sovellukset ovat yhteensopivia Android- ja iOS-laitteiden kanssa. [12]

6.2 Silverlight

Microsoft Silverlight julkaistiin ensimmäisen kerran vuonna 2008 ja viimeisin versio on Silverlight 4.0. Silverlight käyttää xaml- ja .NET-kieliä ja alkuperäisin valmistettiin videoiden toistoon www-sivuille. Silverlightista on tulossa uusi versio 5.0, jonka betaversio julkaistiin huhtikuussa 2011. Kaikki selaimet tukevat Silverlightin jotakin versiota, pois lukien Opera.

Silverlightissa käytetään datagrid-elementtejä objektien sijoitteluun. Luokkakirjastot tarjoavat luokat ja objektit animaatioiden luomiseen ja uusimmassa betaversiossa on tuki graafiselle 3D-mallintamiselle.

Ääniä ja videoita lisätään Silverlightiin mediaelementin avulla. Mediaelementti tukee esimerkiksi tiedostoja muodossa: RGBA, WMV, H264, MPEG, WAV ja MP3.

Silverlight on yhteensopiva Visual Studion ja Expression Studio-työkalujen kanssa ja sillä on tuki Windows Phone ja Nokia Symbianin selaimille. Visual Studio ja Expression Blend integroituvat yhteen, ja esimerkiksi käyttöliittymän toteuttaminen onnistuu kahdella ohjelmalla; Visual Studiossa ohjelmakoodin kirjoittaminen ja Blendissä käyttöliittymän graafinen puoli. [13]

6.3 HTML5

HTML (Hyper Text Markup Language) on kieli, jolla esitetään sisältöä www-sivuilla. Kieltä kehittävät W3C HTML Working Group ja WHATWG. HTML-kieli on ollut käytössä 1990 vuodesta alkaen ja HTML5 on uusin versio. Kielelle ominaisia piirteitä ovat HTML-tagit, jotka merkitään rivin alkuun aloitustagilla ja loppuun lopetustagilla, esimerkiksi <html> ja </html>. Vuonna 2006 alkoi uusimman version kehitys ja kehitystyö on edelleen käynnissä. HTML5 ei ole vielä standardi ja siksi se ei ole vielä yhteensopiva kaikkien selaimien kanssa. HTML-kieleen lisätään usein CSS- ja JavaScript-kieltä, jotka asettavat esimerkiksi tyylimäärityksiä. HTML5:ssä on edeltäjäänsä parempi virheenkäsittely ja kehitys kielellä pitäisi olla laiteriippumatonta.

HTML5 tarjoaa uusia elementtejä, joiden avulla saadaan videoita ja ääniä osaksi verkkosivua. Videoita, kuvia ja ääniä pystytään ohjaamaan mediatapahtumankäsittelijöillä. Canvas-elementti mahdollistaa grafiikan piirtämisen www-sivuille ja elementillä pystytään kontrolloimaan jokaista pikseliä. Elementillä pystyy piirtämään esimerkiksi 2D- ja 3D-objekteja. Canvaksen sisälle kirjoitetaan skriptiä. HTML5:n uusi mediaelementtejä ovat:

- <audio> (multimedia, äänet ja musiikki)
- <video> (videot ja elokuvaleikkeet)
- <source> (mediaelementtien lähteille)
- <embed> (sulautettu sisältö, esimerkiksi plug-in).

HTML5 tarjoaa standardin näyttää videoita ja toistaa ääntä www-sivuilla. Videoiden toistamiseen ei tarvitse ylimääräisiä liitännäisiä osia, mitkä aiemmin ovat osoittautuneet ongelmallisiksi eri selaimille. Video-elementti tukee Ogg-, MPEG 4- ja WebM-videomuotoja. Audio-elementti tukee Ogg Vorbis-, MP3- ja Wav-äänimuotoja.

Web storage on HTML5:n uusi tapa tallentaa tietoa paikallisesti ja se toimii paremmin kuin cookie, kun tallennettavan tiedon määrä on suuri. Tietoa tallennetaan localStorage- tai sessionStorage-elementteihin. LocalStorage-elementti varastoi tietoa ilman aikarajoitusta ja sessionStorage yhden session ajan.

Draggable-attribuutti on uusi toiminto, jolla voidaan vetää ja pudottaa (drag & drop) elementtejä www-sivuilla. [14] [15]

7 TYÖN TULOKSET

Projekti, johon opinnäytetyöni liittyi, on yhä käynnissä. Opinnäytetyöni kattoi vain tietokannan suunnittelun ja toteutuksen projektin ensimmäisten määritysten osalta. Projekti alkoi useilla määrittelyä koskevilla palavereilla, joista sain kerättyä kaiken tarvittavan tiedon käsiteanalyysin muodostamiseen. Projektin puolelta tuli toivomus EAV-mallin mukaan ottamisesta ja valinnat tietokantapalvelimeen ja tietokannan rakennustyökaluun. Opinnäytetyö ei aiheuttanut kustannuksia.

Koska projekti oli vasta ensimmäisten määrittelyiden aikana alussa eikä ohjelmakoodin kirjoittamista ollut aloitettu, tietokanta tullee muuttumaan. Jos projektin edetessä määrittelyt muuttuvat radikaalimmin, ei tietokantaa pysty käyttämään.

Opinnäytetyön haasteita oli aika, jonka takia työ venyi, sekä suuri tietokanta. Tietokanta ei varmaan ikinä ole valmis, vaan sitä voi aina parannella, pyöritellä toisin ja tehostaa. Tietokantaa muutettiin useaan kertaan ja joka kerta tauluja muodostui lisää.

Relaatiomalliksi muuttaminen alkoi kasvattaa tietokannan kokoa, koska tietokanta oli suunniteltu toimimaan mahdollisimman monipuolisesti, joten monesta moneen -suhteita oli paljon.

Tietokannan vaatimusten määrittelyssä ja suunnittelussa onnistuttiin mielestäni hyvin, koska suunnittelu oli hyvin vaiheistettu. Määrittelypalavereissa mietittiin ja suunniteltiin projektiryhmässä mitä vaatimuksia pelillä on ja mihin pelin tuli pystyä. Näin tietokannan vaatimukset ja tarvittavat kohteet hahmottuivat. Tietokannan suunnittelu tehtiin vaihe vaiheelta, jolloin kohteiden ympärillä oleva alkoi rakentua ja syntyi uusia ideoita tarvittavista kohteista.

Ongelmia oli tietokannan toteutuksessa, koska koko relaatiomallia ei ollut testattu testidatalla. Tietokannan toteutuksessa ilmenneet muutokset kirjattiin ja muutettiin suunnitelmiin. Muutokset eivät olleet isoja ja hoituivat yleensä uudella taululla ja viiteyhteyden vaihtamisella toiseen tauluun.

Ongelmia oli myös Management Studion käytössä. Ohjelma ei toiminut aina vaivattomasti, kun valmiina olevaan tauluun halusi tehdä muutoksia. Taulu piti poistaa ja tehdä uudestaan.

Jatkossa olisi hyvä muistaa, että testidatalla tietokannan testaaminen suunnittelun ja toteutuksen eri osavaiheissa olisi hyvä tehdä. Datan syöttämiseen ja hakemiseen voitaisiin tehdä sovellus, jolla saisi relaatioiden yhteydet ja tallennettavan tiedon testattua.

Virtuaalivanhus-projektissa tämän hetkinen tilanne on se, että määritykset ovat muuttuneet ja tietokantaan tallennettavien tietojen määrä on pienentynyt, joten tietokannan taulujen määrä tulee olemaan pienempi. Analyysi-osio ja hyvinvointialueiden pisteet on jätetty kokonaan pois. Projekti on alkuperäisessä aikataulussaan. Pelin toteutus on aloitettu Unity3D-työkalulla.

Tietokantaa voisi jatkossa kehittää esimerkiksi lisäämällä Tyyppi-taulun, joka määrittäisi minkä tyyppinen video tai äänileike on. Tämä tieto helpottaisi esimerkiksi ohjelmakoodin kirjoituksessa. Huoneiden ja kotien graafista rakentamista ei tietokannassa ole otettu huomioon, vain huonekalujen sijainti huoneissa.

LÄHTEET

1 Heikniemi, J. 2005. *Tietokannat tutuiksi*. [verkkodokumentti] viitattu 21.5.2011. http://www.mbnet.fi/lukusali/juttu.aspx?juttuid=854

2 Lahtonen, T. 2003. *SQL*. Toolkit. Vantaa: Dark Oy.

3 Hernandez, M J. 1997*. Tietokannat – suunnittelu ja toteutus*. Jyväskylä: Gummerus Kirjapaino Oy

4 Vieira, R. 2009. *Professional Microsoft SQL Server 2008 Programming*. Indianapolis, Indiana: Wiley Publishing Inc.

5 W3Schools:n www-sivut. Viitattu 8.6.2011. http://www.w3schools.com

6 MSDN:n www-sivut. Viitattu 8.6.2011. http://msdn.microsoft.com

7 Microsoftin www-sivut. Viitattu 8.6.2011. http://www.microsoft.com

8 Hovi, A., Huotari, J. & Lahdenmäki, T. 2005. *Tietokantojen suunnittelu ja indeksointi*. Jyväskylä: Docendo Finland Oy

9 Huotari, J. 2009. *Taulujen määrittely ja muuttaminen*. Jyväskylän ammattikorkeakoulu. [verkkodokumentti] Viitattu 7.5.2011. http://homes.jamk.fi/~huojo/opetus/IIO10200/SQL-opas_DDL.pdf

10 Keskikiikonen, M. 2001. *AB+ Trainer – Tietokannat Access*. Jyväskylä: Gummerus Kirjapaino Oy.

11 Marenco, L., Nadkarni, P., Skoufos, E., Shepherd, G. & Miller, P. 1999. *Neuronal database integration: the Senselab EAV data model*. [verkkodokumentti] viitattu 21.5.2011. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2232788/

12 Adoben www-sivut. Viitattu 8.6.2011. http://help.adobe.com

13 Silverlightin www-sivut. Viitattu 8.6.2011. http://www.silverlight.net/

14 Kuosmanen, S. 2011. *Korttilajittelusovellus HTML5-tekniikoin*. Mikkelin ammattikorkeakoulu.

15 Whatwg:n www-sivut. Viitattu 8.6.2011. http://wiki.whatwg.org

Opinnäytetyön

Projektisuunnitelma

12.6.2011

VERSIOHISTORIA

Versio	Pvm	Kuvaus
0.1	6.7.2010	Dokumentti luotu, tekstiä lisätty
0.2	27.7.2010	Korjaukset tehty
0.3	15.8.2010	Lisää tekstiä
0.4	16.8.2010	Viimeistelyt: budjetti, 4.3
0.5	19.8.2010	Viittausten lisäys, ulkoasu
0.6	30.4.2011	Muutettu aikataulua
0.7	31.5.2011	Muutettu aikataulua
0.8	6.6.2011	Muutettu aikataulua
0.9	8.6.2011	Muutettu aikataulua
1.0	9.6.2011	Hyväksytty versio
1.1	10.6.2011	Korjattu kuvien ristiviitteet
1.2	10.6.2011	Päivitetty päivämäärät
1.3	10.6.2011	Muutettu aikataulua
1.4	11.6.2011	Tehty korjauksia
1.5	11.6.2011	Päivitetty päivämäärät
1.6	11.6.2011	Päivitetty aikataulu
1.7	12.6.2011	Tehty korjauksia
1.8	12.6.2011	Päivitetty päivämäärät
1.9	12.6.2011	Päivitetty aikataulu
2.0	12.6.2011	Hyväksytty versio

Sisältö

1 Johdanto 4
1.1 Yleiskuvaus 4
1.2 Tuote 4
1.3 Suunnitelman ylläpito 4
1.4 Kirjallisuusluettelo 4
2 Projektin organisointi 6
2.1 Projektin vaiheistus 6
2.2 Projektiryhmä 6
2.3 Sidosryhmien kuvaus 7
3 Projektin ohjaaminen 8
3.1 Tavoite 8
3.2 Oletukset, riippuvuudet, reunaehdot 8
3.3 Riskien hallinta 8
3.4 Seuranta ja ohjaus 10
3.5 Henkilöresurssien käytön suunnittelu 10
4 Tekniikka 11
4.1 Menetelmät ja työkalut 11
4.2 Dokumentointi 11
4.3 Tuotteenhallinta 11
4.4 Laadunvarmistus 11
5 Vaiheet, aikataulut ja budjetti 12
5.1 Projektin osittaminen 12
5.2 Ajan käyttö projektissa 12
5.3 Budjetti ja resurssien allokointi 12
5.4 Aikataulu 13

1 Johdanto

1.1 Yleiskuvaus

Projektisuunnitelmassa kuvataan opinnäytetyön aihe, tekniseen toteutukseen ja aikatauluihin liittyvät asiat.

Johdanto kerrotaan opinnäytetyön aiheesta.

alla käydään läpi projektin vaiheistus ja projektiryhmään kuuluvat henkilöt.

alla tarkastellaan projektin tavoitteita ja projektiin liittyviä riskejä.

allakerrotaan teknisistä ratkaisuista.

alla tarkastellaan projektin aikataulua ja budjettia.

1.2 Tuote

Virtuaalivanhus on Savonia-ammattikorkeakoulun digitaalisen median kehityskeskuksen projekti, jonka tarkoituksena on tukea opiskelua ja innostaa terveydenhoitoalan opiskelijoita vanhustyön opiskeluun. Virtuaalivanhus on opetuspeli, jonka avulla tehostetaan vanhustyön opiskelua. Pelissä pelaaja eli opiskelija menee vanhuksen luo kotikäynnille ja toimii virtuaalihoitaja suorittamalla erilaisia tehtäviä. Opettajan osuus on luoda pelitilanne. Minun osuuteni projektista on tietokannan suunnittelu ja toteutus, josta teen opinnäytetyöni. Opinnäytetyöhön kuuluu myös tutkimus mikä olisi sopivin ohjelmointiympäristö käyttöliittymälle.

1.3 Suunnitelman ylläpito

Suunnitelmaa päivitetään tarvittaessa. Suunnitelman päivitykset kirjataan versiohistoriaan.

1.4 Kirjallisuusluettelo

Internet: www.scholar.google.fi

Internet: http://academics.rmu.edu/faculty/wu/infs4240a/book/appendix-B.pdf

Internet: http://www.microsoft.com/sqlserver/2008/en/us/

Internet: http://msdn.microsoft.com/en-us/library/bb418432(v=SQL.10).aspx

Kirja: Professional Microsoft SQL server 2008 programming, Rob Vieira, 2009

Kirja: Microsoft SQL server 2008 administration, Brian Knight, 2008

Kirja: Microsoft SQL server 2008 administration, Steve Seguis, 2008

2 Projektin organisointi

2.1 Projektin vaiheistus

Projekti vaiheistetaan määrittely-, suunnittelu- ja toteutus-vaiheisiin. Määrittelyvaiheeseen kuuluvat opinnäytetyön projektisuunnitelman, projektin määrittely- ja suunnitteludokumenttien kirjoittaminen. Suunnitteluvaiheeseen kuuluvat esitutkimus Pelastusopiston Yhteistoimintaharjoitus-projektin tietokantaan ja aiheen kirjallisuuteen sekä itse tietokannan suunnittelu. Toteutusvaiheessa tehdään tietokanta valmiiksi.

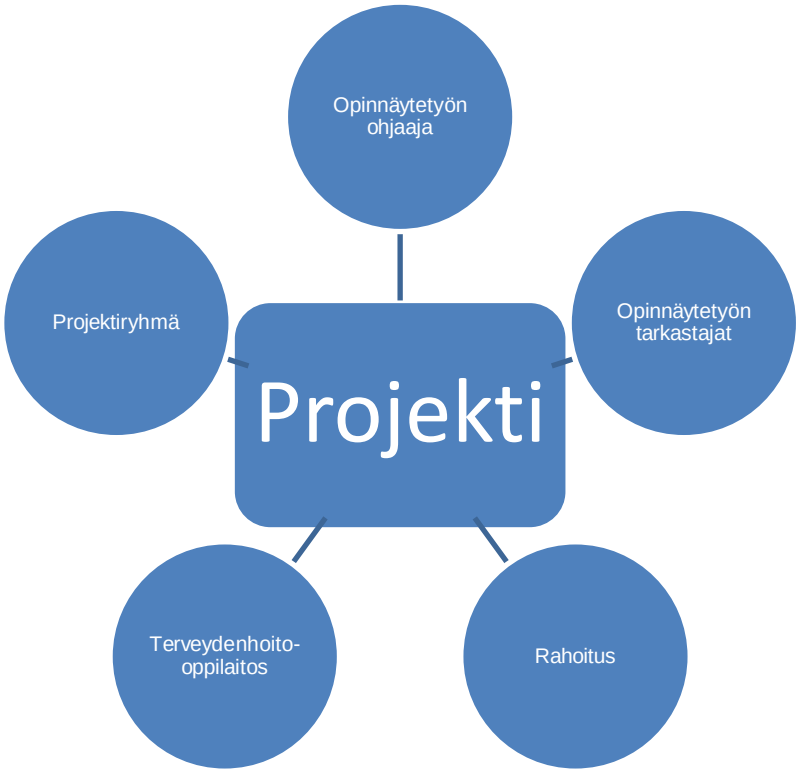
2.2 Projektiryhmä

Projektiryhmän vetäjänä toimii Hannu Karppinen. Käyttöliittymästä vastaa Helena Meriläinen. Mallinnuksesta vastaa Hannu Hoffrén. Koodauksesta ja tietokannasta vastaa Mikko Pääkkönen. Opinnäytetyön ohjaajana toimii Sami Lahti. Alla olevassa taulukossa (Taulukko 1) on projektiryhmän jäsenten yhteystiedot.

Taulukko 1 Projektiryhmän yhteystiedot

Nimi	**Puhelinnumero**	**Sähköposti**
Hannu Karppinen	044 - 7855572	hannu.karppinen@savonia.fi
Helena Meriläinen	044 - 7855598	helena.merilainen@savonia.fi
Hannu Hoffrén	044 - 7855568	hannu.hoffren@savonia.fi
Mikko Pääkkönen	044 - 7856228	mikko.paakkonen@savonia.fi
Sami Lahti	044 - 7856337	sami.lahti@savonia.fi
Annika Immonen	040 - 5568309	annika.immonen@student.savonia.fi

2.3 Sidosryhmien kuvaus



kuvataan projektin sidosryhmät.

Kuva 1 Sidosryhmät

3 Projektin ohjaaminen

3.1 Tavoite

Tavoite on saada tietokanta suunniteltua mahdollisimman monipuoliseksi ja muokattavaksi, jotta muutosten teko myöhemmin olisi vaivatonta ja nopeaa. Virtuaalivanhus-projektin deadline on vuonna 2012, joten tekemäni tietokanta on projektin ensimmäinen versio ja muuttunee myöhemmin projektin edetessä. Tavoitteenani on ottaa kaikki syksyyn 2010 mennessä tulleet asiat huomioon ja tehdä mahdollisimman hyvä ja valmis tietokanta. Tavoitteena on opinnäytetyön valmistuminen syksyllä 2010.

3.2 Oletukset, riippuvuudet, reunaehdot

Projektilla ei ole riippuvuuksia muihin projekteihin.

3.3 Riskien hallinta

Taulukko 2 kuvataan projektin mahdollisia riskejä ja miten ne ennaltaehkäistään.

Taulukko 2 Riskien hallinta

Riski	Vahinko	Todennäköisyys	Priorisointi	Toimenpide
USB-tikku häviää	Tietoa häviää	3	5	Varmuuskopiointi
Aikataulu pettää	Opinnäytetyön valmistuminen viivästyy	3	4	Tehokas työnteko, aikataulun seuraaminen, hyvä määrittely ja suunnittelu
Tekniset ongelmat	Tietoa häviää	2	4	Varmuuskopiointi
Tietokanta on sekava	Tietokannan tulkitseminen on vaikeaa, muun projektiryhmän työ vaikeutuu	3	3	Hyvä suunnittelu

Opinnäytetyö ei mene läpi	Ei tule opintopisteitä, työn joutuu tekemään uudestaan	1	5	Työhön panostaminen
Asiakas ei ole tyytyväinen	Työn joutuu tekemään uudestaan	3	4	Yhteydenpito asiakkaaseen säännöllisin väliajoin

USB-tikku häviää

USB-tikku voi hävitä, missä on kaikki opinnäytetyöhön liittyvät dokumentit ja työt. Varautuen häviämiseen, varmuuskopioidaan tikun sisältö. Tavaroiden säilyttämisessä huolellisuutta lisättävä.

Aikataulu pettää

Opinnäytetyön valmistuminen viivästyy tai seminaaripäivää pitä siirtää. Toimenpiteenä seruataan aikataulua säännöllisesti, tehdään töitä tehokkaasti. Myös hyvä määrittely ja suunnittelu ennen itse työn aloittamista on tärkeää.

Tekniset ongelmat

Teknisiä ongelmia saattaa esiintyä projektin aikana. Ongelmat aiheuttavat aikatauluun viivästymistä ja pahimmillaan tiedon häviämistä. Toimenpiteenä tietojen varmuuskopiointi usein, jotta mahdollisimman vähän tietoa häviäisi ongelman sattuessa.

Tietokanta on sekava

Tietokannasta voi tulla sekava, jolloin se olisi vaikeasti ymmärrettävissä muulle projektiryhmälle ja jatkossa hankalampi jatkokehittää. Toimenpiteenä hyvä tietokannan suunnittelu, sekä yhteistyö yrityksen ohjaajan/asiakkaan kanssa, jotta tietokannasta tulisi vaatimuksia noudattava.

Opinnäytetyö ei mene läpi

Jos opinnäytetyö ei mene läpi, seuraukset on, että opintopisteitä ei tule ja työn joutuu tekemään uudestaan. Toimenpiteenä työhön panostaminen ja keskittyminen.

Asiakas ei ole tyytyväinen

Suunnitelmat muuttuvat usein ja asiakas haluaa tehdä muutoksia kantaan. Toimenpiteenä pidetään asiakkaaseen usein yhteyttä, jotta isoilta muutoksilta vältyttäisiin.

3.4 Seuranta ja ohjaus

Opinnäytetyön ohjaajana toimii Sami Lahti. Ohjaajan kanssa ollaan viikottain yhteydessä opinnäytetyön valmistumisesta. Yrityksen puolelta vastaavana toimii Hannu Karppinen ja ohjaajana tietokannan teossa Mikko Pääkkönen. Aikataulua tarkastellaan viikoittain, jotta aikataulussa pysyttäisiin.

3.5 Henkilöresurssien käytön suunnittelu

Opinnäytetyön ohjaajalle on varattu käytettäväksi 25 tuntia. Minun työni on 15 opintopisteen laajuinen, mikä vastaa 405 tuntia.

4 Tekniikka

4.1 Menetelmät ja työkalut

Tietokannan suunnittelussa käytetään Microsoft Office Visiota, toteutuksessa Microsoft SQL Server 2008:a ja dokumentoinnissa Microsoft Office Wordia.

4.2 Dokumentointi

Määrittelyvaiheessa kirjoitetaan opinnäytetyön projektisuunnitelma sekä projektin määrittely- ja suunnitteludokumentit. Opinnäytetyöpäiväkirjaan kirjataan tehdyt tunnit ja tapahtumat. Palavereista kirjoitetaan pöytäkirja, joka lähetetään opinnäytetyön ohjaajalle.

4.3 Tuotteenhallinta

Jokaisesta dokumentista löytyy versiohallinta. Myös tietokantaan tehdään erillinen versiohallinta. Dokumentteihin lisätään tekstiä vain minun toimesta. Ohjaajat voivat jättää kommentteja dokumentteihin tarvittaessa. Tietokannan muokkausoikeus on vain minulla. Kun tietokanta on valmis, luovutetaan se Digitaalisen median kehityskeskukselle. Tietokantaa tullaan päivittämään ja kehittämään jatkossa.

4.4 Laadunvarmistus

Työ hyväksytetään yrityksellä ennen lopullista valmistumista. Tietokantaa testataan syöttämällä tietoja ja mahdollisesti koodin kanssa.

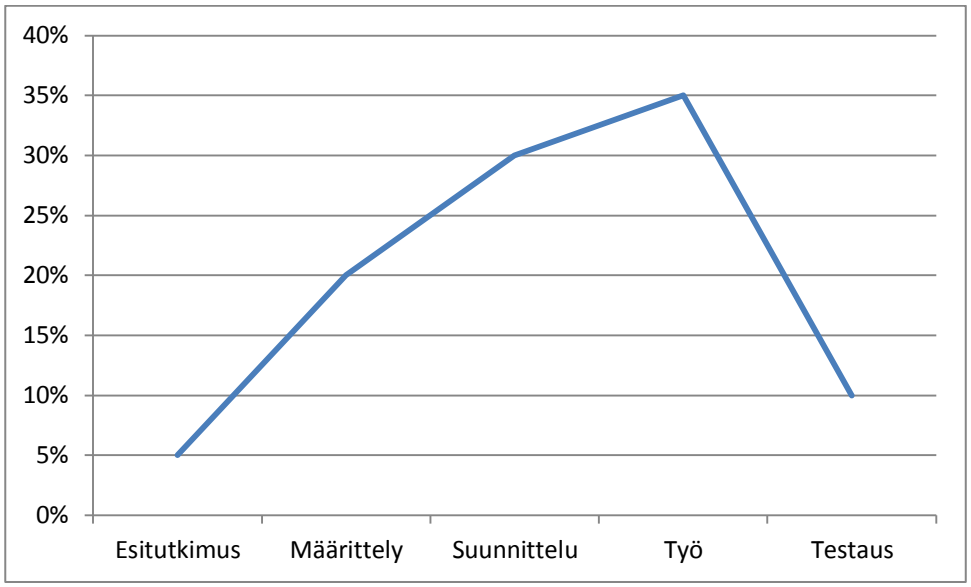
5 Vaiheet, aikataulut ja budjetti

5.1 Projektin osittaminen

Projektin osittaminen on esitetty projektin aikataulussa, joka on esitetty kohdassa 5.4 Aikataulu.

5.2 Ajan käyttö projektissa

Kuva 2 esitetään resurssien tässä tapauksessa ajan käyttö projektin eri vaiheissa. Esitutkimukseen käytetään 5 % koko projektin ajasta. Määrittelyyn menee 20 % ja suunnitteluvaiheeseen 30 %. Itse työn eli tietokannan tekoon käytetään 35 % ja työn testaukseen 10 %.



Kuva 2 Ajan jakautuminen opinnäytetyössä

5.3 Budjetti ja resurssien allokointi

Seuraava taulukko (Taulukko 3) esittelee projektin kulut.

Taulukko 3 Projektin budjetti

Meno		**Hinta**
Ohjelmistokulut	Microsoft SQL 2008 Server + Management Studio	5999 €
	Microsoft Office (Word)	699 €
	Mirosoft Office (Visio)	593 €

Laitteistokulut	Tietokone	800 €
Opinnäytetyöjulkaisu		30 €
Yhteensä		8121 €

5.4 Aikataulu

Opinnäytetyön teko alkoi 22.6. starttipalaverilla. Ensimmäinen työn arvioitu valmistumisaika oli marraskuussa 2010. Opinnäytetyö valmistui kesällä 2011. Visiolla tehty projektin aikataulu on liitteenä. Alla olevassa taulukossa (Taulukko 4) on opinnäytetyön aikataulu.

Taulukko 4 Projektin aikataulu

	Tehtävän nimi	**Alkaa**	**Loppuu**	**Edeltäjät**
1	Projektin starttipalaveri	ti 22.6.2010	ti 22.6.2010	
2	Projektisuunnitelma	ke 23.6.2010	ma 16.8.2010	1
3	Määrittelydokumentti	to 8.7.2010	ma 16.8.2010	1
4	Teorian kirjoitus	ma 26.7.2010	~~to 23.9.2010~~ ke 8.6.2011	
5	Työn suunnittelu	ma 2.8.2010	~~pe 17.9.2010~~ la 30.4.2011	
6	Työn toteutus	~~ma 20.9.2010~~ su 1.5.2011	~~pe 29.10.2010~~ su 29.5.2011	
7	Korjaus	~~ma 1.11.2010~~ su 17.4.2011 ti 7.6.2011	~~to 18.11.2010~~ ke 19.4.2011 ke 8.5.2011	7;5
8	Viimeistely	~~pe 19.11.2010~~ ma 30.5.2011	~~ke 8.12.2010~~ to 9.6.2011 su 12.6.2011	8
9	Seminaari	~~pe 24.9.2010~~ pe 26.11.2010	~~pe 24.9.2010~~ pe 26.11.2010	

Määrittelydokumentti

12.6.2010

Versiohistoria

Versio	Pvm	Muutokset
0.1	19.7.2010	Dokumentti luotu
0.2	12.8.2010	Lisätty tekstiä
0.3	13.8.2010	Lisätty tekstiä ja kaavioita
0.4	15.8.2010	Lisätty tekstiä
0.5	16.8.2010	Lisätty laitteistorajoitteet
0.6	19.8.2010	Korjattu viitteet, ulkoasu
0.7	3.9.2010	Toimintojen kuvat ja tekstit lisätty
0.8	7.9.2010	Muokattu ohjaajan huomautukset
0.9	9.2.2011	Muokattu tekstiä
1.0	9.6.2011	Hyväksytty versio
1.1	10.6.2011	Tehty korjaukset
1.2	10.6.2011	Päivitetty päivämäärät
1.3	10.6.2011	Muokattu ulkoasua
1.4	11.6.2011	Korjattu kuvien ristiviitteitä
1.5	11.6.2011	Päivitetty päivämäärät
1.6	11.6.2011	Korjattu ulkoasua
1.7	12.6.2011	Korjattu ulkoasua
1.8	12.6.2011	Päivitetty päivämäärät
1.9	12.6.2011	Viimeistely
2.0	12.6.2011	Hyväksytty versio

Sisällysluettelo:

1. Johdanto 5
1.1. Tarkoitus 5
1.2. Tuote 5
1.3. Viitteet 5
1.4. Yleiskatsaus dokumenttiin 5
2. Yleiskuvaus 6
2.1. Ympäristö 6
2.2. Toiminta 6
2.3. Käyttäjät 6
2.4. Yleiset rajoitteet 6
2.5. Oletukset ja riippuvuudet 6
3. Tiedot ja tietokanta 7
3.1. Tiedot 7
3.2. Tietokanta 7
4. Pelin kulku 8
4.1. Opettaja 8
4.2. Opiskelija 10
5. Pelin toiminnot 12
5.1. Opettaja 12
5.1.1. Luo peli 12
5.1.2. Tarkastele analyysejä 14
5.2. Opiskelija 14
5.2.1. Pelaa peliä 15
5.2.2. Tarkastele analyysia 16
6. Muut ominaisuudet 17
6.1. Suorituskyky 17
6.2. Käytettävyys, toipuminen, turvallisuus ja suojaukset 17
6.3. Ylläpidettävyys 17
6.4. Siirrettävyys, yhteensopivuus 17

7. Suunnittelurajoitteet 18

7.1. Laitteistorajoitteet 18

7.2. Ohjelmistorajoitteet 18

7.3. Muut rajoitteet 18

1. Johdanto

1.1. Tarkoitus

Määrittelydokumentin tarkoitus on määritellä Virtuaalivanhus-projektiin toteutettava järjestelmä ja tietokanta. Määrittelydokumentissa esitellään ohjelmiston toiminnot, vaatimukset ja rajoitukset. Dokumentti tullaan liittämään kokonaan tai osittain insinöörityöjulkaisuun. Määrittelydokumentin perusteella suunnitellaan tekniset toteutukset.

1.2. Tuote

Virtuaalivanhus on hoitoalan opiskelijoille tarkoitettu opetuspeli. Opetuspelin tarkoituksena on kehittää ja monipuolistaa vanhustyön opetusta. Kehittämisen toivotaan myös lisäävän opiskelijamääriä ja innostavan opiskelijoita valitsemaan vanhustyön suuntautumisekseen. Opetuspelissä opiskelija menee vanhuksen luo, esimerkiksi kotikäynnille, ja hoitaa vanhusta suorittamalla erilaisia tehtäviä. Opiskelija saa valinnoista ja suorittamistaan tehtävistä pelin lopussa palautteen.

1.3. Viitteet

Digitaalisen median kehityskeskuksella on omat dokumentit Virtuaalivanhus-projektille.

1.4. Yleiskatsaus dokumenttiin

Kappaleessa 1 käydään läpi tuotteen ideaa.

Kappaleessa 2 katsastetaan yleiskuvaus peliin.

Kappaleessa 3 kerrotaan tietokannasta.

Kappaleessa 4 kuvaillaan opettajan ja opiskelijan toimintoja sekä pelin kulkua kaavioiden avulla.

Kappaleessa 5 kerrotaan pelin muista ominaisuuksista.

Kappaleessa 6 kerrotaan rajoitteista.

2. Yleiskuvaus

2.1. Ympäristö

Peli tulee toimimaan Internetissä ja/tai SecondLifessa. Peliä voidaan pelata opetuksen yhteydessä tunnilla tai kotona, esimerkiksi kotitehtävänä. Selaimena käy Internet Explorer tai Mozilla Firefox.

2.2. Toiminta

Sovelluksen toiminta on suunniteltu yhteistyössä Terveys- ja hoitoalan ammattilaisten kanssa.

2.3. Käyttäjät

Pelin käyttäjinä tulevat toimimaan opettajat, jotka määrittävät erilaisia pelitilanteita ja opiskelijat, jotka pelaavat peliä. Mahdollisuus myös kahden opiskelijan samanaikaiseen pelaamiseen, jolloin toinen opiskelijoista toimii hoitajana ja toinen vanhuksena.

2.4. Yleiset rajoitteet

Sovellus vaatii toimiakseen Internetin selaamiseen kykenevän tietokoneen. Käyttäjän tunnistaminen tapahtuu koulun käyttäjätunnuksella. SecondLife vaatii oman käyttäjätunnuksen.

2.5. Oletukset ja riippuvuudet

Oletuksena on, että selain on päivitetty ja Flash-lisäosa asennettu. Pelissä pärjääminen edellyttää vanhushoidon ennalta opiskelua.

3. Tiedot ja tietokanta

3.1. Tiedot

Sovellukseen tallennetaan ennen peliä pelitilanteen tiedot, esimerkiksi hahmon ikä, nimi, sukupuoli ja pelin pituus. Pelin aikana sovellus tallentaa tietoja pelaajan valinnoista ja tekemistä toiminnoista. Tietokantaan voidaan tallentaa myös kuvia eri tehtävistä, esimerkiksi haava, tyhjä jääkaappi, silmälasit. Tieto liikkuu Internet-selaimen ja tietokannan välillä.

3.2. Tietokanta

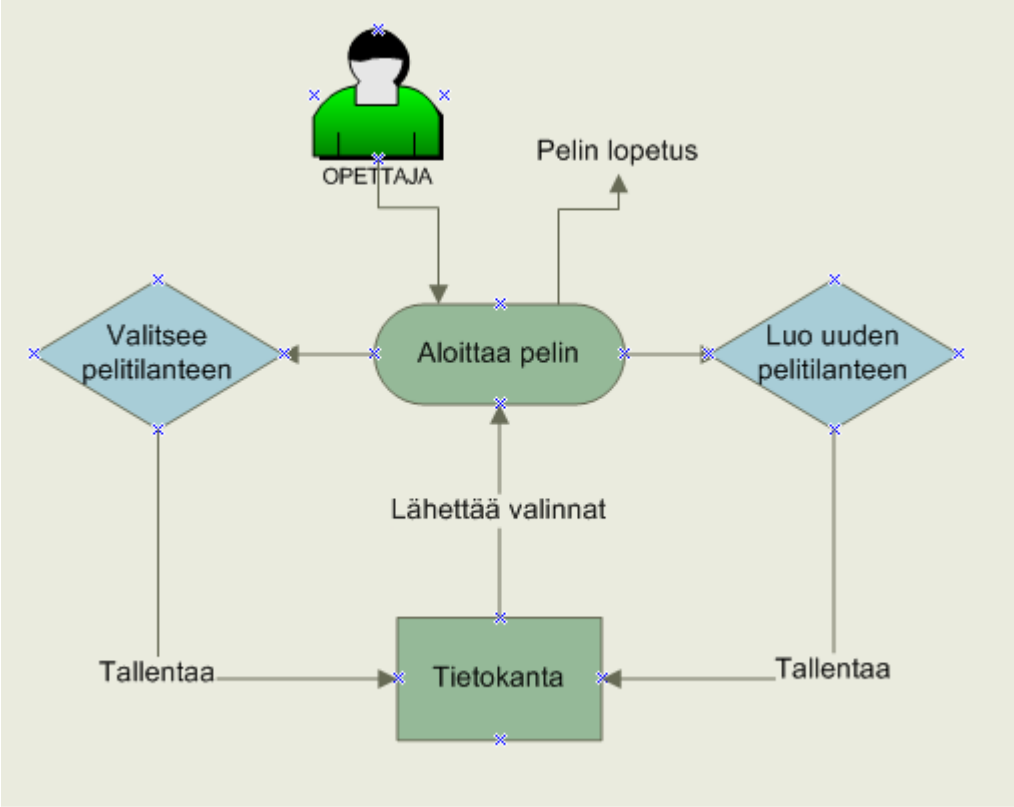
Tietokantapalvelimena käytetään Microsoft SQL Server 2008 R2:sta. Riippuu tietokannan koosta ja käyttäjämääristä, mikä on sopiva tietokantapalvelin. Tietokannassa on paljon tauluja. Käyttäjämääriä ei ole määritelty.

4. Pelin kulku

4.1. Opettaja

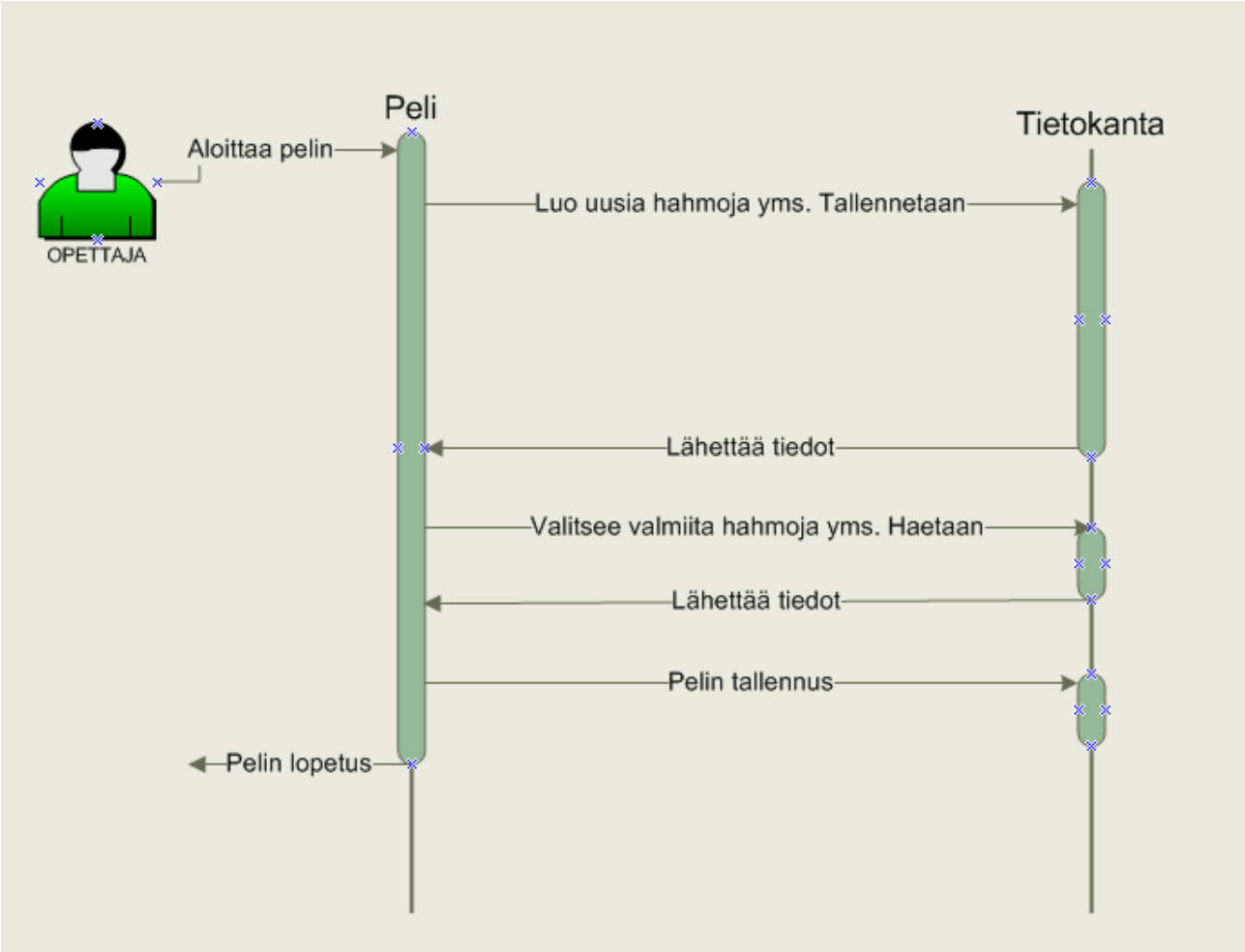
Seuraavissa kuvissa käydään läpi opettajan pelin kulkua.

Kuva 1 opettaja avaa pelin Admin-osion, jossa määritellään pelin asetukset. Opettaja voi valita valmiista hahmoista tai luoda uuden hahmon. Hahmon lisäksi pelin asetuksiin määritetään esimerkiksi hahmon tarina, hahmon ominaisuudet, ympäristö ja pelin kulkuun vaikuttavat asiat. Määrittelyn jälkeen, luodaan uusi pelitilanne ja valinnat tallennetaan tietokantaan.



Kuva 1 Vuokaavio pelin kulusta opettajan roolissa

Kuva 2 näytetään sama toiminto kuvattuna sekvenssikaaviona, missä nähdään hyvin ajan kuluminen. Uusien hahmojen luomiseen menee enemmän aikaa kuin valmiiden valitsemiseen.

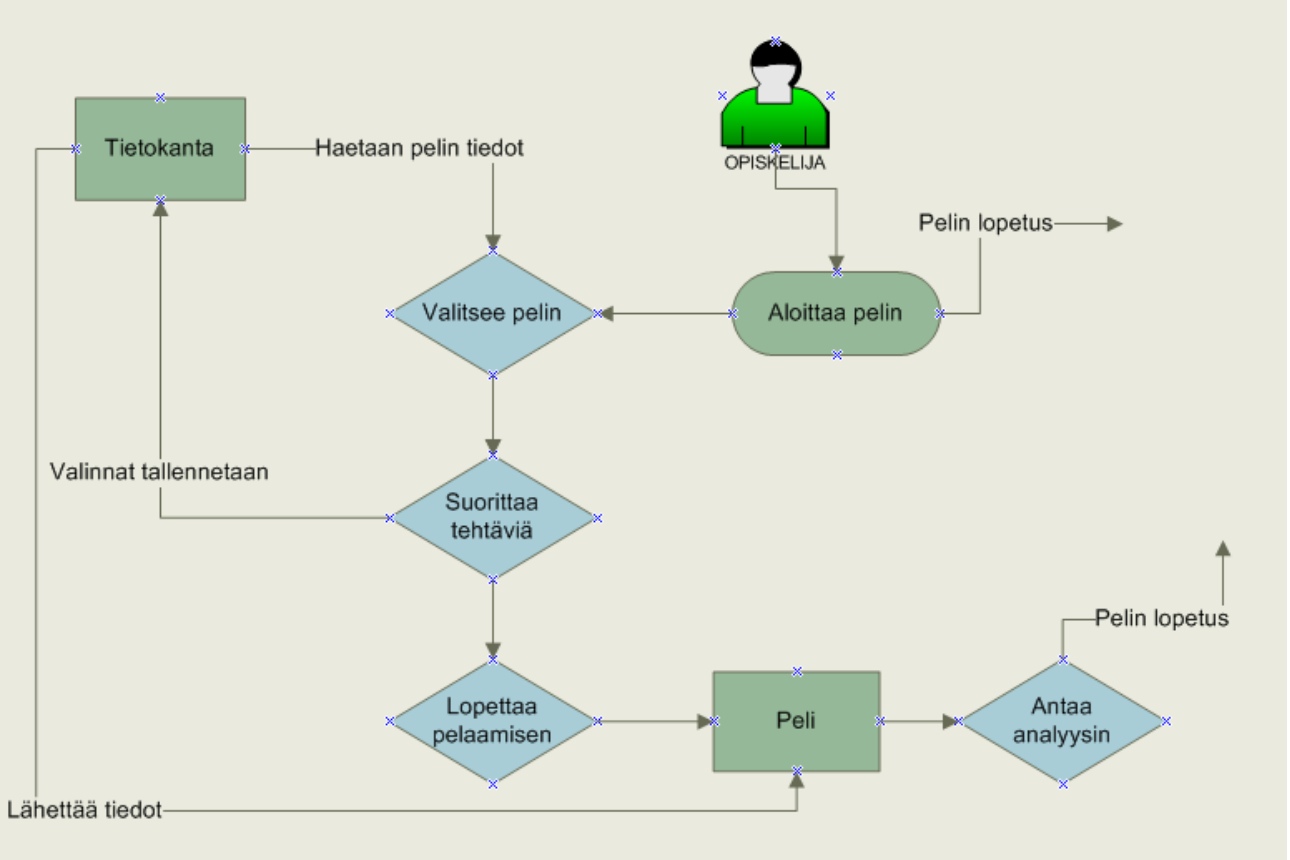


Kuva 2 Sekvenssikaavio opettajan roolista

4.2. Opiskelija

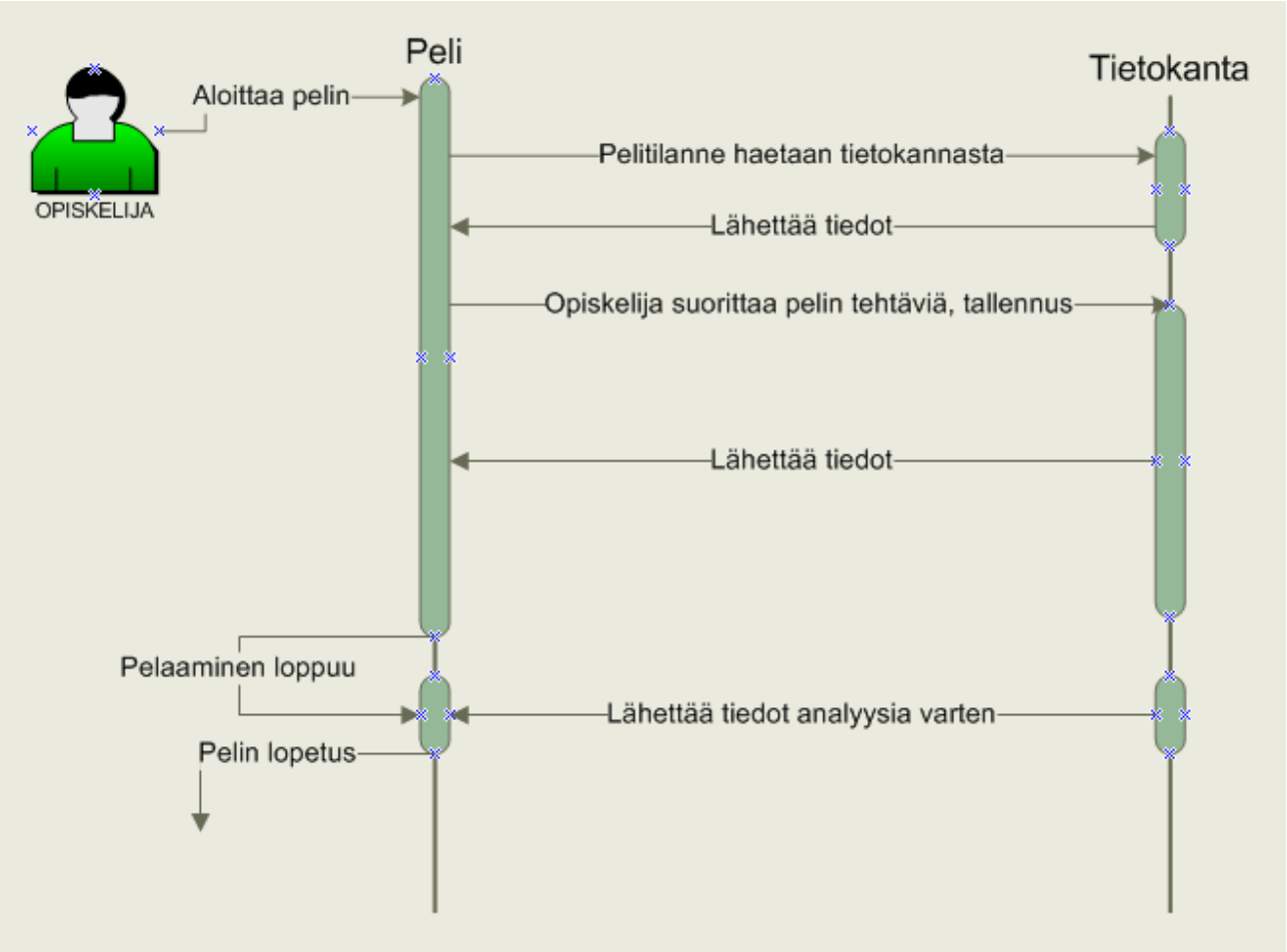
Seuraavissa kaavioissa käydään läpi opiskelijan pelin kulkua.

Kuva 3 opiskelija avaa pelin, jolloin tietokannasta haetaan peliin opettajan määrittelemät tiedot. Opiskelija menee esimerkiksi kotikäynnille hahmon luo ja suorittaa ennalta määritettyjä tehtäviä tai pelaa peliä vapaasti. Suoritetuista tehtävistä ja toiminnoista tallennetaan tiedot tietokantaan. Kun pelaaja lopettaa pelaamisen, tietokannasta haetaan tiedot tehtävistä ja toiminnoista. Peli antaa palautteen miten hyvin pelaaja suoriutui.



Kuva 3 Vuokaavio pelin kulusta opiskelijan roolissa

Kuva 4 näytetään sama toiminto kuvattuna sekvenssikaaviona.



Kuva 4 Sekvenssikaavio opiskelijan roolista

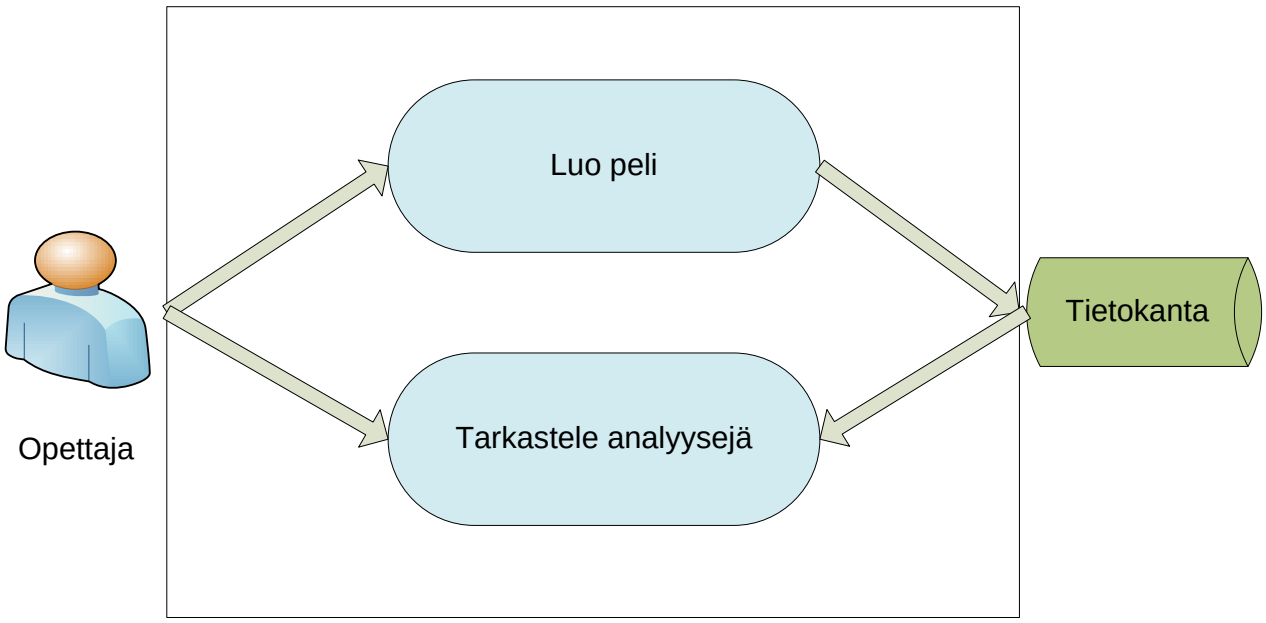
5. Pelin toiminnot

5.1. Opettaja

Seuraavissa kuvissa esitellään karkeasti opettajan toimintoja pelissä. Tarkemmat kuvat ja määritykset tulevat suunnitteludokumenttiin.

Pelin alkutilanteessa opettaja voi luoda pelin, jota opiskelijat pelaavat tai tarkastella analyysejä opiskelijoiden peleistä.

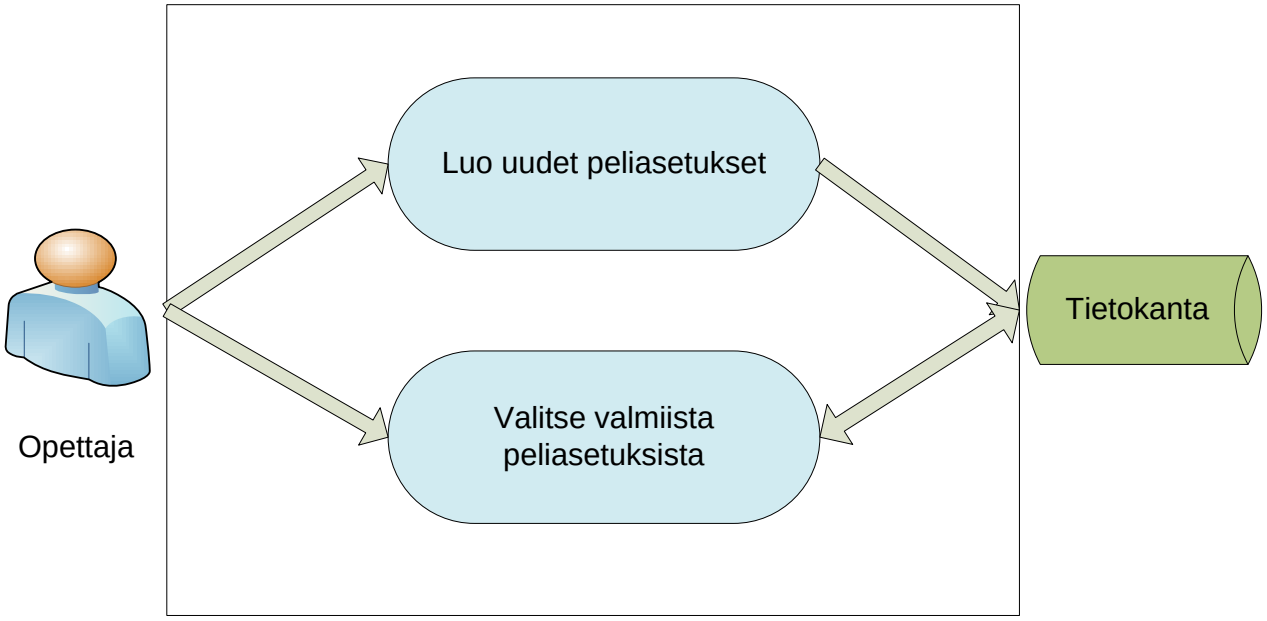
Kuvassa 5 esitellään opettajan toiminnot pelissä.



Kuva 5 Opettajan toiminnot pelissä

5.1.1. Luo peli

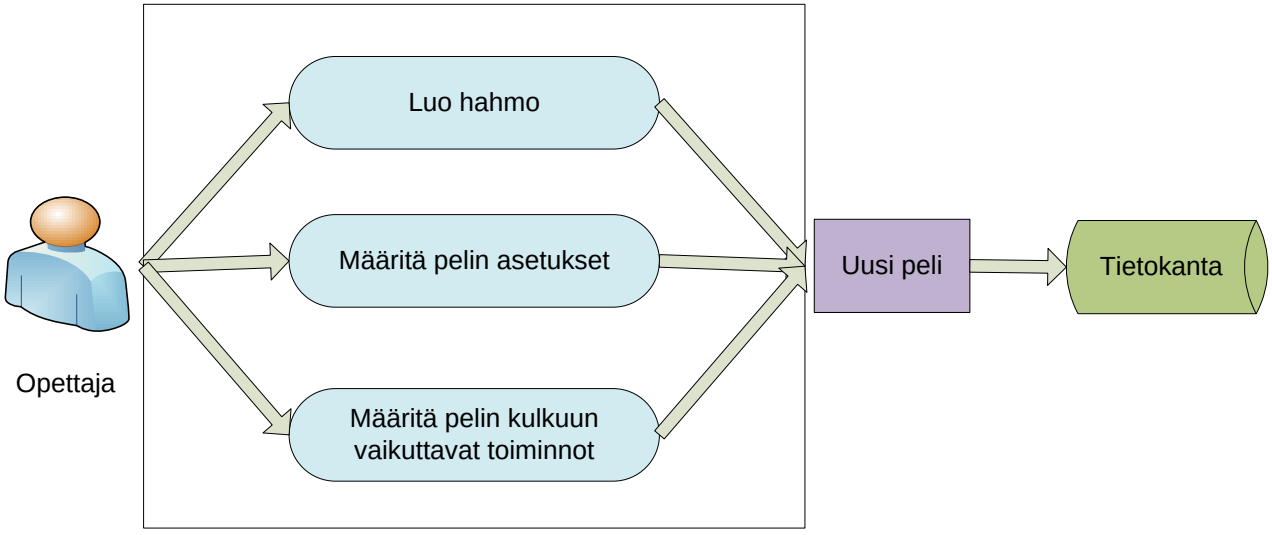
Opettaja voi luoda uuden pelin luomalla uudet peliasetukset tai valitsemalla valmiista peliasetuksista. Kuvassa 6 näytetään opettajan mahdollisuudet uuden pelin luomiseen.



Kuva 6 “Luo peli”-toiminnon valinnat

5.1.1.1. Luo uudet peliasetukset

Opettaja voi luoda uuden pelin uusilla peliasetuksilla, näin pelistä tulee yksilöllinen ja opetuksen tarpeita vastaava. Opettaja määrittelee hahmon, peliasetukset ja pelin kulkuun vaikuttavat toiminnot. Kuvassa 7 tarkastellaan uusien peliasetusten luomista.



Kuva 7 "Luo uudet peliasetukset"-toiminnon valinnat

5.1.1.1.1. Luo hahmo

Hahmon luomiseen tarvitaan esimerkiksi sukupuolen, iän ja kunnon määrittäminen.

5.1.1.1.2. Määritä pelin asetukset

Pelin asetuksiin määritetään esimerkiksi pelin pituus ja pelaavien opiskelijoiden määrä.

5.1.1.1.3. Määritä pelin kulkuun vaikuttavat toiminnot

Pelin kulkuun vaikuttavia toimintoja ovat esimerkiksi hahmon terveystilanne, luonne ja mittarit, jotka mittaavat hahmon hyvinvointia.

5.1.1.2. Valitse valmiista peliasetuksista

Tietokannassa on valmiina muutamia valmiita hahmoja, pelitilanteita ja pelin kulkuun vaikuttavia toimintoja. Opettaja voi luoda pelin valitsemalla valmiita peliasetuksia.

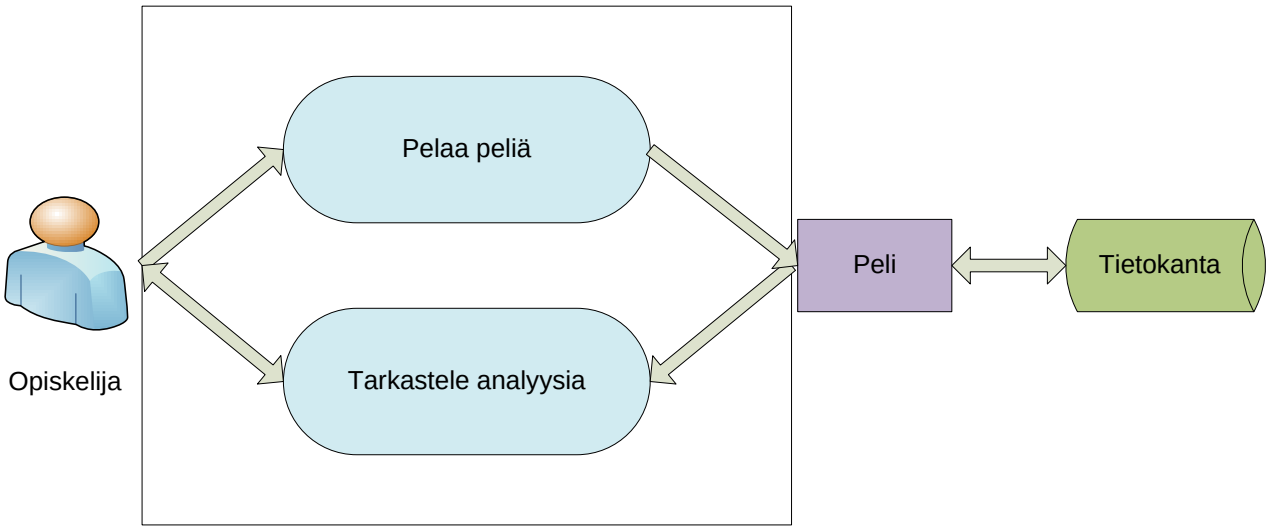
5.1.2. Tarkastele analyysejä

Opettaja voi tarkastella opiskelijoiden saamia analyyseja opettajan luomista peleistä. Analyysi kertoo miten hyvin opiskelija on suoriutunut tehtävistä ja mitä on jäänyt huomioimatta. Analyysi voi antaa koosteen diagrammeilla opettajalle.

5.2. Opiskelija

Seuraavissa kuvissa esitellään karkeasti opiskelijan toimintoja pelissä. Tarkemmat kuvat ja määritykset tulevat suunnitteludokumenttiin.

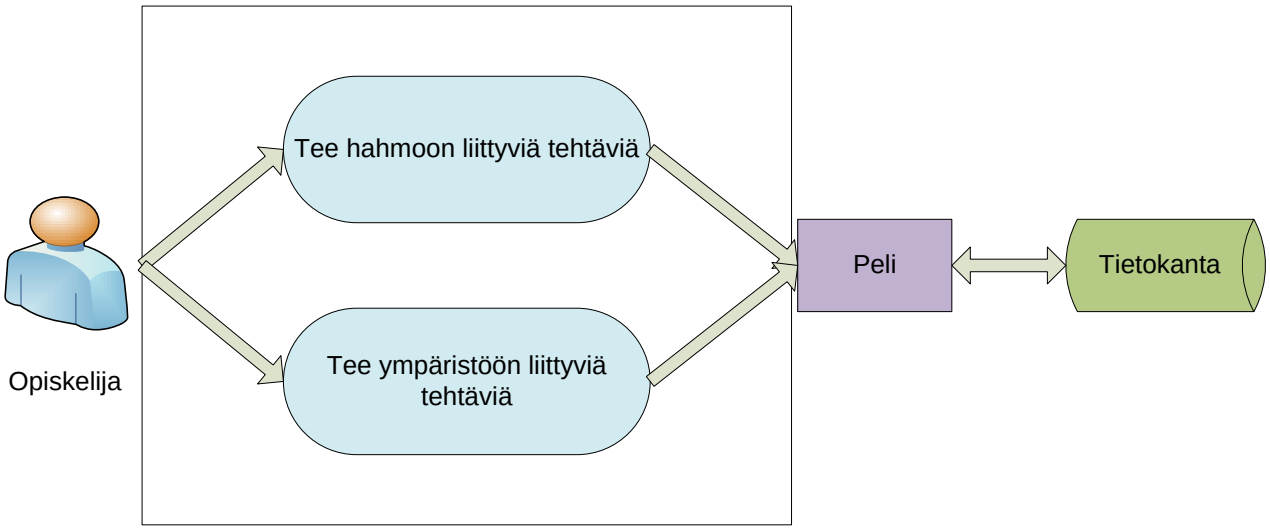
Opiskelijan lähtötilanteessa opiskelija voi pelata peliä tai tarkastella analyysia pelaamistaan peleistä. Mahdollisesti opiskelijalle voisi tulla toiminto myös pelin luomiseen. Alla olevassa kuvassa (kuva 8) nähdään opiskelijan toiminnot pelissä.



Kuva 8 Opiskelijan toiminnot pelissä

5.2.1. Pelaa peliä

Opiskelija voi suorittaa erilaisia tehtäviä pelatessaan peliä. Tehtävät ja toiminnot liittyvät hahmoon ja ympäristöön. Tehtäviin voi tulla vihjeitä tapahtumilla, esimerkiksi jos hahmolla on nälkä, voi ilme muuttua tai hahmo voi kertoa olevansa nälkäinen. Kaikkiin tehtäviin ei vihjeitä tule ja opiskelijan on itse keksittävä hahmon hyvinvointia parantavat toiminnot. Mittarit näyttävät hyvinvoinnin tilan. Hahmolla voi olla esimerkiksi sosiaalisuus-mittari alhaalla, jolloin opiskelijan pitää jutella hahmolle. Kuvassa 9 tarkastellaan opiskelijan tehtäviä pelissä.



Kuva 9 "Pelaa peliä"-toiminnon valinnat

5.2.1.1. Tee hahmoon liittyviä tehtäviä

Opiskelija suorittaa erilaisia hahmoon liittyviä tehtäviä, esimerkiksi verenpaineen mittaaminen, juttelu ja peseminen.

5.2.1.2. Tee ympäristöön liittyviä tehtäviä

Opiskelija suorittaa myös ympäristöön liittyviä tehtäviä, esimerkiksi jääkaapin täyttäminen ja television säätäminen.

5.2.2. Tarkastele analyysia

Opiskelija saa analyysin pelistä pelin pelaamisen jälkeen. Analyysissa näytetään miten opiskelija selviytyi tehtävistä ja mitä jäi huomioimatta. Opiskelija pääsee näkemään myös jälkeenpäin saamiaan analyyseja.

6. Muut ominaisuudet

6.1. Suorituskyky

Suorituskyky on riippuvainen Internet-yhteyden nopeudesta ja tietokantapalvelimen vasteajasta.

6.2. Käytettävyys, toipuminen, turvallisuus ja suojaukset

Käyttäjän tiedot ovat suojattu käyttäjätunnuksella ja salasanalla. Käyttäjän tunnistus tapahtuu Savonia-amk:n omalla tunnistuspalvelulla. Tietokantaan ei tallenneta henkilökohtaisia tietoja käyttäjästä. Pelin tietoja pääsee tarkastelemaan pelin luoja eli opettaja ja pelaaja eli opiskelija, joissakin tapauksissa pelaajia voi olla useampia.

6.3. Ylläpidettävyys

Pelitilanteiden luojat eli opettajat luovat, muokkaavat ja poistavat omia pelitilanteitaan. Sovelluksen ylläpitäjät vastaavat isommista muokkauksista.

6.4. Siirrettävyys, yhteensopivuus

Sovelluksella tulisi olla mahdollisuus SecondLifeen liittämiseen.

7. Suunnittelurajoitteet

7.1. Laitteistorajoitteet

Sovelluksen vähimmäisvaatimukset: Käyttöjärjestelmänä Windows Vista tai uudempi ja Internetyhteys 2Mb/s tai nopeampi.

7.2. Ohjelmistorajoitteet

Vaatii toimiakseen Internet Explorer 7 tai Mozilla Firefox 3 ja uudemmat versiot.

7.3. Muut rajoitteet

Käyttäjä tarvitsee Savonia-amk:n käyttäjätunnuksen ja SecondLife-tunnuksen.