VAASAN YLIOPISTO
TEKNILLINEN TIEDEKUNTA
TIETOTEKNIIKAN LAITOS

Petteri Kaikkonen

TIETOKANTAHAKUJEN TEHOSTAMINEN INDEKSOINNILLA

Case hyvinvointikyselyn tietokantajärjestelmä

Tietotekniikan pro gradu -tutkielma

Multimediajärjestelmien ja teknisen viestinnän koulutusohjelma

VAASA 2008

SISÄLLYSLUETTELO

VAASAN YLIOPISTO
Teknillinen tiedekunta
Tekijä: Petteri Kaikkonen
Tutkielman nimi: Tietokantahakujen tehostaminen indeksoinnilla: Case hyvinvointikyselyn tietokantajärjestelmä
Ohjaajan nimi: Jari Töyli
Tutkinto: Kauppatieteiden maisteri
Laitos: Tietotekniikan laitos
Oppiaine: Tietotekniikka
Koulutusohjelma: Multimediajärjestelmien ja teknisen viestinnän koulutusohjelma
Opintojen aloitusvuosi: 2001
Tutkielman valmistumisvuosi: 2008 Sivumäärä: 84

TIIVISTELMÄ:

Tämä työ käsittelee tietokannan luomisprosessia ja indeksoinnin vaikutusta tietokantahakujen nopeuteen. Työssä käsitellään tietokantajärjestelmän arkkitehtuuria, tietokannan luomisprosessia, sekä indeksointia.

Arkkitehtuuria tutkitaan vertaamalla MySQL tietokannan hallintajärjestelmän vastaavuutta teoriassa esiteltyyn kolmitasoiseen arkkitehtuurikuvaukseen. Hyvinvointikyselyn tietokanta luodaan teoriassa esitettyä prosessikuvausta noudattaen suomalaisia pankkikonttoreita vastaavaksi ja tietokannan tietoja indeksoidaan teoriapohjaisten suositusten perusteella.

Tietokannan luomisprosessia ja indeksointisuositusten paikkansapitävyyttä tutkitaan kvalitatiivisella tutkimusmenetelmällä, jonka tukena käytetään mittaustuloksia. Lopputulosten perusteella prosessinmukaisen etenemisen todetaan olevan luonteeltaan syklistä, ja iteraatiokierrosten määrän olevan riippuvainen suunnittelijan kokemuksesta. Surrogaattiavaimia välttävän tietorakenteiden suunnittelutyylin todetaan parantavan tietokannan automaatti-indeksoinnin riittävyyttä. Mittaustulosten perusteella todetaan tietokantahakujen nopeutuminen indeksien johdosta ja indeksisarakkeiden keskenäisen järjestyksen vaikuttavan hakunopeuksiin.

AVAINSANAT: tietokanta, ER-kaavio, normalisointi, indeksointi

UNIVERSITY OF VAASA
Faculty of technology
Author: Petteri Kaikkonen
Topic of the Master's Thesis: Boosting database queries by means of indexing: Case employee's welfare database system
Instructor: Jari Töyli
Degree: Master of Science in Economics and Business Administration
Department: Department of Computer Science
Major subject: Computer Science
Degree Programme: Degree Programme in Multimedia systems and Technical Communication
Year of Entering the University: 2001
Year of Completing the Master's Thesis: 2008 Pages: 84

ABSTRACT:

The intention of this thesis is to study the process of database creation and to find out how properly designed indexing can impact response times in database queries. Areas covered in this thesis are database architecture, database creation process and indexing.

Chosen database management systems architecture is compared to three-layer description drawn from theories. Theories are found out to be rather adequate to describe modern database management systems, even though the theories themselves are aged.

Employee's welfare database system creation and indexing are done by following a process description joined from literature of multiple authorities. Aim of the database is to represent Finnish bank offices employee structure at sufficient level. Correctness of process description and adequacy of indexing recommendations are studied via qualitative methodology, supported by measurements.

Nature of database creation process is found to be cyclic and the relation between the number of iterations and planner's experience is found to be true. Avoidance of surrogate keys in database planning is found to enhance the quality of automatic indexing. Increased indexes show clear quickening in database query speeds and order between index columns is found to affect query speeds as well.

KEYWORDS: database, ER-diagram, normalization, indexing

1. JOHDANTO

Tämän tutkimuksen taustalla vaikuttavat jo pidemmän aikaa mediassa pyörineet ajatukset työntekijöiden hyvinvoinnista ja työssäjaksamisesta. Projekti sai alkunsa opiskelijavaihdossa tapaamieni ruotsalaisopiskelijoiden kanssa käydyistä keskusteluista, joissa vertailtiin kiinalaisten työpaikkojen oloja Skandinavian vastaaviin. Mietimme keinoja, joilla työympäristön hyvinvointia voitaisiin seurata mahdollisimman tehokkaasti. Kantavana ajatuksena projektin taustalla on se, että seurannan tulisi tapahtua mahdollisimman vaivattomasti ja ilman henkilökunnan suuria ajallisia panostuksia.

Työhyvinvointia seuraava informaatiotyökalu, jonka yhtenä osana tämän tutkimuksen kohteena oleva tietokantajärjestelmä toimii, tarjoaisi johtavissa asemissa oleville henkilöille mahdollisuuden reagoida työympäristön viihtyvyyttä laskeviin tekijöihin heti niiden ilmennyttyä ja tätä kautta tarjoaisi mahdollisuuden pitää yllä työpaikan hyvää ilmapiiriä. Työviihtyvyyden seuranta ei sinällään ole uusi ajatus; yrityksissä on jo vuosikausia seurattu työntekijöiden viihtyvyyttä erilaisin kyselyin. Ongelmana on mielestämme kyselyiden harva taajuus. Jos hyvinvointikysely toteutetaan kerran tai kaksi kertaa vuodessa, niin työnjohto pystyy reagoimaan ongelmiin pahimmillaan vuoden viiveellä. Mikäli johdolle tarjotaan viikoittaista tai kuukausittaista tietoa raporttien muodossa, voidaan ongelmakohtiin puuttua huomattavasti lyhyemmällä viiveellä.

1.1 Tutkimusongelma ja tavoite

Projektiin osallistuvilla henkilöillä on vaihteleva tietämys tietokantajärjestelmän toimintaperiaatteesta ja erittäin pinnallinen tietämys tehokkaan tietokantajärjestelmän luontiprosessista. Tutkimusongelma on siis luonteeltaan kaksijakoinen. Ensimmäiseksi pitää selvittää se, kuinka oikeaoppinen tietokantarakenne luodaan. Toinen tutkittava ongelma on tiettyjen optimointikeinojen, tässä tapauksessa indeksoinnin, hyödyllisyyden selvittäminen hyvinvointikyselytietokannan toiminnassa.

Tutkimuksen tavoitteena on siis luoda tietokantajärjestelmä, joka toimii sekä tiedollisesti oikein, että on toiminnaltaan tehokas. Tavoite on tarkoitus saavuttaa useista lähteistä yhdistellyn teoriapohjan avulla, johon kerätään tietoa tietokantajärjestelmän

arkkitehtuurista, tietokannan suunnitteluprosessista, sekä tietokannan suorituskyvyn tehostamiskeinoista.

1.2 Tutkimuksen rajaus

Tutkimuksessa ei oteta kantaa toiminnan tehostamiseen laitteistoresursseja parantamalla. Tietokannan toiminnan tehostamisen tutkimisessa pyritään optimoimaan tietokantajärjestelmän suorituskyky tietyllä, muuttumattomalla laitteistoasetelmalla. Tehostaminen suoritetaan muokkaamalla tietokantajärjestelmän tietorakenteita ja tiedonhakukeinoja. Tämä rajaus on seurausta siitä, että hyvinvointikyselyjärjestelmän prototyyppi tullaan laajemmassa testausvaiheessa todennäköisesti asentamaan kolmannen osapuolen palvelimelle, joiden laitteistokokoonpanoon on rajalliset vaikutusmahdollisuudet.

Tutkimuksessa ei myöskään oteta kantaa siihen, ovatko henkilöstölle esitetyt kysymykset tarkoituksenmukaisia ja mittaavatko ne oikeita asioita. Suorituskyvyn parantamisessa käytetään ainoastaan kuvitteellista esimerkkiraporttia, jonka tarkoituksena on toimia tietokantahakujen perustana.

1.3 Tutkimusmenetelmä

Oman tutkimukseni kannalta tärkein tutkimusmenetelmä on tapaustutkimus, jota myös yleisesti kutsutaan kvalitatiiviseksi tutkimukseksi. Kvalitatiivisen tutkimuksen piirteisiin kuuluu se, että tutkimuksen kohdetta pyritään tutkimaan mahdollisimman kokonaisvaltaisesti. Täydellistä objektiivisuutta on kuitenkin hankala saavuttaa, sillä tutkijan omat arvolataukset vaikuttavat lopputulokseen. Yleisimpänä tavoitteena kvalitatiivisessa tutkimuksessa onkin löytää uusia tosiasioita tutkimuskohteesta. (Hirsjärvi, Remes & Sajavaara 1997: 152)

Tutkimusta täydentävänä metodina käytän kvantitatiivista tutkimustapaa, jolla tarkastelen indeksoinnin yhteydessä tapahtuvia muutoksia hakuajoissa. Tämä tutkimustapa tarjoaa kvalitatiivista, suhteellista tutkimustapaa tarkempia tietoja mahdollisista muutoksista hakuajoissa. Konkreettiset tulostiedot mahdollisista muutoksista toimivat myös päätöksenteon apuvälineenä, kun tietokannan tietorakenteiden lopullista muotoa päätetään.

1.4 Odotettavissa olevat tulokset

Kirjallisuuskatsauksen perusteella tietokannan luontiprosessia on kehitelty toimivammaksi jo 1970-luvulta asti. Tekniikan ja menetelmien kehittymisen myötä uskon luontiprosessin olevan nykyään jo sellaisella tasolla, että siitä on karsittu valtaosa epäolennaisista toimenpiteistä ja epäloogisista suoritusjärjestyksistä. Indeksointia käsittelevää kirjallisuutta on huomattavasti vähemmän ja aihetta on käsitelty lähinnä teorian näkökulmasta. Tutkimustuloksista on siis mahdollista tehdä päätelmiä, jotka tukevat tai kumoavat teoriassa esitetyt hakuaikoja parantavat vaikutukset. Uskon, että indeksoinnista saatavat mittaustulokset tukevat teoriassa esitettyjä nopeutusvaikutuksia.

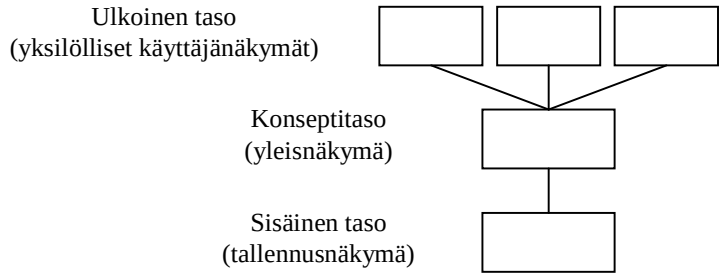
Teoriapohjasta kerättyjen menetelmien pohjalta implementoitavan tietokannan oletan olevan tietorakenteiden toiminnan suhteen mahdollisimman tehokas, sekä käyttäjä- tai laitteistopohjaisille vikatilanteille immuuni. Vikatilanteilla tarkoitan tässä yhteydessä sellaisia tapauksia, joissa tietorakenteet ajautuvat epäloogiseen tilaan virheellisen tietorakennesuunnittelutyön vuoksi.

2. TIETOKANTAJÄRJESTELMÄN ARKKITEHTUURI JA SUUNNITTELUPROSESSI

Tietokantajärjestelmän toiminnan ja arkkitehtuurin ymmärtäminen on olennainen osa toimivan tietokannan luomista. Kun suunnittelijalla on käsitys valitun tietokannan hallintajärjestelmän arkkitehtuurista, eli hallintajärjestelmän sisältämistä komponenteista ja näiden keskinäisistä suhteista, on mahdollista hyödyntää tietokantajärjestelmää tehokkaammin ja useammilla eri tavoilla. Seuraavissa luvuissa on esitelty kirjallisuudessa yleisesti esiintyvä kolmitasoinen arkkitehtuuri, tietokannan hallintajärjestelmän sijoittuminen tässä kolmijaottelussa, sekä tietokannan hallintajärjestelmän sisäinen rakenne. Tietokannan hallintajärjestelmään tulen tulevissa luvuissa viittamaan myös lyhenteellä TKHJ.

2.1 Kolmitasoinen arkkitehtuuri

Tietokannan sisältämä tieto voidaan jakaa näkymiensä perusteella kolmeen päätasoon: ulkoiseen tasoon, konseptitasoon ja sisäiseen tasoon. Sisäisellä tasolla tarkoitetaan käyttöjärjestelmä- ja laitteistotason näkymää tallennetusta tiedosta, eli näkymää tallennusjärjestelmään tallennetuista biteistä ja tavuista. Konseptitasolla puolestaan tarkoitetaan näkymää tietokantaan tallennetusta tiedosta kokonaisuudessaan, eli tietoa tietokannan relaatioista (tauluista), attribuuteista (sarakkeista) ja monikoista (riveistä). Ulkoiseen tasoon luetaan ne tietokannan osa-alueet, jotka näkyvät käyttäjille. Ulkoiset näkymät vaihtelevat käyttäjäoikeuksien mukaan, eli ylläpitäjällä voi olla näkymä kaikkiin relaatioihin ja asiakkaalla/työntekijällä näkymä voi rajoittua vain johonkin tietyn relaation tiettyyn osa-alueeseen. Kolmitasoinen arkkitehtuuri on esitetty kuvassa 1. (Date 1982: 17-18)

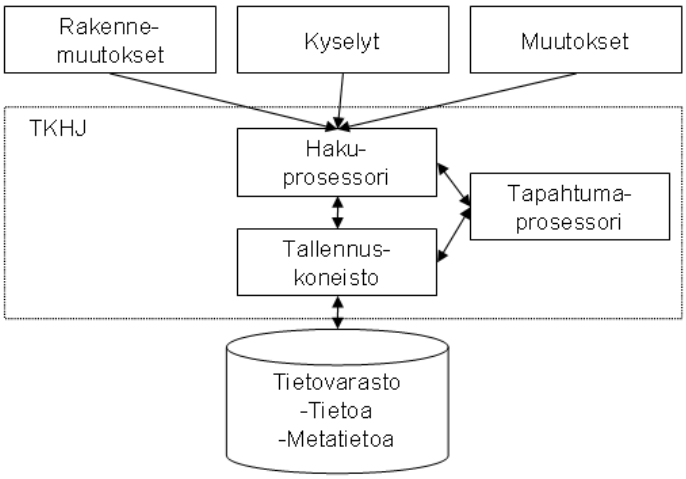


Kuva 1. Arkkitehtuurin kolme tasoa.

Tietokannan hallintajärjestelmän keskeisin toiminta-alue sijaitsee konseptitasolla. Seuraavassa luvussa on käyty läpi TKHJ: sisäistä rakennetta ja toimintaa.

2.2 Tietokannan hallintajärjestelmä

TKHJ toimii linkkinä käyttäjien ja tallennusjärjestelmän välissä halliten muun muassa tietojen indeksointia, tallennusta tallennusjärjestelmässä ja käyttäjille näkyviä näkymiä käyttäjäoikeuksien mukaan (Ricardo 1990: 128-133). TKHJ tarkoittaa siis loogisesti yhteenkuuluvien ja tallennettujen tietojen joukkoa, eli tietokantaa, hallinnoivaa ohjelmistoa. Hallinnointi tapahtuu jollain tietyllä tietokantakielellä, esimerkiksi SQL:llä. Tunnettuja esimerkkejä TKHJ:stä ovat Oracle, DB2, MySQL ja Access. Tietokannan hallintajärjestelmä voidaan edelleen jakaa kolmeen osaan: hakuprosessoriin, tapahtumaprosessoriin ja tallennuskoneistoon. TKHJ:n sisäinen rakenne on esitetty kuvassa 2, joka mukailee Ullman ja Widomin (1997) esittämää TKHJ:n rakennetta. (Ullman ja Widom 1997: 7; Hovi, Huotari & Lahdenmäki 2005: 4)



Kuva 2. TKHJ:n sisäinen rakenne.

2.2.1 Tietovarasto

Tietovarasto on tiedon tallennuspaikka. Tietovarastossa säilytetään tiedon lisäksi myös metatietoa, eli tietoa tiedosta. Tietovaraston metatietoihin lukeutuvat muunmuassa tiedot relaatioiden nimistä, relaatioiden sisältämistä attribuuteista, sekä näiden attribuuttijoukkojen mahdollisista arvoalueista. TKHJ mahdollistaa usein myös indeksien rakentamisen relaatioista. Indeksit ovat osa tallennettua tietoa, ja tiedot siitä, millä arvojoukoilla on indeksi, on osa tietovaraston metatietoa. (Ullman ym. 1997: 7-8)

2.2.2 Tallennuskoneisto

Tallennuskoneisto toimii linkkinä tietovaraston ja hierarkian ylemmiltä tasoilta tulevien pyyntöjen välissä. Yksinkertaisimmillaan tallennuskoneisto tarkoittaa käyttöjärjestelmän tiedostojärjestelmää, jossa säilytetään tietoja siitä, miten tarvittavat tiedot ovat järjestetty tallennusjärjestelmään. Joissain tapauksissa hallintajärjestelmät kontrolloivat levynkäyttöä suoraan ja voivat näin parantaa suorituskykyä tai kiertää tiedostojärjestelmän rajoituksia. Esimerkki tiedostojärjestelmän rajoituksista on Windows-käyttöjärjestelmissä esiintyvä fat32-tiedostojärjestelmä, jossa tiedoston maksimikooksi on rajoitettu 4 gigatavua (NTFS.com 2007). Tallennuskoneisto koostuu kahdesta pääkomponentista: tiedostohallinnasta (file manager) ja puskurimuistista (buffer manager). Tiedostohallinta pitää lukua tiedostojen sijainnista tallennusjärjestelmässä ja puskurimuisti hallinnoi

tallennusjärjestelmästä luettujen levylohkojen pysymistä keskusmuistissa. (Ullman ym. 1997: 9-10)

2.2.3 Hakuprosessori

Hakuprosessori on kyselyjen, tulosteiden ja tietokantaa koskevien manipulaatioiden välittämiseen tarkoitettu komponentti. Sen tehtävänä on välittää käyttäjien ja ohjelmistojen esittämät toimintopyynnöt tallennuskoneistolle. Nimestään huolimatta hakuprosessori ei suorita pelkästään hakuja tietokannasta, vaan se välittää käyttäjien esittämiä toimintopyyntöjä tietokantaan liittyen. Toimintopyynnöt voivat liittyä tietojen hakuun, lisäämiseen tai poistamiseen. Usein hakujen prosessoinnin hankalin osuus on niiden optimointi, eli hakujen saattaminen sellaiseen muotoon, että ne kuormittavat laitteistoa mahdollisimman vähän. Hakuprosessia voidaan tehostaa käyttämällä tietokannan indeksointia, jolloin järjestelmä pystyy aloittamaan tiedonhaun oikeasta paikasta. (Ullman ym. 1997: 10-11)

2.2.4 Tapahtumaprosessori

Tapahtumaprosessori on TKHJ:n komponentti, joka vastaa käyttäjien suorittamien toimintopyyntöjen suoritusjärjestyksestä ja samanaikaisuuden hallinnasta. Sen tehtävänä on varmistaa, että tietokanta pysyy mahdollisimman yhtenäisenä monien käyttäjien muokatessa samaa tietoa samanaikaisesti. Tyypillinen TKHJ antaa käyttäjän niputtaa useita toimintoja yhdeksi tapahtumaksi ja suorittaa tämän tapahtuman kokonaisuutena. (Ullman ym. 1997: 12)

Ullman ym. (1997: 12) määrittelevät tapahtumaprosessorille ja sen suorittamille toiminnoille neljä päävaatimusta, joista käytetään yleisesti englanninkielistä lyhennettä ACID:

1. Atomisuus (Atomicity): Vikatilanteiden välttämiseksi käyttäjän toimintopyyntö on suoritettava kokonaisuudessaan, tai sitä ei suoriteta ollenkaan. Esimerkiksi tilanne, jossa käyttäjä siirtää rahaa verkkopankissa tililtä toiselle. Ainoastaan veloitettavan tilin toiminto kirjautuu tietokantaan, mutta siirretyt varat eivät kirjaudu kohdetilille.
2. Ristiriidattomuus (Consistency): Tietokannan pitää pysyä johdonmukaisessa ja ristiriidattomassa tilassa, vaikka sen sisältämiä tietoja muokkaa useita käyttäjiä

samanaikaisesti. Ristiriidattomuussääntö esimerkiksi elokuvateatterin paikkavarausjärjestelmälle voi olla se, ettei samaa paikkaa myydä kahdelle ihmiselle yhtäaikaisesti.

3. Toimintojen eristäminen (Isolation): Vaikka kaksi samaan tietoon kohdistuvaa toimintopyyntöä tapahtuisikin samanaikaisesti, on niiden vaikutukset pystyttävä eristämään toisistaan. Edellämainitussa elokuvateatterin järjestelmässä tämä tarkoittaa sitä, että ainoastaan toinen samanaikaisista varauksista jää voimaan ja toinen hylätään.
4. Tiedon vikasietoisuus (Durability): Kun toimintopyyntö suoritetaan, se pitää pystyä kirjaamaan tietokantaan myös vikatilanteen (sähkökatko, laitteistorikko) sattuessa.

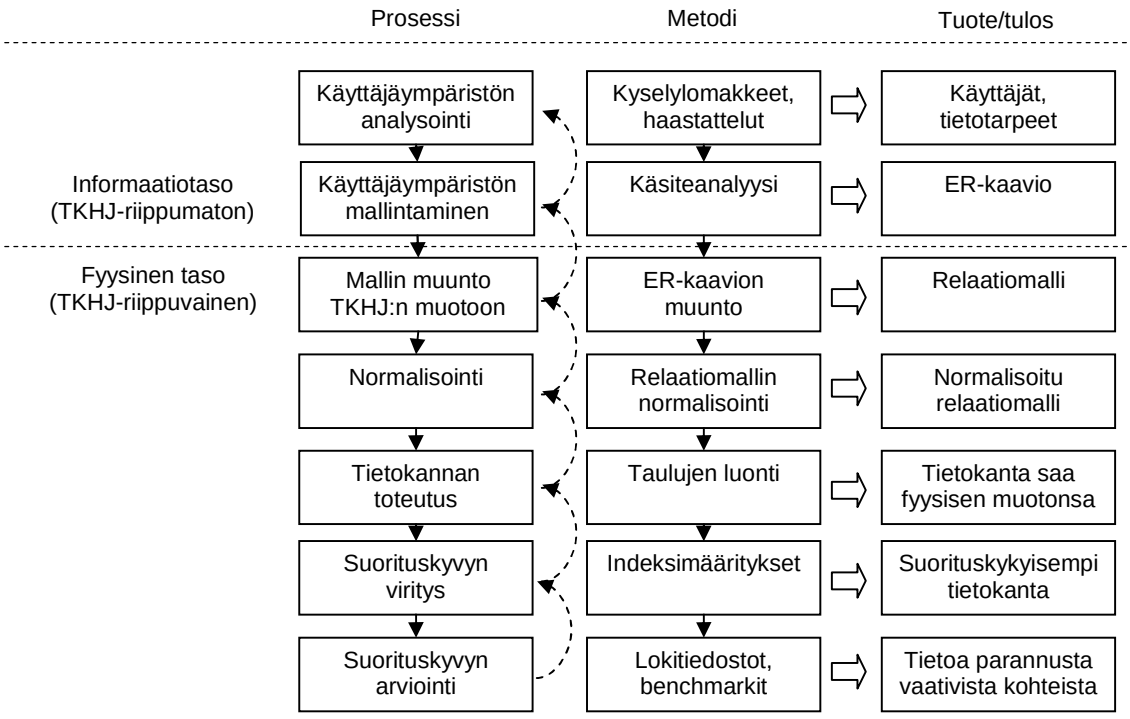
Edellämainitut neljä vaatimusta voidaan täyttää käyttämällä lukitusta, lokitiedostoja ja vikasietoista suoritusjärjestystä tietojen päivityksessä. Lukituksessa tapahtumaprosessori lukitsee ne solut, rivit, sarakkeet tai relaation, johon käyttäjän toimintopyynnössä esittämät muutokset halutaan tehtävän. Kun muutosten kohteena olevat tiedot ovat vain yhden toiminnon kohteena, voidaan välttää tilanteet, jossa kaksi toimintopyyntöä kohdistuu samanaikaisesti samaan tietoon. Vikasietoisuutta voidaan kasvattaa kirjaamalla käyttäjän tekemät toimintopyynnöt ensin järjestelmän lokitiedostoon ja vasta tämän jälkeen itse tietokantaan. Vaikka toimintopyynnön suorittaminen vaatisikin vain väliaikaistaulujen tai väliaikaismuistin käyttöä, niin vikatilanteessa tiedot on saatavilla lokitiedostoista. Samalla periaatteella voidaan myös suorittaa loppuun tietokannan päivitykset, mikäli kesken toiminnon on sattunut jokin vikatilanne. (Ullman ym. 1997: 12-14)

2.3 Tietokannan suunnitteluprosessi

Tietokannan suunnittelu on Pratt ja Adamskin (1987: 231) mukaan kaksivaiheinen prosessi, joka voidaan jakaa informaatiotason suunnitteluun ja fyysisen tason suunnitteluun. Ensimmäisessä vaiheessa kerätään kaikki saatavissa oleva informaatio käyttäjien informaatiotarpeista ja suunnitellaan tietokanta, joka täyttää nämä tarpeet mahdollisimman hyvin. Suunnittelun tämä vaihe on riippumaton siitä, mikä tietokannan hallintajärjestelmä toteutuksessa otetaan käyttöön.

Kummankin vaiheen huolellinen toteuttaminen on lopputuotteena syntyvän tietojärjestelmän toiminnan kannalta kriittistä. Informaatiotasolla täydellisesti suunnitellun tietojärjestelmän suorituskyky voi kaatua huonoon fyysisen tason suunnitteluun, eikä huonoa informaatiotason suunnittelua voi korvata hyvällä järjestelmätason toteutuksella. (Pratt ja Adamski 1987: 231)

Hovi, Huotari ja Lahdenmäki (2005: 24-25) ovat kuvanneet tietokannan suunnitteluprosessia samantyylisellä kaaviolla ja he nimittävät prosessia suunnitteluputkeksi. Prosessi etenee käsiteanalyysin kautta relaatiomalliin ja edelleen tietokannan fyysiseen toteutukseen. Prosessikuvauksessa on otettu huomioon myös relaatiotietokannan toiminnan ja tehokkuuden kannalta kriittiset osa-alueet, eli relaatioiden normalisointi ja indeksointi. Olen kuvannut prosessia kuvalla 3, joka on yhdistetty Pratt ym. (1987: 233), Ricardon (1990: 57), sekä Hovi ym. (2005: 24-25) kuvauksista.



Kuva 3. Yhdistetty kuvaus relaatiotietokannan suunnitteluprosessista.

Prosessia tarkasteltaessa on hyvä tiedostaa sen syklinen luonne. Prosessin edetessä kasvavan tietomäärän perusteella voidaan askelissa palata takaisinpäin tekemään tarvittavia parannuksia ja korjauksia. Tätä loputtoman parantelun sykliä rajoittavat kuitenkin reaalimaailmassa yrityksen kehitystyöhön budjetoima rahamäärä, suunnittelijoiden tietämys aihealueesta ja valittu TKHJ.

Suunnitteluprosessin ”maalina” voidaan pitää tietokantaa, joka täyttää Pratt ym. (1987: 232-233) määrittelemät funktionaaliset tarpeet fyysisten rajoitusten puitteissa. Tällaisia funktionaalisia tarpeita ovat: tulostettavat raportit, tuettavat kyselyt, päivitysprosessit, suoritettavat laskelmat, järjestelmään sisällytettävät rajoitukset, käytössä olevien termien synonyymit, sekä muut ulkoisille järjestelmille lähetettävät tulosteet. Fyysisiä rajoituksia puolestaan ovat: käsitteiden esiintymistiheys, raporttien tulostamistiheys, raporttien rivimäärät, tietoturvavaatimukset, käyttäjäoikeudet, sekä kyselyiden ja päivitysten vasteaikavaatimukset.

2.3.1 Informaatiotason suunnittelu

Informaatiotason suunnittelussa suunnittelijan tavoitteena on kartoittaa ja mallintaa tulevan tietokantajärjestelmän toimintaympäristö ja dokumentoida järjestelmälle asetetut vaatimukset ja rajoitukset. Informaatiotason suunnittelua ei sidota mihinkään tiettyyn laitteisto- tai ohjelmistotason ratkaisuun, joten sen ajatellaan olevan TKHJ:stä riippumaton prosessin osa.

2.3.1.1 Käyttäjäympäristön analysointi

Informaatiotason suunnittelun ensimmäisessä vaiheessa tarkoituksena on käyttäjäympäristön analysointi. Mikäli käytössä on jokin vanha järjestelmä, on syytä analysoida sen toiminta. Kun suunnittelijalla on kattava kuva käytössä olevasta järjestelmästä, hän voi aloittaa yhteistyön nykyisten ja tulevien käyttäjien kanssa heidän tarpeidensa määrittämiseksi. Tämän vaiheen tuloksena suunnittelijalla on mahdollisimman kattava kuva käyttäjien tiedollisista ja funktionaalisista tarpeista. (Ricardo 1990: 57-58; Elmasri ja Navathe 2000: 42)

2.3.1.2 Käyttäjäympäristön mallintaminen

Käyttäjäympäristön mallin perusteella suunnittelija voi luoda tarkan mallin tietokannan loogisesta rakenteesta, jossa kuvataan tietokannan käsitteet, suhteet ja ominaisuudet. Prosessin tässä vaiheessa suunnittelijan täytyy ottaa huomioon kvalitatiivisten seikkojen lisäksi myös se, miten tietokantaa tullaan käyttämään. Suunnitelmaan tulisi siis sisällyttää myös tiedot käyttötarkoituksista ja suunnitterelurajoituksista. Tämän vaiheen tuloksena saadaan tietokannan spesifikaatiot. (Ricardo 1990: 58)

2.3.2 Fyysisen tason suunnittelu

Fyysisen tason suunnittelussa tietokanta saatetaan askeleittain lopulliseen toimivaan muotoonsa. Tason ensimmäinen askel on TKHJ:n valinta. Valinta suoritetaan informaatiotason suunnitelmasta saadun spesifikaation ja suunnitelijan tietojen perusteella.

2.3.2.1 Loogisen mallin muunto TKHJ:n muotoon

Kun TKHJ on valittu, on vuorossa käyttäjäympäristöstä luodun loogisen mallin muunto TKHJ:n muotoon. Tämän tutkimuksen tapauksessa muunto tapahtuu siis ER-kaaviosta relaatiomalliin. Relaatiomallin taulut eivät pysty kuvailemaan reaalimaailman rakenteita kovinkaan hyvin, joten suunnittelu ja mallinnus on syytä tehdä järjestyksessä: käyttäjäympäristön analysointi $\rightarrow$ käyttäjäympäristön mallinnus $\rightarrow$ muunto relaatiomuotoon. Tällä tavoin toteutettuna suunnitteluprosessissa on helpompi palata taaksepäin. (Ricardo 1990: 58; Ullman ym. 1997: 25; Lewis, Bernstein & Kifer 2002: 89-90)

2.3.2.2 Normalisointi

Normalisointivaiheessa relaatiomallista poistetaan tiettyjä sääntöjä noudattaen turha toisteisuus. Tämän vaiheen tarkoituksena on poistaa tiettyjä päivityksiin, poistoihin ja kyselyihin liittyviä ongelmakohtia. Normalisointiprosessin tavoitteena voidaan pitää kolmatta normaalimuotoa, Boyce-Codd –normaalimuotoa tai neljättä normaalimuotoa. (Pratt ym. 1987: 233-234; Elmasri ym. 2000: 484; Hovi ym. 2005: 25).

2.3.2.3 Tietokannan toteutus

Tietokannan toteutuksessa normalisoidut relaatiot saatetaan fyysiseen muotoonsa, eli tauluiksi, palvelimen vaatimalla tavalla. Tällaisia vaatimuksia voivat olla esimerkiksi

laitteisto- ja ohjelmistorajoitukset (Ricardo 1990: 58). Pratt ym. (1987: 233) ovat listanneet fyysisessä suunnittelussa huomioitavia osa-alueita, joihin kuuluvat muun muassa:

- Kaikkien käsitteiden ilmentymämäärät
- Raporttien tulostamistiheys
- Jokaisen raportin rivien määrä
- Vasteaikavaatimukset jokaiselle kyselylle
- Vasteaikavaatimukset päivitystoiminnoille
- Tietoturvarajoitukset ja käyttäjänäkymät

Tietokannan kvalitatiivisen toiminnan, eli sinne tallennettavien ja sieltä saatavien tietojen, oikeellisuus tarkastetaan fyysisen suunnittelun tässä vaiheessa. Arviointia varten voidaan kehittää lopputuotetta kuvaava prototyyppi (esimerkiksi implementoidaan jokin tietty osa tietokannasta), josta saadun palautteen perusteella voidaan havaita tietokannan väärin toimivat osa-alueet ja tehdä niihin parannuksia ennen suorituskyvyn viritystä. Kvalitatiivisen analyysin sivutuotteena saadaan arvokasta tietoa esimerkiksi vasteajoista, jota voidaan puolestaan hyödyntää prosessin seuraavassa vaiheessa. (Ricardo 1990: 58; Hovi ym. 2005: 29)

2.3.2.4 Suorituskyvyn viritys

Tietokannan suorituskykyä voidaan virittää paremmaksi jo ennen implementointia, sekä syklisenä prosessina suorituskyvyn arvioinnin kanssa. Suunnittelija voi arvioida esimerkiksi haluttujen raporttien perusteella mahdollisia tietomäärien liikkeitä ja pyrkiä minimoimaan siirrettävän tiedon määrää koodaamalla tai pakkaamalla. Prosessori- ja levykuormituksen vähentämiseksi tietokannalle luodaan tässä vaiheessa riittävät indeksimääritykset. (Hovi ym. 2005)

2.3.2.5 Suorituskyvyn arviointi

Suorituskyvyn arviointivaiheessa arvioidaan järjestelmän suorituskykyä informaatiosuunnittelun vaiheessa määriteltyjen arviointikriteerien perusteella. Tarkastelussa voidaan siis tarkastella järjestelmän suorituskykyä prosessorikuormituksen, tallennusjärjestelmän tilan, keskusmuistin tai tietoverkkokaistan kuormituksen suhteen. Testaukseen voidaan käyttää ennalta sovittuja testiskenaarioita. Mikäli järjestelmän

toiminnasta saatuun palautteeseen ollaan arvioinnin perusteella tyytyväisiä, suunnittelija saattaa järjestelmän toimintakuntoon. (Ricardo 1990: 58)

3. KÄYTTÄJÄYMPÄRISTÖN ANALYSOINTI JA MALLINNUS

Tietokannan suunnitteluprosessin ensimmäinen kokonaisuus on informaatiotason suunnittelu, joka koostuu käyttäjäympäristön analysoinnista ja mallintamisesta. Olen seuraavissa luvuissa käynyt läpi näiden vaiheiden läpivientiä, sekä esitellyt läpivientiä helpottavia työkaluja.

3.1 Käyttäjäympäristön analysointi

Suunnitteluprosessi aloitetaan käyttäjäympäristön ja tietotarpeiden analysoinnilla, eli määrittelyillä siitä, kuka tarvii mitäkin tietoa ja mitä tietoa tietokantaan tullaan säilömään. Määrittelyihin sisällytetään myös tiedonsiirtovolyymejä ja tilatarpeita koskevia vaatimuksia. Olemassa olevan järjestelmän analysointia helpottavat tiedon liikkeitä kartoittavat tietovuokaaviot, ja tulevaisuutta koskevien käyttäjävaatimusten hahmottamisessa auttavat kyselylomakkeet.

3.1.1 Tietovuokaaviot

Nykyisen järjestelmän analysointia helpottavat tietovuokaaviot (Data Flow Diagram), jossa selvitetään mistä tieto on lähtöisin, missä se käsitellään ja minne se lähetetään prosessoinnin jälkeen. Kaavioon merkitään kaikki henkilöt, laitteet ja ohjelmistot, jotka syöttävät tietoa järjestelmään, tai saavat sitä järjestelmästä. (Ricardo 1990: 59; Elmasri ym. 2000: 42)

Tämän tutkimuksen tarkoituksena ei kuitenkaan ole analysoida mitään olemassa olevaa järjestelmää, vaan luoda täysin uusi tietokanta. En tule hyödyntämään tietovuokaaviota selvittäessäni käyttäjäympäristöä, joten aiheen syvällisempi läpikäyminen ei ole tarpeen.

3.1.2 Kyselylomakkeet

Tietovuokaaviot ovat hyödyllisiä jo olemassa olevan ympäristön analysoinnissa, mutta niillä on hankala kerätä tietoa tulevasta käyttäjäympäristöstä. Nykyistä järjestelmää analysoidessa voi ongelmaksi muodostua myös kaikkien funktionaalisten riippuvuuksien hahmottaminen. Tarvitaankin siis tietoa tulevaisuuden tarpeista, jossa apuna voidaan käyttää kyselylomakkeita. Ollakseen hyödyllinen suunnitteluprosessille, kyselylomakkeen

täytyy sisältää, tai sen perusteella on pystyttävä selvittämään, seuraavat asiat: (Pratt ym. 1987: 265-267)

- Jokaisesta käsitteestä käytettävä termi ja kuvaus. Myös käytössä olevat synonyymit on väärinymmärrysten minimoimiseksi dokumentoitava.
- Jokaisen käsitteen jokainen attribuutti, sekä lyhyt kuvaus attribuutin merkityksestä. Jokainen attribuuttia koskeva ehto tai rajoitus on dokumentoitava (esimerkiksi attribuutti ”Sukunimi” on oltava 3-20 kirjainta, ”Ikä” 1-3 numeroa, jne.). Dokumentointiin tulisi sisällyttää myös attribuuttien määräytymisperusteet, sekä mahdolliset tietoturvamääritykset.
- Kaikki suhteet, suhteisiin osallistuvat käsitteet, sekä suhteen laatu (yksi-yhteen, yksi-moneen tai moni-moneen). Myös kaikki suhteisiin vaikuttavat rajoitukset ja arviot suhteiden kardinaliteetista on syytä selvittää.
- Attribuuttien väliset funktionaaliset riippuvuudet. Selvittämistä helpottavat kysymykset, kuten: ”Jos tiedämme asiakasnumeron, mitä muuta tiedämme?”. Mikäli vastauksena on esimerkiksi asiakkaan nimi, niin nimi on funktionaalisesti riippuvainen asiakasnumerosta.
- Tietoa siitä, minkälaisia päivityksiä, raportteja ja muita toimintoja järjestelmän halutaan suorittavan. On myös hyödyllistä selvittää näiden toimintojen arvioidut tapahtumatiheydet ja volyymit. Hyödyllisiä kysymyksiä tässä vaiheessa ovat esimerkiksi raporttien sisältöön, keskimääräiseen pituuteen, maksimipituuteen ja raportointitiheyteen liittyvät kysymykset.

Kyselyn perusteella suunnittelijalla pitäisi olla kattava kuva seuraavan vaiheen mallinnukseen vaadittavista käsitteistä, suhteista ja ominaisuuksista. Tapahtumatiheuksistä ja volyymeista saatujen vaatimusten avulla myös laitteistolta vaadittu suorituskyky on selvillä ja mahdolliset laitteistohankinnat ja –päivitykset voidaan aloittaa.

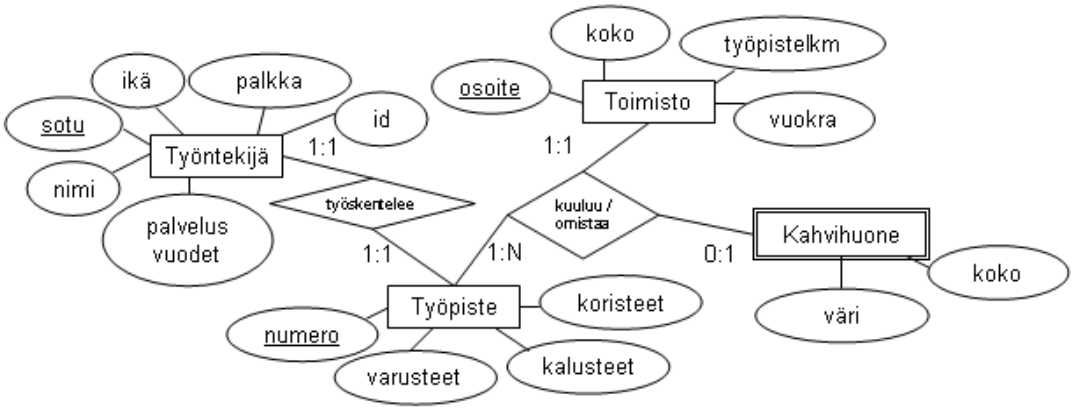
3.2 Käyttäjäympäristön mallinnus

Käyttäjäympäristön mallinnuksessa edellisestä vaiheesta saadut tiedot saatetaan havainnollisempaan muotoon jollain valitulla mallinnustyökalulla. Tällaisia mallinnustyökaluja ovat esimerkiksi ER-kaavio, UML (Unified Modeling Language), ODL

(Object Definition Language), verkostomalli ja hierarkiamalli. Verkosto- ja hierarkiamallit ovat periaatteessa ODL:n rajoittuneita versioita ja olivat käytössä kaupallisissa tietokantajärjestelmissä 70-luvulla, joten niitä voidaan tässä yhteydessä pitää vanhentuneina. UML, ODL ja ER-kaavio ovat lähinnä vaihtoehtoisia ympäristön kuvaustapoja, joten valinta näiden kahden välillä riippuu enemmänkin suunnittelijan mieltymyksistä ja KTHJ:n tyypistä. Tässä tutkimuksessa olen valinnut mallinnusmetodiksi Chen-notaatiota mukailevan ER-kaavion. (Chen 1976)

3.2.1 ER-kaavio

ER-kaavion kehitti ja esitteli Chen vuonna 1976 ja siitä on kehittynyt eri kuvaustapoineen, eli notaatioineen, yksi yleisimmistä ympäristönmallintamistavoista. ER-kaavioon on tavoitteena mallintaa tulevaan järjestelmään liittyvät käsitteet (entities), käsitteiden välillä vaikuttavat suhteet (relations), sekä käsitteisiin liittyvät ominaisuudet (attributes). Käsitteiden ominaisuuksien joukosta voidaan jo suunnittelun tässä vaiheessa poimia jokin käsitejoukon esiintymiä yksilöivä ominaisuus, jota kutsutaan avaimeksi. ER-kaaviossa käsitteitä merkitään suorakaiteilla, suhteita salmiakkikuvioilla ja ominaisuuksia ellipseillä. Avainominaisuuksien teksti on alleviivattu. Kuvassa 4 on esitetty kuvitteellinen mallinnus työntekijän työympäristöstä. (Chen 1976)



Kuva 4. Esimerkki Chenin notaatiolla tehdystä ER-kaaviosta**.**

ER-kaavioon on mahdollista sisällyttää tietoa käsitteiden välisten suhteiden luonteesta ja lukumäärästä, eli kardinaliteetista. Chenin notaatiossa kardinaliteetti, joka siis ilmaisee myös suhteen luonteen, ilmaistaan käsitteen vieressä olevalla numerolla. Chen-notaatiossa kardinaliteetti ilmoitetaan muodossa: ”minimiarvo: maksimiarvo”. Aiemmassa kuvassa 4 kardinaliteetti ilmaisee yhden työntekijän työskentelevän yhdessä työpisteessä ja yhden toimiston omistavan 1-N kappaletta työpisteitä. (Chen 1976, Hovi ym. 2005: 32-78)

ER-kaavioon on mahdollista sisällyttää myös heikkoja käsitteitä, eli käsitteitä, joiden esiintyminen ja tunnistaminen riippuu toisesta käsitteestä. Esimerkkikaaviossa Kahvihuone-käsitteen ilmeneminen riippuu sen sisältävän Toimisto-käsitteen olemassaolosta. Toimistolla ei siis välttämättä ole kahvihuonetta, tai sillä voi olla yksi kahvihuone. Käsitteitä, joiden perusteella heikot käsitteet määräytyvät, kutsutaan vastaavasti vahvoiksi käsitteiksi. (Ricardo 1990: 161-162)

Hyvin luotu ER-kaavio kuvaa ympäristön operaatiot kattavasti ja on joustava uusien informaatiotarpeiden varalta. Kaavioon on siis sisällytetty kaikki käsitteet, suhteet ja ominaisuudet, ja siihen voidaan lisätä niitä tarpeen vaatiessa. Mallin tulisi tukea myös useita näkymiä ja olla riippumaton TKHJ:n käyttämästä kuvauksesta. Ricardon (1990) mukaan mallin tulisi olla riippumaton fyysisestä toteutuksesta, mutta suunnittelijan tulisi ottaa huomioon mahdolliset rajoitukset. (Ricardo 1990: 56-58)

3.3 Relaatioiden suunnittelu

Seuraava vaihe tietokannan suunnitteluprosessissa on ER-kaavion muunto relaatiomuotoon. Relaatiomallin esitteli ensimmäistä kertaa Codd artikkelissaan ”A relational model for large shared databanks” vuonna 1970 (Codd 1970).

Projektin tämä vaihe on erittäin suoraviivainen toimenpide, jossa jokainen ER-kaavion käsite muodostaa relaation (taulun) ja jokainen käsitteen ominaisuus muodostaa attribuutin (sarakkeen) kyseiseen relaatioon. Relaation rakenteen, eli skeeman, määrittelyn jälkeen jokainen kyseisen käsitteen ilmentymä lisätään relaatioon omaksi monikokseen. Mikäli jonkin monikon kohdalla ei tietoa jonkin attribuutin arvosta ole saatavilla, ja mikäli relaation skeema sen sallii, tulee solun arvoksi null. Null on erityinen relaatiomallin merkintä arvolle, jota ei tiedetä. Heikkojen käsitteiden kohdalla muunto tapahtuu samalla

tyylillä, mutta lisäattribuuttien määritys saattaa olla tarpeen rivien identifioimiseksi. Heikosta käsitteestä tehdyn relaation vierasavaimeksi voidaan ottaa esimerkiksi määräävän vahvan relaation avain. Kuva 5 on esimerkki relaatiomuotoon käännetystä ER-kaavion käsitteestä. (Ricardo 1990: 209-210; Lewis ym. 2002: 92)

Työntekijä					
sotu	ikä	nimi	palkka	palvelusvuodet	id
150877-x	30	Antti	2500	3	94857
300466-y	41	Aila	3400	10	94586
070955-z	51	Roy	3750	20	87869

Kuva 5. ER-kaavion käsite “Työntekijä” käännettynä relaatiomuotoon.

Relaatiot voidaan muodostaa myös kahdella muulla tavalla, joita Ricardo (1990: 249-250) kutsuu analyysiksi ja synteesisksi. Analyysissä kaikki ER-kaavion käsitteet ja ominaisuudet kerätään yhdeksi suureksi ”alkurelaatioksi”, joka pilkotaan pienempiin osarelaatioihin normalisointisääntöjen mukaisesti. Synteesissä puolestaan kerätään vain käsitteiden ominaisuudet yhteen, ja niistä muodostetaan tietyn normalisointiasteen relaatioita funktionaalisten riippuvuuksien perusteella.

Pratt ym. (1987: 76) ovat esittäneet relaatiolle, joka on kaksiulotteinen taulu, seuraavat määritelmät:

1. Taulun arvot ovat yksittäisiä
2. Jokaisella attribuutilla (sarakkeella) on yksilöllinen nimi
3. Kaikki arvot tietyssä sarakkeessa ovat saman attribuutin arvoja
4. Sarakkeiden järjestyksellä ei ole väliä
5. Jokainen rivi on yksilöllinen
6. Rivien järjestyksellä ei ole väliä

3.3.1 Funktionaaliset riippuvuudet

Funktionaalisella riippuvuudella tarkoitetaan relaation attribuutin määräytymistä jonkin toisen attribuutin perusteella. Koska suurin osa relaatioiden päivitysongelmista johtuu epätarkoituksenmukaisista funktionaalisista riippuvuuksista, on tärkeätä ymmärtää niiden rakentumisperusteet ja merkitykset. (Pratt ym. 1987: 136)

Relaatiossa R sijaitseva attribuutti Y on funktionaalisesti riippuvainen attribuutista X silloin, kun jokaista X-attribuutin arvoa vastaa vain yksi Y:n arvo (Date 1982: 240; Ricardo 1990: 137). Täydellä funktionaalisella riippuvuudella tarkoitetaan sitä, että Y on riippuvainen ainoastaan koko X:stä. Mikään X:n alijoukko X' ei siis pysty yksistään määrittelemään Y:tä (Date 1981: 241-242; Elmasri ym. 2000: 488). Funktionaaliset riippuvuudet ilmaistaan kaavalla X→Y ja ne ovat olennainen osa relaation avainten määrittelyä ja normalisointiprosessia.

Kuvassa 6 on esitetty esimerkkirelaatio Työntekijä. Esimerkkirelaation attribuutteja ovat sotu, ikä, nimi, palkka, palvelusvuodet ja id. Funktionaalisen riippuvuuden määritelmän mukaan on olemassa vain yksi sotu → id, sillä tiettyä sotu-arvoa voi vastata vain yksi id-arvo. Sama pätee myös ikä, nimi, palkka ja palvelusvuodet –arvoihin, eli sotu → ikä, nimi, palkka, palvelusvuodet, id.

Ikä, nimi, palkka tai palvelusvuodet -attribuutit eivät voi toimia funktionaalisesti määräävinä, sillä useilla henkilöillä voi olla näissä attribuuteissa sama arvo. Attribuutin palvelusvuodet arvo "10" ei siis määrää yksittäistä attribuutin id arvoa, nimi-attribuutin arvo "Roy" ei määrittele yksittäistä sotu-attribuutin arvoa, ja niin edelleen.

Työntekijä					
sotu	ikä	nimi	palkka	palvelusvuodet	id
150877-x	30	Antti	2500	3	94857
131212-r	44	Roy	3400	10	89445
300466-y	41	Aila	3400	10	94586
112484-w	30	Mikko	2500	3	88527
070955-z	51	Roy	3750	20	87869

Kuva 6. Esimerkki relaation funktionaalisista riippuvuuksista.

3.3.2 Avaimet

Relaatiossa esiintyvillä avaimilla tarkoitetaan sellaista rivikohtaista, uniikkia tietoa, jonka perusteella relaation rivit voidaan erottaa toisistaan (Date 1982: 87). Avaimen käsite perustuu funktionaaliseen riippuvuuteen, eli johonkin tiettyyn (avain) attribuuttiin ja muiden attribuuttien funktionaaliseen riippuvuuteen tästä avainattribuutista. Relaatiosta ei välttämättä ole löydettävissä yksittäistä yksilöivää attribuuttia, mutta aina on löydettävissä

attribuuttijoukko, joka täyttää avaimen määritelmän. Väite perustuu luvussa 3.3 esitettyyn relaation määritelmään, jonka mukaan relaatiossa ei saa olla kahta samanlaista riviä.

Avainta voidaan verrata esimerkiksi normaalimaailman sosiaaliturvatunnukseen, jonka perusteella henkilöt voidaan erottaa toisistaan, vaikka heillä olisi esimerkiksi sama nimi tai osoite. Nykyisissä TKHJ:ssä on mahdollista luoda rivitunnisteita myös keinotekoisesti. Näiden surrogaattiarvojen luontiperustana voidaan käyttää kellonaikaa, siemenlukutaulua tai jotain muuta vastaavaa satunnaisluvun generointia. (Hovi ym. 2005: 115)

3.3.2.1 Superavain

Superavain on sellainen attribuutti tai joukko attribuutteja, joka yksilöi käsitteen. Superavain voi olla esimerkiksi oppilaitoksen jokaisen oppilaan yksilöllinen opiskelijanumero. Superavain voi myös sisältää useita attribuutteja ja superavaimen sisältämä attribuutti voi itsessään olla superavain. Tällainen superavain sisältää esimerkiksi attribuutit: opiskelijanumero, pääaine. (Ricardo 1990:151)

3.3.2.2 Kandidaattiavain

Satunnaisesti eteen tulee tilanne, jossa on löydettävissä useita attribuutteja tai attribuuttiyhdistelmiä, jotka aikaansaavat yksilöivän vaikutuksen. Tässä tapauksessa näitä attribuutteja tai attribuuttiyhdistelmiä kutsutaan relaation kandidaattiavaimiksi. (Date 1982: 88)

Ricardon (1990: 151) määritelmän mukaan kandidaattiavain on sellainen superavain, joka ei sisällä enempää arvoja, kuin on yksilöinnin kannalta tarpeellista. Toisin sanoen kandidaattiavain on sellainen superavain, jonka yksikään osajoukko ei itsessään ole superavain. Kaikista kandidaattiavaimista valitaan yksi, joka tämän jälkeen toimii relaation pääavaimena. Kandidaattiavainta, joka ei ole pääavain, kutsutaan vaihtoehtoavaimeksi. (Date 1982: 88; Pratt ym. 1987: 139; Ricardo 1990: 152)

Kandidaattiavaimiin viitataan satunnaisesti termillä sekundääriavain, joka on harhaanjohtavaa. Päinvastoin kuin pääavain, joka siis on valittu kandidaattiavainten joukosta, sekundääriavain ei välttämättä ole yksilöivä relaatiossa R. Yksilöllisyyden puutteen vuoksi sekundääriavainta ei voida käyttää pääavaimena, mutta sitä voidaan

hyödyntää muilla tavoilla relaatiolle suoritettavissa kyselyissä. Joukkoja yksilöiviin attribuutteihin palataan ryvästävän yksitasoindeksin yhteydessä. (Ricardo 1990: 152)

3.3.2.3 Pääavain

Pratt ym. (1987: 138) ovat esittäneet seuraavan määritelmän pääavaimelle:
Attribuutti A (tai attribuuttijoukko) on pääavain relaatiolle R silloin, kun

1. Kaikki attribuutit relaatiossa R ovat funktionaalisesti riippuvaisia A:sta
2. Mikään alijoukko A:ssa (olettaen, että A on attribuuttijoukko) ei omaa kohdan 1 mukaista ominaisuutta

Tämän määritelmän perusteella aiemmin funktionaalisten riippuvuuksien alla esitellyn Työntekijä-relaation pääavaimena ei voida käyttää attribuutteja palkka tai palvelusvuodet, sillä nämä attribuutit eivät yksiselitteisesti määrää työntekijän muita attribuutteja. Yhdistelmäavain {ikä, nimi} voi saada aikaan hetkellisen yksilöivän vaikutuksen, mutta vaikutus häviää, jos yritykseen palkataan kaksi samanikäistä ja –nimistä henkilöä. Luvussa 3.3 esiteltyjen relaation määritelmien perusteella yksilöivä luonne syntyy viimeistään silloin, kun relaation monikon kaikki attribuutit otetaan mukaan avaimeen. Tässä yhteydessä pääavainta on tarkasteltu lähinnä relaation näkökulmasta, kiinnittämättä huomiota siihen, että yleensä monikot edustavat jotain reaalimaailman käsitettä ja ovat jo perusluonteensa takia eroteltavissa toisistaan. Paras vaihtoehto pääavaimelle onkin käyttää varmasti yksilöiviä attribuutteja, kuten sosiaaliturvatunnusta, yksilöllistä työntekijänumeroa tai uniikkeja surrogaattiarvoja. (Date 1982: 87-88; Hovi ym. 2005: 115)

Date (1982: 89-90) on esittänyt seuraavat vaatimukset pääavaimelle:

1. Avaineheys: Yksikään pääavaimen attribuuteista ei voi olla null, eli tuntematon arvo. Tämä sääntö perustuu siihen, että pääavain on se attribuutti, joka yksilöi monikon. Mikäli attribuutilla ei ole arvoa, ei monikkokohtaista yksilöintiä voi tapahtua. Mikäli pääavain on attribuuttijoukko, kaikkien attribuuttien tulisi olla arvoltaan ei-null. Ricardo (1990: 152) yhtyy tähän vaatimukseen.
2. Viite-eheys: Jos relaatiossa R_1 on arvojoukko A, niin A:n kaikkien arvojen täytyy olla joko *null* tai B. B olkoon jonkin rivin pääavain relaatiossa R_2. Mikäli siis jossain relaatiossa A on viitteitä relaatioon B, ei pitäisi olla sallittua poistaa tietoja

relaatiosta B, sillä tällöin relaatioon A syntyy orporivejä (Hovi ym. 2005: 9-10). Mikäli A:n arvo määräytyy B:n mukaan, sitä kutsutaan *vierasavaimeksi*.

3.3.2.4 Pääavaimen määrittäminen suhteen perusteella

Levene ja Loizou (1999: 74) ovat määritelleet pääavaimen valintaperusteet käsitteiden välisten suhteiden perusteella: olkoon relaatiossa R vaikuttavat käsitteet K_1 ja K_2. Kun K_1:n avain on A_1 ja vastaavasti K_2:n avain A_2, valitaan R:n pääavain seuraavin perustein:

- Jos R on moni-moneen, on pääavain $A_1 \cup A_2$.
- Jos R on yksi-moneen, on pääavain A_2.
- Jos R on moni-yhteen, on pääavain A_1.
- Jos R on yksi-yhteen, joko A_1 tai A_2 voi olla pääavain.

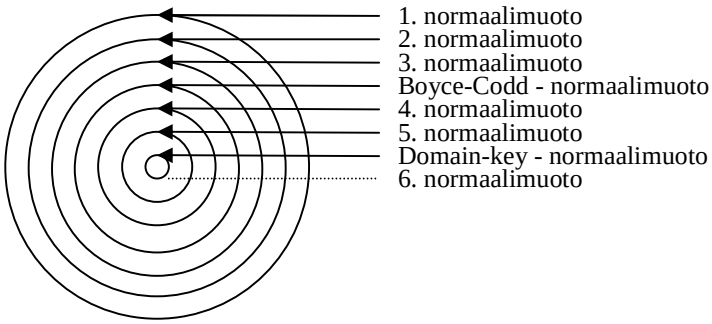
3.4 Normalisointi

Normalisointi on työkalu, jonka avulla pystytään vastaamaan tietokannan suunnittelun peruskysymykseen ”Miten päätetään siitä, mitkä relaatiot ovat tarpeellisia ja mitä attribuutteja näihin relaatioihin sisällytetään?”. Normalisoinnilla pystytään tehokkaasti välttämään tiedon moninkertaista tallentamista eri relaatioihin, joka puolestaan poistaa muun muassa tiedon lisäyksiin, poistamisiin ja päivityksiin liittyviä ongelmia. Edellä mainittujen ongelmien poistuminen tai väheneminen auttaa tietokannan tietojen oikeellisuuden säilyttämisessä. (Date 1982: 237; Elmasri ym. 2000: 484)

Hovi ym. (2005: 86) mukaan normalisointi on menetelmä, jolla tietorakenteita voidaan jalostaa parempaan tallennusmuotoon. Normalisoinnista koituviksi eduiksi he listaavat tietojen toiston vähentymisen, päivitysten tehostumisen, helpomman yhdenmukaisuuden säilyttämisen, sekä muutosjoustavuuden.

Codd esitteli vuonna 1971 normalisoinnin kolme ensimmäistä muotoa tutkimusraportissaan ”*Further Normalization of the Data Base Relational Model*” (Codd 1971; Elmasri ym. 2000: 483). Sittemmin normalisoinnista on esitetty korkeamman asteen muotoja, kuten Boyce-Codd –normaalimuoto, 4. ja 5. normaalimuoto, Domain-key –normaalimuoto, ja viimeisimpänä ajalliseen jatkumoon kantaaottava 6. normaalimuoto.

Normalisointiprosessia on kuvattu alla olevassa kuvassa 7, jossa sisempi kehä tarkoittaa aina ulompaa kehää korkeampaa normalisointiastetta.



Kuva 7. Normalisointikehät.

Normalisointi tapahtuu luomalla alkurelaatiosta normaalimuotoisia projektioita. Projektiot voidaan jälkeenpäin yhdistää luonnollisella liitoksella ja tuloksena on alkurelaatio. Prosessi on siis kaksisuuntainen ja siinä ei häviä tietoa. Elmasri ym. lisäävät varoituksen ylimääräisen tiedon ilmestymisestä luonnollisissa liitoksissa, joka on seurausta huolimattomasti suunnitelluista liitoksista. Tällaisessa tapauksessa normalisoitujen relaatioiden luonnollisen liitoksen tuloksena olevassa relaatiossa on enemmän rivejä kuin pitäisi, ja osa riveistä sisältää väärää tietoa. (Date 1982: 246; Elmasri ym. 2000: 484)

Hovi ym. (2005: 94) mukaan normalisointia ei yleensä tarvitse jatkaa kolmatta normaalimuotoa edemmäs. Elmasri ym. (2000: 484) puolestaan kertovat suunnittelun tavoitteena olevan yleisesti joko Boyce-Codd –normaalimuodon tai neljännen normaalimuodon. Otan tämän tutkimuksen yhteydessä relaatioiden tavoitemuodoksi kolmannen normaalimuodon, joita voidaan edelleennormalisoida Boyce-Codd –normaalimuotoon tai neljänteen normaalimuotoon, mikäli tietokannan toiminta sitä analysointivaiheessa vaatii.

3.4.1 Ensimmäinen normaalimuoto

Jotta relaatio olisi ensimmäisessä normaalimuodossa (1NF), on sen kaikkien tietojen oltava atomisessa muodossa. Tällä tarkoitetaan sitä, että jokainen taulun solu sisältää vain yhden tiedon ja tätä tietoa ei voida enää jakaa pienempiin osiin. Relaatio ei myöskään saa sisältää

toistuvia kenttiä tai ryhmiä. Pratt ym. (1987: 76) esittämien relaation vaatimusten perusteella jokainen relaatio on automaattisesti 1NF. Tämä sääntö ei kuitenkaan päde sisäkkäisiin relaatiomalleihin tai olio-relaatiomalleihin, jotka molemmat sallivat normalisoimattomia relaatioita (Elmasri ym. 2000: 485). (Date 1982: 243; Ricardo 1990: 229)

3.4.2 Toinen normaalimuoto

Relaatio on toisessa normaalimuodossa (2NF) silloin, kun se on 1NF ja kaikki attribuutit, jotka eivät ole avainattribuutteja, ovat täysin riippuvaisia pääavaimesta (Date 1981: 246; Ricardo 1990: 232). Pratt ym. (1987: 142) määritelmän mukaan ei-avainattribuuttien täytyy olla riippuvaisia *koko* pääavaimesta. Tähän määritelmään yhtyvät myös Hovi ym. (2005: 91):

> *”Jos taulussa on moniosainen perusavain, niin kaikkien sarakkeiden tulee olla funktionaalisesti riippuvaisia koko perusavaimesta (eikä siis vain osa-avaimesta).”*

3.4.3 Kolmas normaalimuoto

Relaatio on kolmannessa normaalimuodossa (3NF) silloin, kun se on 2NF ja yksikään arvo ei ole transitiivisesti riippuvainen pääavaimesta (Date 1981: 248). 3NF tavoitteena on siis eliminoida relaatiosta transitiiviset riippuvuudet A→B→C, joissa attribuutti A määrittää attribuutin B, joka puolestaan määrittää attribuutin C (Ricardo 1990: 234). Hovi ym. (2005: 93) ovat kiteyttäneet kolmannen normaalimuodon sanoihin:

> *”Jokaisen sarakkeen pitää olla funktionaalisesti riippuvainen vain perusavaimesta (eikä siis mistään ”tavallisesta” sarakkeesta).”*

3.4.4 Boyce-Codd –normaalimuoto

Boyce-Codd normaalimuoto, eli BCNF, oli tarkoitettu yksinkertaistetuksi muodoksi kolmannesta normaalimuodosta, mutta siitä tulikin ehdoiltaan tiukempi. Alkuperäisen määritelmän mukaan relaatio on BCNF silloin, kun relaatio on 3NF ja kaikki relaation determinantit ovat kandidaattiavaimia. Determinanteiksi Pratt ym. (1987: 146) määrittelevät attribuutit tai attribuuttijoukot, jotka määräävät jonkin toisen attribuutin. (Date 1982: 249)

BCNF tavoitteena on jakaa 3NF-relaatiot siten, että niistä poistetaan turha toisteisuus. Tällaista toisteisuutta voivat aiheuttaa attribuutit, joilla on vain rajallinen määrä mahdollisia arvoja. (Elmasri ym. 2000: 493-494)

3.4.5 Neljäs normaalimuoto

Relaatio R, jossa ovat arvot A, B ja C, on voidaan normalisoida neljänteen normaalimuotoon (4NF) silloin, kun relaatiossa esiintyy moniarvoriippuvuus (esimerkiksi A→→B) ja kaikki R:n attribuutit ovat funktionaalisesti riippuvaisia A:sta. Moniarvoriippuvuus ilmaistaan kaavalla R.A→→R.B | R.C. (Date 1982: 258-259)

Pratt ym. (1987: 156) määritelmän mukaan relaatio on neljännessä normaalimuodossa silloin, kun se on 3NF (BCNF) ja siinä ei esiinny moniarvoriippuvuuksia.

Date:n (1982: 259) mukaan R on parempi jakaa funktionaalisten riippuvuuksinsa suhteen (A→B→C) siten, että tuloksena ovat A→B ja B→C –projektiot. Sama pätee myös moniarvoriippuvuuksiin siten, että jaon tulisi edetä A→→B ja B→→C.

3.4.6 Denormalisointi

Normalisoinnin ensisijaisena tarkoituksena on helpottaa tietojen päivittämistä ja poistaa mahdollisimman tehokkaasti mahdollisten virhetilojen syntyminen. Normalisoinnin tuloksena on kuitenkin enemmän relaatioita lähtötilanteeseen verrattuna, joten tietoja haettaessa on tehtävä useiden taulujen liitoksia. Liitokset puolestaan hidastavat järjestelmän vasteaikaa, joka on joissain tapauksissa tärkeämpää kuin päivittämisen helppous. Tällaisissa tietokannoissa relaatiot voidaan tahallisesti denormalisoida, eli yhdistellä suuremmiksi kokonaisuuksiksi, joissa tietoa toistetaan tahallisesti. Edellämainitun kaltainen denormalisointi voi parantaa järjestelmän toimintaa esimerkiksi tietovarastokannoissa, joissa päivitys on toissijaista vasteaikaan verrattuna. (Hovi ym. 2005: 95-97)

4. INDEKSOINTI

Indeksoinnilla tarkoitetaan järjestetyn luettelon luomista relaatiosta. Relaatiotietokannan tapauksessa indeksi luodaan valituista relaation attribuuteista ja tallennetaan erityiseen indeksitauluun. Indeksitauluun tallennetaan hakuavaimet ja hakuavaimia vastaavien arvojen sijainti indeksoitavassa relaatiossa. Indeksin vertauskuvana voidaan pitää kirjan sisällysluetteloa, johon kerätään kirjan lukijoita kiinnostavat otsikot ja näiden otsikoiden sijainti kirjassa.

Indeksin tarve johtuu relaation luonteesta: relaation rivit ovat vain arvojoukko, joka voidaan esittää missä tahansa järjestyksessä (Date 1982: 83-84). Hakuehtoja vastaavan tiedon etsiminen vaatii siis indeksoimattomassa relaatiossa jokaisen rivin läpikäyntiä (Elmasri ym. 2000: 156). Indeksien käyttö ei vaadi tietokannan suunnittelijalta kyselyjen uudelleenmuotoilua eivätkä ne vaikuta kyselyiden tulosteisiin, mutta ne voivat laskea tietyn kyselyn vasteaikaa tunneista sekunteihin (Lewis ym. 2002: 325). Ullman ym. (1997: 292) määrittelevät indeksoinnin hyödyksi huomattavan vasteajan paranemisen kyselyissä, ja huonoiksi puoliksi kasvavan vasteajan ja monimutkaisuuden tietojen lisäyksissä, päivityksissä ja poistoissa.

4.1 Indeksin hakuavaimet

Indeksitaulun sisältämiä hakuavaimia ei pidä sekoittaa relaatioiden avaimiin, sillä niitä eivät koske samat säännöt. Indeksitaulu voi siis sisältää useita rivejä, joissa toistuu sama hakuavain. Esimerkiksi oppilaitoksen opiskelijatietokannasta luotu indeksi, jossa tietty opiskelijanumero voi osoittaa useisiin eri paikkoihin/tietoihin opiskelijatietokannassa. Tietokannan indeksi voidaan tallentaa joko osaksi taulutiedostoa tai erilliseksi indeksitiedostoksi. Indeksin integroinnilla voidaan pienentää tietokannan viemää tallennustilaa, mutta nykypäivän tallennustilan hinnoilla kokorajoitusta ei voida pitää tietokannan ja indeksien päällimmäisenä suunnittelurajoituksena (Lewis ym. 2002: 339; Hovi ym. 2005: 146). Nykyiset tietokannanhallintajärjestelmät tukevat sekä automaattista indeksointia että indeksien manuaalista luontia. (Lewis ym. 2002: 338)

4.2 Indeksien rakenne

Indeksin hakuavain voi olla jokin yksittäinen attribuutti tai attribuuttijoukko (Ullman ym. 1997: 291). Indeksi voidaan luoda mistä tahansa relaation osasta ja samasta relaatiosta voidaan luoda useita päällekkäisiä indeksejä. Yleisimpiä indeksityyppejä edustavat järjestetyt yksitasoindeksit ja B^+-puut. Indeksejä voidaan luoda myös muilla metodeilla, esimerkiksi hash-koodilla. (Elmasri ym. 2000: 155)

4.2.1 Yksitasoindeksit

Järjestetyn yksitasoindeksin idea on hyvin lähellä kirjojen lopusta löytyvää indeksiä, jossa tärkeät termit ja sivunumerot ovat listattuna aakkosellisessa järjestyksessä. Tiettyä termiä haettaessa ei siis tarvitse selata kirjaa läpi sivu sivulta, vaan indeksin avulla voidaan hypätä oikeaan kohtaan. Yksitasoindeksi voi olla luonteeltaan primäärinen, sekundäärinen tai ryvästetty. Indeksi voidaan luoda joko tiheänä (dense), jolloin jokainen hakuavain osoittaa yhteen tietoon, tai se voidaan luoda harvana (sparse), jolloin hakuavaimet osoittavat tiettyyn lohkoon. (Elmasri ym. 2000: 156-157)

Primäärinen yksitasoindeksi luodaan indeksoitavan relaation primääriavaimen perusteella. Luodussa indeksitaulussa on kaksi kenttää: hakuavain ja osoitin tietoa vastaavaan levylohkoon. Indeksitaulun sisältämät hakuavaimet määrittyvät indeksoitavan relaation varaamien levylohkojen perusteella: mikäli indeksoitava relaatio varaa 8 levylohkoa, niin jokaisen levylohkon ensimmäinen pääavain otetaan indeksin hakuavaimeksi. Indeksoitavan relaation levylohkojen ensimmäisiä arvoja kutsutaan lohkoankkureiksi. Primäärinen yksitasoindeksi on luonteeltaan harva, sillä indeksi ei sisällä jokaista indeksoitavan relaation pääavainta. Koska indeksiin tallennetaan vain osa tiedosta, on sen varaama tila levylohkoina myös huomattavasti indeksoitavaa relaatiota pienempi. (Elmasri ym. 2000: 157)

Elmasri ym. (2000) ovat määrittelyssään lähteneet liikkeelle siitä olettamuksesta, että relaatio on järjestetty aakkosellisesti pääavaimen mukaan. Tämä kuitenkin sotii relaation määritelmiä vastaan, jossa relaation rivien järjestyksellä ei ollut merkitystä. Tällaisesta järjestäytymättömästä relaatiosta pääavaimen perusteella luotava indeksi, joka siis sisältää viittauksen jokaiseen pääavaimeen, on luonteeltaan tiivis. Myös tiiviin indeksin

tilavaatimukset ovat alkurelaatiota pienemmät, sillä siihen tallennetaan vain osa indeksoitavan relaation tiedoista.

Kuten primäärinen yksitasoindeksi, myös sekundäärinen yksitasoindeksi on järjestetty tiedosto, jossa on kaksi kenttää. Sekundäärinen yksitasoindeksi voidaan luoda jonkin ei-järjestävän kentän mukaan, ja samasta relaatiosta voidaan luoda useita sekundääri-indeksejä. Mikäli indeksi luodaan kandidaattiavaimen perusteella, on tuloksena tiheä indeksi (eli kaikki kandidaattiavaimen arvot ovat mukana indeksissä). Jos taas indeksoinnin perustana on jokin ei-avainkenttä, voi tuloksena olla tiheä tai harva indeksi. Kummassakaan tapauksessa sekundääri-indeksi ei tue lohkoankkurointia, sillä järjestäytymättömän relaation tapauksessa on mahdotonta tietää, missä lohkossa haettu tieto sijaitsee. (Elmasri ym. 2000: 162)

Mikäli taulussa olevat tiedot on ryhmiteltävissä jonkin tietyn arvon perusteella, on mahdollista luoda ryvästetty indeksi. Hovi ym. (2005: 156) mukaan ryvästävä indeksi vaikuttaa myös taulun rivien järjestykseen, sillä TKHJ pyrkii järjestämään taulun rivit indeksirivien mukaan. Ryvästetyssä indeksissä on hakuavain jokaiselle eriävälle arvolle ja osoitin ensimmäiseen lohkoon, jossa arvo esiintyy. Ryvästetty indeksi on harva indeksi, sillä indeksi ei sisällä viittauksia jokaiseen yksittäiseen arvoon, vaan ainoastaan niihin ryvästyksen perustana oleviin arvoihin, jotka eroavat toisistaan. (Elmasri ym. 2000: 159-161)

Ryvästetty yksitasoindeksi voidaan luoda esimerkiksi työntekijätietoja sisältävästä relaatiosta, johon on tallennettu työntekijöiden nimet. Indeksi luodaan nimien perusteella siten, että indeksin hakuavaimiksi otetaan nimet, jotka alkavat A:lla, B:llä, C:llä, jne. Indeksin hakuavaimet muodostuvat seuraavasti: hakuavain A ja osoitin ensimmäiseen lohkoon, jossa esiintyy A:lla alkava nimi, hakuavain B ja osoitin ensimmäiseen lohkoon, jossa esiintyy B:lla alkava nimi, jne. Ryvästetyssä indeksissä ei voida hyödyntää lohkoankkureita, ellei jokainen eriävä arvo ala uudesta lohkosta (Elmasri ym. 2000: 162).

4.2.2 Monitasoiset indeksit

Edellä esiteltyjä yksitasoindeksejä tehokkaampaan indeksointiin, ja tätä kautta tehokkaampaan tiedonhakuun, päästään hyödyntämällä monitasoindeksejä.

Monitasoindeksin ideana on pienentää kyselyn kohteena olevaa tietojoukkoa lohkomalla se pienempiin osasiin.

Elmasri ym. (2000: 166-167) mukaan monitasoinen indeksi luodaan siten, että relaatiosta luodaan järjestetty, tiheä indeksi, jota kutsutaan monitasoindeksin ensimmäiseksi tasoksi. Toisen tason indeksi luodaan indeksoimalla ensimmäisen tason indeksi primäärisellä tavalla, eli toinen taso sisältää hakuavaimen ensimmäisen indeksitason varaamien lohkojen ensimmäisistä arvoista ja osoittimen kyseiseen arvoon. Indeksointia jatketaan primäärisellä tavalla luomalla indeksitasoja 2,3,4...n siihen asti, että indeksin n:s taso mahtuu yhteen levylohkoon.

Monitasoindeksi voidaan suunnitella dynaamisesti muuttuvaksi siten, että indeksi pienenee tai kasvaa indeksoitavan taulun mukaan. Esimerkkinä tällaisista dynaamisesti laajenevista ja supistuvista indekseistä ovat B- ja B^+-puut. Indeksin varaamat lohkot pidetään tällöin 50-100% käytössä, jolloin lohkoissa on useimmiten tilaa indeksilisäyksille. Keskimäärin indeksin lohkot ovat 69% käytössä, joten suurin osa indeksilisäyksistä voidaan tehdä ilman indeksin uudelleenjärjestelyä. B-puilla toteutetut indeksit ovat Hovi ym. Mukaan yleisin indeksien toteutustapa (Hovi ym. 2005: 153). (Elmasri ym. 2000: 187)

4.3 Hash-indeksit

Hash-indeksissä indeksitaulu sisältää hakuavaimen ja osoittimen tiedon sisältävään ”ämpäriin” (bucket). Ämpäri on jollain tietyllä perusteella valittu tallennustila-avaruuden osa, esimerkiksi levylohko tai joukko perättäisiä lohkoja. Hash-koodaus voidaan tehdä staattisena, eli määritellään M-määrä ämpäreitä, tai dynaamisena, jolloin ämpärien määrä on muuttuva. Staattinen hash-koodaus aiheuttaa ongelmia silloin, kun tallennettavan tiedon määrä on huomattavasti pienempi kuin ämpäreille varattu tila, tai kun tiedon määrä ylittää tilavaraukset. (Elmasri ym. 2000:142-147, 186)

4.4 Indeksointi ennen tietokannan käyttöönottoa

Puutteellinen indeksointi ei ole katastrofaalista, sillä tietokannan indeksit voidaan saattaa ajan tasalle myös käyttöönoton jälkeen. Tämä on kuitenkin riskialtista ja aiheuttaa

järjestelmään käyttökatkon siksi ajaksi, kun tietokannan hallintajärjestelmä suorittaa uuden indeksin luontia. Tietokannan indeksointi tulisi siis saattaa optimaaliselle tasolle jo ennen järjestelmän käyttöönottoa. Indeksoinnin kolme yleisintä puutetta ovat: (Hovi ym. 2005: 162)

1. indeksien vähyys
2. indeksien puutteellinen sisältö
3. indeksien sisällön väärä järjestys

Hovi ym. esittelevät kaksi hyödyllistä työkalua puutteellisen indeksoinnin havaitsemiseen, joita he nimittävät karkeaksi siiviläksi ja pikaennusteeksi. Karkea siivilä on vaivaton ja nopea työkalu, kun taas pikaennusteen tekeminen vaatii enemmän tietoa ja työtä ongelmakohtien selvittämiseksi. Vastaavasti karkea siivilä tuottaa vain karkean tason arvion indeksoinnin riittävyydestä, ja pikaennuste paljastaa tarkemmin indeksien ja taulujen suunnittelusta aiheutuvat tehokkuusongelmat. (Hovi ym. 2005: 214-215)

4.4.1 Karkea siivilä

Karkeassa siivilässä suunnittelijan täytyy selvittää vain se, ovatko kaikki SQL-kyselyn predikaatit samassa indeksissä. Predikaatit ovat kyselyissä esiintyviä hakuehtoja, jotka ilmaistaan WHERE-lausekkeella. Jos kaikki predikaattisarakkeet ovat indeksisssä, mutta suorituskyky ei parannu halutulla tavalla, voidaan indeksiin lisätä kaikki loputkin taulun sarakkeet. Pelkästään predikaattisarakkeet sisältävää indeksiä Hovi ym. kutsuvat minimipaksunnokseksi, ja kaikki sarakkeet sisältävää indeksiä he nimittävät maksimipaksunnokseksi. (Hovi ym. 2005: 214)

Karkean siivilän käytöllä voidaan havaita suuri osa indeksointipuutteista. Mikäli indeksoinnin jatkuvaan ”parantelu ja testaus” –sykliin ei ole aikaa, Hovi ym. suosittelevat indeksointitavaksi maksimipaksunnosta, jonka rittävyyttä voidaan arvioida toiminnan aikana kerättävistä naularaporteista. (Hovi ym. 2005: 214)

4.4.2 Pikaennuste

Karkeaa siivilää tarkempia tuloksia saadaan käyttämällä pikaennustetta. Pikaennusteen tuloksena saadaan paikallisvaste, joka tarkoittaa käyttäjän näkemää palvelimen vasteaikaa

ilman tietoverkko ym. viiveitä. Viiveitä aiheuttavat tietokantapalvelimen prosessointinopeus, sekä monikäyttäjäympäristössä yleinen jonotus. Paikallisvaste on palveluajan ja jonoajan summa. (Hovi ym. 2005: 215-216)

Paikallisvaste voidaan selvittää kaavalla: $PV = HS \times 10ms + PS \times 0{,}1ms$, jossa: PV on paikallisvaste, HS on hajasipaisujen määrä, ja PS on peräkkäissipaisujen määrä. Sipaisulla tarkoitetaan tietokantajärjestelmän yhden indeksi- tai taulurivin tutkimista. Ensimmäisen rivin tutkiminen vaatii hajasipaisun, joka on käytännössä sama asia kuin pyörivältä kiintolevyltä tehtävä hajalevyluku. Hajasipaisun kesto on 10 millisekuntia. Peräkkäissipaisu on ensimmäistä riviä seuraavien rivien tutkimisesta käytetty nimitys, ja sen järjestelmäkustannus on huomattavasti pienempi (0,1 millisekuntia). (Hovi ym. 2005: 216)

Hovi ym. (2005: 218) esimerkkitapausta mukailevassa tapauksessa huomataan jo karkeaa siivilää käyttämällä indeksin riittämättömyys. Pikaennusteella voidaan puolestaan selvittää indeksiparannuksista koituvat ajalliset säästöt. Oletetaan, että taululle suoritetaan seuraava kysely:

SELECT asiakasnro, etunimi FROM asiakastietotaulu WHERE sukunimi = :sukunimi AND kunta = :kunta

Seuraavin olettamuksin:

1. Asiakastietotaulussa on miljoona riviä
2. Ainoa sopiva indeksi sisältää sarakkeet sukunimi ja etunimi
3. Sukunimi-predikaatin läpäisykerroin (hakuehtoa vastaavien rivien määrä) on 1%
4. Kunta-predikaatin läpäisykerroin on 10%
5. Tulostaulun koko on 1 000 riviä (pahin tapaus)
6. Perusavain-indeksi *asiakasnro* on ryvästävä indeksi ja asiakastietotaulun rivit ovat asiakasnro-järjestyksessä.

Indeksistä on näillä olettamilla luettava 10 001 peräkkäistä riviä, eli $1 \times 10ms + 10000 \times 0{,}1ms = 1010ms$. Taulusta on luettava 10 000 ei-peräkkäistä riviä, sillä kunta-sarake puuttuu indeksistä. Luvun kesto on $10000 \times 10ms = 100s$. Kokonaispaikallisvaste kyselylle on siis $100s + 1010ms = 101s$. Indeksiä paksuntamalla

paikallisvaste voidaan tiputtaa yli sadasta sekunnista yhteen sekuntiin. (Hovi ym. 2005: 218; 226-228)

Pikaennuste perustuu seuraaviin laitteistotason oletuksiin (Hovi ym. 2005: 219-220):

- Suorittimen jonotusaika on merkityksetön; käytössä useita suorittimia ja kuormitus on kohtuullisella tasolla.
- Kiintolevyjen käyttöaste on 15-35%.
- Muu jonotus on merkityksetöntä.
- Lajitteluaika on huomattavasti pienempi kuin lajiteltavien rivien hakuaika.
- Indeksien yläsivut ovat keskusmuistissa / välimuistissa, mutta lehti- ja taulusivut haetaan kiintolevyltä.
- Suorittimien ja kiintolevyjen nopeus on vuoden 2002 yläpään tasoa (suoritin noin 100 MIPS, kiitolevyn pyörimisnopeus 10 000 rpm ja hakuaika enintään 5 ms)

4.5 Taululiitosten indeksointi

Yksi vanhimmista taululiitoksiin liittyvistä optimointikeinoista on liitossarakkeiden indeksointi. Kuten taulukohtaisissa indeksisuosituksissa, myös liitoksissa indeksien riittävä paksuus on olennaista vasteajan minimoimisessa. Taululiitoksia vaativien SQL-kyselyiden vasteajat voivat parhaimmassakin tapauksessa olla normalisoiduilla tauluilla huomattavasti denormalisoituja tauluja korkeampia. (Hovi ym 2005: 260-261)

4.6 Indeksointi tietokannan käyttöönoton jälkeen

Tietokantajärjestelmän käyttöönoton jälkeen suorituskykyä ja vasteaikoja voidaan seurata jäljitys- ja seurantajärjestelmillä. Nämä järjestelmät luovat vasteajoista niin kutsutun ”naularaportin”, jossa poikkeuksellisen pitkät vasteajat ovat ryhmiteltynä ohjelmakohtaisesti. Raportista on löydettävissä toistuvasti hitaat ohjelmat, sekä satunnaisesti erittäin hitaat ohjelmat. Raportin perusteella voidaan jäljittää hitauden syynä olevat seikat, esimerkiksi hitaat SQL-lauseet. Hitauden syy voi olla myös muualla, kuin huonosti suunnitelluissa tauluissa tai puutteellisessa indeksoinnissa. Suuret

tiedonsiirtoryöpyt voivat ruuhkauttaa optimaalisesti suunnitellun järjestelmän, joka puolestaan näkyy kohonneina odotusaikoina paikallisvasteajassa. (Hovi ym. 2005: 210)

Raportissa voi esiintyä tuhansia nauloja sadoista eri ohjelmista, ja tämän tietomäärän läpikäyminen voi olla turhan työlästä. Huomio kannattaakin kohdistaa ensimmäisenä niihin tapahtumiin, joiden paikallisvastetta voidaan todennäköisesti lyhentää indeksointia parantamalla. Näiden tapahtumien tunnusmerkkinä voidaan pitää kahta seikkaa: levypalveluaika on pitkä ja levyluvut kohdistuvat pääosin tauluihin. (Hovi ym. 2005: 238-239)

Tulen seuraavissa luvuissa käymään läpi edellisten lukujen 2-4 teorioita käytännön sovelluksen kannalta. Teorioiden sovittamisessa käytäntöön pyrin noudattamaan luvussa 2.3 esitettyä prosessikuvausta.

5. HYVINVOINTIKYSELYN TIETOKANTAJÄRJESTELMÄN ARKKITEHTUURI

Kuten luvussa kaksi esitettiin, voidaan tietokantajärjestelmän arkkitehtuuri jakaa kolmeen tasoon näkymiensä perusteella. Järjestelmäympäristössä toimivan komponentin, henkilön, tai ohjelman näkemä palanen suuremmasta kokonaisuudesta siis muodostaa malliin yhden tason. Käyn seuraavaksi läpi tämän tutkimuksen kohteena olevan hyvinvointikyselyn tietokantajärjestelmän tasot, sekä näiden tasojen keskeiset komponentit.

5.1 Ulkoinen taso

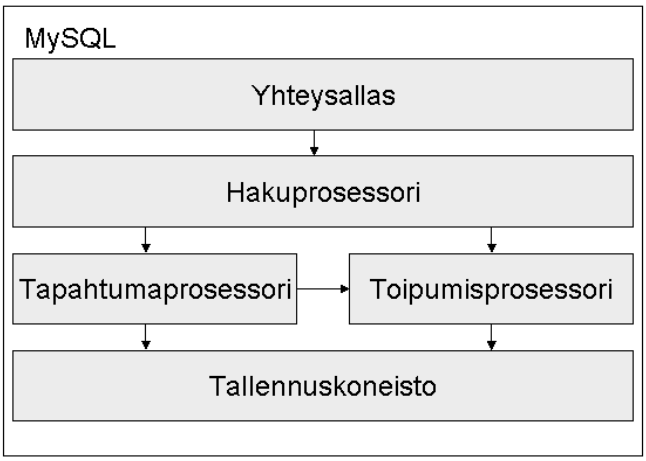
Tämän tutkimuksen yhteydessä ulkoinen taso koostuu neljästä yksilöllisestä käyttäjänäkymästä. Nämä ulkoisen tason näkymät ovat:

- Työntekijä-näkymä, joka on tarkoitettu ainoastaan kysymysten tarkasteluun, oman vastauksen antamiseen ja rajoitettuun omien tietojen päivitykseen
- Ryhmänjohtaja-näkymä, joka on tarkoitettu järjestelmästä saatavien raporttien tarkasteluun
- Ylläpitäjä-näkymään, jolla on oikeudet nähdä ja muokata kaikkia tietokannan tietoja, sekä
- Kysymysten päivittäjä –näkymään, jolla on oikeudet päivittää WWW-kyselylomake –relaation tietoja.

Ulkoisen tason siis muodostavat järjestelmän ulkopuoliset toimijat, jotka näkevät tietokannasta vain taulut tai jonkin rajatun kappaleen tauluista. Yksilölliset käyttäjänäkymät voidaan määritellä joko käyttäjäryhmille yksilöidyillä vastauskäyttöliittymillä (www-sivu tai erillinen ohjelma), tai ne voidaan määritellä TKHJ:n näkymät-osiossa. Mikäli näkymät määritellään TKHJ:n sisällä, niitä varten täytyy luoda jokaista neljää ryhmää vastaava käyttäjäprofiili, jonka perusteella näkymät tauluihin määritellään.

5.2 Konseptitaso

Konseptitason näkymä on TKHJ:n, eli MySQL:n, pääasiallinen toiminta-alue. Konseptitasolla tarkoitetaan järjestelmätason näkymää tietokantakoneiston toimintaan ja koko tietokannan tietoon. Kuvassa 8 on esitetty MySQL:n arkkitehtuuri, joka on yhdistelty kolmen eri lähteen kuvauksista. (McGraw-Hill/Osborne 2004; Bannon, Chin & Kassam 2002; MySQL AB 2007b)



Kuva 8. MySQL:n arkkitehtuuri.

5.2.1 Yhteysallas

Yhteysallas on menetelmä, jolla pieni määrä tietokantayhteyksiä voidaan jakaa suuremmalle osalle käyttäjiä. Yhteysaltaalla voi olla esimerkiksi 15-20 yhteyttä tietokantaan, jolloin näitä yhteyksiä voi käyttää jopa 600 tietokannan ulkopuolista prosessia. Tämä perustuu siihen, että jonkin tietyn prosessin tarvitsema yhteys on vain pienen ajan aktiivisessa käytössä, jolloin yhteys voidaan jakaa useamman prosessin kesken. (MySQL AB 2007c)

5.2.2 Hakuprosessori

Hakuprosessori purkaa tietokantajärjestelmälle suoritetut haut MySQL:n ymmärtämään muotoon ja varmentaa, että haun suorittaneella taholla on oikeudet suorittaa haku. Tämän jälkeen haun optimoija ohjaa haun suorituskomponentille, joka luo suunnitelman haun

tehokkaimmalle suoritustavalle. Optimoija käyttää todennäköisyyksiin perustuvaa optimointia, joka on mahdollista ohittaa. Optimoijan ohittaminen voi olla hyödyllistä, mikäli haun suorittajalla on tietoa haun tehokkaimmasta suoritustavasta. (McGraw-Hill/Osborne 2004)

5.2.3 Tapahtumaprosessori

Tapahtumaprosessori pitää tietokannan johdonmukaisessa ja ristiriidattomassa tilassa, vaikka sen sisältämiä tietoja muokkaa useita käyttäjiä samanaikaisesti. Jokainen tietokannan taulu, jonka tietoja muokataan, käyttää omaa tapahtumaprosessoriaan.

5.2.4 Toipumisprosessori

Toipumisprosessorin tehtävänä on tallentaa muokattuja tietoja ja lokitiedostoja mahdollisten virhetilanteiden varalta. Tällaisia virhetilanteita ovat esimerkiksi vain osittain suoritetut tietojen lisäykset ja indeksien puutteellinen/virheellinen sisältö. Tallennuskoneistoista vain InnoDB ja BDB hyödyntävät kaikkia luvussa kaksi määriteltyjä ACID-ominaisuuksia. (McGraw-Hill/Osborne 2004)

5.2.5 Tallennuskoneisto

MySQL:n tallennuskoneisto toimii yhteistyössä käyttöjärjestelmän kanssa ja se on vastuussa kaikkien järjestelmätietojen, käyttäjätaulujen, indeksien ja lokien kirjoittamisesta tietokantapalvelimen tallennusjärjestelmään. MySQL mahdollistaa eri tallennuskoneiston käytön jokaiselle eri relaatiolle/taululle.

MySQL:n tapauksessa mahdollisia tallennuskoneistoja ovat muun muassa (MySQL 2007b):

- Nopeutta ja täystekstihakua korostava *MyISAM* (MySQL:n oletustallennuskoneisto)
- Samanaikaisuuden hallintaan erikoistunut *InnoDB*
- Keskusmuistissa toimiva *Memory*

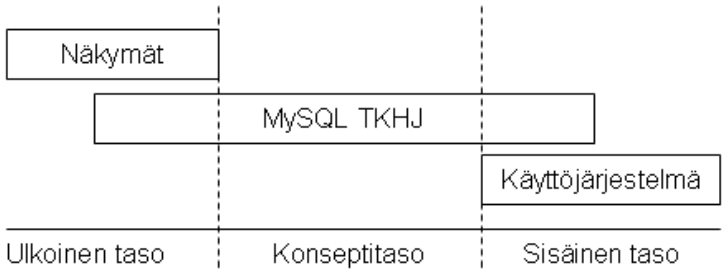
InnoDB mahdollistaa mielivaltaisen kokoiset taulut vaikka käyttöjärjestelmän tiedostojärjestelmä ei sallisikaan esimerkiksi yli 2GB tiedostoja. Tämä johtuu siitä, että InnoDB hajauttaa taulun tarvittaessa useampaan 2GB tiedostoon. (MySQL AB 2007a)

5.3 Sisäinen taso

Sisäisen tason näkymä koostuu laitteiston näkemistä biteistä ja tavuista, jotka tallennetaan tallennusjärjestelmään. Prototyyppilaitteiston tapauksessa tämä tarkoittaa tiedostojärjestelmän näkymää, joka testikoneessa on yksittäiselle kiintolevylle asennetun Ubuntu-linuxin ext3-tiedostojärjestelmän näkymä kiintolevylle kirjoitetuista biteistä.

5.4 Kolmitasoisen arkkitehtuurin uusi graafinen kuvaus

Aiemmin luvussa kaksi esitetty kolmitasoinen kuvaus ei mielestäni tuo tarpeeksi hyvin esille sitä seikkaa, että MySQL hämärtää rajoja arkkitehtuurin tasojen välillä. Selkeämpi kuvaus siitä, miten MySQL sekä määrittelee ulkoisen tason yksilöllisiä käyttäjänäkymiä, että osaltaan voi kiertää sisäisen tason tiedostojärjestelmän rajoituksia, on esitetty kuvassa 9.



Kuva 9. Päällekkäisyyksien kuvaus kolmitasoisessa arkkitehtuurissa.

6. RELAATIOIDEN LUONTI INFORMAATIOTASON SUUNNITTELUN PERUSTEELLA

Kuvaan seuraavissa luvuissa hyvinvointikyselyn tietokannan taulurakenteiden muodostumisprosessia. Suunnittelussa pyrittiin noudattamaan luvussa kaksi esitettyä prosessia fyysiseen toteutukseen asti.

6.1 Käyttäjäympäristön analysointi

Käyttäjäympäristön analysointia ja mallinnusta varten tarvittiin tietoa käyttäjäympäristön käsitteistä, käsitteiden ominaisuuksista, sekä käsitteiden välisistä suhteista. Projektimme tässä vaiheessa meillä ei kuitenkaan ollut vielä virallista testikäyttäjäyritystä, joten esimerkkinä käytettävän pankkikonttorin henkilöstörakenne koostettiin pankkikonttoreille tehdyn kyselyn perusteella.

6.1.1 Henkilöstömäärä ja –hierarkiakysely

Kyselyssä pyydettiin pankkikonttorissa johtavissa asemissa olevia henkilöitä kuvaamaan pankkikonttorin kokonaishenkilöstömäärää, jaottelemaan työntekijät johtaviin ja suorittaviin henkilöihin osastokohtaisesti, sekä erittelemään osa-aikaiset ja täysipäiväiset työsuhteet. Kysely lähetettiin sähköpostitse Oulun Nordean, Osuuspankin ja Sampo Pankin konttorijohtajille. Sähköpostikyselyn lisäksi kysymyksiin pyydettiin vastausta myös puhelimitse ja pankkikonttorissa käymällä, ja kysymyksissä pyrittiin saamaan vastauksia samoihin kysymyksiin kuin sähköpostikyselyssäkin. Vastausten perusteella oli tarkoitus koostaa keskivertoa pankkikonttoria vastaava kokonaisuus, joka henkilöstömääriltään, -hierarkialtaan, sekä muilta ominaisuuksiltaan vastaisi riittävän hyvin reaalimaailman pankkikonttoria.

Varsinaisen testikäyttäjäympäristön puutteen vuoksi tyydyimme kehittämään varsinaisen käsitteitä, ominaisuuksia ja suhteita selvittävän kyselylomakkeen tulevaisuuden tarpeita silmällä pitäen. Henkilöstömääräkyselyn perusteella voisimme tarjota tulevaisuuden prototyyppitestaajille niin kutsutun ”luurangon”, jonka ympärille testaajakohtaiset tarpeet voitaisiin yksilöidä kehitellyn kyselylomakkeen perusteella. Kyselylomakkeen sisällössä

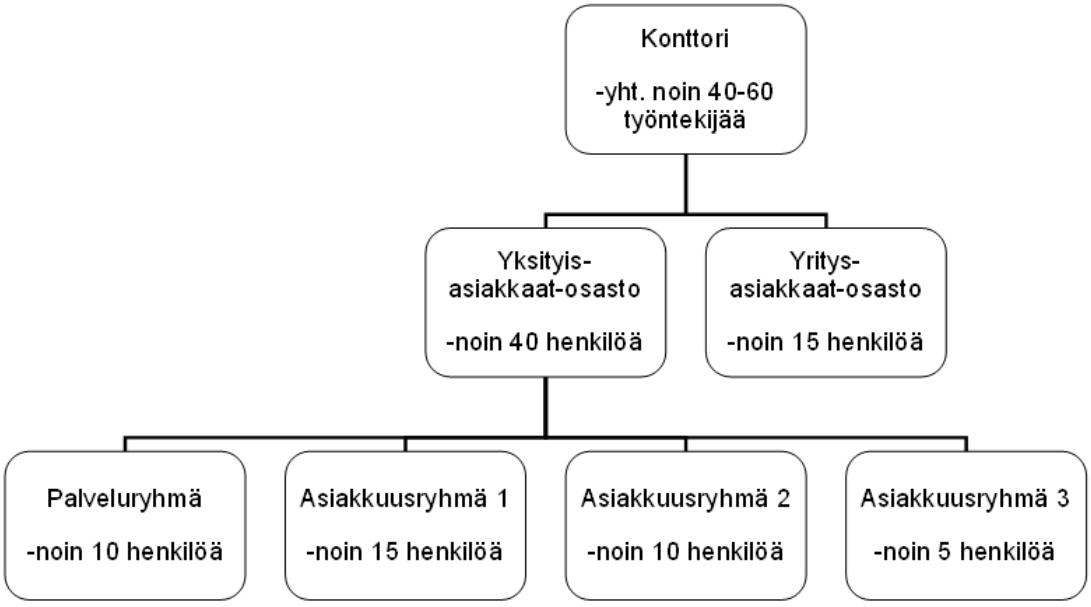
pyrittiin huomioimaan mahdollisimman hyvin luvussa kolme esitellyt vaatimukset. Kyselylomake on esitelty liitteissä (LIITE 1. KYSELYLOMAKE).

6.1.2 Kyselyn vastausten analysointi

Kyselymme vastausten perusteella pankkikonttorien, joihin viitataan tästä eteenpäin myös nimellä konttori, henkilöstömäärä liikkuu yleisesti noin 10-60 henkilön välillä. Sampo Pankin osalta kyselyymme vastanneen henkilön mukaan 10 henkilöä on yleinen määrä pienelle laitakaupungin tai maakunnan konttorille ja heidän suurin konttorinsa, joka sijaitsee Oulussa, työllistää noin 55-60 henkeä. Osuuspankin keskimääräiseksi henkilöstömääräksi saatiin sähköpostivastauksen mukaan noin 18 henkeä per konttori, johon ei ole laskettu yritysasiakkaiden asioita hoitavia pankkitoimihenkilöitä. Suurten keskustakonttorien voidaan olettaa työllistävän tuplaten tämän verran henkilöstöä ja laitakaupungin palvelupisteiden henkilöstömäärä on oletettavasti noin puolet. Nordean osalta kyselyymme ei vastattu, joten konttorikohtaisen henkilöstömäärän voidaan vain olettaa liikkuvan samoissa lukemissa.

6.1.3 Pankkikonttorin henkilöstörakenne

Konttorin henkilöstömäärä ja sisäinen hierarkia pyrittiin koostamaan mahdollisimman hyvin 40-60 –henkistä, suurehkoa pankkikonttoria vastaavaksi. Tällä tavoin tulevan tietokannan suunnittelussa on varauduttu suurimpaan mahdolliseen konttorikohtaiseen kuormitukseen jo suunnitteluvaiheessa, sillä koko henkilöstön oletetaan vastaavan hyvinvointikyselyyn. Konttorin sisäinen henkilöstörakenne on esitetty kuvassa 10.



Kuva 10. Pankkikonttorin sisäinen henkilöstörakenne.

Henkilöstörakennekaaviosta selviää, että pankkikonttorin henkilöstö koostuu:

- Yksityisasiakkaiden osastosta, henkilöstömäärä noin 40
- Erilaisista palveluryhmistä yksityisasiakkaiden osastossa, henkilöstömäärä noin 5-15 per ryhmä
- Yritysasiakkaiden osastosta, henkilöstömäärä noin 15

6.1.4 Hyvinvointikysely

Hyvinvointikyselyn rakenteeksi suunniteltiin 10-15 kohtaa sisältävä, kaksi kertaa viikossa täytettävä kyselylomake. Kyselylomakkeeseen voidaan sisällyttää kysymyksiä ja väittämiä, joista noin kymmeneen vastataan numeroilla ja 1-2 kysymystä täytetään tekstimuotoisena. Esimerkkikyselyssä voisi siis olla 10 väittämää, joihin työntekijä vastaa numeroilla 1-10. Numero 1 tarkoittaa vastaajan olevan täysin eri mieltä ja numero 10 täysin samaa mieltä väittämän kanssa. Näistä kyselyistä saatavan numeerisen informaation pohjalta voidaan luoda trendianalyysejä tietyltä ajanjaksolta. Trendianalyysejä varten vastausten ohella tallennetaan myös vastauspäivämäärä. Lisäksi kyselylomakkeeseen voidaan sisällyttää 1-2 tekstimuotoista vastausta vaativaa kysymystä, joiden pohjalta analyysiä tekevät tekevät

henkilöt saavat tarkemman selvityksen syistä ja seurauksista. Kun vastaukset pysyvät sisällöltään samanlaisina, pystytään tietokantajärjestelmän tilatarpeen kehittymistä seuraamaan suhteellisen tarkasti, ja mahdolliset laitteistohankinnat suorittamaan jo ennen tallennustilan loppumista.

6.1.5 Järjestelmästä saatavat raportit

Tietokantajärjestelmästä saatavat raportit ovat yksi tärkeimmistä järjestelmän toiminnoista. Hyvinvointikyselyn tulosten pohjalta luotavat raportit voivat olla sisällöltään hyvinkin erilaisia, riippuen siitä mitä tunnuslukuja raportin tilaaja haluaa vastausten perusteella tarkastella. Käyttäjäympäristön mallinnusta varten luotiin yleisraportti, jossa ilmoitetaan osastokohtaisesti seuraavat tiedot:

- Osaston nimi
- Kysymyksen numero
- Vastausten keskiarvo
- Vastausten mediaani
- Vastausjoukon suurin arvo
- Vastausjoukon pienin arvo
- Vastausten lukumäärä

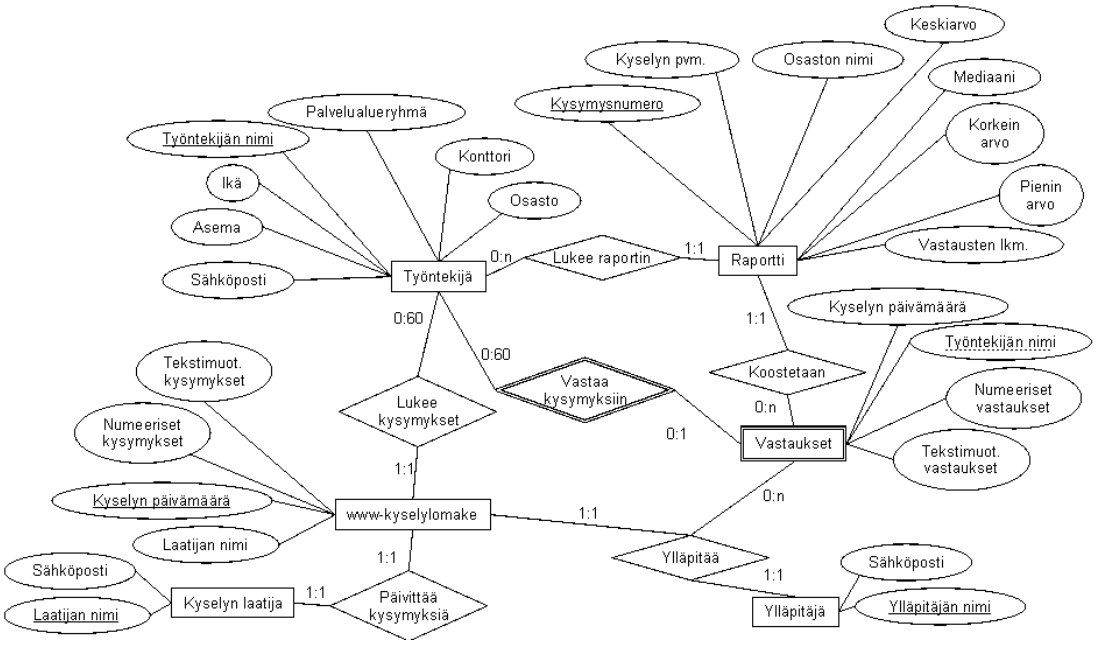
Esimerkkiraportti on kuvattu liitteissä (LIITE 2. ESIMERKKIRAPORTTI).

6.1.6 Käyttäjäympäristön käsitteiden, ominaisuuksien ja suhteiden määrittelyt

Käsitteiden ja niiden ominaisuuksien määrittelyn apuna käytettiin lomaketta, johon kerättiin yhden käsitteen attribuutit määrittelyineen. Attribuuteista kerättyjä määrittelyjä olivat siitä käytetty termi, kuvaus, rajoitukset, määräytymisperuste, sekä luku- ja kirjoitusoikeudet. Määrittelyssä pyrittiin kattamaan mahdollisimman hyvin luvussa 3.1.2 määritellyt analysoitavat osa-alueet. Esimerkkilomake käsitteestä ”Työntekijä” on liitteissä (LIITE 3. KÄSITEMÄÄRITTELY). Analysoinnissa ilmenneet käsitteet, näiden attribuutit, sekä käsitteiden väliset suhteet on esitetty käyttäjäympäristön mallinnuksen yhteydessä, jossa niitä on kuvattu sekä graafisesti, että sanallisesti.

6.2 Käyttäjäympäristön mallinnus ER-kaaviolla

Käyttäjäympäristön analysoinnin jälkeen vuorossa on käyttäjäympäristön mallinnus. Analysointivaiheessa luotujen käsitekuvausten (määrittelylomakkeiden) perusteella tulevan tietokantajärjestelmän toimintaympäristöstä voidaan koostaa TKHJ-riippumaton, konttorikohtainen ER-kaavio. ER-kaavion suhteiden kardinaliteetit on mallinnettu yhtä konttoria vastaavasta tilanteesta. Mallintamisen tuloksena syntynyt alustava ER-kaavio on esitetty kuvassa 11.



Kuva 11. Hyvinvointikyselyyn liittyvän käyttäjäympäristön ER-kaavio.

Suunnitteluprosessin edetessä on syytä muistaa prosessin syklinen luonne; prosessin edetessä ja tietomäärän karttuessa voidaan aina palata taaksepäin tarkentamaan alussa luotua ER-kaaviota. Tulen viemään suunnitteluprosessia läpi alustavan ER-kaavion pohjalta ja esittelen lopullisen ER-kaavion suunnitteluprosessin lopuksi.

6.2.1 Käsitteiden, ominaisuuksien ja suhteiden selitykset

ER-kaavion tärkein funktio on saattaa käyttäjäympäristön analysoinnista saadut tiedot havainnollisempaan muotoon. Tässä vaiheessa myös käsitteiden välisten suhteiden hahmottaminen helpottuu visuaalisen esitystavan myötä.

Käyttäjäympäristön käsitteiksi määriteltiin:

- Työntekijä
- www-kyselylomake
- Vastaukset
- Raportti
- Kyselyn laatija
- Ylläpitäjä

Työntekijä-käsitteeseen määriteltyjen ”Palvelualueryhmä”, ”Osasto”, ”Konttori” ja ”Ikä” –attribuuttien perusteella raportteihin voidaan tehdä erilaisiin ikään tai toimipisteeseen perustuvia jaotteluja. Esimerkkinä tällaisesta jaottelusta ovat palvelualueryhmä-, konttori- tai ikäkohtaiset vastausten keskiarvot. Työntekijästä tallennetaan myös attribuutti ”Työntekijän nimi”, joka valittiin alustavasti käsitteen sisällä työntekijäkohtaisen yksilöivän vaikutuksen aikaansaavaksi ominaisuudeksi. ”Asema”-attribuutti tallennetaan raporttioikeutusten perustaksi, sillä vain esimiesasemassa olevilla henkilöillä on oikeus saada raportteja järjestelmästä. ”Sähköposti”-attribuutilla puolestaan haluttiin varmistaa helppo yhteydenotto tarvittaessa. Työntekijä-käsite on yhteydessä www-kyselylomake –käsitteeseen moni-yhteen -suhteella. Suhteen kardinaliteetti vaihtelee 0 ja 60 välillä riippuen siitä, kuinka moni työntekijä lukee hyvinvointikyselyn kysymykset.

Hyvinvointikysely toteutetaan www-lomakkeella, jonka sisällön päivittämisestä vastaa kyselyn laatija. Kyselylomakkeeseen liittyy kyselyn avainattribuutti ”Päivämäärä”, laatijan ilmaiseva ”Laatijan nimi”-attribuutti, sekä teksti- ja numeromuotoisten kysymysten attribuutit. Työntekijä-käsitteeseen yhdistävän suhteen kardinaliteetti on 1, sillä työntekijät voivat tarkastella ainoastaan hyvinvointikyselyä, joka on kyseisellä hetkellä aktiivinen.

Kyselyn laatijasta tallennetaan attribuutit ”Sähköposti” ja ”Laatijan nimi”. Näistä attribuuteista jälkimmäinen toimii kyselyn laatijan tunnisteena, ja ensimmäisen avulla kyselyn laatijaan voidaan tarvittaessa ottaa yhteyttä. ”Kyselyn laatija” –käsitteen ”www-kyselylomake” –käsitteeseen yhdistävän suhteen kardinaliteetti on 1, sillä vain yksi henkilö on kerrallaan vastuussa yhden kyselylomakkeen kysymysten luomisesta ja päivittämisestä.

Työntekijän vastaukset siirtyvät ”Vastaukset”-käsitteeseen, jonka attribuutteja ovat ”Numeeriset vastaukset”, ”Tekstimuotoiset vastaukset” ja ”Kyselyn päivämäärä”. Näistä kaksi ensimmäistä kertovat työntekijän vastaukset kysymyslomakkeen vastauksiin, ja viimeinen mahdollistaa vastausten yhdistämisen tiettyihin kysymyksiin. ”Työntekijä”-käsitteen ja ”Vastaukset”-käsitteen välisen suhteen kardinaliteetti on minimissään 0 (kukaan ei vastaa kyselyyn) ja maksimissaan 60 (kaikki vastaavat kyselyyn), joka tekee käsitteestä heikon. Käsitteen heikko luonne perustuu siihen olettamaan, että vastausta ei pystytä yhdistämään vastaajaan ilman asianmukaista viittausta (vierasavainta). Käsitteen avainattribuuttina käytetään ”Työntekijän nimi”-attribuuttia, joka saadaan vierasavaimena ”Työntekijä”-käsitteeltä.

”Ylläpitäjä”-käsite, jolla tarkoitetaan koko järjestelmän ylläpitäjää, sisältää attribuutit ”Ylläpitäjän nimi” ja ”Sähköposti”. Kuten ”Kyselyn laatija” –käsitteessä, myös Ylläpitäjä:ssä attribuutit ovat tarkoitettu tunnistukseen ja yhteydenottoon. ER-kaaviosta on jätetty pois ”Ylläpitää”-suhteen yhteys ”Työntekijä” ja ”Raportti”-käsitteisiin kaavion selkiyttämiseksi. ”Ylläpitäjä”-käsite on kuitenkin määritelty sellaiseksi käsitteeksi, jolla on oikeudet koko järjestelmän laajuiseen tietojen lisäämiseen, poistamiseen ja päivittämiseen.

”Raportti”-käsitteen attribuutit lasketaan ”Vastaukset”-käsitteen attribuuteista. Käsitteen attribuuteiksi määriteltiin ”Kysymysnumero”, ”Kyselyn päivämäärä”, ”Osaston nimi”, ”Keskiarvo”, ”Mediaani”, ”Korkein arvo”, ”Pienin arvo” ja ”Vastausten lukumäärä”. Yhden raportin sisältämät osastokohtaiset tiedot lasketaan n –määrästä vastauksia ja n-kappaletta työntekijöitä voi tarkastella raporttia. Käytännössä raporttia tarkastelevien työntekijöiden lukumäärä on kuitenkin jotain 0 ja 10 väliltä, sillä raporttia pystyvät lukemaan ainoastaan esimiesasemassa olevat työntekijät. ”Kyselyn päivämäärä” ja ”Kysymysnumero” –attribuuteilla voidaan tarvittaessa selvittää ne kysymykset, joihin työntekijät ovat vastanneet. ER-kaavioon mallinnetussa raportissa ei hyödynnetty tekstimuotoisten vastausten tuomaa lisäarvoa, mutta ne haluttiin jättää järjestelmään siltä

varalta, että kyselyn laatija voisi niitä tarvittaessa hyödyntää. Tällaisessa tapauksessa ”Raportti”-käsitteen attribuutteihin lisättäisiin myös mahdolliset tekstimuotoiset vastaukset. Tämä toimii hyvänä esimerkkinä ER-kaavion muutosjoustavuudesta.

6.3 ER-kaavion muunto relaatioiksi

Kuten teoriapohjaisessa prosessitarkastelussa todettiin, on seuraavana vuorossa mallinnetun käyttäjäympäristön muunto TKHJ:n muotoon. Tämän tutkimuksen yhteydessä kyseessä on siis ER-kaavion muunto relaatiomuotoon. Tässä vaiheessa jokainen ER-kaavion käsite muunnetaan relaatioksi (tauluksi), ja jokainen käsitteen ominaisuus muunnetaan kyseisen taulun attribuutiksi (sarakkeeksi). Kuvan 12 esimerkkirelaatiossa ER-kaavion käsite ”Työntekijä” on muunnettu relaatiomuotoon. Muiden käsitteiden relaatiomuodot löytyvät liitteistä (0LIITE 4. KÄSITTEIDEN RELAATIOMUODOT).

TYÖNTEKIJÄ						
Työntekijän nimi	Ikä	Konttori	Osasto	Palvelualueryhmä	Asema	Sähköposti

Kuva 12. Käsite “Työntekijä” muunnettuna relaatiomuotoon.

Relaation skeemaan liittyvät tekijät on jo pääosin määritelty käsitemäärittelylomakkeissa (LIITE 3. KÄSITEMÄÄRITTELY), joten niitä ei tarvi määritellä uudestaan. Tällaisia skeemaan liittyviä määriteltyjä ominaisuuksia ovat esimerkiksi attribuuttikohtaisesti tallennettava merkkimäärä, luku- ja kirjoitusoikeudet, sekä null-arvojen sallinta.

6.3.1 Funktionaaliset riippuvuudet ja avainattribuuttien valinta

Käsitteille määriteltiin alustavat avainattribuutit jo ER-kaavion luontivaiheessa. Avainattribuuttien lopullinen määrittely on kuitenkin paikallaan, sillä jotkin aiemmin määritellyistä avainattributeista saattavat aiheuttaa ongelmia. Esimerkkinä tällaisista ongelmista on riviä yksilöivän vaikutuksen lakkaaminen. Relaatioiden kandidaattiavaimet määritellään funktionaalisten riippuvuuksien perusteella, ja näistä kandidaattiavaimista yksi valitaan relaation pääavaimeksi. Seuraavissa luvuissa määrittelen jokaisen aiemmin luodun

relaation pääavaimen. Relaatiot määriteltyine avainattribuutteineen ovat liitteissä (LIITE 5. RELAATIOIDEN AVAINATTRIBUUTIT).

6.3.1.1 Relaatio 1: Työntekijä

Relaation avaimeksi alustavasti valittu ”Työntekijän nimi”-attribuutti määrää funktionaalisesti relaation muut attribuutit; avainominaisuus on siis tältä osin kunnossa. Attribuutti menettää kuitenkin avainominaisuutensa, mikäli konttorissa työskentelee kaksi samannimistä henkilöä.

Vaihtoehtoisesti relaation avain voitaisiin toteuttaa attribuuttiyhdistelmänä, joka toisi ainakin väliaikaisen yksilöivän vaikutuksen relaation riveille. Esimerkki tällaisesta yhdistelmäavaimesta on attribuuttien ”Työntekijän nimi” ja ”Ikä” yhdistelmä. Relaation kandidaattiavaimen määritelmän täyttää myös ”Sähköposti”-attribuutti. Ongelmana tämän attribuutin kohdalla on siihen taatut kirjoitusoikeudet, eli työntekijän oikeus muuttaa yhteystietojaan. Attribuutti ei myöskään ole staattinen, vaan voi muuttua esimerkiksi henkilön vaihtaessa sukunimeään. Edellä mainitut seikat vaikeuttavat, tai tekevät jopa mahdottomaksi, vastausaktiivisuuden seurannan ja luotettavien raporttien luomisen vastausaineistosta.

Ongelmatilanteiden välttämiseksi relaatiolle luodaan 6-numeroinen surrogaattiavain ”TyöntekijäId”, jonka arvo lisääntyy aina yhdellä, kun relaatioon lisätään yksi rivi. Tällöin ”TyöntekijäId” on aina (tai ainakin 999 999 rivinlisäykseen asti) uniikki relaatiossa ”Työntekijä”, joka puolestaan takaa sen avainominaisuuden säilymisen myös suurilla monikkomäärillä. Relaatio, johon uusi avainattribuutti on lisätty, on esitetty kuvassa 13.

TYÖNTEKIJÄ							
TyöntekijäId	Työntekijän nimi	Ikä	Konttori	Osasto	Palvelualueryhmä	Asema	Sähköposti

Kuva 13. Työntekijä-relaation uusi avainattribuutti.

6.3.1.2 Relaatio 2: www-kyselylomake

Avainattribuutiksi valittu ”Kyselyn pvm.” määrittää funktionaalisesti muut attribuutit, joten se valitaan relaation pääavaimeksi. Attribuutti on myös uniikki, sillä tietyllä päivämäärällä luodaan vain yksi kysely. Kysely on samansisältöinen kaikille pankkikonttoreille. Relaatiolla ei ole muita kandidaattiavaimia, sillä laatijan nimi ja kysymykset voivat olla samoja useissa kyselyissä.

6.3.1.3 Relaatiot 3 ja 4: Kyselyn laatija ja Ylläpitäjä

Kuten ”Työntekijä”-relaatiossa, myös näissä relaatioissa nimi-attribuutin luonne vaatii erityistoimia. ”Kyselyn laatija”-relaatioon luodaan surrogaattiavain ”LaatijaId”, ja ”Ylläpitäjä”-relaatioon surrogaattiavain ”YlläpitäjäId”. Kummankin relaation surrogaattiavaimet luodaan samalla periaatteella kuin ”Työntekijä”-relaatiossa.

6.3.1.4 Relaatio 5: Vastaukset

ER-kaavion luonnin yhteydessä todettiin, että ”Vastaukset”-käsite on heikko ja sen avainattribuutti otetaan vierasavaimena ”Työntekijä”-isäntäkäsitteeltä. ”Työntekijä”-relaation pääavain määriteltiin surrogaattiavaimeksi (TyöntekijäId), joka korvasi aiemman avainattribuutin. ”Vastaukset”-relaation pääavaimeksi tuleva vierasavain ”TyöntekijäId” ei kuitenkaan voi toimia relaation pääavaimena yksin, sillä sen yksilöivä luonne häviää kun relaatioon tallennetaan useamman kuin yhden kyselyn vastaukset. Ongelma voidaan poistaa ottamalla lisää attribuutteja avaimeen. Koska yksi työntekijä voi vastata yksittäiseen (tietyn päivämäärän) hyvinvointikyselyyn vain kerran, on luonnollista valita lisäattribuutiksi ”Kyselyn pvm.”. Yhdistelmäavain {TyöntekijäId, Kyselyn pvm.} on varmasti yksilöivä, joten se on relaation pääavain.

6.3.1.5 Relaatio 6: Raportti

”Raportti”-relaation attribuutti ”Kysymysnumero” määrittää funktionaalisesti kaikki muut arvot, joten se voi toimia relaation pääavaimena. Jos raporttiin haluttaisiin keskiarvoja useamman kyselykerran osalta, niin avainattribuutiksi pitäisi lisätä esimerkiksi ”Kyselyn pvm.”, tai luoda erillinen surrogaattiavain.

Hernandez mainitsee kirjassaan (2000: 190), että ideaalitaulun (relaation) ei tulisi sisältää laskettuja kenttiä. Hovi ym. (2005: 82) mainitsevat kuitenkin, että nykyisiin operatiivisiin tietokantoihin lisätään yhä enemmän tietovaraston tyyppisiä ominaisuuksia, sillä tämä yksinkertaistaa ja helpottaa raportointia huomattavasti. Tietovarastotyyppisen relaation ylläpitäminen aiheuttaa erityisvaatimuksia muun muassa tietokannan yhtenäisyydelle ja tietojen ajantasaisuudelle.

6.4 Relaatioiden normalisointi

Kun relaatiot on luotu ja niiden pääavaimet määritelty, on aika poistaa mahdolliset päivitysongelmat normalisoinnin avulla. Normalisoinnin perustana käytän luvussa kolme esiteltyjä normalisointikehiä ja tavoitteenani on ollut kolmas normaalimuoto. Käsittelen seuraavissa luvuissa normalisointia kuvin ja esimerkein. Kaikkien relaatioiden normalisoidut muodot ovat liitteissä (LIITE 6. NORMALISOIDUT RELAATIOT).

6.4.1 Työntekijä-relaatio

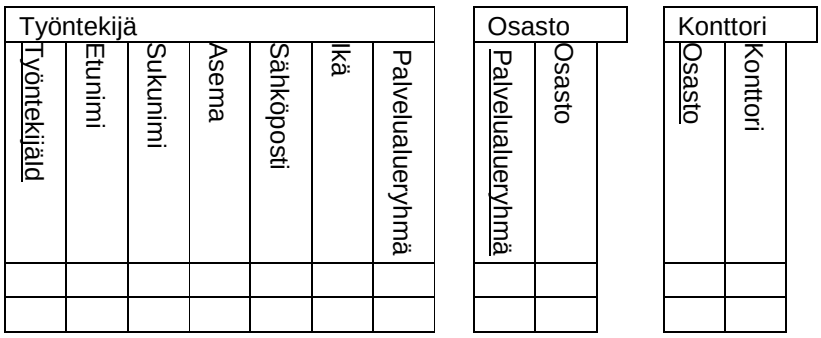
Ensimmäisen normaalimuodon vaatimuksena on tietojen atomisuus ja toistuvien ryhmien poistaminen. Työntekijä-relaatiossa ongelmatilanteen aiheuttaa ”Työntekijän nimi”-attribuutti. Attribuuttiin liittyvä ongelma on sen luonne; toisaalta attribuutti sisältää atomisen tiedon (henkilön nimen), mutta toisaalta nimi koostuu etu- ja sukunimestä. Normalisoinnin tarkoituksena on ehkäistä päivitykseen liittyviä ongelmia, joten on parempi jakaa attribuutti tekijöihinsä (etu- ja sukunimi). Jaettu attribuutti on esitetty kuvassa 14.

TYÖNTEKIJÄ								
<u>TyöntekijäId</u>	Etunimi	Sukunimi	Ikä	Konttori	Osasto	Palvelualueryhmä	Asema	Sähköposti

Kuva 14. Työntekijä-relaatio ensimmäisessä normaalimuodossa.

Jotta relaatio täyttäisi toisen normaalimuodon vaatimuksen funktionaalisista riippuvuuksista, täytyy kaikkien ei-avain attribuuttien olla funktionaalisesti riippuvaisia koko pääavaimesta. Relaatio ”Työntekijä” on toisessa normaalimuodossa, sillä pääavain yksiselitteisesti määrittää jokaisen relaation attribuuteista.

Kolmatta normaalimuotoa varten relaatiosta täytyy eliminoida kaikki transitiiviset riippuvuudet. Ei-avain attribuuttien täytyy olla funktionaalisesti riippuvaisia vain pääavaimesta, eikä mistään normaalista sarakkeesta. Relaatio ei täytä tätä vaatimusta, sillä attribuutti TyöntekijäId määrittää attribuutin Palvelualueryhmä, joka puolestaan määrittää attribuutin Osasto. Transitiivinen riippuvuus etenee vielä attribuutista Osasto attribuuttiin Konttori, sillä tietty osasto voi sijaita vain tietyssä konttorissa. Relaatio saatetaan kolmanteen normaalimuotoon pilkkomalla se osarelaatioihin ”Työntekijä”, ”Osasto” ja ”Konttori”. Normalisoitu relaatio on esitetty kuvassa 15.



Kuva 15. Relaatio “Työntekijä” saatettuna kolmanteen normaalimuotoon.

6.4.2 www-kyselylomake –relaatio

Relaation nimi muutetaan yksinkertaisemmaksi ja kuvaavammaksi nimeksi ”Kysymys” ja attribuutti ”Laatijan nimi” vaihdetaan laatijan yksiselitteisesti ilmaisevaan ”LaatijaId”-attribuuttiin. Jotta relaatio olisi ensimmäisessä normaalimuodossa, on siitä poistettava myös toistuvat ryhmät. Tämä tapahtuu korvaamalla toistuvat ryhmät yhteisillä attribuuteilla ”Kysymysnumero”, ”Kysymystyyppi” ja ”Sisältö”. Relaation pääavain vaatii tämän toimenpiteen jälkeen lisämäärittelyjä, sillä attribuutti ”Kyselyn pvm.” ei enää yksiselitteisesti määrittele kaikkia relaation attribuutteja useampia kyselyjä kattavalla ajanjaksolla. Ongelma voidaan ratkaista ottamalla avaimeen toiseksi attribuutiksi ”Kysymysnumero”. Mahdollisten ohjelmointiongelmien välttämiseksi ”Kyselyn pvm.”-attribuutin nimi muutetaan muotoon ”KyselyPvm”. Ensimmäiseen normaalimuotoon saatettu relaatio on esitetty kuvassa 16.

Kysymys				
Kysymysnumero	KyselyPvm	LaatijaId	Kysymystyyppi	Sisältö

Kuva 16. Relaatio “Kysymys” ensimmäisessä normaalimuodossa.

Toisen normaalimuodon vaatimus siitä, että attribuutit ovat funktionaalisesti riippuvaisia koko pääavaimesta, voidaan toteuttaa jakamalla ”Kysymys”-relaatio kahteen osarelaatioon ”Kysymys” ja ”KyselynLaatija”. Ilman jakamista myös pelkkä pääavaimen osa ”KyselyPvm” riittää määrittelemään attribuutin ”LaatijaId”. Relaatio on tämän jälkeen myös kolmannessa normaalimuodossa, sillä siinä ei esiinny transitiivisia riippuvuuksia.

6.4.3 Vastaukset-relaatio

Relaation nimi muutetaan yksinkertaisempaan muotoon ”Vastaus”. Toistuvat ryhmät eliminoidaan luomalla attribuutit ”Vastausnumero”, ”Vastaustyyppi” ja ”Sisältö”. Avainattribuuttien määrittelyn yhteydessä (luku 6.3.1) määritelty yhdistelmäavain {TyöntekijäId, Kyselyn pvm.} ei enää tämän muutoksen jälkeen täytä avaimelle esitettyjä vaatimuksia rivikohtaisesta yksilöllisyydestä, joten lisäattribuuttien määrittely on tarpeen. Attribuuttiyhdistelmä {TyöntekijäId, KyselyPvm, Vastausnumero} täyttää tämän vaatimuksen, sillä tietty työntekijä voi vastata tiettyyn kysymykseen vain kerran tiettynä päivämääränä. Relaatio on nyt ensimmäisessä normaalimuodossa, sillä sen arvot ovat atomisia ja siinä ei ole toistuvia ryhmiä.

Toisen normaalimuodon vaatimus siitä, että kaikki ei-avain attribuutit ovat riippuvaisia koko pääavaimesta, ei täyty attribuutin ”Vastaustyyppi” kohdalla. Pääavaimen osa {KyselyPvm, Vastausnumero} riittää määrittämään attribuutin, joten se on siirrettävä omaksi relaatiokseen. Normalisoinnin tuloksena ovat relaatiot ”VastausTyyppi” ja ”VastausSisältö”. Normalisoidut relaatiot ovat myös kolmannessa normaalimuodossa.

6.4.4 Ylläpitäjä ja kyselyn laatija -relaatiot

Kummassakin relaatiossa nimen sisältävä attribuutti vaatii toimenpiteitä. Attribuutti jaetaan etu- ja sukunimen sisältäviin attribuutteihin, kuten tehtiin aiemmin ”Työntekijä” relaation kohdalla. Tämän jaottelun jälkeen voidaan huomata, että tietokanta sisältää samantyyppisiä tietoja useissa eri relaatioissa. Skeeman yksinkertaistamiseksi voidaan palata askel

taaksepäin ja yhdistää kaikki tietokantaympäristössä työskentelevien henkilöiden tiedot samaan relaatioon, jonka jälkeen syntynyt relaatio voidaan uudelleennormalisoida normalisointisääntöjen mukaisesti.

Kolmanteen normaalimuotoon saatetusta ”Työntekijä”-relaatiosta, sekä ”Ylläpitäjä” ja ”Kyselyn laatija” –relaatioista yhdistetään uusi ”Henkilö”-relaatio. ”TyöntekijäId”, ”YlläpitäjäId” ja ”LaatijaId”-attribuutit yhdistetään ”HenkilöId”-attribuutiksi. Yhdistetty relaatio on esitetty kuvassa 17.

Henkilö						
HenkilöId	Etunimi	Sukunimi	Asema	Sähköposti	Ikä	Palvelualueryhmä

Kuva 17. Yhdistetty “Henkilö”-relaatio.

Ongelmia aiheuttavat tiedon lisäyksessä syntyvät null-arvot, sillä kyselyn laatijasta tai tietokannan ylläpitäjästä ei tallenneta ikätietoja, eivätkä he välttämättä ole minkään palvelualueryhmän alaisuudessa. Ongelman ratkaisemiseksi ”Henkilö”-relaatio jaetaan edelleen kolmeen relaatioon ”HenkilöNimi”, ”HenkilöIkä” ja ”Palvelualueryhmä”.

6.4.5 Raportti

Relaatio on ensimmäisessä normaalimuodossa, sillä sen sisältämät tiedot ovat atomisia ja se ei sisällä toistuvia ryhmiä. Myös kolmannen normaalimuodon vaatimukset täyttyvät, sillä yksikään attribuutti ei ole riippuvainen pääavaimen osasta tai toisesta normaalista kentästä. Tämän relaation osalta normalisoinnilla ei kuitenkaan ole niin suurta merkitystä, sillä se on ensisijaisesti tehty vähentämään raportoinnin aiheuttamaa järjestelmäkuormitusta.

Tietokannan luomisprosessin syklisyys on vaikuttanut alkuperäiseen ER-kaavioon. Iteraatiokierrosten seurauksena syntynyt normalisoitu ER-kaavio on esitetty liitteissä (LIITE 7. ER-KAAVIO NORMALISOIDUISTA RELAATIOISTA).

7. TIETOKANNAN FYYSINEN TOTEUTUS

Paperilla tapahtuneen relaatioiden muodostamisen jälkeen taulurakenteet voidaan saattaa fyysiseen muotoonsa tietokantapalvelimelle. Käyn seuraavissa luvuissa läpi käytetyt tallennuskoneistot ja indeksointitavat, tietokantaan syötetyn testitiedon luontiprosessin, sekä tutkimuksessa käytetyn palvelimen laitteisto- ja ohjelmistokokoonpanon.

7.1 Käytetyt tallennuskoneistot ja indeksointitavat

MySQL mahdollistaa useiden erilaisten tallennuskoneistojen käytön ja jokaisen tallennuskoneiston indeksointitavoissa on pieniä eroja. Oletustallennuskoneistona MySQL käyttää MyISAM-tallennuskoneistoa, mutta käyttäjä ei ole sidottu tähän tallennuskoneistoon. Esittelen seuraavissa luvuissa kolmen hyödynnettävän tallennuskoneiston ominaisuuksia ja käyn läpi niiden mahdollistamat indeksointitavat.

7.1.1 MyISAM-tallennuskoneisto

Suurimmassa osassa tämän tietokannan tauluja ei tarvita suurta samanaikaisuuden hallintaa tietojen lisäyksen osalta, joten ne toteutetaan MyISAM-tallennuskoneistolla. MyISAM-tallennuskoneiston etuna on MySQL AB:n mukaan hakujen nopeus ja täystekstihaut (MySQL AB 2007a: 772).

MyISAM-tallennuskoneisto mahdollistaa 64 indeksiä per taulu ja yhdessä indeksissä voi olla 16 saraketta. Erillisellä käynnistysoptiolla indeksejä voi olla 128 kappaletta per taulu. Indeksit on mahdollista luoda ainoastaan järjestettyinä B-puina. (MySQL AB 2007a: 773-779)

7.1.2 InnoDB-tallennuskoneisto

Vastauksia tallentavat taulut (VastausTyyppi ja VastausSisältö) saattavat joutua hetkellisesti kovankin kuormituksen kohteeksi, joten ne toteutetaan InnoDB-tallennuskoneistolla. InnoDB-tallennuskoneisto täyttää luvussa kaksi esitellyt ACID-vaatimukset, joten sen pitäisi pystyä pitämään tietokannan tiedot eheinä myös tietokannan ruuhkautuessa. Mikäli taululle ei ole määritelty pääavainta, MySQL etsii taulusta ensimmäisen yksilöllisiä, ei null arvoja sisältävän sarakkeen (kandidaattiavaimen) ja

InnoDB käyttää sitä pääavaimena. Mikäli taulussa ei tällaista saraketta ole, InnoDB luo taululle automaattisesti kasvavan 6-tavuisen pääavaimen. (MySQL AB 2007a: 812-813)

InnoDB-tallennuskoneisto käyttää pääavaimen perusteella luotavaa ryvästävää indeksointia, jossa indeksi tallennetaan B-puuksi. InnoDB tarkkailee jatkuvasti taulun indeksin perusteella suoritettavia hakuja, ja mikäli se päättelee hakujen hyötyvän B-puun perusteella luotavasta hash-indeksistä, se luo tällaisen automaattisesti. MySQL:n ohjekirja ei kerro sitä kuinka monta indeksiä tai saraketta per indeksi tämä tallennuskoneisto sallii. (MySQL AB 2007a: 812-813)

7.1.3 Memory-tallennuskoneisto

Tietokantajärjestelmästä saatava raportointi päätettiin toteuttaa erillisellä relaatiolla. Tämä taulu toteutetaan pelkästään keskusmuistin varassa operoivalla Memory-tallennuskoneistolla, sillä sen hyödyllisyys järjestelmän kannalta vaatii nopeutta hauissa ja päivityksissä. Taulun tiedot häviävät esimerkiksi palvelimen virhetilanteessa tai uudelleenkäynnistyksessä, mutta nämä tiedot voidaan luoda uudelleen vastausten perusteella.

Pelkästään keskusmuistissa toimivat Memory-tallennuskoneiston taulut käyttävät indeksoinnissaan oletusarvoisesti hash-indeksejä, mutta niissä on mahdollista hyödyntää myös B-puu –indeksointia. Taululla voi olla 32 indeksiä ja jokainen indeksi voi sisältää 16 saraketta.

7.2 Testitiedon luomisessa käytetty menetelmä

Tietokantahakuja varten tietokantaan tarvittiin reaalimaailman käyttötilannetta vastaava tietomäärä. Tiedot luotiin käyttämällä php-skriptejä, eli php-kielellä toteutettuja web-ohjelmia. Php-skripteillä tietokantaan automaattisesti syötettävää tietoa voitiin hallita niin, että tietokannan tietoihin saatiin todellisuutta vastaavaa vaihtelevuutta.

Tietojen syöttäminen aloitettiin pankkikonttorien ja työntekijöiden luonnilla. Php-skriptin yksi kierros syötti yhteen pankkikonttoriin 5-60 työntekijää. Luontisilmukkaa toistettiin niin kauan, että toinen kahdesta ehdosta täyttyi: 6600 työntekijää tai 330 pankkikonttoria.

Näin ollen käyttäjämäärä vastasi Osuuspankista saatuja tietoja Suomen konttori- ja työntekijämääristä.

Tämän jälkeen tehtiin php-skripti, joka loi vastaukset tietokantaan. Tässä tapauksessa jokainen luotu työntekijä ”vastasi” hyvinvointikyselyyn jokaisena kyselykertana. Tietoa luotiin vuoden aikaväliä kuvaavalta ajanjaksolta. Kyselyn vastaukset koostuivat kymmenestä numeromuotoisesta ja kahdesta tekstimuotoisesta vastauksesta per työntekijä per kyselykerta. Vastaukset muodostuivat siten, että jokaisen työntekijän jokaiseen numeromuotoiseen kysymykseen antama vastaus oli 1-10 tasaisesti jakautuneella todennäköisyydellä, ja jokaiseen tekstimuotoiseen kysymykseen syötettiin vakioitu 255-merkin maksimimittainen teksti. Reaalimaailman vastausten hajontaa olisi kuvannut paremmin normaalijakauma, mutta tämänkaltaisen hajonnan toteuttamista ei pidetty tarpeellisena; jonkin tietyn arvon esiintyminen voidaan joka tapauksessa laskea todennäköisyyden perusteella. Tekstimuotoisilla vastauksilla vastaustaulun kokoa kasvatettiin niin, että sen koko oli vuoden vastausten jälkeen suurin mahdollinen.

7.3 Tietokantapalvelimen laitteisto ja käytetyt ohjelmat

Tietokantapalvelimessa käytettiin Intelin Core2Quad 6600 –prosessoria, joka toimi 2,4 gigahertsin taajuudella. 800 megahertsin taajuudella toimivaa DDR2-tyypin keskusmuistia palvelimeen asennettiin 4 gigatavua. Kiintolevynä käytettiin Seagaten 160 gigatavuista, sata-liitäntäistä, ST316002 3AS –mallia. Sekä prosessorissa, että kiintolevyssä, oli 8 megatavua omaa välimuistia.

Käyttöjärjestelmänä toimi 64-bittinen Ubuntu Linux 7.10 ”Gutsy Gibbon” kernel-versiolla 2.6.22-14-generic. Tiedostojärjestelmänä käytettiin ext3:a. Työpöytäympäristönä käytössä oli Gnome versio 2.20.1. Tietokannan hallintajärjestelmänä oli MySQL 5.0.45-Debian_1ubuntu3.1-log ja tietokantakyselyt suoritettiin MySQL Query Browserilla, jonka versio oli 1.2.5beta.

8. SUORITUSKYVYN VIRITYS JA ARVIOINTI

Kun tietokanta on saanut fyysisen muotonsa, on aika tarkastella sen suorituskykyä. Karkealla siivilällä pystytään tekemään pikainen arvio taulujen indeksoinnin riittävyydestä tietylle haulle. Perusohjeena on tarkistaa kaikkien hakuehtojen, eli predikaattien, löytyminen samasta indeksistä. Pikaennusteella voidaan puolestaan suorittaa yksityiskohtaisempia laskelmia hakujen kestoista ja tätä kautta laskea parannettujen indeksien vaikutus hakujen kestoon. Kumpikin metodi esiteltiin luvussa neljä.

MySQL luo oletuksena MyISAM ja InnoDB tauluille B-puu –indeksin, johon sisältyvät kaikki pääavaimen attribuutit. Tähän primääriseen indeksiin ei oletusasetuksilla voi lisätä sarakkeita, eikä niitä voi myöskään poistaa. Tauluille voidaan kuitenkin luoda lisäindeksejä kuten tallennuskoneiden esittelyn yhteydessä todettiin. Arviointimenetelmien hyödyllisyyden vertailussa käytettiin kahta tietokantahakua ja lisäksi tehtiin vuosiraportin koostava haku php-ohjelmointikielellä, jolla saatiin yleiskuva järjestelmän suorituskyvystä. Käyn seuraavissa luvuissa läpi käytetyt hakulauseet, indeksoinnin riittävyyden arvioinnin karkealla siivilällä, hakuaikojen arvioinnin pikaennusteella, indeksisarakkeiden järjestyksen vaikutuksen haun nopeuteen, sekä raportin koostamisen php-skriptillä.

8.1 Käytetyt hakulauseet

Indeksien riittävyyden ja indeksisarakkeiden järjestyksen vaikutusten arvioinnissa käytettiin kahta SQL-hakulausetta. Lauseista ensimmäinen sisältää yhden hakuehdon, eli predikaatin, jolle annettiin vakioarvo 2. Toinen hakulause sisältää tämän lisäksi arvoalueen määrittävän predikaatin. Ensimmäisen hakulauseen avulla testattiin pikaennusteen paikkansapitävyyttä ja toisella hakulauseella kahden indeksin välisiä nopeuseroja. Käytetyt hakulauseet olivat:

1. SELECT * FROM VastausSisalto WHERE Sisalto = '2';
2. SELECT * FROM VastausSisalto WHERE Sisalto = '2' AND KyselyPvm BETWEEN '2007-04-28' AND '2007-09-07';

8.2 Karkea siivilä

Karkeaa siivilää käytettäessä täytyy käytetyistä hakulauseista selvittää hakupredikaatit. Hauissa käytettyjen SQL-lauseiden predikaatit olivat:

1. Sisalto = '2'
2. Sisalto = '2' AND KyselyPvm BETWEEN '2007-04-28' AND '2007-09-07'

Ensimmäisen hakulauseen predikaatti ei ole indeksissä, sillä ”Sisältö”-sarake ei ole osa taulun pääavainta. Indeksipuute todennettiin MySQL:n konsolissa ajetulla EXPLAIN-komennolla. Komennon tuloste on esitetty kuvassa 18.

```
mysql> EXPLAIN SELECT * FROM VastausSisalto WHERE Sisalto = '2';
+----+-------------+----------------+------+---------------+------+---------+------+---------+-------------+
| id | select_type | table          | type | possible_keys | key  | key_len | ref  | rows    | Extra       |
+----+-------------+----------------+------+---------------+------+---------+------+---------+-------------+
|  1 | SIMPLE      | VastausSisalto | ALL  | NULL          | NULL | NULL    | NULL | 9541701 | Using where |
+----+-------------+----------------+------+---------------+------+---------+------+---------+-------------+
1 row in set (0.01 sec)
```

Kuva 18. Ensimmäisen hakulauseen indeksipuute.

Indeksipuute korjattiin luomalla hakua vastaava indeksin minimipaksunnos. Indeksilisäys toteutettiin luomalla uusi B-puu –indeksi ”Minimipaksunnos”, johon lisättiin ainoastaan Sisalto-sarake. Minimipaksunnoksen vaikutus todennettiin EXPLAIN-komennolla. Komennon tuloste on esitetty kuvassa 19.

```
mysql> EXPLAIN SELECT * FROM VastausSisalto WHERE Sisalto = '2';
+----+-------------+----------------+------+-----------------+-----------------+---------+-------+--------+--------------------------+
| id | select_type | table          | type | possible_keys   | key             | key_len | ref   | rows   | Extra                    |
+----+-------------+----------------+------+-----------------+-----------------+---------+-------+--------+--------------------------+
|  1 | SIMPLE      | VastausSisalto | ref  | Minimipaksunnos | Minimipaksunnos | 257     | const | 590724 | Using where; Using index |
+----+-------------+----------------+------+-----------------+-----------------+---------+-------+--------+--------------------------+
1 row in set (0.05 sec)
```

Kuva 19. Ensimmäistä hakulausetta vastaava indeksin minimipaksunnos.

Pääavaimen perusteella luotu indeksi oli riittämätön myös toiselle hakulauseelle. Ensimmäisen hakulauseen perusteella luotu minimipaksunnos ei myöskään ollut riittävä, sillä käytetyt predikaatit ”Sisalto” ja ”KyselyPvm” eivät olleet samassa indeksissä. Hakulausetta varten luotiin kaksi uutta indeksiä: Pvm_Sisalto ja Sisalto_Pvm. Indeksisarakkeiden järjestyksen vaikutusta hakunopeuteen on tutkittu tarkemmin luvussa 8.4.

8.3 Pikaennuste

Pikaennusteella voidaan laskea karkeaa siivilää tarkemmin lisäindeksoinnin tuoma hyöty. Lasken tässä luvussa pikaennusteet luvussa 8.1 esitellylle hakulauseelle numero 1 Hovi ym. (2005: 216) määrittelemällä kaavalla ja vertaan hakuajoista saatuja tuloksia laskukaavasta saatuihin arvoihin. Pikaennusteiden kohteena oli ”VastausSisalto”-taulu, jossa on 9541440 riviä. Pikaennuste hakuajoisa oli seuraava:

Ensimmäinen hakulause ilman lisäindeksointia: hakupredikaatin läpäisykerroin 1/12 ja ei sopivaa indeksiä. Taulusta luetaan noin 795120 ei-perättäistä riviä, jolloin haun kesto on 795120*10ms= 7951200ms = 7951,2 sekuntia
Ensimmäinen hakulause lisäindeksoinnin jälkeen: Indeksistä luetaan yksi hajasipaisu ja 795120 perättäistä riviä, jolloin haun kesto on 1*10ms+795120*0,1ms= 79522ms = 79,522 sekuntia

Tietokannan ”VastausSisalto”-taululle suoritettiin ensimmäinen hakulause sekä ennen lisäindeksointia, että sen jälkeen. Hakujen nopeus mitattiin kolmella perättäisellä haulla. Hakujen tuloksista on laskettu rivikohtainen hakuaika jakamalla haun kesto haetulla rivimäärällä. MySQL ilmoittaa haun keston muodossa ”minuutit:sekunnit.tuhannesosat”, joka muutettiin laskukaavaa varten millisekunneiksi. Taulukoissa 1 ja 2 on esitetty ensimmäisen hakulauseen kestot ennen lisäindeksointia ja lisäindeksoinnin jälkeen:

Taulukko 1. Ensimmäisen hakulauseen kestot ennen lisäindeksointia.

Hakunumero	Haetut rivit	Kesto	Laskukaava	Hakuaika/rivi
1	794961 kpl	0:22.4545s	22454,5/794961	~0,030ms
2	794961 kpl	0:05.0964s	5096,4/794961	~0,006ms
3	794961 kpl	0:05.2400s	5240/794961	~0,007ms

Taulukko 2. Ensimmäisen hakulauseen kestot lisäindeksoinnin jälkeen.

Hakunumero	Haetut rivit	Kesto	Laskukaava	Hakuaika/rivi
1	794961 kpl	0:01.1364s	1136,4/794961	~0,001ms
2	794961 kpl	0:00.0446s	44,6/794961	~0,00006ms
3	794961 kpl	0:00.1836s	183,6/794961	~0,0002ms

Parannukset hakuajoissa osoittavat selvästi indeksoinnin tuoman nopeuslisän, sillä indeksoinnin jälkeen rivikohtaiset hakuajat olivat 30-100 kertaa nopeampia. Hakutuloksista nähdään selvästi myös MySQL:n hakuvälimuistin vaikutus. MySQL tallentaa välimuistiin edellisiä hakuja, jolloin seuraava samanlainen haku on ensimmäistä hakukertaa huomattavasti nopeampi.

8.4 Indeksisarakkeiden järjestyksen vaikutus

Indeksisarakkeiden keskenäisen järjestyksen vaikutusta testattiin ”VastausSisalto”-taululla, johon oli luotu ”Pvm_Sisalto” ja ”Sisalto_Pvm” –indeksit. MySQL:n valinta kahden indeksin välillä oli ”Sisalto_Pvm”, joka todettiin hakulauseelle numero 2 annetun EXPLAIN-komennon tulosteesta. Komennon tuloste on esitetty kuvassa 20.

```
mysql> EXPLAIN SELECT * FROM VastausSisalto WHERE Sisalto = '2' AND KyselyPvm BETWEEN '2007-04-28' AND '2007-09-07';
+----+-------------+---------------+-------+-----------------------+-------------+---------+------+--------+--------------------------+
| id | select_type | table         | type  | possible_keys         | key         | key_len | ref  | rows   | Extra                    |
+----+-------------+---------------+-------+-----------------------+-------------+---------+------+--------+--------------------------+
|  1 | SIMPLE      | VastausSisalto | range | Pvm_Sisalto,Sisalto_Pvm | Sisalto_Pvm | 260     | NULL | 420648 | Using where; Using index |
+----+-------------+---------------+-------+-----------------------+-------------+---------+------+--------+--------------------------+
1 row in set (0.00 sec)
```

Kuva 20. Indeksisarakkeiden järjestyksen vaikutus MySQL:n indeksivalintaan.

Indeksisarakkeiden keskenäisen järjestyksen merkitystä hakunopeuteen tutkittiin mittaamalla hakulausekkeen numero 2 kestoa käytettäessä MySQL:n valitsemaa ”Sisalto_Pvm” –indeksiä ja pakottamalla toisessa hakutapauksessa MySQL käyttämään ”Pvm_Sisalto” –indeksiä. Sisalto_Pvm –indeksiä käytettäessä hakuajat olivat:

Taulukko 3. Toisen hakulauseen kestot käytettäessä alkuperäistä indeksivalintaa.

Hakunumero	Haetut rivit	Kesto	Laskukaava	Hakuaika/rivi
1	291840 kpl	0:00.3870s	387/291840	~0,001ms
2	291840 kpl	0:00.3614s	361,4/291840	~0,001ms
3	291840 kpl	0:00.3855s	385,5/291840	~0,001ms

MySQL pakotettiin käyttämään Pvm_Sisalto –indeksiä, jolloin hakuajat olivat:

Taulukko 4. Toisen hakulauseen kestot pakotetulla indeksillä.

Hakunumero	Haetut rivit	Kesto	Laskukaava	Hakuaika/rivi
1	291840 kpl	0:02.0230s	2023/291840	~0,007ms
2	291840 kpl	0:02.7928s	2792,8/291840	~0,010ms
3	291840 kpl	0:02.2251s	2225,1/291840	~0,008ms

Taulukoissa 3 ja 4 esitetyistä tuloksista on nähtävissä indeksisarakkeiden vaikutus haun nopeuteen. Pvm_Sisalto –indeksiä käytettäessä hakuaika oli keskimäärin 2,3470 sekuntia, joka on yli kuusinkertainen hakuaika alkuperäiseen indeksivalintaan verrattuna. Ero voidaan selittää indekseissä olleiden rivimäärien erolla: Pvm_Sisalto –indeksissä oli 2699580 riviä ja Sisalto_Pvm –indeksissä oli 420648 riviä. MySQL osaa siis näiden tulosten perusteella valita tehokkaimman käytettävän indeksin sillä oletuksella, että kyseinen indeksi on olemassa.

8.5 Vuosiraportti

Vuosiraportin koostaminen toteutettiin erillisellä php-skriptillä, joka on esitetty liitteissä (LIITE 9. VUOSIRAPORTIN PHP-SKRIPTI). Erillisen ohjelmointirajapinnan kautta toteutettu haku simuloi järjestelmäympäristöä MySQL Query Browseria paremmin, sillä hyvinvointikyselyn käyttöliittymät on tarkoitus toteuttaa Apache-internetpalvelinohjelmistolla ja php-skripteillä. Vuosiraportin koostava php-skripti ajettiin ensin perustilassa olevalle tietokannalle, jossa oli vain pääavaimiin perustuvat indeksit, ja sen jälkeen sama skripti ajettiin tietokannalle, johon oli luotu edellisessä luvussa mainitut Pvm_Sisalto ja Sisalto_Pvm –indeksit. Lisäksi VastausTyyppi-taululle luotiin indeksi, joka sisälsi Vastaustyyppi-sarakkeen. Lisäindeksoinnilla pyrittiin kattamaan vaatimus taululiitosten indeksoinnista, joka esitettiin luvussa 4.5.

Skripti ajettiin kolme kertaa peräkkäin kummassakin tapauksessa. Perusindeksoinnilla tehtyjen hakujen kestot on esitetty taulukossa 5.

Taulukko 5. Vuosiraportin koostamisajon kesto ilman lisäindeksointia.

Ajonumero	Kesto	Luodut rivit
1	5701.27948403 sekuntia	1200 kpl
2	0.100069999695 sekuntia	1200 kpl
3	0.097069978714 sekuntia	1200 kpl

Ensimmäinen haku kesti noin 95 minuuttia. Haun tuloksena syntyi 1200-rivinen tuloste, jossa oli listattuna vuoden jokainen kyselykerta ja kyselykerran numeromuotoisten vastausten keskiarvo. Kyselykertoja oli 120 kappaletta ja numeromuotoisia vastauksia per kyselykerta oli 10 kappaletta.

Sama skripti ajettiin tietokannalle, johon oli tehty lisäindeksointi. Ennen hakua MySQL käynnistettiin uudestaan, jotta hakuvälimuisti saatiin tyhjennettyä vertailukelpoisia tuloksia varten. Hakujen kestot lisäindeksoinnin jälkeen on esitetty taulukossa 6.

Taulukko 6. Vuosiraportin koostamisajon kesto lisäindeksoinnin jälkeen.

Ajonumero	Kesto	Luodut rivit
1	73.160020113 sekuntia	1200 kpl
2	0.0864350795746 sekuntia	1200 kpl
3	0.102666139603 sekuntia	1200 kpl

Vuosiraportin koostamisessa koostamisaika laski noin 1,5 tunnista hieman reiluun minuuttiin. Kummassakin tapauksessa voidaan nähdä hakuvälimuistin nopeuttava vaikutus. Hakuvälimuistin vaikutus kuitenkin lakkaa, mikäli tauluhin kirjoitetaan muutoksia tai jos MySQL käynnistetään uudestaan.

9. POHDINTA JA JOHTOPÄÄTÖKSET

Tietokantajärjestelmää luotaessa kolmitasoisen arkkitehtuurin kautta tapahtuva tarkastelu helpottaa järjestelmäympäristön suunnittelun alkuvaihetta ja ohjaa suunnittelijan huomion olennaisiin osa-alueisiin. Järjestelmäympäristön jakaminen kolmeen pääosaan voi osaltaan helpottaa suuren kokonaisuuden hallintaa ja varmistaa, että mahdollisimman moni tietokantajärjestelmän toimivuuden kannalta olennainen osa-alue tulee huomioitua. Kolmijaon kautta voidaan käynnistää päätöksenteko käytettävästä laitteistosta, tietokannan hallintajärjestelmästä, sekä tarvittavista käyttöoikeuksista. Tarkastelu tarjoaa myös luontevan käynnistysimpulssin informaatiotason suunnittelulle.

Tietokantaan tallennettavan tiedon koostumuksen selvittäminen ja tietokannan taulurakenteista päättäminen etenee tästä tutkimuksesta saadun kokemuksen perusteella syklisesti. Käsitteiden, ominaisuuksien ja suhteiden selvittämisen työmäärää lisää huomattavasti jos tarkkailija joutuu toimimaan analysoitavan järjestelmän ulkopuolella ns. ”passiivisen tarkkailijan” roolissa. Analyysin sujuvaa läpivientiä varten tarvitaan konkreettista tietoa järjestelmäympäristön sisältä, esimerkiksi analysoitavan ympäristön hyvin tunteva asiantuntijatiimi, joka yhdessä analysoijan kanssa kartoittaa tarvittavat käsitteet ja ominaisuudet. Suorittamani käsiteanalyysin perusteella suurin ongelma ei niinkään ole käsitteiden selvittämisessä, vaan pikemminkin oleellisten ominaisuuksien ja käsitteiden välisten suhteiden hahmottamisessa. Käsiteanalyysin iteraatiokierrosten määrään vaikuttaa analysoijan kokemus käsiteanalyysistä ja tietokannan luomisprosessista; on todennäköistä, että useita tietokantoja tällä tavalla luonut henkilö osaa kiinnittää analysointi- ja mallinnusvaiheessa huomiota tarvittaviin ominaisuuksiin kokematonta henkilöä paremmin.

Kuvaustapana ER-kaavio on riittävän joustava ja tukee iteratiivisia analysointi-mallinnus -kierroksia. Luomassani ER-kaaviossa oli esitetty tietoa käsitteiden lisäksi käsitteiden välisten suhteiden kardinaliteetista ja suhteen tyypistä. Yleisenä järjestelmän kuvauksena tiedot voivat olla hyödyllisiä, mutta en koe niiden tuovat taulurakenteisiin suuntavassa suunnitteluprosessissa merkittävää lisäarvoa. Käyttämäni metodi, jossa ER-kaavion informaatiota täydennettiin käsitteiden ominaisuuksia kuvaavilla taulukoilla, tarjoaa tarvittaessa saman informaation ilman sitä vaaraa, että ER-kaavio on liian täynnä informaatiota. Mielestäni järkevin tapa mallinnuksessa on vastaavissa yhteyksissä käyttää

ER-kaaviossa ainoastaan käsitteitä ilmaisevia suorakaiteita ja käsitteiden välisten suhteiden luonteen ilmaisevia yksi-yhteen, yksi-moneen tai moni-moneen –viivoja. Kaiken muun lisäinformaation, esimerkiksi relaatioiden arvioidut monikkomäärät, voi kirjata tarvittaessa erilliselle käsitekuvauslomakkeelle.

Käsiteanalyysin perusteella suoritettu relaatioiden muodostaminen ja relaatioiden normalisointi on periaatteessa erittäin suoraviivainen operaatio. Kokemattomalla suunnittelijalla relaatioiden muodostaminen, funktionaalisten riippuvuuksien hahmottaminen, avainattribuuttien valinta ja relaatioiden normalisointi vaatii todennäköisesti useita uudelleensuunnittelukierroksia. Kokemuksen karttuessa suunnittelija todennäköisesti oppii normalisoimaan taulurakenteita yhä aikaisemmassa vaiheessa ja näin ollen vähentää syklisyyttä suunnittelussaan.

Tietokantaan luotavien taulurakenteiden suunnittelussa on mielenkiintoista huomata avainattribuuttien valinnan vaikutus automaattisesti luotavaan indeksointiin. Jos relaatioiden suunnittelun yhteydessä käytetään monesta attribuutista koostuvan pääavaimen sijaan yksittäistä surrogaattiavainta, niin automaattisesti luotava indeksi perustuu ainoastaan tähän yksittäiseen attribuuttiin. Monisarakkeisen pääavaimen omaavan taulun primääri-indeksi on siis automaatti-indeksoinnin jälkeen paksumpi, kuin yksittäistä saraketta pääavaimenaan käyttävän taulun primääri-indeksi. Tietokantahakujen predikaatit löytyvät siis todennäköisesti luonnollisia attribuutteja pääavaimenaan käyttävän taulun indeksistä ja näin ollen haut ovat nopeita. Surrogaattiavainta pääavaimenaan käyttävän taulun tapauksessa automaattisesti luotava indeksointi on ohuempi, joka puolestaan voi johtaa huonompiin vasteaikoihin tietokantahauissa. Johtopäätöksenä on se, että taulurakenteita ja avainattribuutteja mietittäessä kannattaa suosia moniosaisia pääavaimia. Moniosaisen pääavaimen perusteella luotava automaatti-indeksointi on jo valmiiksi hyvällä tasolla, joka johtaa puutteellista indeksointia parempiin vasteaikoihin tietokantahauissa.

Indeksoinnin paksuuden vaikutusta hakuaikoihin mitattiin hyvinvointikyselyn vastauksia koskevilla hauilla. Tuloksista kävi selkeästi ilmi, mikä on tarpeeksi paksun indeksin vaikutus tietokantahakujen vasteaikoihin. Hovi ym. (2005) antamissa esimerkeissä indeksejä hyödyntävien hakujen nopeus on noin sadasosa indeksoimattomien hakujen kestoista. Mittausteni perusteella indeksoidut haut ovat noin 20-100 kertaa indeksoimattomia nopeampia. Luvussa 8.3 esitettyjen mittaustulosten välinen suhde

laskettiin kaavalla: haun kesto ennen lisäindeksointia / haun kesto lisäindeksoinnin jälkeen. Tulosten välinen suhde on esitetty taulukossa 7.

Taulukko 7. Indeksoidun haun nopeus suhteessa indeksoimattomaan hakuun.

	1. haku	2. haku	3. haku
Ennen lisäindeksointia	0:22.4545s	0:05.0964s	0:05.2400s
Lisäindeksoinnin jälkeen	0:01.1364s	0:00.0446s	0:00.1836s
Suhde	~19,8	~114,3	~28,5

Hakutuloksista lasketut rivikohtaiset hakuajat eivät vastaa Hovi ym. (2005: 216) asettamia pikaennusteen laskennan viitearvoja. Ero voidaan selittää nopeutuneilla prosessoreilla ja keskusmuisteilla, sekä kehittyneemmillä välimuisteilla. Olen verrannut Hovi ym. (2005: 219-220) käyttämiä laitteisto-oletuksia käyttämäni palvelimen arvoihin taulukossa 8. Tarkemmat tiedot käytetystä laitteistokokoonpanosta ovat liitteissä (LIITE 8. TIETOKANTAPALVELIMEN LAITTEISTOKOKOONPANO).

Taulukko 8. Hovi ym. laitteistoarvojen vertaus käytetyn laitteiston suorituskykyyn.

	Hovi ym. laitearvot	Käytetyt laitteistoarvot
Suoritin	käytössä useita, noin 100MIPS/suoritin	käytössä 4-ytiminen suoritin, noin 4800MIPS/ydin
Kiintolevy	10 000 rpm, hakuaika max 5 millisekuntia	7 200 rpm, hakuaika 7,85 millisekuntia
Kiintolevyjen käyttöaste	noin 15-35%	noin 0-5%

Indeksirivien keskenäisen järjestyksen vaikutusta hakujen nopeuteen tutkittiin luomalla samalle taululle kaksi samat sarakkeet sisältävää indeksiä. Toisessa indeksissä sarakkeet olivat päinvastaisessa järjestyksessä ensimmäiseen verrattuna. MySQL:n hakukäyttäytymistä selittävän EXPLAIN-komennon tulosteesta todettiin todennäköisyyksiin perustuvan MySQL:n hakuoptimoijan käyttävän ensisijaisesti

vähemmän rivejä sisältävää indeksiä. Kun sama haku suoritettiin niin, että MySQL pakotettiin käyttämään enemmän rivejä sisältävää indeksiä, hakunopeudet nousivat noin 7-10 –kertaisiksi.

Vuosiraportin koostavan php-ohjelman suoritusaikoja mitattiin sekä perusindeksoinnilla, että lisätyllä indeksoinnilla. Ilman lisäindeksointia suoritettu vuosiraportin koostaminen kesti noin 95 minuuttia ja lisätyllä indeksoinnilla koostamiseen meni 1 minuutti 15 sekuntia. Ilman lisäindeksointia suoritettu koostaminen vie siis noin 76-kertaisen ajan lisäindeksoituun verrattuna. Tällaisella nopeutumisella on suuri merkitys esimerkiksi sellaisessa tilanteessa, jossa tietokantaan tehdään samalla muita luku- tai kirjoitusoperaatioita. Mikäli raportti koostetaan hiljaisempana vuorokaudenaikana, esimerkiksi yöllä, nopeudella ei ole niin suurta merkitystä.

10. YHTEENVETO

Tässä tutkimuksessa selvitettiin työntekijöiden työhyvinvointia ja –viihtyvyyttä seuraavan informaatiotyökalun osaksi tulevan tietokantajärjestelmän luontiprosessia. Tutkimuksen tavoitteena oli selvittää se tapa, jolla tehokkaasti toimiva tietokanta luodaan. Tarkastelun perustana käytettiin yhdistettyä prosessimallia, jossa oli pyritty huomioimaan kaikki olennaiset vaiheet ja toimenpiteet, jotka suunnittelija käy läpi tietokantaa toteuttaessaan.

Tietokannan toimintaympäristön tarkastelu aloitettiin kolmitasoisen arkkitehtuurin määrittelyllä. Tutkimuksessa todettiin, että valittu tietokannan hallintajärjestelmä, MySQL, vastasi suurimmaksi osaksi teoriassa esitettyjä kuvauksia hallintajärjestelmien koostumuksesta. Arkkitehtuurin selvittämisen todettiin myös antavan suunnittelijalle hyvän yleiskuvan käytettävästä käyttö- ja tiedostojärjestelmästä, tietokannan hallintajärjestelmästä ja tarvittavista käyttäjäoikeuksista.

Varsinainen tietokantaan tallennettavien tietojen kerääminen ja organisointi aloitettiin informaatiotason suunnittelulla, joka koostui käyttäjäympäristön analysoinnista ja mallintamisesta. Analysointia varten tarvittavia tietoja kerättiin Oululaisten pankkikonttorien esimiehiltä puhelinsoitoilla, sähköpostikyselyillä ja käymällä paikan päällä. Kerättyjen tietojen perusteella mallinnettiin pankkikonttoria kuvaava käyttäjäympäristö ER-kaaviolla. Riittävän tarkalla tasolla olevan kuvauksen luomisen todettiin vaativan useita iteraatiokierroksia ja suunnittelusyklien määrän arvioitiin olevan riippuvainen suunnittelijan kokemuksesta.

Suunnitteluprosessin toinen päävaihe, fyysisen tason suunnittelu, aloitettiin kääntämällä ER-kaavio valitun tietokannan hallintajärjestelmän muotoon. Käännöksen todettiin olevan suoraviivainen toimenpide, jossa syklisyyttä aiheuttivat eniten normalisoinnissa suoritetut toimenpiteet. Normalisoinnin yhteydessä relaatioiden attribuutteja jaettiin normalisointisääntöjen mukaisesti pienempiin osakokonaisuuksiin. Normalisoinnin yhteydessä joitakin relaatioita normalisoitiin, denormalisoitiin ja normalisoitiin uudestaan. Syklisyyden syynä oli tietorakenteiden selkiytyminen ja olennaisten suhteiden hahmottuminen.

Indeksoinnin vaikutusta tietokannan tauluille suoritettavien kyselyiden nopeuteen selvitettiin esimerkkihauilla. Indeksoinnin selvityksen yhteydessä selvitettiin myös indeksisarakkeiden keskenäisen järjestyksen ja indeksin sisältämän rivimäärän vaikutus hakujen nopeuksiin. Tuloksista pääteltiin moniosaisen pääavaimen sisältävien taulujen indeksoinnin olevan automaattisesti hyvälla tasolla, indeksoinnin parantavan hakunopeuksia huomattavasti, ja indeksisarakkeiden keskenäisen järjestyksen vaikuttavan hakujen nopeuteen.

LÄHTEET

Bannon, Ryan, Alvin Chin, Faryaaz Kassam & Andrew Roszko (2002). *MySQL Conceptual Architecture*.

Chen, Peter Pin-Shan (1976). *The Entity-Relationship Model-Toward a Unified View of Data*. ACM Transactions on Database Systems. Vol. 1. No. 1. maaliskuu 1976. sivut 9-36.

Codd, E. F. (1970). *A relational model for large shared databanks*. Communications of the ACM. Vol. 13. No. 6. kesäkuu 1970. sivut 377-387.

Codd, E. F. (1971). *Further Normalization of the Data Base Relational Model*. IBM Research Report RJ909, San Jose, California.

Date, C. J. (1982), *An Introduction to Database Systems*. Addison-Wesley. 574s. ISBN 0-201-14471-9.

Elmasri, Ramez & Shamkant B. Navathe (2000). *Fundaments of database systems*. Yhdysvallat: Addison-Wesley. 955s. ISBN 0-201-54263-3.

Hernandez, Michael J. (2000). *Tietokannat – suunnittelu ja toteutus*. Jyväskylä: Gummerus Kirjapaino Oy. 442s. ISBN 951-826-137-7

Hirsjärvi, Sirkka, Pirkko Remes ja Paula Sajavaara (1997). *Tutki ja kirjoita*. Helsinki: Kustannusosakeyhtiö Tammi. 436s. ISBN 951-26-5113-0

Hovi, Ari, Jouni Huotari & Tapio Lahdenmäki (2005). *Tietokantojen suunnittelu & indeksointi*. Jyväskylä: Docendo Finland Oy. 356s. ISBN 951-846-262-3.

Levene, Mark & George Luizou (1999). *A guided tour of relational databases and beyond*. Englanti: Springer-Verlag London Limited. 625s. ISBN 1-85233-008-2.

Lewis, Philip M., Arthur Bernstein & Michael Kifer (2002). *Databases and transaction processing: an application-oriented approach*. Yhdysvallat: Addison-Wesley. 1014s. ISBN 0-201-70872-8.

McGraw-Hill/Osborne (2004). *A Technical Tour of MySQL*. http://www.devshed.com/c/a/MySQL/A-Technical-Tour-of-MySQL/

MySQL AB (2007a). *MySQL Reference Manual*. http://downloads.mysql.com/docs/refman-5.0-en.a4.pdf

MySQL AB (2007b). *MySQL 5.0 Pluggable Storage Engine Architecture*. http://www.mysql.com/common/images/PSEA_diagram.jpg

MySQL AB (2007c). *Connection pooling with MySQL Connector/J*. http://dev.mysql.com/tech-resources/articles/connection_pooling_with_connectorj.html

NTFS.com (2007). *NTFS.com NTFS vs FAT32 FAT16 FAT.Comparing.Performance*. http://www.ntfs.com/ntfs_vs_fat.htm.

Pratt, Philip J. & Joseph J. Adamski (1987). *Database Systems: Management and Design*. Boston: Boyd & Fraser Publishing Company. 757s. ISBN 0-87835-227-9

Ricardo, Catherine M. (1990). *Database systems: principles, design and implementation*. New York: Macmillan Publishing Company. 576s. ISBN 0-02-399665-X.

Ullman, Jeffrey D. & Jennifer Widom (1997). *A First Course in Database Systems*. Upper Saddle River, New Jersey: Prentice Hall. 470s. ISBN 0-13-861337-0.

LIITE 1. KYSELYLOMAKE

1. Mitä käsitteitä tulevan tietokantajärjestelmän ympäristössä on? (esim. Toimisto, johtaja, työntekijä, jne.) Listaa vain olennaiset asiat, jotka liittyvät lopulliseen järjestelmään. Kirjaa myös kaikki käsitteistä virallisessa käytössä olevat synonyymit.
2. Minkälaisia ominaisuuksia listatuilla käsitteillä on? Listaa vain oleelliset ominaisuudet, eli tiedot, joita tietokantaan halutaan tallentaa. (esim. Työntekijän sosiaaliturvatunnus, ikä, palkka, työntekijänumero, toimiston tunnus, jne.)
3. Lisää edellä listattuihin ominaisuuksiin lyhyt kuvaus, sekä mahdolliset rajoitukset. (esim. Ikä = työntekijän ikä, mahdolliset arvot 10-100 vuotta, toimiston tunnus = toimistokohtainen, yksilöllinen numero- ja kirjainsarja, pituus 10 merkkiä, jne.)
4. Mikäli ominaisuus määräytyy jonkin toisen ominaisuuden perusteella, kirjaa ylös k.o. ominaisuuden määräytymisperuste. (esim. Palkka määräytyy palvelusvuosien perusteella siten, että jos työntekijän työvuodet ovat alle 5, niin palkka on 2500€/kk, johtajan titteli on vuoropäälikkö, joten alaisten lukumäärä on 5-20, toimiston koko on alle 200 neliömetriä, joten työpisteiden määrä on maksimissaan 20, jne.)
5. Listaa mahdolliset tietoturvarajoitukset. (esim. Kenellä on oikeus tarkastella työntekijän palkkatietoja tai sosiaaliturvanumeroa)
6. Kirjaa ylös käsitteiden väliset suhteet. (esim. Yhdessä toimistossa on yksi toimistopäälikkö, yksi toimistopäälikkö vastaa 2-20 työntekijästä, jne.) Mikäli suhteiden lukumääristä tai rajoituksista on arvio, niin listaa ne. (esim. Yhdessä toimistossa työskentelee 2-50 työntekijää ja 1-3 johtajaa, yhdellä työntekijällä voi olla vain yksi pomo, yksi työntekijä voi olla kirjoilla vain yhdessä toimistossa, jne.)
7. Listaa ominaisuuksien väliset riippuvuudet. (esim. ”jos tiedämme toimiston koodin, tiedämme myös sen koon” tai ”jos tiedämme työntekijän sosiaaliturvatunnuksen, tiedämme hänen kuukausipalkkansa” jne.)
8. Mitkä ovat tallennettavien tietojen muuttuvia osia, jotka ovat siis useimmiten päivitysten kohteena? (katso kysymys numero 2)
9. Kuinka usein näitä tietoja päivitetään ja kuka suorittaa päivitykset? (esim. Johtaja päivittää palkkatiedot kerran kuussa, kaikkien henkilöiden ikä päivitetään automaattisesti kerran vuodessa, työntekijä päivittää viihtyvyyskyselyn vastaukset kerran viikossa, jne.)
10. Minkälaisia raportteja järjestelmästä halutaan? (esim. Johtaja haluaa alaistensa viihtyvyydestä koostetun raportin kerran viikossa, divisioonan johtaja haluaa toimistokohtaisen raportin kerran kuussa, jne.)
11. Mikä on haluttujen raporttien sisältö? (esim. kerran viikossa tulostettavaan raporttiin halutaan kaikki työntekijän tiedot ja vastaukset yksittäisiin kysymyksiin, kerran kuussa tulostettavaan raporttiin halutaan kysymyskohtaiset numeraaliset keskiarvot, jne.)

LIITE 2. ESIMERKKIRAPORTTI

Raportti: osasto X

Kysymys #	Keskiarvo	Mediaani	Korkein	Alin	Vastauksia
1	4,1	3	9	1	10
2	4,5	3,5	10	2	10
3	5,1	4,5	10	2	10
4	2,6	2,5	5	1	10
5	4,9	6	8	1	10
6	6	6	10	1	10
7	4,1	4	8	2	10
8	5,1	4,5	9	2	10
9	4,7	5	9	1	10
10	5,9	6,5	9	1	10

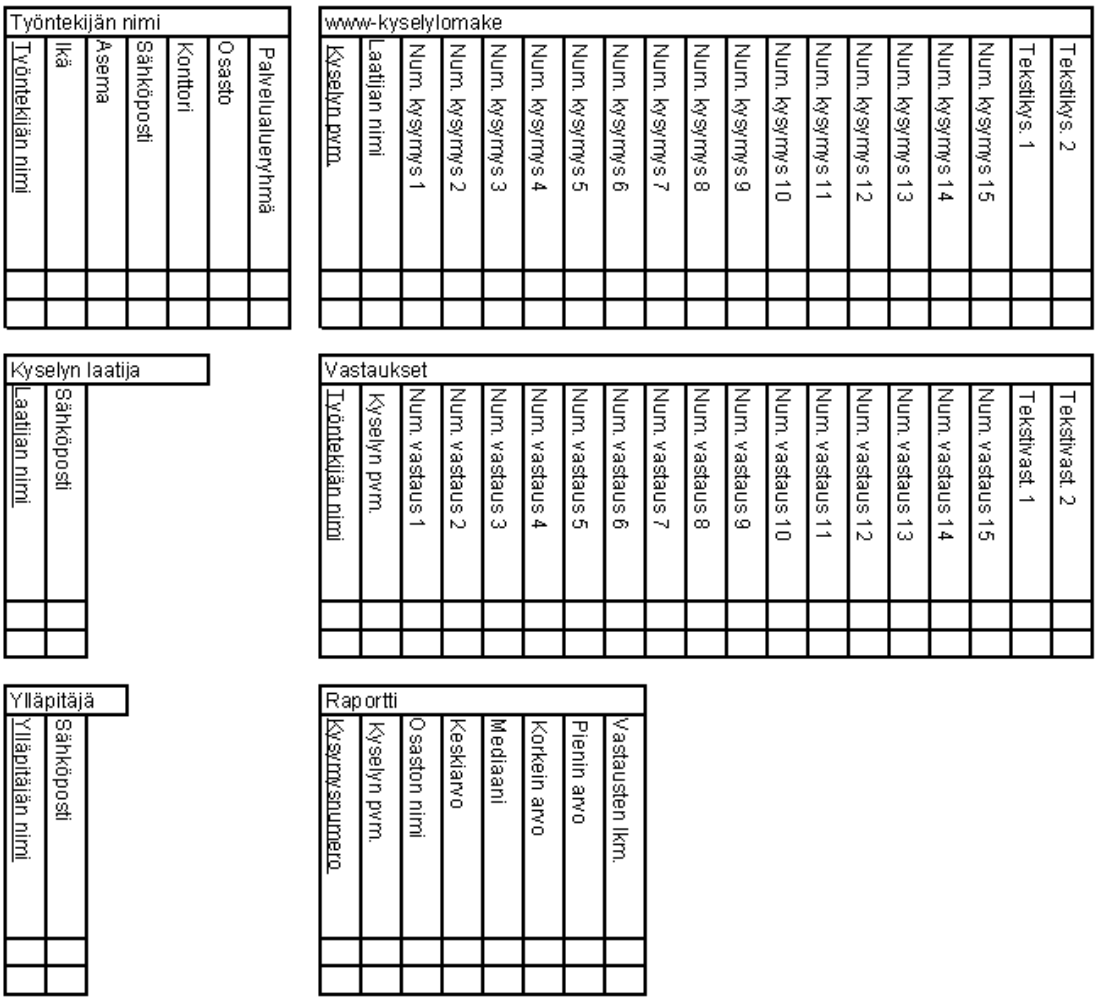
Raportti: osasto Y

Kysymys #	Keskiarvo	Mediaani	Korkein	Alin	Vastauksia
1	5,8	6	9	3	10
2	6,4	7	9	3	10
3	4,7	4,5	10	1	10
4	3,7	3	7	1	10
5	3,9	4,5	7	1	10
6	4,8	4,5	10	1	10
7	4,9	4,5	8	2	10
8	4,3	3,5	9	1	10
9	5	5	9	1	10
10	4,4	4	7	1	10

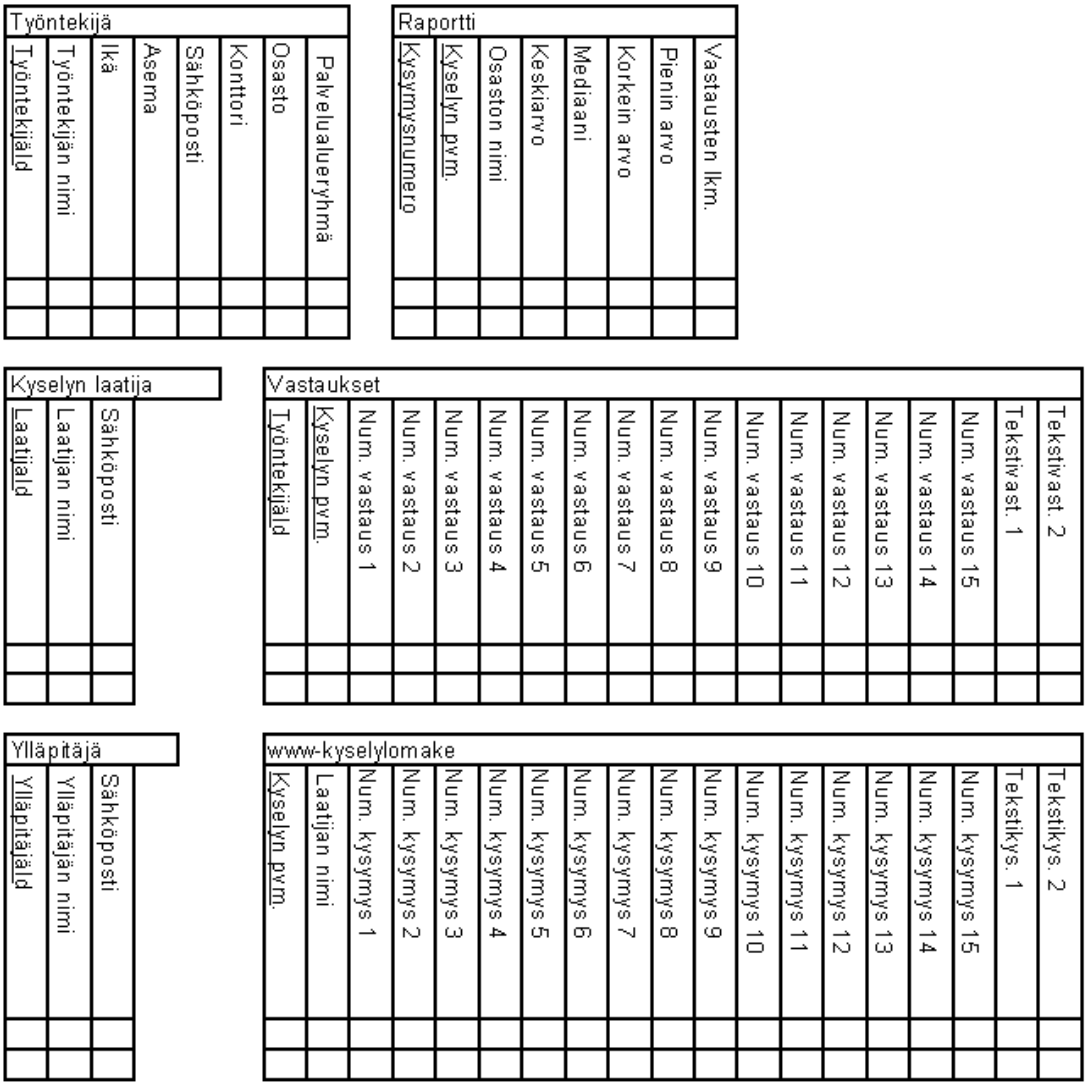
LIITE 3. KÄSITEMÄÄRITTELYLOMAKE

KÄSITE: TYÖNTEKIJÄ					
ATTRIBUUTIN NIMI	KUVAUS	RAJOITUKSET	MÄÄRÄYTY-MISPERUSTE	LUKU-OIKEUDET	KIRJOITUS-OIKEUDET
Työntekijän nimi	Työntekijän virallinen nimi	Maksimissaan 30 merkkiä. Ei saa olla Null	Virallinen todistus (ajokortti, passi, ym.)	Työntekijä, ylläpitäjä	Ylläpitäjä
Ikä	Työntekijän ikä	Arvo täytyy olla 15-100. Ei saa olla Null.	Virallinen todistus (ajokortti, passi, ym.)	Työntekijä, ylläpitäjä	Ylläpitäjä
Asema	Työntekijän asema konttorissa	Ei saa olla Null.	Työsopimus tai muu vastaava tosite	Työntekijä, ylläpitäjä	Ylläpitäjä
Sähköposti	Työsähköposti-osoite	Ei saa olla Null.	Etunimi.sukunimi@konttori.fi	Työntekijä, ylläpitäjä	Työntekijä, ylläpitäjä
Konttori	Pankkikonttori, jossa työntekijä työskentelee / on kirjoilla	Virallinen konttorin nimi. Ei saa olla Null.	Työsopimus tai muu vastaava tosite	Työntekijä, ylläpitäjä	Ylläpitäjä
Osasto	Osasto, jossa työntekijä työskentelee / on kirjoilla	Mahdolliset arvot: Yksityis-asiakkaat, yritysasiakkaat. Ei saa olla Null.	Työsopimus tai muu vastaava tosite	Työntekijä, ylläpitäjä	Ylläpitäjä
Palvelualue-ryhmä	Palvelualue-ryhmä, jossa työntekijä työskentelee / on kirjoilla	Mahdolliset arvot: 1, 2, 3, 4, YA (YritysAsiak-kaat). Ei saa olla Null.	Työsopimus tai muu vastaava tosite	Työntekijä, ylläpitäjä	Ylläpitäjä

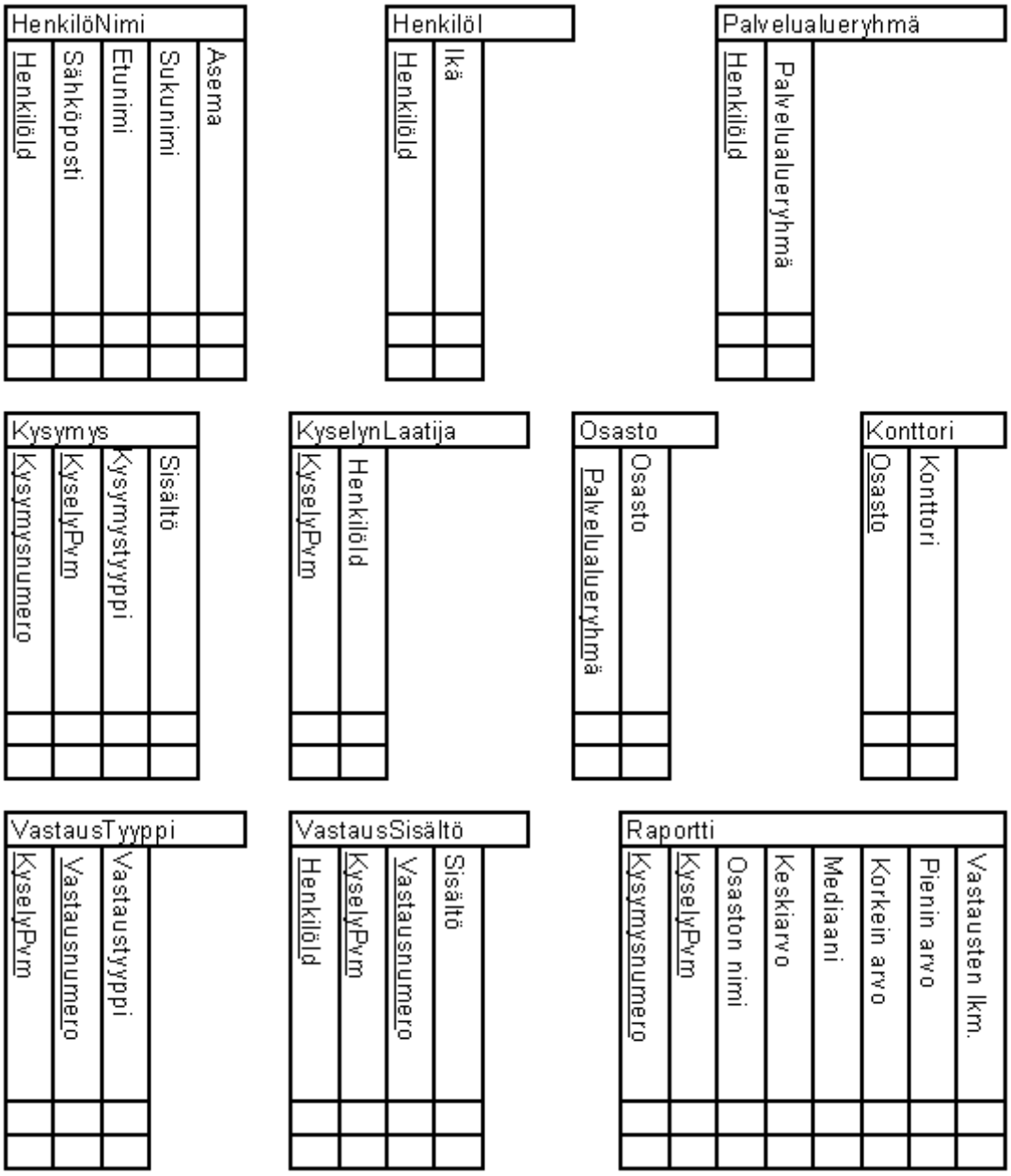
LIITE 4. KÄSITTEIDEN RELAATIOMUODOT



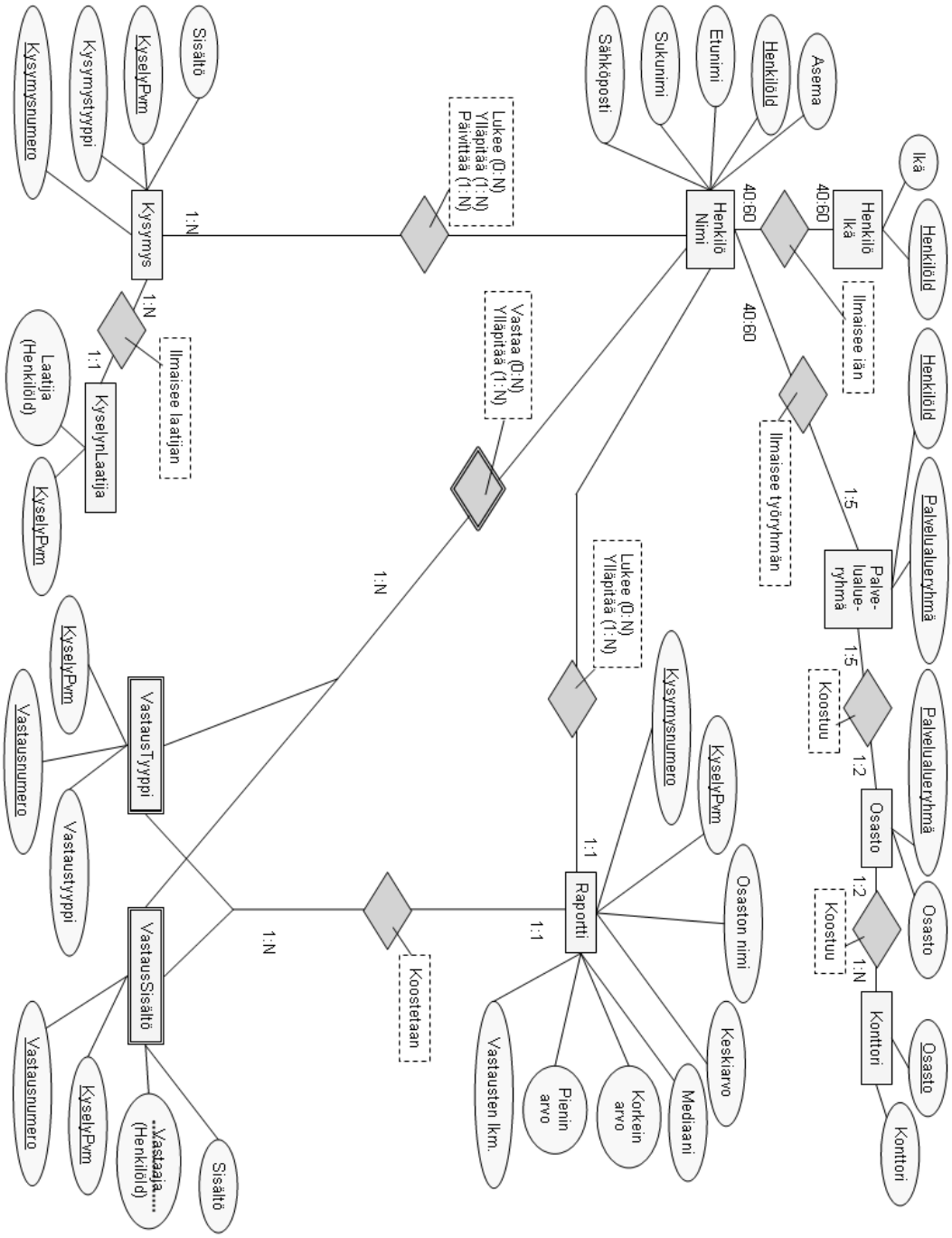
LIITE 5. RELAATIOIDEN AVAINATTRIBUUTIT



LIITE 6. NORMALISOIDUT RELAATIOT



LIITE 7. ER-KAAVIO NORMALISOIDUISTA RELAATIOISTA



LIITE 8. TIETOKANTAPALVELIMEN LAITTEISTOKOKOONPANO

- Kiintolevy: Seagate ST316002 AS3, SATA-liitäntä, 8 megatavun välimuisti, hakuajat 7,75ms, 7,96ms ja 7,83ms => keskiarvo on 7,85ms. Testattu Roadkil's Disk Speed –ohjelmalla[1].
- Prosessori, jossa neljä ydintä. Malli Intel Core2Quad 6600. Kellotaajuus 2,4GHz. ~4800 MIPS (selvitetty Ubuntu Linuxin konsolikäskyllä 'cat /proc/cpuinfo').
- DDR2-tyyppistä keskusmuistia 4 gigatavua. Nopeus 800MHz.

[1] http://roadkil.net/DskSpeed.html

LIITE 9. VUOSIRAPORTIN PHP-SKRIPTI

```
<?php
$link = mysql_connect('localhost', '***', '***');
if (!$link) {die('Could not connect: ' . mysql_error());}
echo 'Connected successfully' . '<br/>';

$db = mysql_select_db("Gradu", $link);
if (!$db) {die('Could not select database: ' . mysql_error());}
echo 'DB selected successfully' . '<br/>';

$time_start = microtime(true);
$result = mysql_query("SELECT KyselyPvm, Vastausnumero FROM VastausTyyppi
WHERE Vastaustyyppi = 'numero'", $link);

print "<table border=1><tr>
    <td><strong>Kyselypäivämäärä</strong></td>
    <td><strong>Kysymysnumero</strong></td>
    <td><strong>Minimi</strong></td>
    <td><strong>Maksimi</strong></td>
    <td><strong>Keskiarvo</strong></td>
    </tr>\n";

$rivimaara = 0;
while ($row = mysql_fetch_array($result)) {
    $Pvm = $row['KyselyPvm'];
    $Vastausnumero = $row['Vastausnumero'];
    $result2 = mysql_query("SELECT KyselyPvm, Vastausnumero, MIN(Sisalto) as
Minimi, MAX(Sisalto) as Maksimi, AVG(Sisalto) as Keskiarvo FROM VastausSisalto
WHERE KyselyPvm = '$Pvm' AND Vastausnumero = '$Vastausnumero' GROUP BY
KyselyPvm", $link);
        while ($row2 = mysql_fetch_array($result2)) {
```

```
                echo "<tr><td>" . $row2['KyselyPvm'] . "</td><td>" .
$row2['Vastausnumero'] . "</td><td>" . $row2['Minimi'] . "</td><td>" . $row2['Maksimi'] .
"</td><td>" . $row2['Keskiarvo'] . "</td></tr>\n";
        $rivimaara++;
        }
}

print "</table>";

$time_end = microtime(true);
$time = $time_end - $time_start;
echo "Query took ". "<font color=red>" . $time . "</font>" . " seconds<br />\n";
echo $rivimaara . " Rows";

mysql_close($link);
?>
```