Lappeenrannan teknillinen yliopisto
Teknistaloudellinen tiedekunta
Tietotekniikan tutkinto-ohjelma

Kandidaatintyö

Timo Ruokolainen

Relaatiotietokannan tiedonhallintajärjestelmän valintakriteerit

Työn tarkastaja ja ohjaaja: TkT Erja Mustonen-Ollila

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
Teknistaloudellinen tiedekunta
Tietotekniikan tutkinto-ohjelma

Timo Ruokolainen

Relaatiotietokannan tiedonhallintajärjestelmän valintakriteerit

Kandidaatintyö

2014

42 sivua, 7 kuvaa, 2 taulukkoa

Työn tarkastaja ja ohjaaja: TkT Erja Mustonen-Ollila

Hakusanat: tiedonhallintajärjestelmä, relaatiotietokanta, valintakriteeri

Tässä työssä käsitellään lähinnä relaatiomallia hyödyntäviä tiedonhallintajärjestelmiä. Tiedonhallintajärjestelmä hallitsee yleisesti tietokannan luontia, käyttöä ja muutoksia ja relaatiomallia käyttävät tiedonhallintajärjestelmät ovat jo 1970 -luvulta lähtien olleet hallitseva trendi tietokantamarkkinoilla. Työssä otetaan huomioon neljä eri tiedonhallintajärjestelmä-tyyppiä, jotka ovat keskitetyt, hajautetut, tietovarasto ja operatiiviset tiedonhallintajärjestelmät. Työssä selvitetään, miten näitä tiedonhallintajärjestelmiä voi verrata ja mitkä valintakriteerit vaikuttavat niiden valintaan.

ABSTRACT

Lappeenranta University of Technology
Faculty of Technology and Management
Computer Science Degree Program

Timo Ruokolainen

Selection criteria for a relational database management system

Bachelor of Science Thesis

2014

42 pages, 7 figures, 2 tables

Supervisor and examiner: Associate professor Erja Mustonen-Ollila

Keywords: database management system, relational database, selection criteria

This Bachelor of Science thesis covers database management systems that are based on the relational data model. In generally, relational database management system controls the creation, usage and changes of a relational database. There are many different types of databases and database management systems available, but from 1970s the relational database management systems have been the dominant trend on the database market. This thesis examines four types of database management systems that is centralized, and distributed database management systems, and data warehouse and operative database management systems. The goal of the thesis is to find out how to compare these four previously mentioned database management systems, and what are the selection criteria which can affect the choice of a database management system.

SISÄLLYSLUETTELO

LYHENNELUETTELO

ACID	Tietokantatapahtumien neljä tärkeää ominaisuutta. Lyhenne tulee sanoista: atomicity, consistency, isolation ja durability
ANSI-SPARC-arkkitehtuuri	THJ-arkkitehtuuri. Koostuu kolmesta eri tasosta: ulkoinen taso (external level), käsitteellinen taso (conseptual level) ja sisäinen taso (internal level)
Arkkitehtuuri	Määrittelee THJ:n rakenteen
Centralized database	Keskitetty tietokanta. Itse tietokanta sijaitsee yhdessä paikassa
DAR	Data access requirements. Kyselyihin liittyvät tiedonhakuvaatimukset
Data and Metadata	Sisältää varsinaisen tiedon ja kuvaustiedon tarkoittaen kuvausta tiedon rakenteesta
Data mart	Tietovaraston osa. Data martit tarjoavat käyttäjille helpomman pääsyn tärkeään tietoon ja säilytyspaikan kuvaustiedolle
DDBS	Distributed database system eli hajautettu tiedonhallintajärjestelmä
Discipline-specific database	Tietokanta, joka keskittyy tietyn aihepiirin tietoon
Distributed database	Hajautettu tietokanta. Tieto on jaettu usean paikan ja tietokannan hallintajärjestelmän kesken

ETL	Extraction-transformation-loading. Työkaluja tiedon syöttämistä varten tietovarastoon
General-purpose database	Tietokanta, joka sisältää useaa eri tyyppistä tietoa, joita hyödynnetään eri tarkoituksiin
Hakemisto	Aputietorakenne, joka nopeuttaa tiedonsaantia
MDBS	Multidatabase system. Hajautettu arkkitehtuuri, johon kuuluu usea tietokannan hallintajärjestelmä
ODBC	Open Database Connectivity. Standardi avoin rajapinta tietokannoille
OLAP	Online analytical processing. Käytetetään tietovaraston tiedon analysointiin
OLTP	On Line Transaction Processing. Tietojärjestelmä, joka käytetään operatiivisissä tietokannoissa
Oliomalli	Tietomalli, jossa tieto kuvataan olioina ja luokkina
Olio-relaationaalinen malli	Tietomalli, joka laajentaa relaatiomallia monilla olio konsepteilla
Operational database	Tietokanta, joka tukee jokapäiväistä liiketoimintaa
Operators	SQL-kyselykielen operaatioita tietokannan määrittelyyn ja manipulointiin
P2P	Peer-to-Peer eli vertaisverkko

Query Processor	Kyselynkäsittelijä, optimoi kyselyiden ja päivitysten suorittamista
QoE	Quality of Experience. Käyttäjätyytyväisyyden mitta, liittyen tiettyyn tuotteeseen tai palveluun
Relational model	Relaatiomalli. Tietomalli, joka kuvaa tiedon tauluina eli relaatioina
Relaatiotietokanta	Tietokanta, jossa tieto näkyy käyttäjälle tauluina
SQL	Structured Query Language on tietokantakyselyjen tekoon tarkoitettu ohjelmointikieli, jolla voi manipuloida tauluissa sijaitsevaa tietoa
Storage Manager	Muistinhallitsija , hoitaa tiedonsiirron keskusmuistin ja tietokannan välillä
Tieto	Jotain, mikä on esitetty tietokannassa kirjaimina, numeroina, kuvina jne
Tietovarasto	Päätöksentekoa tukeva tietokanta. Sisältää tietoa operatiivisista tietokannoista ja muista tietolähteistä, kuten tiedostoista
THJ	Tiedonhallintajärjestelmä. Luo tietokannan ja ylläpitää sitä. Suojaa käyttäjää laitteistotason yksityiskohdilta
Transaction manager	Tapahtumakäsittelijä, ohjaa samanaikaisten käyttäjien operaatioiden lomittumista (samanaikaisuudenhallinta) ja pitää huolen, että päivitykset ovat atoomisia

1 JOHDANTO

Tämä kandidaatintyö on kirjallisuustutkimus, jossa kirjallisuusmateriaali on kerätty tiedonhallintajärjestelmiin liittyvistä julkaisuista ja kirjoista. Työssä keskitytään relaatiotietokantojen tiedonhallintajärjestelmien valintakriteerien esittämiseen asiakkaan näkökulmasta. Seuraavissa alaluvuissa esitellään aiheen tausta, tavoitteet ja rajaukset ja työn rakenne.

1.1 Tausta

Tieto on organisaation yksi tärkeimmistä omaisuuksista. Tieto liittyen asiakkaisiin, työntekijöihin, tilauksiin ja tositteisiin ovat kaikki tärkeitä ja mahdollistavat yrityksen olemassaolon. Seuraamalla tärkeimpiä kasvu- ja suorituskykymittareita yritys voi taata taktisen ja strategisen päätöksenteon onnistumisen. Tästä syystä organisaation hallussa olevaa tietoa ei tule käsitellä huolimattomasti. Tämän sukupolven tiedonhallintajärjestelmät ovat helppoja käyttää (Coronel ym., 2011, s. 11) ja tämän vuoksi useat tietokoneita hallitsevat yritykset voivat yliarvioida taitonsa tietokantaa suunniteltaessa (Coronel ym., 2011, s. 11).

Tiedonhallintajärjestelmä tarkoittaa ohjelmistoa, joka on fyysisen tietokannan ja käyttäjän välissä. Kaikki käyttäjältä tulevat käskyt tai pyynnöt menevät tiedonhallintajärjestelmän kautta. Tiedonhallintajärjestelmän yksi tärkeimmistä tarkoituksista on suojata käyttäjää laitteistotason yksityiskohdilta. Ohjelmointikieli toimii samalla periaatteella. Se suojaa ohjelmoijaa laitteistotason konekieleltä. Tiedonhallintajärjestelmä tarjoaa käyttäjälle näkymän tietokannasta, joka on jokseenkin laitteistotason yläpuolella (Date, 2001, s. 7).

Tietokantatyyppejä ja tiedonhallintajärjestelmiä on tarjolla useita. Työssä käsitelläänkin lähinnä relaatiomallia hyödyntäviä tiedonhallintajärjestelmiä. Daten (2001) mukaan melkein kaikki tietokantatuotteet 1970-luvulta lähtien kuuluvat relaatiomalliseen lähestymistapaan ja ovat edelleen hallitseva trendi markkinoilla.

1.2 Tavoitteet ja rajaukset

Tässä työssä käsitellään relaatiotietokannan tiedonhallintajärjestelmän valintakriteereihin vaikuttavia seikkoja asiakkaan näkökulmasta. Työssä keskitytään relaatiomalliin perustuviin tiedonhallintajärjestelmiin (Date, 2001, ss. 21-22) ja niiden valintakriteereihin, yritysten eri mahdollisuuksiin tiedonhallintajärjestelmän valinnassa ja tiedonhalintajärjestelmien vertailuun. Oliotietokantojen tiedonhallintajärjestelmät ja olio-relaationaaliset tiedonhallintajärjestelmät jätetään aiheen ulkopuolle. Työssä ei myöskään esitellä tarkemmin yksittäisten tietokantatoimittajien tuotteita, vaan otetaan huomioon olemassa olevat teknologiat.

1.3 Työn rakenne

Luvussa kaksi esitetään kirjallisuuskatsaus aihepiirin sisältä. Siinä kerrotaan relaatiomallista ja relaatiotietokannan hallintajärjestelmistä. Teoriaosuuden jälkeen luvussa määritellään tutkimusongelmat ja kerrotaan miten tutkimusongelmia lähdetään ratkaisemaan.

Kolmannessa luvussa esitellään tietokantatyypit ja tiedonhallintajärjestelmien (THJ)-tyypit: keskitetyt THJ:t, hajautetut THJ:t, tietovarastot ja operatiiviset THJ:t.

Neljännessä luvussa esitellään valitut valintakriteerit ja verrataan THJ-tyyppejä, jotka esiteltiin kolmannessa luvussa.

Viidennessä luvussa esitellään työn johtopäätökset.

2 TEORIAOSUUS

Tässä osuudessa selvitetään työhön liittyvät termit ja kerrotaan tausta tietoa aiheesta. Sekä esitellään työn tutkimusongelmat ja kerrotaan miten tutkimusongelmiin on lähdetty etsimään ratkaisuja.

2.1 Tiedonhallintajärjestelmä

Tietomalli määrittää sen miten tieto kuvataan. Tietomalli on abstrakti, looginen määritelmä tietorakenteista, tiedon operaatioista jne. Nämä yhdessä muodostavat absraktin koneen, jonka kanssa käyttäjät vuorovaikuttavat. Tietomalli on myös pysyvän tiedon malli eli loogisen tietokannan malli, joka kuuluu jollekin organisaatiolle (Date, 2008, s. 38). Relaatiomalli on yksi tietomalleista, jossa tieto näkyy käyttäjälle tauluina eli relaatioina (Codd, 1970, s. 377).

Tiedonhallintajärjestelmä (THJ) (engl. DMS eli database management system) hallitsee tietokannan luontia, käyttöä ja muutoksia. Siihen sisältyy structure query language (sql) -kyselykieli, se hallitsee transaktioita, hoitaa automaattisesti triggerien toiminnan, hallitsee eheyttä jne. THJ on on tehokas työkalu suuren tietomäärän luomiseen ja hallitsemiseen. THJ pitää huolen tiedon säilymisestä ja tietoturvallisuudesta. THJ mahdollistaa suurien tietomäärien tallentamisen ja tarjoaa tietorakenteita, joilla tietoon pääsee käsiksi tehokkaasti. Tallennettu tieto on riippumatonta sitä käyttävistä prosesseista. Toisin kuin tavallisessa tiedostojärjestelmässä, THJ kyky hallita tietoa ei rajoitu vain tiedon kirjoittamiseen ja lukemiseen. Voimakkaan kyselykielen avulla käyttäjä tai ohjelma pystyy hakemaan ja manipuloimaan tietoa. THJ mahdollistaa samanaikaisen tiedon käytön ja tukee tiedon eristämistä siten, että kyselyt näennäisesti suoritetaan yksi kerrallaan. Tietokannan tapahtumat ovat atoomisia, mikä tarkoittaa sitä, että ne suoritetaan joko kokonaan tai ei ollenkaan (Ullman ja Widom, 2002, ss. 1-2).

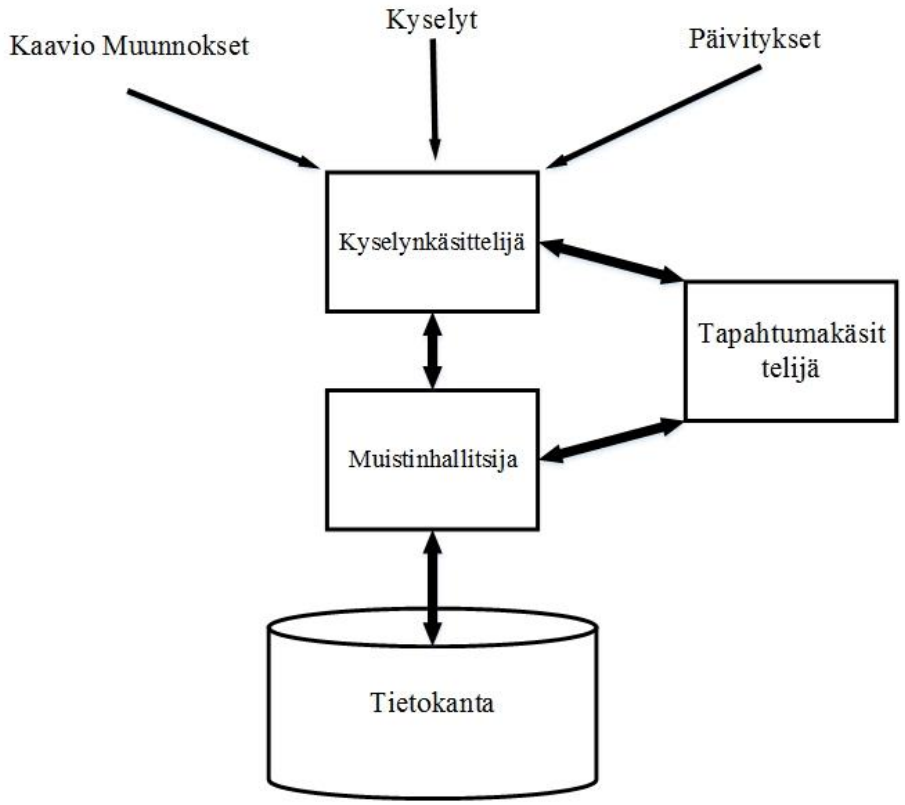
2.1.1 Tiedonhallintajärjestelmän rakenne

Tyypillisellä tiedonhallintajärjestelmällä on seuraavat komponentit (kuva 1) (Ullman ja Widom, 2002, s. 8):

- *Tietokanta (Data and Metadata)* sisältää varsinaisen tiedon ja kuvaustiedon tarkoittaen kuvausta tiedon rakenteesta.
- *Muistinhallitsija (Storage Manager)* hoitaa tiedonsiirron keskusmuistin ja tietokannan välillä.
- *Kyselynkäsittelijä (Query Processor)* optimoi kyselyiden ja päivitysten suorittamista.
- *Tapahtumakäsittelijä (Transaction manager)* ohjaa samanaikaisten käyttäjien operaatioiden lomittumista (samanaikaisuudenhallinta) ja pitää huolen, että päivitykset ovat atoomisia.

Tiedonhallintajärjestelmän syötteisiin (Inputs) kuuluu:

- *Kyselyt* yleisen kyselyliittymän välityksellä, esimerkiksi SQL-kyselyt. Lisäksi sovelluskohtainen liittymä, esimerkiksi lomakkeet.
- *Päivitykset.*
- *Tietokantakaavion muutokset* kuten attribuutin lisääminen tauluun (Ullman ja Widom, 2002, s. 8).



Kuva 1. Tiedonhallintajärjestelmän osat (Ullman ja Widom, 2002, s. 8).

2.2 Relaatiomalli

Relaatiomalli on tietokantamalli, joka kuvaa tietokannan rakenteen. Relaatiotietokantaan kuuluu kolme tasoa: ulkoinen taso, käsitteellinen taso ja sisäinen taso. Ulkoinen taso kuvaa tietokannan käyttäjäryhmän tai sovelluksen näkökulmasta. Käsitteellinen taso on tietokannan looginen taso, joka kuvataan relaatiomallin mukaan. Sisäinen taso määrittää tietokannan fyysisen rakenteen eli kuvaa miten tiedot on tallennettu tietokantaan (Mustonen-Ollila, 2013). Relaatiomallilla kuvataan tieto tauluina. Relaatiomallilla on vain yksi konsepti tiedon mallintamiseen nimeltä relaatio. Relaatio tarkoittaa kaksiulotteista taulua, johon tietokannan tiedot sijoitetaan (Codd, 1970; Garcia-Molina ym., 2002, s. 61). Kuvassa 2 on esimerkki relaatiotietokanta. Kuvasta nähdään miten tieto voidaan esittää tauluina. Relaatiomallin avulla THJ kykenee manipuloimaan tietoa operaatioiden avulla. (Date, 2001, s. 57). Relaatiomalli tukee SQL (Structured Query Language) – ohjelmointikieltä, jolla voi kirjoittaa yksinkertaisia, mutta voimakkaita ohjelmia, joilla

manipuloidaan tauluissa sijaitsevaa tietoa. Lähes kaikki tietokantatuotteet perustuvat relaatiomalliin (Date, 2001, s. 21).

Työntekijä

TYONTEKIJA_ID	TT_NIMI	OSASTO_ID	PALKKA
T1	Kallio	O2	40 000
T2	Vaalte	O1	43 000
...	...	...	...

Osasto

OSASTO_ID	OS_NIMI	BUDJETTI
O1	Myynti	5 milj
O2	ATK	3 milj
...	...	...

Kuva 2. Relaatiotietokannan kaksi taulua.

Kuvassa 2 nähdään, miten taulut muodostuvat. Taulut ovat relaatiotietokannassa perusrakenteita, joihin tiedot sijoitetaan. Ensimmäinen taulu sisältää tietoa työntekijöistä ja toinen taulu osastoista. Taulut on otsikoitu sen perusteella, minkälaista tietoa ne sisältävät. Taulut muodostuvat vaakasuorassa riveistä eli tietueista (Date, 2008, s. 144), pystysuorassa sarakkeista eli attribuuteista, ja yksittäistä alkiota taulussa kutsutaan kentäksi (Relaatiotietokantatermit). Esimerkiksi työntekijä-taulussa yksi rivi sisältää tiedot yhdestä työntekijästä. Relaatiotietokannassa jokaisella sarakkeella on oltava oma nimi, ja nimen täytyy olla uniikki (Whitehorn ja Marklyn, 2007, s. 13, 18-19).

Jokainen taulu kuvaa jotain oikean maailman objektia, jos kuvan 2 taulut olisivat oikeassa tietokannassa, jokainen rivi työntekijä-taulussa vastaisi oikeaa työntekijää. Näillä objekteilla tai tauluilla voi olla myös eri tyyppisiä suhteita toistensa kanssa. Näitä mahdollisia suhteita on neljä: yhden suhde moneen (1:N), yhden suhde yhteen (1:1), monen suhde moneen (N:M) tai ei suhdetta ollenkaan. Näiden suhteiden avulla taulut on mahdollista yhdistää toisiinsa (Whitehorn ja Marklyn, 2007, s. 81).

Avaimet ovat tärkeä osa tauluja ja mahdollistavat yhteydet taulujen välillä. Pääavaimet ovat osa taulujen rakennetta. Relaatiotietokannan jokaisessa taulussa täytyy olla pääavain. Pääavain muodostuu yhdestä tai useammasta attribuutista, ja jokaisen pääavain kentän arvon täytyy olla uniikki. Esimerkiksi kuvassa 2 Työntekijä-taulun pääavaimeksi voidaan valita ”TYONTEKIJA-ID”-attribuutti. Pääavain antaa jokaiselle työntekijälle uniikin arvon, jolla työntekijät voidaan tunnistaa. Pääavaimen lisäksi taulujen yhdistämiseen tarvitaan vierasavaimia, vierasavaimen arvot otetaan toisen taulun pääavaimesta. Kuvassa 2 tauluilla on yhden suhde moneen (1:N) suhde. Taulujen pääavaimet ovat ”TYONTEKIJA_ID” ja ”OSASTO_ID” attribuutit. Jotta yhden suhde moneen suhde voidaan muodostaa, tarvitaan pääavain ja vierasavain. Jos ajatellaan jälleen oikean maailman objekteja, työntekijöitä voi olla useampi yhtä osastoa kohden. Osasto-taulu on siis suhteessa se puoli, joita on yksi monta kohden ja työntekijä-taulu se puoli, joita voi olla useampi yhtä kohden. Osasto-taulun pääavaimen arvot tulevat työntekijä-taulun vierasavaimeksi. Nyt jokainen työntekijä voidaan tunnistaa kuuluvaksi johonkin tiettyyn osastoon (Whitehorn ja Marklyn, 2007, s. 86, 93).

2.3 Relaatiotietokannan tiedonhallintajärjestelmä

Relaatiotietokannan tiedonhallintajärjestelmän määritelmä sisältää vähintään seuraavat asiat (Date, 2001, s. 52).

1. Tieto näkyy käyttäjälle vain tauluina (table). Kuvassa 2 nähdään kaksi taulua, työntekijä- ja osastotaulut.
2. Käyttäjän käytössä oleviin operaatioihin (operators) kuuluvat ne, jotka luovat uusia tauluja vanhoista tauluista. Operaatioihin täytyy vähintään kuulua SELECT-, PROJECT- ja JOIN-operaatiot. Nämä ovat tiedon manipulointiin tarkoitettija operaatioita, jotka kuuluvat SQL-kyselykieleen (Date, 2001, s. 66).

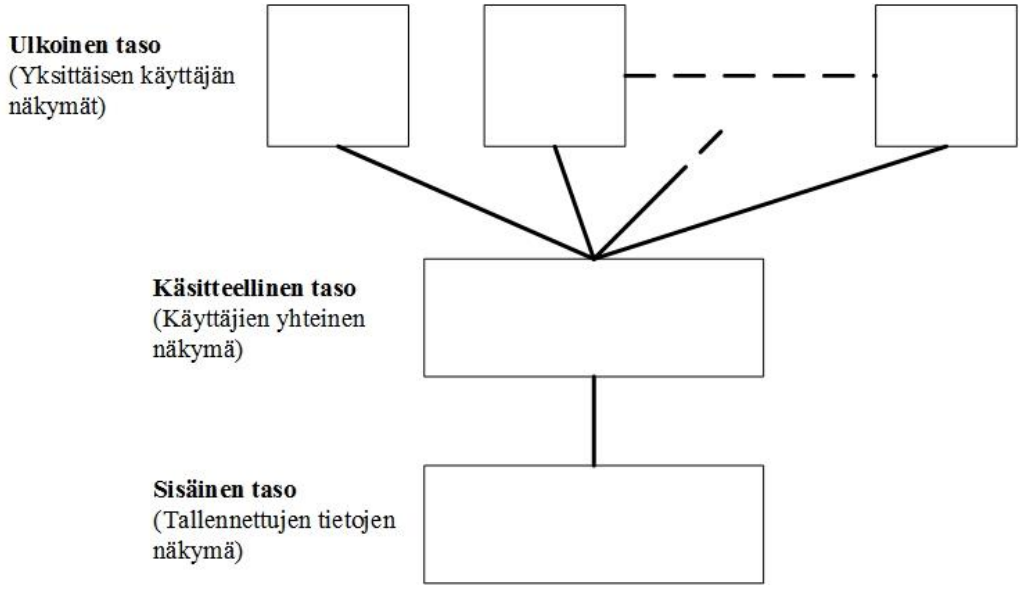
Yllämainitut operaatiot määritellään seuraavasti: SELECT valitsee määrätyt rivit taulusta; PROJECT valitsee määrätyt sarakkeet taulusta; JOIN yhdistää kaksi taulua yhteisen sarakkeen yhteisten arvojen perusteella (Date, 2001, s. 53).

Jotta THJ:n voi määritellä relaatiojärjestelmäksi, käyttäjän täytyy havaita tietokannan tieto tauluina. Itse tieto voi olla fyysisesti missä muodossa tahansa, kunhan tieto nähdään tai havaitaan tauluina. Tallennetun tiedon yksityiskohdat ovat piilossa käyttäjältä. ANSI-SPARC-termein (Date, 2001, ss. 28-29) Ulkoinen ja käsitteellinen taso ovat relaatiotietokannan tiedonhallintajärjestelmässä relaatio-muodossa, mutta sisäinen ja fyysinen tietokanta eivät ole. Relaatioteorialla ei ole mitään sanottavaa sisäisestä tasosta, vain sillä on merkitystä, että miten tietokanta näkyy käyttäjälle (Date, 2001, s. 55).

Kuvassa 3 näkyy ANSI/SPARC-arkkitehtuuri, joka on yksi THJ-arkkitehtuureista. ANSI/SPARC-arkkitehtuuri koostuu kolmesta eri tasosta: ulkoinen taso eli *external level* (Date, 2001, s. 29), Käsitteellinen taso eli *conseptual level* (Date, 2001, s. 29) ja sisäinen taso eli *internal level* (Date, 2001, s. 29). Useimmat THJ:t eivät erottele kolmea tasoa kokonaan, mutta kuitenkin tukevat ANSI/SPARC-arkkitehtuuria jollakin asteella (Date, 2001, s. 28; Elmasri ja Navathe, 2010, s. 35).

- Ulkoinen taso on yksittäisen käyttäjän taso ja vaikuttaa lähinnä siihen miten käyttäjä näkee tiedon. Ulkoinen taso sisältää useita ulkoisia kaavioita tai käyttäjänäkymiä. Jokainen ulkoinen kaavio määrittelee osan tietokannasta tietylle käyttäjäryhmälle ja piilottaa lopun tietokannasta. Käyttäjä voi olla ohjelmoija tai loppukäyttäjä. Jokaisella käyttäjällä on käytettävissä oma ohjelmointikielensä. Ohjelmoijalle se on jokin tavanomainen ohjelmointikieli. Loppukäyttäjälle se on yleensä kyselykieli (Date, 2001, s. 29, 31; Elmasri ja Navathe, 2010, s. 34).
- Käsitteellinen taso on näkymä koko tietokannan sisällöstä ja sillä on käsitekaava, joka kuvaa koko tietokannan rakenteen käyttäjäryhmille. Käsitteellinen taso on muodossa, joka on abstrakti verrattuna siihen, että miten itse tieto on fyysisesti tallennettu. Näkymä on myös erillainen verrattuna siihen, että miten yksittäinen käyttäjä näkee tiedon. Käytännössä käsitteellisen taso on tarkoitettu olevan näkymä tiedosta ilman mitään ulkoisia rajoitteita, esimerkiksi laitteisto tai käytetty ohjelmointikieli eivät vaikuta käsitteelliseen näkymään (Date, 2001, s. 34; Elmasri ja Navathe, 2010, s. 34).
- Sisäisellä tasolla on sisäinen kaavio ja se kuvaa fyysisen tietokannan rakenteen. Sisäinen taso määrittää asioita kuten tietotyypit, indeksit, miten tallennetut kentät kuvataan ja missä fyysisessä järjestyksessä tallennetut tiedot ovat. Sisäinen taso ei

ota huomioon fyysisesti tallennettuja tietoja, kuten lohkoja ja sivuja, eikä myöskään tietylle laitteelle ominaisia sylinterejä ja ura kokoja (Date, 2001, s. 28, 36; Elmasri ja Navathe, 2010, s. 34).



Kuva 3. ANSI/SPARC-arkkitehtuuri (Date, 2001, s. 29).

Relaatiotietokannan tiedonhallintajärjestelmässä kaikki tieto voidaan esittää vain yhdellä tavalla. Tiedot sijaitsevat tietyn sarakkeen kohdalla, tietyssä rivissä ja kuuluvat tauluihin. Relaatiotietokannan tauluissa ei ole osoittimia toisiin tauluihin. Kuvassa 2 nähdään, miten tauluilla on yhteys toisiinsa. Esimerkiksi työntekijätaulussa T1-rivillä on yhteys osastotaulun O2-riviin. Tämä on mahdollista, koska OSASTO_ID-sarake esiintyy molemmissa tauluissa. Toisaalta vaikka hallintajärjestelmä esittää taulut ilman osoittimia, se ei tarkoita, että osoittimia ei ole olemassa fyysisellä tasolla. Näiden määrittelyjen lisäksi kaikki tietokannan arvot ovat atoomisia (atomic). Jokaisessa sarakkeessa, jonkin rivin kohdalla voi olla tasan yksi arvo, joten sillä ei voi olla useita arvoja (Date, 2001, s. 55).

Esimerkkejä relaatiotietokantojen tiedonhallintajärjestelmistä ovat mm. Oracle ™ (Coronel ym., 2011, s. 10), IBM DB2™ (Coronel ym., 2011, s. 10), SQL server ™ (Coronel ym., 2011, s. 10), MySQL™ (Coronel ym., 2011, s. 10), PostqreSQL™ (Mathew ja Stones, 2005, s. 1), Access™ (Coronel ym., 2011, s. 10).

2.4 Tietokannan normalisointi

Suunniteltaessa relaatiotietokantaa on kaksi ääripään vaihtoehtoa. Kaikki tieto voidaan sijoittaa yhteen tauluun, joka on vain vähän tai ei yhtään normalisoitu. Toinen ääripään vaihtoehto on, että jokainen attribuutti sijoitetaan omaan tauluun ja jokainen attribuutti pystyy tallentamaan loputtoman määrän arvoja (Grant, 2012, s. 470).

Yleissääntönä on hyvä yrittää tunnistaa oikean maailman objektit, joita yritetään mallintaa tietokannassa. Objekteja kuten työntekijät, tilaukset, asiakkaat, tuotteet jne. ja sijoittaa näiden tiedot omiin tauluihinsa. Taulujen tullessa monimutkaisemmiksi voi olla vaikea määritellä mitkä kentät menevät mihinkin tauluun. Kyselyiden suorittamisessa voi tulla vastaan ongelmia, jos tieto ei sijaitse oikeassa taulussa. Sama tieto voi myös turhaan sijaita useassa paikassa. Normalisaation päämääränä on vähentää tarpeetonta ja päällekkäistä tietoa, ja siten poistaa tiedon päivitykseen liittyviä ongelmia (Whitehorn ja Marklyn, 2007, ss. 215-216; Date, 2008, s. 116).

Kohtuullinen normalisointi nostaa tietokannan suorituskykyä. Tietokannassa, jossa on usealla sarakkeella varustettuja, isoja tauluja on yleensä merkki siitä, että normalisaatiota ei ole tarpeeksi. Vähä normalisointi aiheuttaa tiedon toistumista, joka voi johtaa vääriin tuloksiin ja alentaa kyselytehokkuutta. Samaan tapaan liika normalisointi ei ole hyväksi kyselyiden tehokkuudelle, liika normalisointi aiheuttaa ylimääräistä taulujen yhdistelyä. Hyvä sääntö on kiinnittää huomiota kyselyihin, joissa yhdistetään enemmän kuin 8-12 taulua (Grant, 2012, ss. 470-471).

2.5 Tutkimusongelmat

Kuten aikaisemmin mainittiin työssä otetaan huomioon relaatiomalliin perustuvat tiedonhallintajärjestelmät. Tutkimuskysymykset selvittävät, mitä vaihtoehtoja on THJ:n valinnassa ja milla kriteereillä niitä voidaan valita ja vertailla. Työssä pyritään vastaamaan seuraaviin tutkimuskysymyksiin:

1. Mitä eri mahdollisuuksia yrityksellä on tiedonhallintajärjestelmän valinnassa?
2. Mitkä ovat oleelliset valintakriteerit tiedonhallintajärjestelmää valittaessa?
3. Miten tiedonhallintajärjestelmiä voi vertailla?

2.6 Ratkaisumenetelmä

Käytän tässä työssä tiedonhallintajärjestelmien tutkimiseen aiheeseen liittyvää kirjallisuutta ja artikkeleita. Kirjallisuutta on etsitty Lappeenrannan tiedekirjastosta, Google Scholarista, Lappeenrannan teknillisen yliopiston (LTY) seuraavista tietokannoista: EBSCO Academic Search Elite, EBSCO Business Source Complete, Elsevier, Emerald Journals, SpringerLink eBooks, SpringerLink eJournals ja Wiley Blackwell Online Library. Työssä on suurimmilta osin käytetty lähteinä LTY:n tietokannoista löytyneitä artikkeleja ja kirjoja, sekä Lappeenrannan tiedekirjaston kirjoja. Google Scholarista on löytynyt myös muutama hyvä lähde. Lähteitä on löytynyt esimerkiksi hakusanoilla: “data warehouse”, ”database systems in businesses”, ”DBMS”, “distributed database systems”, ”relational database”, ”database systems” jne. Hyvän laatuisia lähteitä on käytetty mahdollisimman paljon.

Aluksi tavoitteena on perehtyä syvällisesti tiedonhallintajärjestelmiin, tietokantatyyppeihin ja relaatiomalliin. Selvittämällä mahdolliset tietokantatyypit, voidaan selvittää mitä vaihtoehtoja asiakkaalla on tiedonhallintajärjestelmän valinnassa relaatiomallin puitteissa.

Kirjallisuuteen perehdyttyä verrataan nykyisiä eri tiedonhallintajärjestelmä (THJ)-tyyppejä ja selvitetään, millä kriteereillä näitä THJ-tyyppejä voidaan verrata toisiinsa. Mahdollinen valintakriteeri THJ:lle voi olla esimerkiksi suorituskyky tai tiedon tyyppi.

3 TIEDONHALLINTAJÄRJESTELMÄ-TYYPIT

3.1 Tietokantatyypit

Tiedonhallintajärjestelmällä voidaan rakentaa monia erityyppisiä tietokantoja. Lisäksi monia erilaisia menetelmiä on käytetty tietokantojen luokitteluun. Tietokannat voidaan luokitella käyttäjien määrällä, tiedon sijainnilla, tiedon tyypillä ja tiedon käyttötarkoituksella.

Käyttäjämäärä määrittelee onko tietokanta yhden käyttäjän tietokanta vai usean käyttäjän tietokanta. Yhden käyttäjän tietokanta tukee vain yhtä käyttäjää kerrallaan. Tämän kaltainen yhden käyttäjän tietokanta on *desktop database* (Coronel ym., 2011, s. 9). Usean käyttäjän tietokannat tukevat nimensä mukaan samanaikaisia käyttäjiä. Käyttäjien määrän pysyessä noin alle 50 käyttäjässä, kutsutaan tietokantaa *workgroup databaseksi* (Coronel ym., 2011, s. 9). Käyttäjien määrän noustessa yli 50:n tietokanta voidaan luokitella yrityksille tarkoitetuksi *enterprise databaseksi* (Coronel ym., 2011, s. 9).

Tietokannan sijaintia voi myös käyttää tietokannan tyypin määrittelyyn. Tietokantaa, joka sijaitsee yhdessä paikassa, kutsutaan keskitetyksi tietokannaksi eli *centralized databaseksi* (Coronel ym., 2011, s. 9). Tietokanta, jossa tieto on jaettu usean paikan kesken, kutsutaan hajautetuksi tietokannaksi eli *distributed database* (Coronel ym., 2011, s. 9).

Joissain tapauksissa, kuten tutkimusympäristössä, tietokannan voi määritellä tallennetun tiedon tyypin mukaan (Coronel ym., 2011, s. 9). Näin tietokannat jakautuvat kahteen ryhmään. Ensimmäisessä tyypissä tietokanta sisältää useaa erityyppistä tietoa, joita hyödynnetään eri tarkoituksiin. Tämä tyyppi tunnetaan nimellä *general-purpose database* (Coronel ym., 2011, s. 9). Toisen tyyppisessä tietokannassa on tietoa, mikä keskittyy tiettyyn aihepiiriin. Tämä tyyppi on *discipline-specific database* (Coronel ym., 2011, s. 9). Tämänkaltainen tietokanta voi olla esimerkiksi lääketieteellinen tietokanta, joka sisältää potilastietoja (Coronel ym., 2011, s. 9).

Hernandezin (2003) mukaan THJ:n hallinnoimat tietokannat voidaan jakaa kahteen eri tyyppiin: analyyttisiin tietokantoihin ja operatiivisiin tietokantoihin. Suosituin tapa

määritellä tietokanta on tapa, jolla tietokantaa käytetään tai kuinka nopeasti tietoa tarvitaan. Tämä voi esimerkiksi tarkoittaa jokapäiväistä liiketoimintaa tukevaa tietokantaa, joka voi esimerkiksi sisältää tuotteen tai palvelun maksutapahtumatietoa. Tämän kaltaisen tiedonhallintajärjestelmän pitää tallentaa tietoa tarkasti ja nopeasti. Tätä tyyppiä kutsutaan *operational databaseksi* eli operatiiviseksi tietokannaksi (Coronel ym., 2011, s. 9).

Analyyttiset tietokannat keskittyvät lähinnä historiallisen tiedon tallentamiseen tai sisältävät tietoa, jota käytetään nimenomaan yrityksen taktiseen ja strategiseen päätöksentekoon. Tyypillisesti analyyttiset tietokannat muodostuvat kahdesta osasta: *tietovarastosta* eli *data warehouse* (DW) (Coronel ym., 2011, s. 10) ja *online analytical processing* (OLAP) (Coronel ym., 2011, s. 10) työkaluista. Tietovarasto on erikoistunut tietokanta, joka säilyttää tietoa päätöksenteon tukemista varten. Tietovarasto sisältää historiallista tietoa operatiivisista tietokannoista ja muista ulkopuolisista lähteistä. Online analytical processing (OLAP) tarjoaa ympäristön, jossa on työkaluja monimutkaiseen tiedon analysointiin. OLAP-työkaluja käytetään tiedon hakemiseen, käsittelyyn ja mallintamiseen (Coronel ym., 2011, ss. 9-10). OLAP on 3- dimensionaalinen tietokanta eli kuutio.

3.2 Keskitetyt tiedonhallintajärjestelmät

Tiedonhallintajärjestelmä on keskitetty jos kaikki tieto sijaitsee yhdessä toimipaikassa. Keskitetty THJ voi tukea useita käyttäjiä, kuitenkin THJ ja tietokanta ovat vain yhdessä toimipaikassa (Elmasri ja Navathe, 2010, s. 49). THJ-arkkitehtuurit ovat seuranneet samanlaisia trendejä, kuten tyypilliset tietokonearkkitehtuurit. Vanhemmat arkkitehtuurit käyttivät tehokkaita keskustietokoneita, jotka vastasivat kaikesta prosessoinnista. Keskustietokone vastasi kaikista ominaisuuksista mukaan lukien loppukäyttäjän sovellusohjelmista, käyttöliittymästä ja THJ-toiminnoista. Tämä johtui siitä, että käyttäjät yhdistivät järjestelmään terminaalin kautta. Terminaali toimi näyttöpäätteenä, eikä sillä itsellään ollut prosessointitehoja. Useimmat käyttäjät korvasivat terminaalit henkilökohtaisiin tietokoneisiin vasta laitteiden hinnan laskiessa. Vähitellen THJ:t alkoivat hyödyntämään käyttäjän prosessointitehoja, joka johti asiakas/palvelin THJ-arkkitehtuureihin. Tyypillisesti asiakas ja palvelin ohjelmistot toimivat eri tietokoneella.

Asiakas/palvelin THJ:llä on kaksi arkkitehtuuria: kaksitasoinen ja kolmitasoinen arkkitehtuuri (Elmasri ja Navathe, 2010, s. 44, 46).

3.2.1 Kaksitasoinen asiakas/palvelin-arkkitehtuuri

THJ-arkkitehtuuria kutsutaan kaksitasoiseksi koska ohjelmistokomponentit on jaettu kahteen osaan: asiakkaalle ja palvelimelle. Arkkitehtuurin hyötynä on yksinkertaisuus ja saumaton yhteensopivuus vanhojen järjestelmien kanssa (Elmasri ja Navathe, 2010, s. 47).

Relaatio THJ:ssä ensimmäiset komponentit, jotka siirrettiin asiakkaalle olivat käyttöliittymä ja loppukäyttäjän käyttämät sovellusohjelmat. SQL tarjosi loogisen jaottelun asiakkaan ja palvelimen välillä. SQL-kielen prosessointiin liittyvät kyselyt ja tapahtumat jäivät palvelimen vastuulle. Tämän tyyppisessä arkkitehtuurissa palvelinta kutsutaan usein kyselypalvelimeksi (Elmasri ja Navathe, 2010, s. 46) tai tapahtumapalvelimeksi (Elmasri ja Navathe, 2010, s. 46). Relaatiomallisessa THJ:ssä palvelinta voidaan kutsua myös SQL-palvelimeksi (Elmasri ja Navathe, 2010, s. 46).

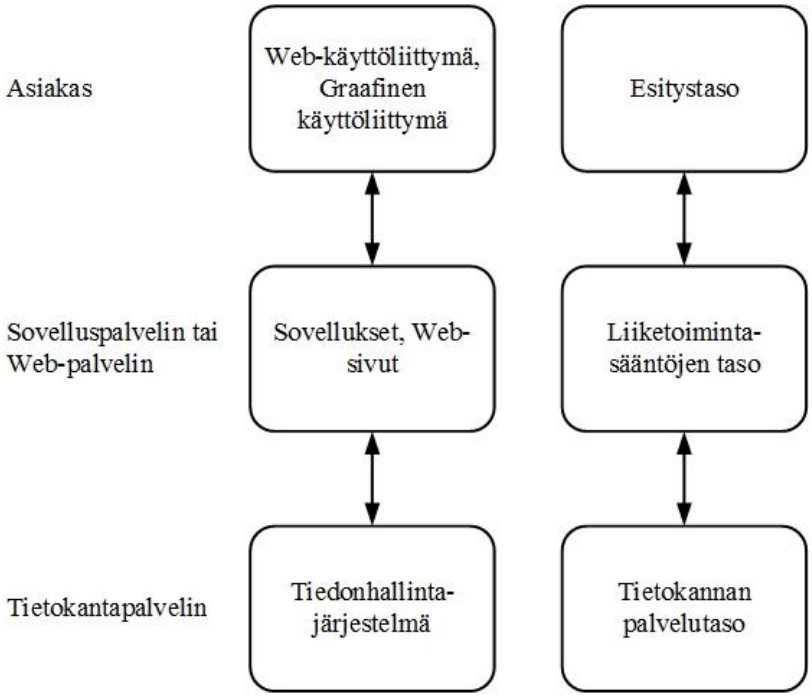
Standardi nimeltä Open Database Connectivity eli ODBC (Elmasri ja Navathe, 2010, s. 47) mahdollistaa kommunikoinnin asiakkaan ja palvelimen välillä. ODBC tarjoaa ohjelmointirajapinnan, jolla asiakas ottaa yhteyden palvelimeen ja yhteyden muodostuttua asiakas voi kommunikoida THJ:n kanssa. Suurin osa THJ-toimittajista tukevat ODBC standardia (Elmasri ja Navathe, 2010, s. 47). Asiakasohjelma ottaa ODBC-ohjelmointirajapinnalla yhteyden palvelimeen ja lähettää kysely- tai tapahtumapyynnöt palvelimelle. Palvelin prosessoi pyynnöt ja lähettää tuloksen takaisin asiakkaalle (Elmasri ja Navathe, 2010, s. 46, 47).

3.2.2 Kolmitasoinen asiakas/palvelin-arkkitehtuuri

Web-sovellukset käyttävät arkkitehtuuria nimeltä kolmitasoinen arkkitehtuuri, joka lisää asiakkaan ja palvelimen väliin välipalvelimen. Välipalvelin voi olla Web-palvelin tai sovelluspalvelin. Palvelimella on omat sovelluksensa ja se säilyttää liiketoimintasääntöjä (business rule), jotka ovat yleensä luonnollisen kielen muodossa, ja esittävät mitä jokin tieto tietokannassa tarkoittaa tai miten sen arvoja on rajoitettu (Date, 2008, s. 21). Palvelin

parantaa myös tietoturvaa, tarkistamalla käyttöoikeudet ennen kuin välittää pyynnöt THJ:lle (Elmasri ja Navathe, 2010, ss. 47-48).

Käyttöliittymä, sovellussäännöt ja tiedonhallinta toimivat kolmena tasona. Kuvassa 4 nähdään kolmitasoisen arkkitehtuurin rakenne. Asiakasohjelma sisältää graafisen käyttöliittymän ja sovelluksiin liittyviä liiketoimintasääntöjä. Välipalvelin ottaa vastaan pyyntöjä asiakkaalta, prosessoi pyynnöt ja lähettää kyselyt ja komennot tietokantapavelimelle. Välipalvelin lähettää palvelimen prosessoiman tiedon takaisin asiakkaalle, jossa voidaan prosessoida tietoa pidemmälle ja tieto esitetään käyttäjälle graafisessa käyttöliittymässä. Esitystaso esittää informaatiota käyttäjälle ja mahdollistaa tiedon syötön. Liiketoimintasääntöjen taso on vastuussa keskimmäisen tason säännöistä ja rajoituksista, ennen kuin tieto lähetetään THJ:lle. Alimpaan tasoon kuuluu kaikki tiedonhallintapalvelut. Keskimmäinen taso voi myös toimia Web-palvelimena, joka ottaa vastaan kaikki kyselytulokset THJ:ltä. Web-palvelin muotoilee tiedot dynaamiseksi web-sivuksi, jota voidaan tarkastella selaimella (Elmasri ja Navathe, 2010, s. 48).



Kuva 4. Kolmitasoinen asiakas/palvelin-arkkitehtuuri
(Elmasri ja Navathe, 2010, s. 48).

Salaustekniikan kehittyminen on mahdollistanut turvallisen salatun tiedonsiirron palvelimelta asiakkaalle, jossa salaus puretaan. Salauksen purkaminen voidaan suorittaa laitteistolla tai kehittyneellä ohjelmistolla. Arkkitehtuurin etuihin kuuluu tehokas tietokannan suojaus, mutta on kuitenkin altis verkon kautta tuleville hyökkäyksille. Tämän lisäksi tiedon tiivistäminen helpottaa suurien tietomäärien siirtoa palvelimelta asiakkaille (Elmasri ja Navathe, 2010, s. 49).

3.3 Hajautetut tiedonhallintajärjestelmät

Hajautettu tiedonhallintajärjestelmä mahdollistaa hajautetun tietokannan hallinnoimisen ja pitää huolen siitä, että tieto on käyttäjille läpinäkyvää eli tiedon hajautuneisuus ei näy suoraan käyttäjälle. Hajautettu tietokanta on joukko tietoverkon yli hajautettuja ja loogisesti yhteydessä olevia tietokantoja. *Distributed database system* eli lyhyesti DDBS (Özsu ja Valduriez, 2011, s. 3) tarkoittaa molempia sekä hajautettua THJ:ää että hajautettua tietokantaa. Hajautetun THJ:n käytöllä on monia hyötyjä ja näistä keskeisimmät ovat seuraavat: hajautetun ja kopioidun tiedon läpinäkyvä hallinta, luotettava tapahtumien suoritus hajautuksen avulla, parantunut suorituskyky ja helpompi järjestelmän laajentaminen (Özsu ja Valduriez, 2011, s. 3, 7). Seuraavaksi esitellään esimerkkejä eri hajautetuista DDBS-arkkitehtuureista.

3.3.1 Asiakas/palvelin-tiedonhallintajärjestelmä

Asiakas/palvelin-tyypin THJ keskittyy tiedon hallinnoimiseen palvelimilla, ja asiakasohjelma keskittyy ohjelmistoympäristöön, josta käyttäjä pääsee käyttöliittymän avulla käsiksi tietokantoihin. Asiakas/palvelin-järjestelmä erottaa toiminnot, joita pitää tarjota palvelimen ja asiakkaan puolesta. Järjestelmän toiminnot jaetaan kahteen osaan: palvelimen toimintoihin ja asiakasohjelman toimintoihin. Tämä mahdollistaa kaksitasoisen arkkitehtuurin, joka helpottaa nykyaikaisen hajautetun tiedonhallintajärjestelmän hallintaa. Relaatiomallin THJ:ssä palvelin tekee suurimman osan tiedonhallintatyöstä. Toisin sanoen, palvelin hoitaa kyselyiden käsittelyn ja optimoinnin, tapahtumien käsittelyn ja fyysisen tason hallinnoinnin. Asiakkaalla on asiakasohjelman ja käyttöliittymän lisäksi THJ:n asiakasmoduuli, joka on vastuussa välimuistissa sijaitsevan tiedon hallinnasta (Özsu ja Valduriez, 2011, ss. 27-28).

Relaatiomallia hyödyntävissä hajautetuissa THJ:ssä asiakasohjelman ja palvelimen välinen kommunikointi tapahtuu SQL-komennoilla. Asiakas lähettää SQL-kyselyt palvelimelle, kuitenkaan yrittämättä ymmärtää niitä. Palvelin tekee suurimman osan työstä ja lähettää tulokset asiakkaalle. Asiakas/palvelin arkkitehtuureja on useita ja niistä yksinkertaisin on tapaus, jossa on vain yksi palvelin ja useita asiakkaita (Özsu ja Valduriez, 2011, s. 29).

3.3.2 Vertaisverkko tiedonhallintajärjestelmä

THJ:en vertaisverkko eli Peer-to-Peer (P2P) -arkkitehtuuri voidaan määritellä siten, että järjestelmässä ei ole tunnistettavissa olevia palvelimia ja asiakkaita.
Pääsääntöisesti P2P teknologiaan pohjautuvat menetelmät voidaan jakaa kolmeen kategoriaan niiden verkkotopologian mukaan (Mushtaq ym., 2006, s. 2).

1. Keskitetty, joka käyttää erillispalvelinta (*dedicated)* (Mushtaq ym., 2006, ss. 1-8).
2. Hybridi, koostuu servereistä ja vertaisverkosta (Mushtaq ym., 2006, ss. 1-8).
3. Täysin hajautettu, ei sisällä serveriä vaan koostuu ainoastaan vertaisverkosta (Mushtaq ym., 2006, ss. 1-8).

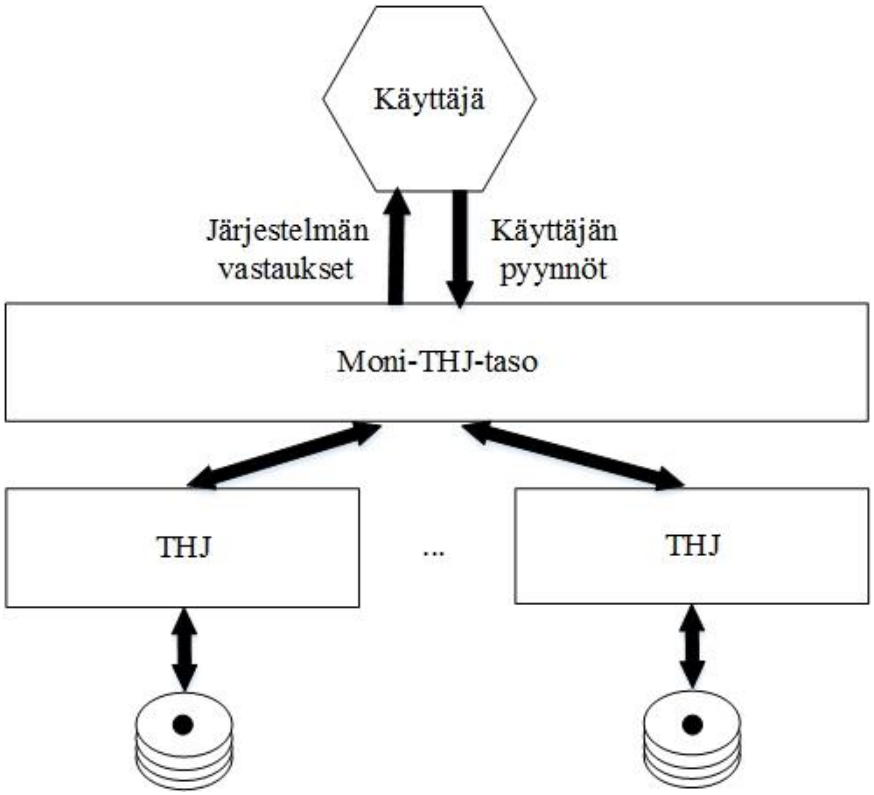
Kuitenkin moderneissa THJ:ssä tietokannan hajautus tapahtuu paljon suuremmassa mittakaavassa. Aikaisemmissa järjestelmissä toimipaikkoja oli vain muutamia. Nykyisissä järjestelmissä toimipaikkoja voi olla tuhansia ja ne voivat olla maantieteellisesti hyvinkin kaukana toisistaan. Nämä maantieteellisesti kaukana sijaitsevat toimipaikat voivat muodostaa keskenään omia klustereita. Tyypillinen vertaisverkon ominaisuus on toimipaikkojen epäyhtenäisyys ja itsenäisyys muista toimipaikoista. Tämä tuo omat ongelmansa ja kokonaan estää joidenkin ratkaisujen käytön. Nykyaikaiset arkkitehtuurit eroavat vanhoista myös siten, että ne ovat hyvin epävakaita. Tyypillisesti hajautetut THJ:t ovat hyvin tarkkaan hallittuja ympäristöjä, joissa uusien toimipaikkojen lisäys tai poisto ei ole yleistä ja nämä operaatiot tehdään huolellisesti. Nykyaikaisessa vertaisverkkoarkkitehtuurissa eri toimipaikat ovat usein ihmisten yksittäisiä tietokoneita. Yksittäiset tietokoneet voivat saapua tai poistua järjestelmästä vapaasti, ja siten hankaloittavat tiedonhallintaa. Tyypilliseltä nykyaikaiselta vertaisverkko THJ:ltä vaaditaan seuraavia ominaisuuksia (Özsu ja Valduriez, 2011, ss. 611-612):

- Autonomia: yksittäinen tietokone voi milloin vain saapua järjestelmään tai lähteä järjestelmästä. Lisäksi sen pitäisi pystyä kontrolloida tietoa, jota se varastoi ja sitä, että mitkä muut järjestelmän tietokoneet voivat varastoida sen tietoa.
- Kyselyiden tarkkuus: kyselykielen täytyy antaa käyttäjän määritellä haut tarpeeksi suurella tarkkuudella. SQL:n kaltainen kyselykieli on tarpeen, jos tieto on tietokannassa monimuotoisessa rakenteessa. Kuten relaatiotietokannassa tauluina.
- Suorituskyky: THJ käyttää tehokkaasti resursseja, kuten tietokoneen tehoja, kaistaa ja tietokantaa. Tehokkaalla resurssien käytöllä kyselyjä pystytään suorittamaan suurempia määriä.
- Käyttäjätyytyväisyys: käyttäjän havaitsema järjestelmän suorituskyky. Käyttäjätyytyväisyyden piiriin lasketaan asioita kuten: kyselytulosten täydellisyys, tiedon yhdenmukaisuus, tiedon saatavuus ja kyselyiden nopeus.
- Virhetoleranssi: suorituskyvyn ja käyttäjien tyytyväisyyden pitää säilyä, vaikka jokin tietokone tai jotkin tietokoneet kaatuisivat.
- Tiedon suojaus: THJ ei tukeudu palvelimiin, joten sen avoin toimintatapa avaa järjestelmän vakaville tietoturvauhille.
- Skaalautuvuus
- Palvelunlaatu
- Digitaaliset vesileimat (Wang ym., 2009, s. 3)
- Serverin kuormitus
- Turvallisuus
- Lisenssin hajautettavuus

3.3.3 Monitietokanta -arkkitehtuuri

Multidatabase systems eli lyhyesti MDBS (Özsu ja Valduriez, 2011, s. 35) tapauksessa yksittäiset THJ:t ovat täysin autonomisia, eivätkä ne tee yhteistyötä riippumatta siitä, että ovatko ne hajautettuja vai eivät. THJ:t eivät välttämättä tiedä toistensa olemassaolosta tai miten kommunikoida toistensa kanssa. THJ:jiä on useita, ja ne kaikki vastaavat omasta tietokannasta. Yksittäinen THJ voi päättää kenellä on oikeus päästä siihen tietokantaan käsiksi, jota se hallitsee. MDBS tarjoaa ohjelmistotason, joka toimii yksittäisten THJ:n kanssa ja mahdollistaa käyttäjille pääsyn eri tietokantoihin. Ohjelmistotaso voi toimia

useissa kohteissa tai sen käyttö voi olla keskitetty yhteen kohteeseen. Yksittäiselle THJ:lle ohjelmistotaso näkyy vain ohjelmana, joka lähettää pyyntöjä ja vastaanottaa vastauksia (Özsu ja Valduriez, 2011, ss. 35-37). Kuvassa 5 nähdään Moni-THJ:n komponentit. Käyttäjä kommunikoi moni-THJ- tason kanssa, joka mahdollistaa yhteyden asiakkaan ja yksittäisen THJ:n kanssa.



Kuva 5. Moni-THJ:n komponentit (Özsu ja Valduriez, 2011, s. 38)

3.4 Tiedon varastointi

Tietovarastot ovat tietokantoja, jotka keräävät tietyin aikavälein tietyn organisaation toiminnoista tietoa. Tietoa kerätään analysointia varten, jonka avulla voidaan selvittää organisaatiolle hyödyllistä strategista informaatiota, kuten tietoa trendeistä jne. Hyödyntämällä tietovarastoja, organisaatiot kehittävät toimintaansa ja kykenevät saavuttamaan päämääränsä helpommin. Esimerkiksi tietovarastoja voidaan käyttää asiakaskäyttäytymisen analysointiin. Analysoidun tiedon avulla organisaatio kykenee ymmärtämään paremmin asiakkaan tarpeita ja odotuksia (Malinowski ja Zimányi, 2009).

Operatiiviset THJ:t eivät kykene vastaamaan tiedon analysoinnin tarpeisiin. Ne tukevat organisaation jokapäiväistä toimintaa, keskittyvät tiedon nopeaan saatavuuteen ja suuren käyttäjäjoukon tukemiseen. Tyypillinen operatiivinen tietokanta säilyttää yksityiskohtaista tietoa, kuitenkin ottamatta huomioon historiallista tietoa. Ne ovat yleensä hyvin pitkälle normalisoituja, joten monimutkaisia kyselyjä suorittaessa niiden suorityskyky on huono. Koko organisaation toiminnan analysointi vaatii myös tiedon integrointia useista lähteistä. Tietovarastot luotiin vastaamaan näihin ongelmiin. Tietovarasto on kokoelma integroitua, historiallista, pysyvää ja tiettyihin aiheisiin perustuvaa tietoa (Malinowski ja Zimányi, 2009, s. 41).

OLAP-kyselyt ovat erittäin tärkeitä tietovarastoille. Yrityksien ja muiden organisaatioiden tietovarastoissa on suuria määriä tietoa. Kyselyillä selvitetään tietovarastosta organisaatiolle tärkeitä toistuvia trendejä. OLAP-prosessissa käytetään monimutkaisia kyselyjä, jotka kokoavat organisaatiolle merkityksellistä tietoa. Päätösentekoa tukevat OLAP-kyselyt tyypillisesti käsittelevät erittäin suuria määriä tietoa, vaikka kyselyn tulokset olisivatkin pieniä. Esimerkki tämän kaltaisesta kyselystä on kysely, joka etsii tuotteita, joiden kokonaismyynti on nousussa tai laskussa (Garcia-Molina ym., 2002, s. 1070).

Tietovaraston luomiseen käytettävän THJ:n pitäisi sisältää toimintoja, jotka avustavat tietovaraston hallintaa. Näihin toimintoihin kuuluu: suuren tietomäärän hallinta, tietovaraston päivitys uusilla tiedoilla eri lähteistä ja monimutkaisten operaatioiden suorittamista, joihin voi kuulua monien taulujen yhdistämistä ja tiedon kokoamista. Näiden tukemista varten tietovaraston THJ:n ominaisuuksia kuten: menetelmiä tiedon tallentamiseen, hakemistoja, funktioita tiedon kokoamista varten, kykyä jakaa tauluja osiin ja tuki samanaikaisille kyselysuorituksille (Malinowski ja Zimányi, 2009, ss. 294-295). Tietovaraston arkkitehtuuri muodostuu seuraavista osista:

- Alimmalla tasolla on ETL-työkalut eli extraction-transformation-loading-työkalut (Malinowski ja Zimányi, 2009, s. 55), joiden tehtävänä on syöttää tietoa järjestelmään operatiivisista tietokannoista ja muista tietolähteistä. Näitä tietolähteitä ovat esimerkiksi tiedostot, jotka voivat olla yrityksen sisäisistä tai

ulkoisista lähteistä. Alimalla tasolla on myös tietokanta, joka on olemassa tiedon integrointia ja muuntamista varten (Malinowski ja Zimányi, 2009, ss. 55-56).

- Seuraavalla tasolla on itse tietovarasto. Yritysluokan tietovaraston lisäksi tasoon kuuluu data martit (Malinowski ja Zimányi, 2009, s. 55), jotka tarjoavat käyttäjille helpomman pääsyn tärkeään tietoon ja säilytys paikan kuvaustiedolle. Data mart yleensä säilyttää esimerkiksi tietyn osaston tarpeisiin erikoistunutta tietoa (Malinowski ja Zimányi, 2009, s. 55, 57).
- OLAP-tasolla on OLAP-palvelin, joka tukee moniulotteistatietoa ja operaatioita. Sen tehtävä on tarjota käyttäjille tietovarastosta haettua moniulotteistatietoa. (Malinowski ja Zimányi, 2009, s. 55, 58).
- Ylimmän tason tehtävä on tiedon analysointi ja visualisointi. Se sisältää asiakasohjelman osia kuten OLAP-, raportointi-, statistiikka- ja tiedonlouhintatyökalut (Malinowski ja Zimányi, 2009, s. 55).

3.5 Operatiiviset tiedonhallintajärjestelmät

Operatiiviset tietokannat ovat osa monia yrityksiä, organisaatioita ja instituutteja. Tämän tyyppistä tietokantaa käytetään ensisijaisesti OLTP eli on-line transaction processing (Hernandez, 2003, s. 4) tilanteissa. Tämän tyyppisissä tilanteissa on tarve päivittäin kerätä, muokata ja säilyttää tietoa. Operatiivisessa tietokannassa tallennettu tieto on dynaamista, toisin sanoen tieto muuttuu jatkuvasti ja kuvastaa yrityksen ajantasaista informaatiota. Vähittäiskauppojen, teollisuusyrityksien ja sairaaloiden kaltaiset organisaatiot hyödyntävät operatiivisia tietokantoja, sillä niiden tieto on jatkuvan muutoksen kohteena (Hernandez, 2003, s. 4).

Operatiivinen tietokanta on suunniteltu perussovellusten käyttöön ja sen tehtävänä on automatisoida toimintoja. Tyypillisesti operatiiviseen tietokantaan syötetään tiedot päätteiltä tai työasemilta. Operatiivisessa THJ:ssä käytetään nykyään melkein pelkästään relaatiotietokantoja (Hovi, 1997, s. 19). Operatiivisella THJ ja tietokannalla on seuraavia ominaisuuksia (Hovi, 1997, s. 20):

- Yleensä päivityspainotteinen
- Paljon päivittäisiä tapahtumia

- THJ:n tehokkuus on tärkeä
- Tietokantaan tehdään muutoksia käyttöliittymän kautta
- Pitkälle normalisoidut tiedot
- Tietojen tulee olla ajantasaisia
- Tietokannassa oleva tieto on dynaamista
- Tiedot ovat alkeisella tasolla
- SQL-käskyt on indeksoitu
- Ei paljon historiallista tietoa, kuten tietovarastoissa
- Tiukat käytettävyysvaatimukset
- Sisältää lokin tapahtumille ja tapahtumat varmistetaan

4 TIEDONHALLINTAJÄRJESTELMÄN VALINTAKRITEERIT

Tässä luvussa esitetään valintakriteerejä, jotka voivat vaikuttaa tiedonhallintajärjestelmän valintaan. Elmasrin ja Navathen (2010) mukaan tiedonhallitajärjestelmä voidaan määritellä seuraavilla kriteereillä: tietomallilla, käyttäjien määrällä, toimipaikkojen määrällä ja hinnalla. Luvussa otetaan myös huomioon tietokannan suojaukseen liittyvät tiedonhallintajärjestelmän ominaisuudet, jotka Daten (2001) mukaan ovat: tietokannan palautus, tapahtumien rinnakkaisuus, tietoturva ja tietokannan eheys. Näiden lisäksi luvussa otetaan huomioon tietokantatapahtumien tärkeät ominaisuudet, joita kutsutaan ACID-ominaisuuksiksi (Date, 2001, s. 379) ja tiedonhallintajärjestelmän suorituskyky (Gillenson, 1990, s. 267).

Taulukossa 1 esitellään ensin kaikki valitut kriteerit. Taukossa valintakriteereillä on lyhyet kuvaukset ja taulukon jälkeen osa kriteereistä kuvataan aliluvuissa tarkemmin. Luvun lopussa vertaillaan eri THJ:ä.

Taulukko 1. Tiedonhallintajärjestelmän valintakriteerit

Valintakriteeri	**Kuvaus**
Suorituskyky	Ohjelman tai järjestelmän toimintanopeus (Gillenson, 1990, s. 267).
Käyttäjätyytyväisyys	Käyttäjän havaitsema järjestelmän suorituskyky. Käyttäjätyytyväisyyden piiriin lasketaan esimerkiksi kyselytulosten täydellisyys, tiedon yhdenmukaisuus, tiedon saatavuus ja kyselyiden nopeus (Özsu ja Valduriez, 2011, s. 612).
Tietokannan varmistus/toipuminen	Vika tai häiriö voi aiheuttaa ongelmia tietokannassa. Tietokannan toipuminen on pääasiassa tietokannan palautusta edelliseen toimivaan tai virheettömään tilaan. Tietokannan varmistus vaatii sitä,

	että mikä tahansa tieto voidaan palauttaa toisen ylimääräisen tiedon avulla, joka sijaitsee eri paikassa (Date, 2001, s. 375).
Tapahtumien rinnakkaisuus	Tyypillisesti THJ mahdollistaa usean tapahtuman päästä käsiksi samaan tietoon. Tämä vaatii THJ:ltä mekanismia, joka pitää huolen siitä, että tiedon samanaikainen käyttö onnistuu ongelmitta. Tiedon lukitus vastaa tapahtumien rinnakkaisuusongelmiin (Date, 2001, s. 391).
Tietoturva	Tietoturva on tiedon suojausta tiedon luvattomalta paljastukselta, muutoksilta tai tuhoamiselta. Tietoturva pitää huolen, että käyttäjien tekemät asiat ovat sallittuja. THJ valvoo, että tietokannan käyttäjät eivät riko niille asetettuja sääntöjä (Date, 2001, s. 416).
Tietokannan eheys	Tiedon eheys viittaa tiedon tarkkuuteen ja paikkaansapitävyyteen. THJ valvoo, että tieto pysyy eheänä kaiken aikaa ja eheyden määrittäviä sääntöjä seurataan. Toisin kuin tietoturvassa, säännöt eivät ole käyttäjäkohtaisia (Date, 2001, s. 440).
Skaalautuvuus	Järjestelmän laajentamisen helppous. Hajautetussa järjestelmässä on mahdollista lisätä uusia THJ:iä ja tietokantoja tietoverkkoon, ja ovat siten helposti laajennettavissa (Özsu ja Valduriez, 2011, s. 500).
THJ:n hinta	Kuinka paljon jokin tietty THJ-tuote maksaa (Elmasri ja Navathe, 2010, s. 50).

Tietomalli	THJ:n valintaan voi vaikuttaa sen käyttämä tietomalli. Tässä työssä kuitenkin käsitellään vain relaatiomalliin perustuvia THJ:ää (Elmasri ja Navathe, 2010, s. 50).
Käyttäjien määrä	THJ:n tukema samanaikaisten käyttäjien määrä (Coronel ym., 2011, s. 9).
Toimipaikkojen määrä	Keskitetyllä THJ:lla kaikki tieto on yhdessä tietokoneessa ja yhdessä toimipaikassa. Hajautetussa järjestelmässä THJ ja tietokannat voivat olla hajautettu usean toimipaikan kesken (Coronel ym., 2011, s. 9; Elmasri ja Navathe, 2010, s. 49).
ACID-ominaisuudet	Tietokantatapahtumien neljä tärkeää ominaisuutta. ACID muodostuu sanoista: atomicity, consistency, isolation ja durability (Date, 2001, s. 379).

4.1 Suorituskyky

Suorituskykyyn eli ohjelmien ja järjestelmien toimintanopeuteen voi vaikuttaa usea eri tekijä. Monet näistä ongelmista voivat liittyä tiedonhallintaan ja asioihin mihin ei voi vaikuttaa tietokantasuunnittelulla. Suorituskykyyn vaikuttavat tietokoneen prosessorin nopeus, kiintolevyjen tiedonsiirron nopeus, eri ohjelmien kilpailu tietokoneen resursseista ja tietenkin itse THJ. Tietokantasuunnittelulla voi kuitenkin vaikuttaa moniin asioihin, jotka vaikuttavat suorituskykyyn. Tämän lisäksi suorituskykyyn vaikuttaa se, että mikä tietty THJ tuote on kyseessä (Gillenson, 1990, s. 267).

Suorituskyvyn ennustamiseen on olemassa useita keinoja. Jotta suorituskykyä voidaan ennustaa tarpeeksi suurella tarkkuudella, täytyy ottaa huomioon suuria määriä informaatiota. Muutakin kuin pelkästään tietokannan rakenne. Esimerkiksi pitää ottaa huomioon, että kuinka usein tietokannassa esiintyy eri tyyppisiä tietoja,

tiedonhallintajärjestelmän tavat kutsua ohjelmia ja resursseista kilpailu muiden ohjelmien kanssa (Gillenson, 1990, s. 290).

Suorituskyvyn hienosäätö on iteratiivinen prosessi, jossa selvitetään suurimmat suorituskyvyn pullonkaulat. Prosessissa arvioidaan muutoksen tuomaa hyötyä ja palataan takaisin ensimmäiseen askeleeseen, kunnes suorituskyky on hyväksyttävä. Suorituskyvyn optimoinnissa on hyvä ottaa huomioon kustannuksien nouseminen, kun yritetään nostaa suorituskykyä pienissä askelissa. On syytä miettiä tarkkaan, onko sijoitus suorituskyvyn optimointiin kannattava. Ennen sijoittamista kannattaa kysyä kaksi kysymystä (Grant, 2012, ss. 4-5):

- Mikä on hyväksyttävä suorituskyky ohjelmalle?
- Onko suorituskyvyn lisäämisestä saatu etu tarvittavan sijoituksen arvoinen?

4.2 Hinta

Nykypäivänä löytyy ilmaisia avoimenlähdekoodin THJ-tuoteita, kuten MySQL (Coronel ym., 2011, s. 10) tai PostgreSQL (Mathew ja Stones, 2005, s. 1), joten THJ:ä on vaikea luokitella hinnan mukaan. Monista relaatiomalliin pohjautuvista tiedonhallintajärjestelmistä on tarjolla 30-päivän kokeiluversioita ja halpoja henkilökohtaisia versioita useilla ominaisuuksilla. Suuret THJ:t myydään modulaarisina eli eri komponentit ovat vastuussa hajautuksesta, tiedon toistamisesta, samanaikaisesta käytöstä, mobiilin tukemisesta jne. Lisäksi niitä myydään lisenssimuodossa eli yhdessä toimipaikassa voi olla käytössä niin monta kopiota ohjelmistosta kuin tarvitaan. Joitakin yhden käyttäjän THJ:ä, kuten Microsoft Access (Coronel ym., 2011, s. 10) myydään yksittäisinä pakeitteina tai ne tulevat tietokoneen mukana. Tietovarastoja, tiedonlouhinta ominaisuuksia ja uusia tietotyyppeja on tarjolla lisämaksusta. On mahdollista maksaa vuosittain jopa miljoonia euroja THJ:en ja tietokantojen asennuksesta ja ylläpidosta (Elmasri ja Navathe, 2010, s. 50).

4.3 Tiedon suojaus

THJ täytyy tarjota laajamittaisia toimintoja eri uhkia vastaan erityisesti tietokannan palautuksen, tapahtumien rinnakkaisuuden, tietoturvan ja eheyden suhteen. Tiedon suojaus tarkoittaa tietokannan suojelemista monilta eri uhilta, uhat voivat olla tahallisia tai tahattomia. Mahdollisuuksia eri uhille on käytännössä loputtomasti. Totuus on, että on olemassa hyvin paljon erilaisia uhkia, joille tietokanta on alttiina. Esimerkiksi näihin kuuluu (Date, 2001, s. 373):

- THJ voi kaatua jonkin kesken jonkin tietyn ohjelman suoritusta, jolloin tietokanta voi jäädä ennalta arvaamattomaan tilaan.
- Kaksi ohjelmaa, joita suoritetaan samaan aikaan voivat häiritä toistensa toimintaa ja siten tuottaa virheellisiä tuloksia.
- Tärkeä tieto voi olla luvattomalle käyttäjälle alttiina tai muutettavissa.
- Tietokannan päivitykset voivat muuttaa tietokantaa väärällä tavalla.

4.3.1 ACID-ominaisuudet

Tietokantatapahtumilla on neljä tärkeää ominaisuutta, näitä ominaisuuksia kutsutaan ACID-ominaisuuksiksi (Date, 2001, s. 379).

- *Atomicity* tarkoittaa tapahtuman atomisuutta. Atoomisessa tapahtumassa suoritetaan kaikki tapahtuman operaatiot tai ei mitään (Date, 2001, s. 379).
- *Consistency* eli eheys tarkoittaa sitä, että tietokannan johdonmukaisuus säilyy muunnoksissa. Tapahtumat muuttavat tietokantaa yhdestä johdonmukaisesta tilasta toiseen johdonmukaiseen tilaan. Siellä ei ole merkitystä onko tietokanta johdonmukaisessa tilassa tapahtuman aikana, kunhan se on johdonmukaisessa tilassa tapahtuman alussa ja lopussa (Date, 2001, s. 379).
- *Isolation* tarkoittaa tapahtumien eristystä. Kaikki tapahtumat ovat eristyksissä toisistaan. Vaikka tietokannassa tapahtuu samaan aikaa useita tapahtumia, yhden tapahtumat operaatiot eivät näy muille tapahtumille, kunnes tapahtuma on suoritettu loppuun (Date, 2001, s. 379).

- *Durability*, eli kestävyys. Tapahtuman ollessa suoritettu sen tekemät operaatiot säilyvät, vaikka THJ kaatuisi (Date, 2001, s. 379).

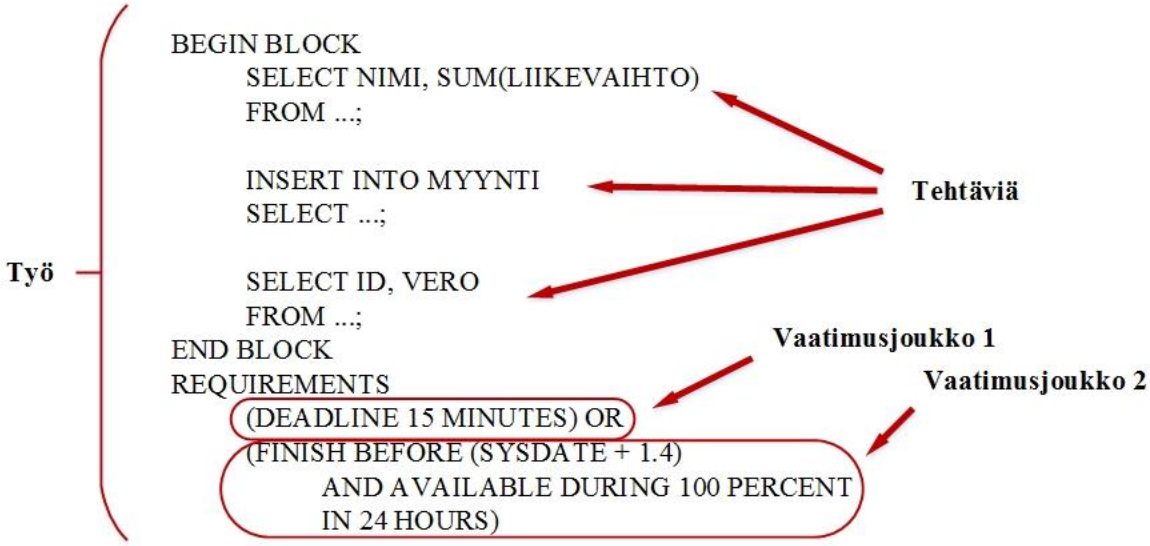
4.4 Käyttäjätyytyväisyys

Nykyvuosina on tullut uusi tarve suunnitella ohjelmistoja ja laitteita, jotka keskittyvät käyttäjätyytyväisyyteen. *Quality of Experience* eli lyhyesti QoE (Carvalho Costa ja Furtado, 2013, s. 86) määrittelee käyttäjäntyytyväisyyden tason. QoE keskittää huomion käyttäjätyytyväisyyteen suorituskyvyn optimoinnin sijaan. Konsepti vaatii THJ:n toimimista käyttäjien odotusten mukaisesti. Tämän lisäksi THJ:n täytyy saavuttaa tavanomaiset kyselyiden suorituskyky- ja aikatauluttamistavoitteet, eli käyttäjän kannalta kelvollisen suorituskyvyn (Carvalho Costa ja Furtado, 2013, s. 86).

Käyttäjätyytyväisyyteen keskittyvä THJ analysoi käyttäjävaatimuksia ja tekee halutut operaatiot vain, jos vaatimukset on mahdollista täyttää. Lisäksi THJ ilmoittaa käyttäjälle, jos vaatimuksia ei ole mahdollista täyttää. Vaatimusten avulla käyttäjien ei tarvitse tuhlata aikaa operaatioiden odottamiseen, joita ei pystytä suorittamaan tyydyttävällä tavalla. Tämän lisäksi suoritus aikaa ei mene hukkaan operaatioihin, joilla ei tule olemaan käytännön hyötyä (Carvalho Costa ja Furtado, 2013, s. 86).

THJ:n ottaessa huomioon käyttäjätyytyväisyyden, THJ voi muokata toimintojaan perustuen käyttäjän odotuksiin. Esimerkiksi jos käyttäjä haluaa tietyn raportin valmistuvan kymmenessä minuutissa, vaikka kuitenkin kaikkien operaatioiden suorittamiseen menee 30 minuuttia. Tavanomainen THJ suorittaa operaatiot parhaansa mukaan, kuitenkaan ottamatta aikavaatimuksia huomioon. Kymmenen minuutin jälkeen käyttäjä huomaa, että raportti ei valmistukaan ajoissa. Käyttäjätyytyväisyyteen keskittyvä THJ kertoo käyttäjälle, kuinka kauan operaatioihin menee aikaa. Tämän jälkeen käyttäjällä on mahdollisuus muuttaa aikavaatimusta, siirtää operaatioiden suorituksen toiselle ajankohdalle tai päättää, että raporttia ei tehdä ollenkaan. Toisena esimerkkinä käyttäjä voi tulla johtopäätökseen, että raportti voidaan muodostaa käyttämällä vain osaa tarjolla olevasta tiedosta. Käyttäjä käyttää vain 80 % tiedosta, ja raportti valmistuu aikavaatimuksen mukaisesti (Carvalho Costa ja Furtado, 2013, s. 87).

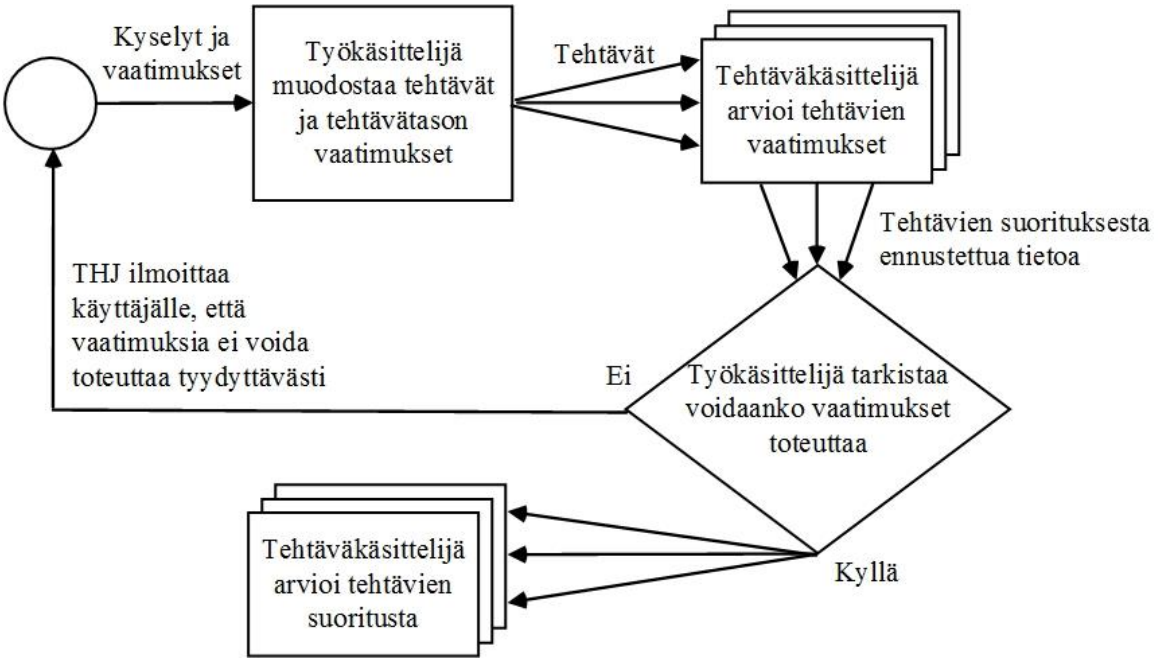
Käyttäjän odotusten huomioimiseen käytetään tiedonhakuvaatimuksia eli *data access requirements* (DAR) (Carvalho Costa ja Furtado, 2013, s. 87). DAR voi liittyä yhteen tietokantakomentoon, komentojoukkoon tai jopa suuren luokan tietokantatapahtumiin. Käyttäjä voi myös sivuuttaa vaatimukset kokonaan siten, että THJ hakee tiedot vain mahdollisimman nopeasti (Carvalho Costa ja Furtado, 2013, s. 87). Kuvasta 6 alla nähdään miten tiedonhakuvaatimukset voivat toimia kyselykielen kanssa. Operaatiossa on ensin kyselyyn kuuluvat tehtävät ja sen jälkeen on määritelty vaatimukset.



Kuva 6. Tiedonhakuvaatimukset

(Carvalho Costa ja Furtado, 2013, s. 90).

Kuvasta 7 alla nähdään käyttäjätyytyväisyyteen suuntautuneen THJ:n toiminta. Prosessi alkaa kaavion oikeasta yläkulmasta, jossa aluksi käyttäjä syöttää THJ:lle kyselyt ja tiedonhakuvaatimukset. Työkäsittelijä jakaa työn tehtäviksi, tässä tapauksessa kolmeksi tehtäväksi. Jokainen tehtävä saa omat tiedonhakuvaatimukset. Tehtäväkäsittelijän tehtävänä on arvioida, ovatko annetut vaatimukset mahdollista täyttää. Tehtäväkäsittelijän muodostama arvio lähetetään työkäsittelijälle. Työkäsittelijä vahvistaa tehtäväkäsittelijän tulokset siitä, että ovatko kyselyt toteutettavissa vaatimusten puitteissa. Työkäsittelijän tullessa tulokseen, että vaatimuksia ei voida täyttää, THJ ilmoittaa tästä käyttäjälle ja prosessi aloitetaan alusta. Tämä prosessi loppuu kun käyttäjän määrittämät vaatimukset voidaan täyttää ja työkäsittelijä lähettää tehtävät tehtäväkäsittelijälle.



Kuva 7. Käyttäjätyytyväisyyteen suuntautunut THJ (Carvalho Costa ja Furtado, 2013, s. 90).

4.5 Tiedonhallintajärjestelmien vertailu

Hajautetun THJ:n etuina ovat hajautetun ja kopioidun tiedon läpinäkyvä hallinta, luotettava tapahtumien suoritus hajautuksen avulla, parantunut suorituskyky ja helpompi järjestelmän laajentaminen. Hajautetussa THJ:ssä jokainen toimipaikka on vastuussa tietystä tietokannan osuudesta, ja siten kilpailu prosessointitehoista ei ole niin suurta kuin keskitetyssä THJ:ssä. Tiedon läpinäkyvän hallinnan etu on, että se tukee monimutkaisten ohjelmien kehittämistä. Järjestelmän laajentaminen on myös helpompaa ekonomisista syistä, tyypillisesti maksaa vähemmän koota järjestelmä useasta pienemmän tehon tietokoneesta kuin yhdestä suuren tehon tietokoneesta (Özsu ja Valduriez, 2011, s. 7, 14-15).

Hajautetun THJ:n huono puoli on, että hajautus tekee järjestelmästä monimutkaisemman. Verrattuna keskitettyyn THJ:än tapahtumien synkronisaatio on huomattavasti vaikeampaa. Tämän lisäksi resurssien kopiointi tuo huomattavasti lisäkuluja. Tiedon hajautus aiheuttaa myös tietoturvaongelmia, kun taas keskitetyssä THJ:ssä tieto sijaitsee yhdessä sijainnissa ja sen etuihin kuuluu helppo yksittäisen tietokannan suojaus (Özsu ja Valduriez, 2011, s. 16; Elmasri ja Navathe, 2010, s. 47).

Keskitetyn tiedonhallintajärjestelmän kaksitasoisen arkkitetuurin hyötynä on yksinkertaisuus ja saumaton yhteensopivuus vanhojen järjestelmien kanssa. Keskitetyssä tiedonhallintajärjestelmässä tieto sijaitsee yhdessä sijainnissa ja yksi suuri uhka on tämän yksittäisen toimipaikan menetys, kun taas hajautetussa THJ:ssä yksittäisen toimipaikan menetys ei ole välttämättä suuri ongelma. Etuihin kuuluu myös tiedontiivistäminen, joka helpottaa suurien tietomäärien siirtoa palvelimelta asiakkaille (Elmasri ja Navathe, 2010, s. 47, 49; Özsu ja Valduriez, 2011, s. 12).

Taulukossa 2 vertaillaan operatiivisia tiedonhallintajärjestelmiä ja tietovarastoja. Taulukosta ilmenevät oleellisimmat erot näiden järjestelmien välillä.

Taulukko 2. Operatiivisten järjestelmien ja tietovarastojen vertailu.
(Malinowski ja Zimányi, 2009, s. 42)

Kuvaus	**Operatiiviset tiedonhallintajärjestelmät**	**Tietovarastot**
Käyttäjätyyppi	Toimistotyöntekijät, operaattorit	Yrityksen johto
Tietokannan käyttö	Ennelta arvattavaa, toistuvaa	Ei ohjattua, tiettyä tarkoitusta varten
Tieto sisältö	Ajankohtaista, yksityiskohtaista	Historiallista, tiivistettyä
Tiedon organisointi	Käyttötarpeiden mukaan	Analysointiongelman mukaan
Tiedon rakenne	Optimoitu pienille tietokantatapahtumille	Optimoitu monimutkaisille kyselyille
Käytön määrä	Korkea	Keskitasosta matalaan
Tiedon haku/saanti	Tiedon luku, päivitys, poisto, sijoitus	Tiedon luku, vain lisäys
Tietueiden määrä per tapahtuma	Vähän	Paljon
THJ:n vasteaika	Lyhyt	Voi olla pitkä

Samanaikaisen käytön aste	Korkea	Matala
Lukkojen käyttö	Välttämätön	Ei tarpeen
Päivitysten määrä	Korkea	Ei yhtään
Tiedon toistuminen	Matala (normalisoidut taulut)	Korkea (ei normalisointia paljon)
Tiedon mallinnus	ER-malli	Moniulotteinen-malli

5 JOHTOPÄÄTÖKSET

Työssä tutkittiin eri relaatiomalliin perustuvia tiedonhallintajärjestelmiä (THJ) sekä niihin liittyviä valintakriteerejä. Taulukosta 1 voidaan huomata, että THJ:t voidaan luokitella hyvin monella tavalla ja THJ:n valintaan vaikuttavat monet valintakriteerit. Valittuja relaatiopohjaisia tiedonhallintajärjestelmiä olivat keskitetyt ja hajautetut tiedonhallintajärjestelmät, tietovarasto ja operatiiviset tiedonhallintajärjestelmät. Taulukosta 2 nähdään, miten tietovarastot ja operatiiviset THJ:t soveltuvat täysin eri tarkoituksiin. Operatiiviset tiedonhallintajärjestelmät on tarkoitettu päivittäiseen toimintaan, joissa tieto muuttuu jatkuvasti ja ne kuvastavat yrityksen ajantasaista informaatiota. Tietovarastot on taas tarkoitettu historiallisen tiedon varastointiin ja tiedon analysointiin, jolla voidaan selvittää hyödyllistä strategista informaatiota. Suorituskykyyn vaikuttaa suoraan tietokannan normalisaation taso, joka vaihtelee paljon varsinkin tietovaraston ja operatiivisen THJ:en välillä.

Hajautetun THJ:n etuina verrattuna keskitettyyn THJ:ään ovat hajautetun ja kopioidun tiedon läpinäkyvä hallinta, luotettava tapahtumien suoritus hajautuksen avulla, parantunut suorituskyky ja helpompi järjestelmän laajentaminen. Keskitetyn THJ:n etuihin kuuluu yksinkertaisuus ja tiedontiivistäminen.

Nykypäivän THJ:ssä käyttäjätyytyväisyys viittaa lähinnä suorituskykyyn ja asioihin kuten kyselyiden täydellisyyteen. Nykyaikaisen THJ:n täytyy vastata tiedon suojauksen tarpeisiin, varsinkin tietokannan palautuksen, tapahtumien rinnakkaisuuden, tietoturvan ja tiedon eheyden suhteen. Käyttäjätyytyväisyyteen keskittyvä THJ ottaa huomioon käyttäjien tarpeet raa’an suorituskyvyn sijaan. Käyttäjätyytyväisyyden tärkeys voi nousta tulevaisuudessa. Tämä tuo uusia haasteita ottaa käyttäjät huomioon uudella tavalla tiedonhakuvaatimusten muodossa.

Työssä otettiin huomioon vain relaatiomallia hyödyntävät tiedonhallintajärjestelmät. Muitakin tietomalleja käyttäviä THJ:ä olisi voinut tutkia, kuten olio-tiedonhallintajärjestelmiä tai olio-relaatio-tiedonhallintajärjestelmiä, mutta ne jätettiin tämän tutkimuksen ulkopuolella aiheen laajuuden takia.

LÄHTEET

Carvalho Costa, R. L. ja Furtado, P. (2013). *Providing Quality of Experience for Users: The Next DBMS Challenge*. Computer Society, Vol. 46. IEEE.

Codd, E.F. (1970). *A Relational Model of Data for Large Shared Data Bank.* Communications of the ACM, Vol. 13, No. 6.

Coronel, C., Morris, S. ja Rob, P. (2011). *Database Systems Design, Implementation and Management*, 10th ed. Cengage Learning: U.S.A.

Date, C. J. (2001). *An Introduction to Database Systems*, 6th ed. Addison-Wesley Publishing Company: U.S.A.

Date, C.J. (2008). *The Relational Database Dictionary,* Extended edition. Apress: U.S.A.

Elmasri, R. ja Navathe, S. B. (2010). *Fundamentals of Database Systems*, 6th ed. Addison-Wesley: U.S.A.

Garcia-Molina, H., Ullman, J. D. ja Widom, J. (2002). *Database Systems: The Complete Book.* Prentice Hall: U.S.A.

Gillenson, M. L. (1990). *Database Step-by-Step.* Wiley-Interscience: U.S.A.

Grant, F. (2012). *SQL Server 2012 Query Performance Tuning.* Apress:USA.

Hernandez, M. J. (2003). *Database Design for Mere Mortals*. Addison-Wesley: U.S.A.

Hovi, A., (1997). *Data Warehousing – Tietovarastotekniikka*. Suomen Atk-kustannus Oy: Suomi.

Malinowski, E. ja Zimányi, E. (2009). *Advanced Data Warehouse Design.* Springer Verlag: Germany.

Mathew, N. ja Stones, R. (2005). *Beginning Databases With PostgreSQL From Novice to Professional*, 2nd ed. Apress: U.S.A.

Mushtaq, M., Ahmed, T., ja Meddour, T.E. (2006). *Adaptive packet video streaming over P2P networks*. ACM International Conference Proceeding Series, Vol. 152, No. 59.

Pooley, R., Coady, J., Linger, H., Barry, C., Lang, L. ja Schneider, C. (2013). *Information Systems Development.* Springer Verlag: U.S.A.

Relaatiotietokantatermit, saatavilla: http://www.cs.helsinki.fi/u/laine/relaatiosanasto/rssuom_termit.html, viitattu [22.9.2013]

Ullman, J.D. ja Widom, J. (2002). *A first course in database systems*. Pearson Education International: U.S.A.

Wang, F., Pan, J. ja Jain, L. C. (2009). *Innovations in Digital Watermarking Techniques.* Springer Verlag: Germany.

Whitehorn, M. ja Marklyn, B. (2007). *Inside Relational Databases*. Springer Verlag: U.K.

Özsu, M.T., Valduriez, P. (2011). *Principles of Distributed Database Systems*. Springer Verlag: U.S.A.