Laadukas ja asiakaslähtöinen relaatiotietokanta

Turun yliopisto
Tietotekniikan laitos
LuK-tutkielma
Tietojenkäsittelytiede
Tammikuu 2025
Heidi Tuurala

TURUN YLIOPISTO
Tietotekniikan laitos

HEIDI TUURALA: Laadukas ja asiakaslähtöinen relaatiotietokanta

LuK-tutkielma, 27 s.
Tietojenkäsittelytiede
Tammikuu 2025

Relaatiotietokannat ovat suosituin käytössä oleva tietokantamalli, minkä takia niiden kehittäminen ja tutkiminen on edelleen ajankohtaista. Ohjelmistojen vaatimukset etenkin tietoturvasta ja pääsynvalvonnasta sekä kilpailu ohjelmistojen ja sovellusten välillä kasvaa. Näin ollen myös loppukäyttäjien vaatimukset ohjelmistoille kasvavat. Tässä tutkielmassa tarkastellaan, miten toteutetaan laadukas relaatiotietokanta sekä miten toteutuksessa voidaan huomioida asiakaslähtöisyys.
Tutkielma on toteutettu kirjallisuuskatsauksena. Tutkielma tarkastelee, millainen on laadukas relaatiotietokanta ja miten toteutuksessa voidaan huomioida asiakaslähtöisyys. Tuloksina nähdään, että laadukas relaatiotietokanta antaa kyselyä vastaavia hakutuloksia. Se on myös johdonmukainen, tietoturvallinen, helppokäyttöinen ja suorituskykyinen. Nämä ominaisuudet tekevät relaatiotietokannasta huomaamattoman ohjelmistoa käytettäessä. Laadukkaan relaatiotietokantojen hallintajärjestelmän toiminnot toteuttavat ACID:n. Laadukkaasti toteutettua relaatiotietokantaa ja relaatiotietokannan hallintajärjestelmää käyttävä ohjelmisto on esteetön, helppokäyttöinen ja suorituskykyinen. Teknisesti laadukkaan relaatiotietokannan ja tietokannan hallintajärjestelmän toteutus ei kuitenkaan vielä takaa ohjelmiston käyttökelpoisuutta. Niiden lisäksi toteutuksessa on huomioitava myös asiakkaiden ja ohjelmistokehittäjien näkemykset. Asiakaslähtöisyys voidaan huomioida optimoimalla tietokantaa ja priorisoimalla sen ominaisuuksia asiakkaiden näkemysten ja toiveiden mukaan.

Asiasanat: Relaatiotietokanta, Relaatiotietokannan hallintajärjestelmä, Asiakaslähtöisyys, Tietokanta, Tietokannan hallintajärjestelmä

Sisällys

1 Johdanto

Jokaiseen käytössä olevaan sovellukseen ja ohjelmistoon on tallennettu joukko tietoa. Tietokannat ja tietokantojen hallintajärjestelmät ovat kehitetty helpottamaan tietojen hallintaa. Tietokantoja on erilaisia ja näistä edelleen tilastojen mukaan maailmanlaajuisesti suosituin on 1970-luvulla kehitetty relaatiotietokanta. Statistan mukaan maailmanlaajuisesti suosituin tietokantojen hallintajärjestelmä oli heinäkuussa 2024 relaatiotietokantojen hallintajärjestelmä markkinaosuuden ollessa 72,8 % [1]. Kolme suosituinta tietokantojen hallintajärjestelmää heinäkuussa 2024 olivat Oracle, MySQL ja Microsoft SQL Server, jotka ovat relaatiotietokantojen hallintajärjestelmiä [2]. Sama tulos nähdään joulukuun 2024 tilastoissa DB-Engines Ranking -sivustolta [3]. Nämä osoittavat, että relaatiotietokannat ovat edelleen suosituimpia käytössä olevia tietokantoja, joten niiden tutkiminen ja kehittäminen on edelleen ajankohtaista.

Ohjelmistojen vaatimukset etenkin tietoturvasta ja pääsynvalvonnasta kiristyvät jatkuvasti sekä kilpailu ohjelmistojen välillä kasvaa. Näin ollen myös loppukäyttäjien vaatimukset ohjelmistoille kasvavat. Tässä tutkielmassa tarkastellaan, millainen on laadukas relaatiotietokanta sekä miten toteutuksessa voidaan huomioida asiakaslähtöisyys.

Tutkielman tutkimuskysymykset ovat:

- TK 1: Millainen on laadukas relaatiotietokanta?
- TK 2: Miten asiakaslähtöisyys voidaan huomioida relaatiotietokannan toteutuksessa?

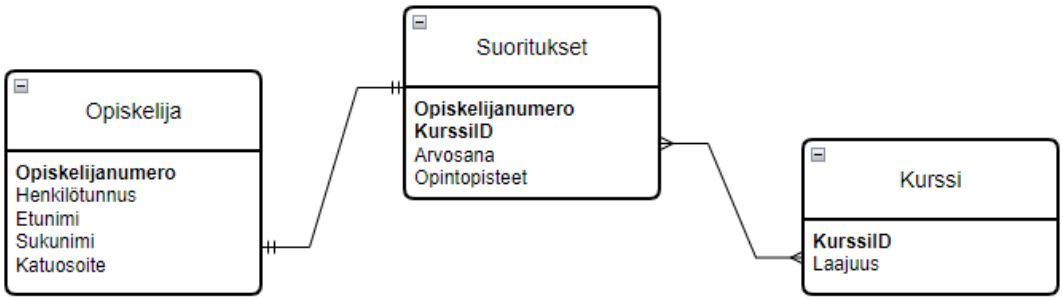
Luvussa 2 on relaatiotietokantojen historiaa, määritellään aiheen kannalta keskeiset

käsitteet ja perehdytään tapaustutkimukseen. Luvussa 3 kerrotaan työssä käytetyistä lähteistä ja aineiston keräämiseen käytetyistä menetelmistä. Luvussa 4 käsitellään relaatiotietokantojen teknisiä ominaisuuksia ja niiden toteutusperiaatteita. Luvussa 5 syvennytään relaatiotietokantojen asiakaslähtöisyyteen. Siinä pohditaan, miten relaatiotietokantojen asiakaslähtöisyyden toteuttaminen vaikuttaa niiden tekniseen toteutukseen. Tutkielman viimeisessä luvussa 6 on yhteenveto tutkielmasta. Lisäksi luvussa tehdään johtopäätökset ja esitellään niiden pohjalta tutkielman tulokset.

2 Relaatiotietokannat

Tietokannat ovat yksi vanhimmista tietotekniikassa käytettävistä teknologioista [4], ja relaatiotietokanta on suosituin käytössä oleva tietokantamalli [1]. Tietokantoihin tallennetaan joukko toisiinsa liittyviä tietoja, ja niissä on tallennetun tiedon lisäksi informaatiota tietojen välisistä suhteista toisiinsa [5]. Tiedot ovat näin ollen johdettavissa muista tietokantaan tallennetuista tiedoista [6] eli ne ovat toisistaan riippuvaisia ja tätä kutsutaan tietojen väliseksi *relaatioksi* (engl. relation). Tietokantojen tarkoitus on helpottaa tiedon käsittelyä niin, että käyttäjän tai käytettävän ohjelman ei itse tarvitse huolehtia tiedon käsittelystä. Tietojen käsittelyssä auttaa tietokannan ja ohjelman käyttäjän välissä toimiva tietokannan hallintajärjestelmä tai relaatiotietokannan hallintajärjestelmä, joka on tietokannan hallintajärjestelmän alatyyppi relaatiotietokannoille. [7]

Tietokannan hallintajärjestelmä (engl. database management system, DBMS) on osa ohjelmistoa, jossa sitä käytetään. Yksinkertaisesti hallintajärjestelmä kääntää loppukäyttäjän syöttämän kyselyn tietokannan vaatimiksi operaatioiksi sekä optimoi tietokantaa. Tietokannan hallintajärjestelmä tarjoaa käyttäjälle pääsyn tallennetun tiedon lukemiseen, poistamiseen ja muokkaamiseen. Yleisiä yritysten käyttämiä relaatiotietokannan hallintajärjestelmiä (engl. relational database management system, RDBMS) ovat esimerkiksi Oracle ja MySQL. Myös relaatiotietokantoja voidaan muokata ja hallita erilaisilla operaatioilla. Operaatioita ovat tiedon lisääminen ja poistaminen sekä tallennettujen tietojen lukeminen, poistaminen ja päivittäminen. [7]



Kuva 2.1: Yksinkertainen esimerkki relaatiotietokannan rakenteesta.

Relaatiotietokannassa (engl. relational database) tiedot tallennetaan tauluihin. Tauluissa tiedot esitetään riveillä ja sarakkeissa. Yleensä relaatiotietokannassa tieto esitetään yhdessä paikassa ja eri taulujen välinen yhteys on merkitty. Relaatiotietokannan rakennetta ja taulujen välisiä yhteyksiä on havainnollistettu kuvassa 2.1. Relaatiotietokantojen selkeän rakenteen ansiosta niihin tallennettua tietoa on helppo käsitellä [4].

Suurin osa esimerkiksi liiketoiminnan datasta on tallennettu relaatiotietokantoihin. Kun käyttää luotto- tai pankkikorttia tai verkkomaksamista, on luultavasti vuorovaikutuksessa relaatiotietokantojen kanssa. [8] Relaatiotietokannat ovat näin ollen todella yleisiä, niitä käyttävät myös suuret yhtiöt ja ne soveltuvat monenlaisen tiedon tallentamiseen. Tämän takia on tärkeää, että tietokannat toteutetaan laadukkaasti ja niiden toimintaan voidaan luottaa.

Kuvassa 2.2 on kuvitteellinen esimerkki relaatiotietokannan taulusta, johon on tallennettu kurssin tiedot. Taulun ensimmäisellä rivillä on taulun nimi eli `Kurssi` ja taulun toisella rivillä on omissa sarakkeissaan kurssin tiedot. Ensimmäisessä sarakkeessa on kurssin yksilöivä tunniste `kurssiID` ja toisessa sarakeessa on kurssin laajuus opintopisteinä. Näiden alle on omiin sarakkeisiinsa merkitty kurssien tiedot.

Kurssi	
KurssiID	Laajuus
1234567	4 op
6422908	8 op
7584902	2 op

Kuva 2.2: Esimerkki yksinkertaisesta relaatiotietokannan `Kurssi`-taulusta.

2.1 Historiaa

Relaatiotietokannan teorian on kehittänyt E.F. Codd 1970-luvulla. Relaatiotietomalli esiteltiin uutena tapana tallentaa tietoa tietokoneeseen. Ideana oli poistaa tietokantaan tallennetun tiedon päällekkäisyyttä ja auttaa välttämään epäjohdonmukaisuutta tietokannan sisällössä. Coddin tutkimus relaatiotietomalleista julkaistiin 1970. Silloin useimpien tietokantajärjestelmien toiminta perustui tiedon esittämiseen tietueiden ja osoittimien yhdistelmällä, tällaista tietokantaa sanottiin *navigointitietokannaksi* (engl. navigational database system). *Osoittimet* (engl. pointers) olivat muistiviitteitä, jotka kertoivat, missä kohtaa tietokoneen muistissa tietue tai taulu sijaitsi. Tällaisten tietokantojen käyttöön liittyi monia haasteita. Käyttäjän piti esimerkiksi tietää, miten tieto oli tallennettu tietokantaan ja miten tallennetut tiedot kytkeytyivät toisiinsa. Navigointitietokannat usein myös sisälsivät osoittimia, joihin oli luotu esivalmisteltuja hakuja. Hauilla, joita ei ollut esivalmisteltu, oli haastavaa tai jopa mahdotonta saada tuloksia. [8][9]

Codd loi tietokantamallin, josta piilotettiin tietokannan fyysisen rakenteen yksityiskohdat käyttäjältä, jotta ohjelmoijien työ helpottuisi ja he olisivat tuottavampia. Tämä teki tietokannasta huomattavasti yksinkertaisemman, koska ohjelmoijien piti jatkossa ymmärtää vain tauluista muodostuva tietomalli. Relaatiotietokannat eivät kuitenkaan aluksi saaneet suurta kannatusta ja niitä kyseenalaistettiin vahvasti. Monet yritykset, mukaan lukien tutkimusta rahoittanut IBM, olivat juuri panostaneet navigointitietokantaan, eikä niistä oltu valmiita luopumaan helposti. Esimerkiksi relaatiotietokantamallin suorituskykyä ja

automaatiota kyseenalaistettiin. Koska relaatiotietokantoihin ei uskottu, niiden toiminta saatiin todistettua vain rakentamalla relaatiotietokantajärjestelmä ja mittaamalla sen suorityskyky. [8]

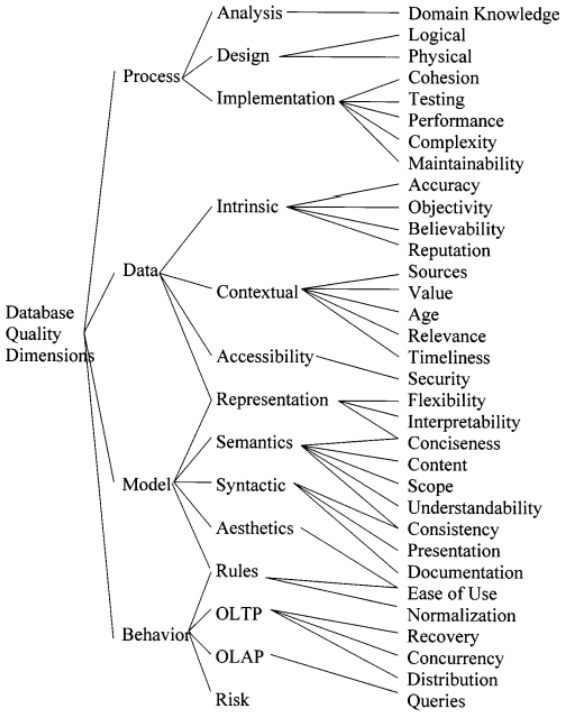
2.2 Laatuattribuutit

Relaatiotietokantojen laatua voidaan tarkastella kahdella tavalla: tarkastelemalla itse relaatiotietokannan laatua tai relaatiotietokantojen hallintajärjestelmän laatua. Christof Ebertin mukaan relaatiotietokannan tiedon luotettavuus (engl. integrity) voidaan taata, kun tietokannan hallintajärjestelmän toiminnot toteuttavat ACID:n [4]. ACID tulee sanoista atomisuus (engl. atomicity), eheys (engl. consistency), eristyneisyys (engl. isolation) ja pysyvyys (engl. durability). ACID:n *atomisuus* tarkoittaa sitä, että toiminto joko suoritetaan kokonaan tai sitä ei suoriteta ollenkaan. Tietokanta on *eheä*, kun se toimii johdonmukaisesti. Johdonmukaisuudella tarkoitetaan, että tapahtuma noudattaa ohjelman protokollaa eli toimii ennalta määritettyjen ohjeiden ja rajoitteiden mukaisesti. Lisäksi tietokanta on eheä, kun sen tila pysyy yhtenäisenä tapahtuman alusta loppuun, tila ei siis voi esimerkiksi muuttua vain osittain. Tapahtumat ovat *eristyneitä*, kun ne eivät häiritse tai estä toisiaan. *Pysyvyys* tarkoittaa tapahtumien pysyvyyttä eli tietokannan tila pysyy samana, kun esimerkiksi laite tai palvelin käynnistetään uudelleen. [10]

Esimerkiksi kun maksaa ostokset pankkikortilla, tietojärjestelmän rahojen siirtymiseen liittyvät prosessit eli tapahtumien sarja joko tapahtuu tai ei tapahdu. Näin ollen ei ole mahdollista, että rahat siirtyisivät vain osittain. Onnistuneen maksutapahtuman jälkeen pankkitilin saldo muuttuu. Jos maksutapahtuma epäonnistuu, tilin saldo ei muutu. Kun tilin saldo on muuttunut, sen tulee pysyä samana, kunnes tilin saldoon tulee uusi muutos. Jotta tietokanta on eheä, maksutapahtuman yhteydessä tietokannan prosessien on tapahduttava joka kerta edellä kuvatulla tavalla, eli järjestelmä toimii johdonmukaisesti.

Toisen näkökulman mukaan relaatiotietokannan toteutuksessa tavoitteena on luoda tietovarasto, joka toimii käyttäjälle fyysisenä ja toiminnallisena mallina hallittavista tietoko-

konaisuuksista. Relaatiotietokannan laatuominaisuudet voidaan John Hoxmeierin mukaan karkeasti jakaa neljään ominaisuuteen, jotka laadukkaasti toteutettuna takaavat myös laadukkaan relaatiotietokannan. [11] Laatuattribuutteja ei voi jakaa yksiselitteisesti omiin kategorioihinsa, koska niiden ominaisuuksissa on jonkin verran päällekkäisyyttä. Tätä on havainnollistettu kuvassa 2.3.



Kuva 2.3: Relaatiotietokannan laadun määrittävät tekijät havainnollistettuna kuvassa. [11]

Laatuominaisuuksia ovat menetelmät (engl. processes), tieto (engl. data), tietomalli (engl. model) ja käyttäytyminen (engl. behavior). Relaatiotietokannan *menetelmien* laadun tarkastelussa seurataan esimerkiksi sitä, että tietokanta toimii ennaltamäärättyjen ohjeistusten ja standardien mukaan. Jos suorituksesta löytyy poikkeama, se ratkaistaan ja prosessia muutetaan tilanteen vaatimalla tavalla. *Tiedon* laadusta puhuttaessa, syötetyn tiedon on esimerkiksi oltava virheetöntä. Jos tiedon laadussa on puutteita, käyttäjät voivat kokea, etteivät he voi luottaa järjestelmään tai he voivat tehdä päätöksiä vääränlaisen tiedon pohjalta. Tiedon laatuun liittyy laadukas *tietomalli* eli se, miten tietoa esitetään käytetyssä ohjelmassa. Laadukkaan tietomallin yksi tärkeä ominaisuus on semanttisuus. Haettaessa

tietoa saadut tulokset ovat relevantteja ja tuovat esitettyyn ongelmaan ratkaisun. Relaatiotietokannan käyttäytymisellä tarkoitetaan sen *käytettävyyttä*. Käytettävyyteen liittyy esimerkiksi se, miten hyviä ja tarkkoja hakutuloksia tietokanta antaa käyttäjälle. Lyhyesti voidaan myös sanoa, että huonosti toteutettua relaatiotietokantaa on haastava ja epämukava käyttää. [11]

Relaatiotietokannan normalisointi estää monia huonon relaatiosuunnittelun aiheuttamia ongelmia [7]. *Normalisointi* (engl. normalization) on relaatiotietokannan muokkaamista niin, että se noudattaa normaalimuotoja. *Normaalimuodot* (engl. normal forms) karsivat päällekkäisiä tietoja ja auttavat välttämään epäjohdonmukaisuutta relaatiotietokannassa. [8][7] Codd kehitti relaatiotietokantaan normaalimuodot, joita oli kolme erilaista – nykyään niitä on jo viisi. Normaalimuodot ovat numeroitu ja suurempi mallin luku kertoo edistyneemmästä normalisoinnista. Normalisoidussa relaatiotietokannassa tieto on tallennettu vain yhteen tauluun, mikä estää tietojen päällekkäisyydet ja tietokannan epäjohdonmukaisuuden. [5][7] Näin ollen yksi relaatiotietokannan toteutuksen tavoite on normalisoitu relaatiotietokanta [11].

Tietokantojen suunnitteluun voivat vaikuttaa esimerkiksi yrityksen vaatimukset, hinta, tallennettava tieto, turvallisuus ja tiedon eheys. Tietokannan suunnitteluun liittyy myös monenlaisia haasteita, kuten tietokannan integrointi toiseen järjestelmään. [11] Integrointi tarkoittaa tietokannan sisällyttämistä rajapinnan eli järjestelmien yhtymäkohdan kautta toiseen järjestelmään. Voidaan siis sanoa, että relaatiotietokannat ovat monimutkainen kokonaisuus, ja jo niiden suunnittelussa on otettava monia asioita huomioon.

2.3 Asiakaslähtöisyys

Asiakaslähtöisyydellä tarkoitetaan asiakkaan tarpeiden ja toiveiden huomioimista [12]. Tutkielmassa esitetyt relaatiotietokantojen laadukkaan toteutuksen määritelmät näkyvät ohjelmiston ja tietokannan käytettävyydessä, jotka vaikuttavat käyttökokemukseen. Näin ollen tarkastellaan asiakaslähtöisyyttä ohjelmiston käyttökokemuksen kautta. *Käyttökoke-*

mus (engl. user experience) on käyttäjän subjektiivinen kokemus ohjelman käytöstä, johon vaikuttavat esimerkiksi ohjelman käyttöliittymä ja käytettävyys. Aiemmin määriteltyjen laatuattribuuttien perusteella hyvä käyttökokemus tarkoittaa huomaamatonta, esteetöntä, suorituskykyistä ja helppokäyttöistä relaatiotietokantaa.

Relaatiotietokannan asiakaslähtöinen toteutus kulminoituu ohjelmiston käyttöliittymään. Hyvä käyttöliittymä ohjaa käyttäjää ja ohjelmisto käyttäytyy intuitiivisesti sekä käyttäjälle ymmärrettävällä tavalla [11]. Tavallisimpia käyttöliittymäsuunnitteluun liittyviä haasteita voidaan välttää *suunnittelumalleilla* (engl. design patterns). Ne tarjoavat ratkaisuja jo aiemmin toteutetuissa käyttöliittymissä todettuihin ongelmiin ja niiden toimivuus perustuu yleensä käytännön kokemukseen. [13] Käyttöliittymäsuunnittelussa tulee huomioida myös tietoturva ja pääsynvalvonta. Hyvin toteutettu käyttöliittymä antaa vain tietoja, joihin käyttäjällä on oikeus ja tietokantaan tallennetut tiedot on salattu väärinkäytösten varalta [14].

2.4 Tapaustutkimus

Hoxmeierin artikkelissa [11] käsitellään tapaustutkimusta, jossa arvioitiin suuren tulostin- ja skannerivalmistajan tuotantotietokannan laatua. Lopputuloksena sovelluksessa oli useita puutteita, jotka tekivät siitä huonolaatuisen ja se sijoittui kokonaislaadun perusteella todella huonosti. Tutkimuksen lopuksi sovellukseen annettiin useita parannusehdotuksia, joilla sovelluksen saisi toimivammaksi. Vaikka tutkimuksessa keskityttiin relaatiotietokannan laadun arvioimiseen, parannusehdotuksia tuli myös sovelluksen käyttöliittymään. Parannusehdotuksissa oli esimerkiksi käyttöliittymän muokkaaminen niin, että se ohjaa käyttäjää toimimaan työvaiheiden mukaisesti. Tähän liittyen puutteita oli raportoitu esimerkiksi siitä, että sovellus antoi käyttäjän syöttää järjestelmään tiedot, jotka olivat siellä jo. Lisäksi raporttien tekemiseen meni monta tuntia, eikä siihen ollut tilanilmaisinta. [11] Jos tietokanta olisi rakennettu hyvin, toiminnot olisivat toimineet nopeasti ja sitä olisi mie-

lekäs käyttää. Hyvällä käyttöliittymäsuunnittelulla olisi voinut vaikuttaa myös siihen, ettei tietokantaan lisätä samoja tietoja uudestaan.

Jakob Nielsen on luonut mallin *Jacob's Law of Internet User Experience* eli vapaasti suomennettuna Jacobin laki internetin käyttökokemuksesta. Mallissa on kyse siitä, että käyttäjät käyvät eri verkkosivuilla ja he muodostavat niiden perusteella mielikuvan, miten verkkosivu toimii. Kun käyttäjä menee uudelle verkkosivulle, verkkosivun tulee vastata tähän käyttäjän mielikuvaan. [15] Tässä tapaustutkimuksessa esiin tuotuihin asioihin mallia voisi soveltaa esimerkiksi tilanilmaisimeen. Kun käyttäjä antaa sovellukselle komennon, niin käyttäjä odottaa saavansa reaktion sovellukselta. Kun ladataan raportti, niin sovelluksen pitäisi antaa esimerkiksi palkki, jossa näkyy, miten pitkällä lataus on. Tutkimuksen mukaan tilanilmaisinta ei ollut, eli käyttäjä ei todennäköisesti saanut mitään tietoa siitä, onko raporttia alettu lataamaan tai miten pitkä aika lataamiseen kuluu. Tapaustutkimuksen tulosten ja parannusehdotusten perusteella voidaan todeta, että käyttäjän ohjaaminen on keskeisessä asemassa, kun halutaan luoda hyvä käyttökokemus. Käyttäjää ohjaamalla voidaan esimerkiksi tarvittaessa paikata tietokannan puutteita tiettyyn pisteeseen asti.

3 Menetelmät

Tässä tutkielmassa on käytetty tutkimusmenetelmänä kirjallisuuskatsausta. Työssä käytetyt lähteet ja hakulausekkeet ovat englanninkielisiä ja ne ovat haettu ACM Digital Library, Web of Science, IEEE Xplore ja Volter -tietokannoista. Lähteissä on kaksi oppikirjaa ja muut lähteet ovat vertaisarvioituja tutkimuksia. Sopivien aineistojen löytäminen vaati useita eri hakutermejä ja erilaisia rajauksia ennen kuin löydettiin tutkielmassa käytetyt lähteet. Useimmat lähteet ovat valittu niiden otsikon ja tiivistelmän perusteella tietokannoista ja osa lähteistä on kerätty tutkielmassa käytettyjen aineistojen lähdeluetteloista tai niitä hyödyntämällä. Relaatiotietokantojen tekniseen toteutukseen liittyvät tutkimukset eivät ole kovin uusia, koska relaatiotietokantojen teoria ja tekniikka on niin vanhaa, että 2000-luvun alun jälkeen julkaistuissa tutkimuksissa käsitellään usein liian yksityiskohtaisia aiheita tutkielman aiheelle.

Lähteitä etsittiin erilaisilla hakulausekkeilla riippuen siitä, mihin aiheeseen lähdettä tarvittiin. Tietokannoista haettiin esimerkiksi hakulausekkeella ”relational database” AND quality, joka tuotti 1000 – 100 000 tulosta riippuen tietokannasta. Riippuen siitä, mihin asiaan haettiin lähdettä, hakulauseke oli yleensä ”relational database” AND *asia jota etsittiin*. Esimerkiksi ”relational database” AND ”customer-oriented” tai ”relational database” AND ACID. Hakulauseke rajattiin tiivistelmään, jonka jälkeen tuloksia oli hyvin vaihtelevasti 20 – 10 000. Jos hakutuloksia oli yli tuhat, selattiin muutaman ensimmäisen sivun otsikot ja niiden perusteella luettiin tiivistelmät. Yhtä aineistojen hakuprosessin osaa

Kuva 3.1: Aineistojen kerääminen hakulausekkeella ”relational database” AND ”user friendly”.

on havainnollistettu kuvassa 3.1. Iso osa normalisointiin liittyvistä aineistoista on kerätty muiden aineistojen lähdeluetteloista tai lähdeluettelojen lähteitä hyödyntämällä.

4 Relaatiotietokantojen laadukas toteutus

Tässä luvussa tarkastellaan tarkemmin sitä, mitä ominaisuuksia on laadukkaalla relaatiotietokannalla: miten se tulee toteuttaa ja miten asiakaslähtöisyys voidaan huomioida relaatiotietokannan toteutuksessa. Laadukas relaatiotietokanta on optimoitu ja toteuttaa laatuattribuutit. Laadukkaan ja asiakaslähtöisen relaatiotietokannan ominaisuuksia tuodaan esiin useissa tutkielmassa käytetyistä lähteistä, joista keskeisimmät on koottu taulukkoon 4.1. Taulukkoon valitut lähteet käsittelevät pääasiassa tutkielman kannalta keskeisiä aiheita eli laatuattribuutteja, optimointia ja asiakaslähtöisyyttä. Aineistot, jotka vain täydensivät tutkielmaa, on jätetty taulukon ulkopuolelle, kuten [16] ja [17].

Kuten luvussa 2 todettiin, ohjelmiston käyttökokemukseen vaikuttaa relaatiotietokannan toteutuksen laatu. Relaatiotietokannan tiedon laatuun edelleen vaikuttaa relaatiotietokannan hallintajärjestelmän laatu. Hyvin toteutettua relaatiotietokantaa on helppo ja vaivaton käyttää, mikä näkyy esimerkiksi ohjelmiston hyvänä suorituskykynä [11]. Verrattuna ennen relaatiotietokantoja käytössä olleisiin navigointitietokantoihin ja niiden hallintajärjestelmiin, relaatiotietokannassa on paremmat edellytykset esimerkiksi suurten tietomäärien käsittelyyn. Tämän mahdollistaa relaatiotietokannan hyvä suunnittelu ja yksinkertaisempi rakenne, jotka tekevät siitä helpomman käyttää eikä resursseja kulu liikaa tietokannan sisällön tai hallintajärjestelmän käytön opetteluun [18].

Tietokantojen hallintajärjestelmät kehitettiin 1970-luvulla vastaamaan loppukäyttäjien

Taulukko 4.1: Tutkielman keskeisimmät aineistot, jotka käsittelevät relaatiotietokannan laatuattribuutteja, optimointia ja asiakaslähtöisyyttä.

	Laatu-attribuutit	Normalisointi	Redundanssi	Asiakas-lähtöisyys
G. Vial, ”Different Databases for Different Strokes”, (2018)	x	x		
C. Delobel, ”Normalization and hierarchical dependencies in the relational data model”, (1978)		x		
E. F. Codd, ”Derivability, redundancy and consistency of relations stored in large data banks”, (1969)			x	x
J. L. Harrington, ”Relational Database Design and Implementation: Clearly Explained, 3rd ed.”, (2009)	x	x	x	
B. W. Wade ja D. D. Chamberlin, ”IBM Relational Database Systems: The Early Years”, (2012)		x		x
N. Banothu ym., ”Big-data: Acid versus base for databa- se transactions”, (2016)	x			x
J. A. Hoxmeier, ”Typology of database quality factor” (1998)	x	x		x
N. Elmasri ym., ”Fundamentals of database systems”, (2011)		x	x	
K. Sugihara ym., ” Semantic Approach to Usability in Relational Database Systems”, (1984)	x			x

rajusti kasvavaan tarpeeseen kaupallisiin ja teollisuuden sovellusohjelmiin. Tietokantojen hallintajärjestelmien toivottiin ratkaisevan yleisimmät ongelmat, joita ohjelmoijat kohtasivat syöttäessään järjestelmään tietoa tai tulostaessaan tietoa järjestelmästä. Sen aikaiset navigointitietokannat olivat kuitenkin rakenteeltaan niin monimutkaisia, että myös niiden hallintajärjestelmiä oli haastava käyttää, eivätkä hallintajärjestelmät sellaisenaan tuoneet toivottua tulosta. Tämän takia kehitettiin relaatiotietokanta ja relaatiotietokantojen hallintajärjestelmä. Tietokantojen hallintajärjestelmät ovat alunperin kehitetty, jotta voidaan vastata loppukäyttäjien, eli tässä tapauksessa ohjelmistoja tuottavien yritysten asiakkaiden, tarpeisiin parempaan tietokantojen tiedonhallintaana. [18] Tietokantojen hallintajärjestelmät eivät kuitenkaan toimi optimaalisesti, jos hallintajärjestelmä tai tietokanta on huonosti suunniteltu ja toteutettu.

4.1 Normalisointi

Codd kehitti teorian, jossa tietokantaan tallennettujen tietojen välillä on relaatio. Relaatio tarkoittaa sitä, että tietokantaan tallennettu tieto voidaan johtaa toisesta tietokantaan tallennetusta tiedosta. Teorian taustalla on matematiikka, joten tiedot esitettiin aluksi matriiseissa ja relaatiot toimivat kuten matematiikan relaatiot. [6]

Codd esitti relaatioiden olevan *redundantteja* (engl. redundant), jos ne toistuivat tietokannassa monta kertaa. Hän määritteli redundanteille relaatioille kaksi eri tyyppiä: vahvan ja heikon redundanssin (engl. redundancy). Coddin paperissa kerrotaan, että tietovarastojen hallinnoimisesta vastaavien henkilöiden tulisi olla tietoisia redundanssista. Redundanssi voi aiheuttaa epäjohdonmukaisuutta, ja sitä voidaan käyttää hyväksi luvattomissa muutoksissa. [6] Toiston poistamiseksi ja epäjohdonmukaisuuden välttämiseksi Codd esitti myöhemmin ratkaisuna normaalimuodot [8].

Normaalimuodossa esitetyt relaatiotietokannan taulut on esitetty mahdollisimman yksinkertaisessa muodossa. Jos sarakkeissa on tallennettuna sama tieto, ylimääräinen tieto poistetaan. Kuvassa 4.1 on havainnollistettu relaatiotietokannan puurakenne ja normali-

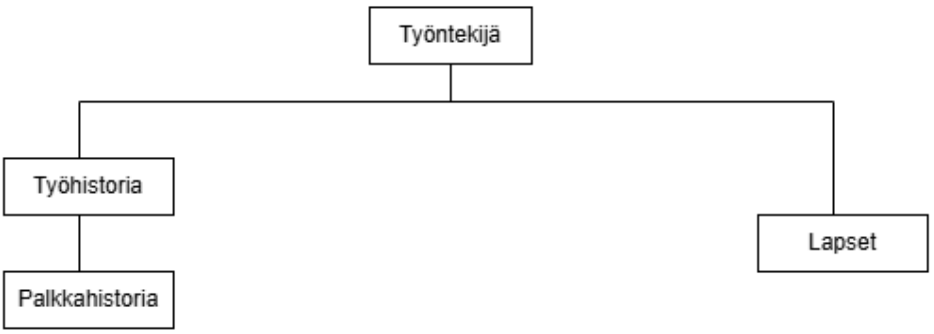
soiminen. Relaatiotietokannassa tiedot ovat järjestetty hierarkisesti toisiinsa nähden, ja puurakenne kuvastaa tätä hierarkiaa. Kuvasta 4.1 nähdään, että (a)-kohdassa normalisoimattomassa muodossa on tallennettu samoja tietoja monta kertaa, kuten `Työhistoria` ja `Lapset`, joiden tiedot on tallennettu `Työntekijä`-taulun lisäksi omiin tauluihinsa. [19]

Normaalimuoto muodostetaan aloittamalla puun ylimmästä taulusta, eli tässä tapauksessa `työntekijä`-taulusta. Valitaan pääavain ja sijoitetaan se jokaiseen suoraan `Työntekijä`-taulun alapuolella olevaan tauluun. [19] *Pääavain* (engl. primary key) on sarake tai yhdistelmä sarakkeita, jotka yksilöivät taulun rivit [20]. `Työntekijä`-tauluun tallennetut tiedot periytyvät `Työhistoria`- ja `Lapset`-tauluihin, joten `Työhistoria`- ja `Lapset`-taulujen tietoja ei tarvitse olla lisäksi tallennettuna `Työntekijä`-tauluun. `Työhistoria`-taulun tiedot periytyvät jälleen `Palkkahistoria`-taululle. Kuvan 4.1 (b)-kohdassa on normalisoidun rakenteen tiedot. [19]

4.2 Laadukkaan relaatiotietokannan teknisiä toteutusperiaatteita

Hyvin suunniteltu relaatiotietokanta on normalisoitu [16]. Normalisoidussa relaatiotietokannassa on relaatiotietokannan perusperiaatteen mukaisesti yksi tieto tallennettu yhteen paikkaan [5]. Kun relaatiotietokannalle tehdään kysely, haluttu tieto haetaan taulusta, johon se on tallennettu. Jos halutaan hakea useaa toisiinsa liittyvää tietoa, jotka on tallennettu eri tauluihin, on tehtävä useita kyselyitä. [20]

Elmasrin ja Navathen esittävät [20], että toisinaan on tarpeen käyttää kontrolloitua redundanssia. *Kontrolloitu redundanssi* (engl. controlled redundancy) tarkoittaa sitä, että vastoin normalisoinnin ideaa, samoja tietoja tallennetaan useaan tauluun. Esimerkiksi jos tiedetään, että relaatiotietokannasta halutaan yleensä löytää opiskelijan nimi ja kurssin numero niin, että mukana on myös kurssin arvosana, opiskelijanumero ja ryhmän tunniste, voidaan hyödyntää kontrolloitua redundanssia. [20]



(a) Normalisoimattomat taulut:
Työntekijä (man#, nimi, syntymäpäivä, työhistoria, lapset)
Työhistoria (päivämäärä, nimike, palkkahistoria)
Palkkahistoria (palkkapäivä, palkka)
Lapset (lapsenNimi, syntymävuosi)}

(b) Normalisoidut taulut:
Työntekijä' (man#, nimi, syntymäpäivä)
Työhistoria' (man#, päivämäärä, nimike)
Palkkahistoria' (man#, päivämäärä, palkkapäivä, palkka)
Lapset' (man#, lapsenNimi, syntymävuosi}

Kuva 4.1: Relaatiotietokannan normalisoitu puurakenne [19]

(a) Arvosanaraportti

Opiskelijanumero	Opiskelijan_nimi	Jakso_ID	Kurssinumero	Arvosana
17	Smith	112	Matikka2410	B
17	Smith	119	CS1310	C
8	Brown	85	Matikka2410	A
8	Brown	92	CS1310	A
8	Brown	102	CS3320	B
8	Brown	135	CS3380	A

(b) Arvosanaraportti

Opiskelijanumero	Opiskelijan_nimi	Jakso_ID	Kurssinumero	Arvosana
17	Brown	112	MATH2410	B

Kuva 4.2: Opiskelijoiden tiedot on tallennettu yhteen tauluun, jotta ne on helpompi löytää. [20]

Sen sijaan, että tehdään monta hakua ja etsitään erikseen yksittäisistä tiedostoista kaikki halutut tiedot, ne voidaan tallentaa yhteen tauluun, mikä nopeuttaa kyselyiden tekemistä ja tiedon hakua. Tälläista taulua on havainnollistettu kuvassa 4.2 kohdassa (a). Menetelmää kutsutaan *denormalisoinniksi* (engl. denormalization). Kuitenkin jos halutaan hyödyntää denormalisointia, tietokantojen hallintajärjestelmän on pystyttävä kontrolloimaan redundanssia, jotta relaatiotietokanta pysyy johdonmukaisena. Jos redundanssia ei kontrolloida, taulusta tulee epäjohdonmukainen ja alkuperäisten taulujen tiedot tallennetaan väärin. [20] Tietokantojen hallintajärjestelmä, joka pystyy kontrolloimaan redundanssia, ilmoittaa, jos samalla arvolla on tallennettu aiemmin tietoja tai tiedot, joita yritetään tallentaa, eivät liity toisiinsa. Kuvan 4.2 kohdassa (b) nähdään, että redundanssia ei ole kontrolloitu ja tauluun on tallennettu virheellisesti opiskelijan Brown opiskelijanumeroksi 17.

Tiedot etsitään relaatiotietokannan tauluista erilaisten tiedot yksilöivien tunnisteiden eli avainten avulla. Tämän takia on tärkeää, että samaa tietoa ei ole tallennettu moneen paikkaan tai se on tehty kontrolloidusti. Avaimia on muutamia erilaisia eri tarkoituksiin. Yksi avaintyypeistä on jo aiemmin mainittu rivit yksilöivä pääavain. Pääavaimen arvo on valittava tarkkaan, koska sillä on vaatimuksia: pääavaimen arvo ei saa olla null eikä sen tule ikinä muuttua. Pääavaimen arvo ei saa olla sellainen, jonka arvo voi olla null, koska

tietokannassa se tarkoittaa tuntematonta arvoa ja toisaalta sen saattaa asettaa useita kertoja, jolloin pääavain ei enää ole uniikki. Pääavaimen arvo ei myöskään voi olla sellainen, joka voi muuttua. Jos pääavaimen arvo muuttuu, voi seurauksena olla, ettei alkuperäistä pääavainta tiedetä ja haluttuja tietoja ei enää löydetä tietokannasta. [7][20]

Tietokannan suunnittelu on tärkeä osa ohjelmistokehitystä [21], koska tietokanta liittyy olennaisesti ohjelmiston suorituskykyyn [8]. *Hyvin muotoiltu* (engl. well-formed) relaatiotietokanta toteuttaa aiemmin määritellyt tekniset vaatimukset liittyen tiedon eheyteen ja laatuun sekä tietokannan normalisointiin ja suorituskykyyn. Toisaalta se ei vielä takaa käyttökelpoista tietokantaa. Hyvin muotoillussa tietokannassa puutteita voi olla esimerkiksi tiedon saavutettavuudessa. [11] Suunnitteluvaiheessa on mietittävä, mitä tietoja tietokantaan täytyy pystyä tallentamaan ja miten tietoja yleensä halutaan tai on helpoin käyttää. Tämän pohjalta suunnitellaan tietokannan rakenteen ja taulujen hierarkian lisäksi se, miten tiedot on järjestetty tietokantaan suhteessa toisiinsa. Jos tiedot, jotka eivät liity toisiinsa, tallennetaan samaan tauluun tai taulun pääavaimeksi valitaan arvo, joka ei täytä pääavaimen kriteereitä, on lopputuloksena heikosti toimiva relaatiotietokanta [20]. Lähtökohtaisesti relaatiotietokannan pitäisi olla normalisoitu, mutta toisinaan on ohjelmiston suorituskyvyn kannalta tarpeen hyödyntää myös redundanssia ja tietokannan denormalisointia.

5 Relaatiotietokantojen asiakaslähtöinen toteutus

Asiakaslähtöisyydessä on monta puolta: asiakkaalle ensivaikutelma yrityksestä ja tuotteesta ovat tärkeitä ostopäätöksen kannalta [17], toisaalta järjestelmän odotetaan olevan suorituskykyinen. Ensivaikutelma muodostuu asiakkaan omien näkemysten ja kokemusten pohjalta ja siihen vaikuttavat erityisesti visuaaliset tekijät. Sitä seuraavat kokemukset liitetään ensivaikutelmaan, minkä takia hyvä ensivaikutelma on tärkeä. [17] Kun puhutaan relaatiotietokannoista, ensivaikutelma ohjelmistosta ja tietokannasta luodaan käyttöliittymällä. Käyttöliittymä käsittää ohjelmiston käytettävyyden ja ulkoasun, joista käyttäjä huomioi ulkoasun ensimmäisenä.

Käyttöliittymä on olennainen osa ohjelmistoa, koska se on lähimpänä käyttäjää. Ohjelmisto, jossa on hyvä käyttöliittymä, on asiakaslähtöinen, koska sitä on miellyttävä käyttää. [22] Käyttöliittymien ominaisuuksista on olemassa valmiita malleja, jotka ovat hyväksi todettuja ja joita kannattaa hyödyntää ohjelmistokehityksessä [13]. Toisaalta toisen tutkimuksen mukaan päätöksentekoa ohjaavat vinoumat eli päätöksentekoa vääristävät omat näkemykset. Ohjelmistokehityksessä esimerkiksi voidaan valita ongelmaan ratkaisuksi tuttu ja turvallinen teknologia, joka ei välttämättä edes ratkaise ongelmaa tai tehdään asiat samalla tavalla, koska se toimi aiemminkin sen sijaan, että mietitään, miten asiat voisi tehdä paremmin. Tähän perustuen tutkimuksen mukaan ohjelmistosuunnittelun ja -kehityksen apuna voi käyttää ohjelmistosuunnitteluprosesseja ja metodologioita, mutta

ne eivät takaa onnistunutta lopputulosta. Näiden rinnalla pitäisi käyttää *pehmeitä taitoja* (engl. soft-skills). Pehmeitä taitoja käyttävä ohjelmistosuunnittelija kerää mahdollisimman paljon olennaista tietoa käyttäjiltä ja ohjelmoijilta, ja punnitsee kerättyjen tietojen pohjalta ristiriitoja ja niiden etuja ja haittoja. [23]

5.1 Relaatiotietokannan asiakaslähtöiset ominaisuudet

Relaatiotietokannan odotetaan olevan suorituskykyinen ja toimivan intuitiivisesti [24]. Sen laadukkaan toteutuksen takaavat laadukkaasti toteutetut menetelmät, tietomalli, käyttäytyminen ja tiedon tallentaminen [11]. Relaatiotietokannan suorituskykyä ja optimointia voidaan parantaa normalisoimalla se mahdollisimman pitkälle sekä suunnittelemalla taulujen hierarkia mahdollisimman toimivaksi [5]. Toisaalta välillä optimointia on tietokannan denormalisointi, jotta kyselystä saadaan nopeampi [20], mutta se voi menettää hyötynsä, jos redundanssin kontrolloinnissa on puutteita [6]. Redundanssi voi tehdä relaatiotietokannasta epäjohdonmukaisen, mikä taas heikentää ohjelmiston käyttökokemusta ja käytettävyyttä.

Tietokannan tiedon hallintaa helpottaa tietokannan hallintajärjestelmä. Tietokannasta ja sen hallintajärjestelmästä on eniten hyötyä, kun molemmat ovat suunniteltu ja toteutettu hyvin. Laadukkaalle relaatiotietokannan hallintajärjestelmälle on määritelty ACID:n mukaiset toimintaperiaatteet [10]. Kuten relaatiotietokannan toteutuksessa, myös tietokannan hallintajärjestelmän toteutuksessa on optimoitava ominaisuuksia, jotta järjestelmä toimii mahdollisimman nopeasti ja tehokkaasti [21].

Banothu ym. [10] mukaan erityisesti isoissa ohjelmistoissa kaikki ACID:n mukaiset ominaisuudet, eli toimintojen atomisuus, eheys, eristyneiyys ja pysyvyys, eivät välttämättä toteudu. Esimerkiksi Amazon.com saattaa laiminlyödä toimintojen eristyneisyyttä näyttämällä kahdelle asiakkaalle, että molemmat ovat saaneet ostoskoriinsa viimeisen tuotteen sillä riskillä, että toinen asiakas jää ilman tuotetta, mutta etuna on, että järjestelmä toimii nopeasti. Todennäköisyys sille, että toinen asiakas jää ilman tuotetta, on kuitenkin todella

pieni, joten Amazon.comille on kannattavampaa ottaa riski, että molemmat tilaavat tuotteen ja mahdollisesti hyvittää se myöhemmin toiselle asiakkaalle kuin tehdä järjestelmästä hidas. [10] Kuten yleensä ohjelmistokehityksessä, kyse on tässäkin tapauksessa optimoinnista ja priorisoinnista.

Asiakkaille halutaan hyvä ensivaikutelma ja käyttökokemus, joten käytetyn aineiston perusteella voidaan päätellä, että Amazon.com on päättänyt priorisoida verkkosivujen nopeutta sen pienen todennäköisyyden edelle, että joutuu hyvittämään asiakkaalle tuotteen. Toisaalta, jos niin käy, tämä todennäköisesti heikentää asiakkaan käyttökokemusta ja mielikuvaa Amazon.comista. Amazon.com-sivustolla kävi vuonna 2023 heinä-joulukuun aikana Statistan mukaan noin 2,5 miljardia kävijää kuukaudessa [25]. Kun puhutaan tällaisista suurista yrityksistä ja niiden ohjelmistoista, yksittäisen asiakkaan huono kokemus on todennäköisesti pienempi ongelma yritykselle ja sen asiakaskunnalle kuin suuren joukon huono kokemus ja sitä mahdollisesti seuraava laajalle levinnyt huono maine. Tämän takia voi toisinaan olla kannattavaa joustaa ennalta määrätyistä ominaisuuksista, ja tehdä ratkaisuja tilanteen vaatimalla tavalla.

Ohjelmistokehityksessä on relaatiotietokannan toteutuksessa otettava huomioon monta asiaa liittyen ohjelmiston optimointiin ja suorituskykyyn, kuten normalisointi ja tietojen päällekkäisyys. Coddin mukaan kehittäjien on itse oltava tietoisia näistä. [6] Relaatiotietokantaa voi kontrolloida itse, mutta ohjelmistojen vaatimukset, ja erityisesti vaatimukset tietoturvasta ja pääsynvalvonnasta, kasvavat koko ajan [21]. Yleisesti tieto- ja kyberturvallisuudesta puhuttaessa sanotaan, että ”ihminen on järjestelmän heikoin lenkki”, millä tarkoitetaan sitä, että ihminen tekee inhimillisiä virheitä ja on helposti harhautettavissa – toisin kuin ohjelmisto. Tämä voi aiheuttaa puutteita ja tietoturvariskejä ohjelmistoon. Tutkielmassa käytetyn aineiston pohjalta, voidaan sanoa, että relaatiotietokannan kontrollointiin on etenkin tietoturvaa ajatellessa hyvä hyödyntää relaatiotietokannan hallintajärjestelmää, joka auttaa kehittäjiä pitämään huolen, että vaatimukset täyttyvät. Hallintajärjestelmä voi tehdä ohjelmistosta turvallisemman käyttää, koska ainakin suosituimpaan re-

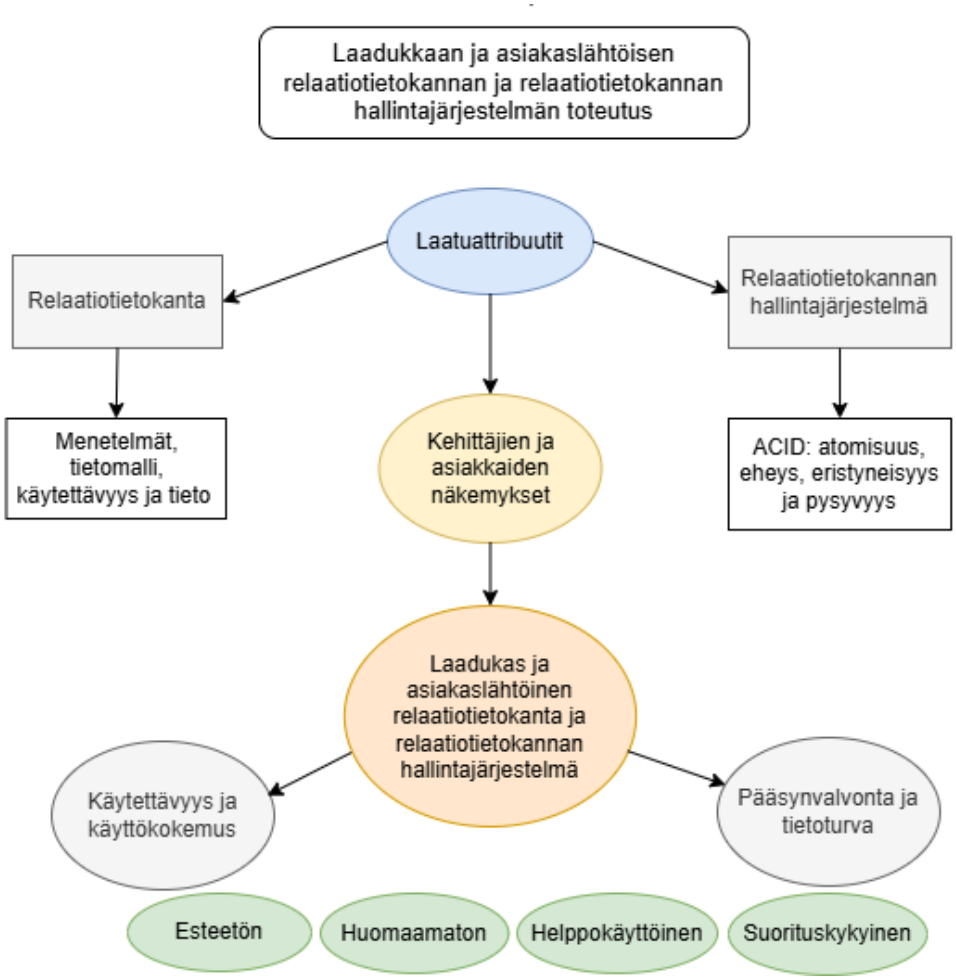
laatiotietokantojen hallintajärjestelmään Oracleen [2], voi esimerkiksi luoda automaatiota, kuten laukaisimia (engl. triggers), vahtimaan tietojen käsittelyä ja tietokannan toimintoja. Lisäksi relaatiotietokannan hallintajärjestelmä käsittelee tietoa nopeasti, mikä parantaa ohjelmiston suorituskykyä ja käyttökokemusta [21].

5.2 Pohdinta: malli relaatiotietokannan laadukaasta ja asiakaslähtöisestä toteutuksesta

Tutkielmassa käytettyjen lähteiden perusteella tiedetään, että on olemassa valmiita malleja, joiden mukaan ohjelmistoja voidaan suunnitella, jotta niihin saadaan hyvä ja helppokäyttöinen käyttöliittymä. Päätöksiä kuitenkin ohjaavat mieltymykset, joten ratkaisuja ongelmiin on tarkasteltava kriittisesti ja välttää automaattisesti luottamasta tuttuun ja turvalliseen ratkaisuun – se ei välttämättä tuo lisäarvoa ohjelmistolle. Ohjelmistosuunnitteluun kannattaa käyttää erilaisia yleisesti hyväksi todettuja metodologioita. Esimerkiksi ATAM eli *riskinhallintaprosessi* (engl. Architecture Tradeoff Analysis Method) on metodologia, joka ohjaa kompromissien tekemisessä ohjelmistokehityksessä [26]. Asiakaslähtöisyys on kuitenkin ohjelmiston käyttäjien eli asiakkaiden näkemysten huomioimista. Amazon.com ei täytä kaikkia ACID:n kriteereitä, mutta se on suorituskykyinen ja toimii intuitiivisesti, joten se antaa käyttäjille hyvän ensivaikutelman ja sen myötä yleensä hyvän käyttökokemuksen. Tämän perusteella voidaan todeta, että yleisesti laadukkaan relaatiotietokannan tai relaatiotietokannan hallintajärjestelmän kriteerit täyttävä ohjelmisto on paperilla varmasti hyvä, mutta se ei välttämättä ole asiakaslähtöinen.

Kirjallisuuteen perustuen loin mallin laadukkaan ja asiakaslähtöisen relaatiotietokannan toteutuksesta. Malli on havainnollistettu kuvassa 5.1. Kuvasta nähdään, että relaatiotietokantaa ja relaatiotietokannan hallintajärjestelmää suunnitellessa on huomioitava laatuattribuutit. Kun laatuattribuutit on huomioitu, kuullaan kehittäjiä ja loppukäyttäjiä. Heidän näkemysten perusteella priorisoidaan laatuattribuutteja ja tietokannan ominaisuuksia.

Isoissa järjestelmissä voidaan priorisoida esimerkiksi järjestelmän nopeutta tiedon eheyden edelle, kun taas pienemmissä järjestelmissä tämä ei välttämättä ole kannattavaa tai edes tarpeellista käyttökokemuksen kannalta. Lopulta saadaan laadukas ja asiakaslähtöinen relaatiotietokanta. Sitä on helppo käyttää, se on suorituskykyinen ja tieto on helposti saatavilla. Nämä takaavat sen, että relaatiotietokanta toimii loppukäyttäjälle huomaamattomasti ohjelmiston taustalla.



Kuva 5.1: Itse luotu malli laadukkaan ja asiakaslähtöisen relaatiotietokannan ja relaatiotietokannan hallintajärjestelmän toteutuksesta.

6 Johtopäätökset

Laadukkaasti toteutettu relaatiotietokanta antaa edellytykset sille, että ohjelmiston menetelmät noudattavat ennalta määrättyä protokollaa, relaatiotietokanta on helppokäyttöinen sekä antaa kyselyihin täsmällisiä ja luotettavia vastauksia. Laadukkaasti toteutetun relaatiotietokannan hallintajärjestelmän toiminnot taas toimivat ACID:n mukaisesti. ACID:n toteuttava relaatiotietokannan hallintajärjestelmä toimii johdonmukaisesti, tietokannan tila pysyy samana, kunnes tallennettuihin tietoihin tehdään muutoksia ja tapahtumat eivät häiritse tai estä toisiaan.

Tietokanta on olennainen osa ohjelmistoa ja näkyy suoraan sen käytettävyydessä. Laadukkaan relaatiotietokannan tai relaatiotietokannan hallintajärjestelmän määritelmän mukaan toteutettu järjestelmä antaa pohjan laadukkaalle ohjelmistolle, mutta ei vielä takaa sen käyttökelpoisuutta tai asiakaslähtöisyyttä. Kun toteutetaan relaatiotietokantaa asiakaslähtöisesti, erilaisia ohjelmistosuunnitteluprosesseja ja metodologioita on hyvä käyttää ohjelmistosuunnittelussa ja -toteutuksessa, mutta niiden rinnalla on tärkeää kuunnella myös ohjelmistokehittäjiä ja asiakkaita eli loppukäyttäjiä. Hyvin muotoiltu relaatiotietokanta ei takaa käyttökelpoista tietokantaa, koska olennaista on, mitä tietoja tietokantaan tallennetaan ja miten tiedot on tallennettu suhteessa toisiinsa. Lisäksi asiakkailta ja ohjelmistokehittäjiltä voi saada näkemyksiä siihen, mitä ominaisuuksia järjestelmässä halutaan tai kannattaa priorisoida.

Ohjelmisto, jossa on käytössä laadukas ja asiakaslähtöinen relaatiotietokanta ja relaatiotietokannan hallintajärjestelmä, ei välttämättä toteuta kaikkia laadukkaan tietokannan

tai sen hallintajärjestelmän toteutusperiaatteita. Ohjelmiston on pääasiassa noudatettava relaatiotietokannoille ja niiden hallintajärjestelmille määriteltyjä laatuattribuutteja, mutta näistä voi toisinaan olla tarpeellista joustaa, jos se parantaa ohjelmiston käyttökokemusta.

Vastauksena ensimmäiseen tutkimuskysymykseen *TK1* voidaan todeta, että laadukas relaatiotietokanta täyttää laatuattribuutit, jotka ovat: relaatiotietokannan menetelmät, tietomalli ja käytettävyys sekä tallennetun tiedon laatu. Laadukas relaatiotietokanta on optimoitu normalisoimalla ja toisinaan voidaan hyödyntää myös redundanssia ja denormalisointia. Denormalisointi vaatii kuitenkin myös relaatiotietokannan hallintajärjestelmän, joka pystyy käsittelemään redundanssia, jotta relaatiotietokanta pysyy johdonmukaisena. Laadukkaan relaatiotietokannan hallintajärjestelmän menetelmät täyttävät ACID:n ominaisuudet eli ne ovat atomisia, eheitä, eristyneitä ja pysyviä.

Toiseen tutkimuskysymykseen *TK2* voidaan vastata, että asiakaslähtöinen relaatiotietokanta on huomaamaton, esteetön, suorituskykyinen ja helppokäyttöinen. Lisäksi relaatiotietokanta on tietoturvallinen ja huolehtii pääsynvalvonnasta. Toisinaan tällaisen tietokannan toteuttaminen tarkoittaa jostain laatuattribuutista, kuten toimintojen eristyneisyydestä, joustamista. Relaatiotietokantaa kehitettäessä on kuunneltava asiakkaita ja ohjelmistokehittäjiä. Ohjelmistossa, jossa on asiakaslähtöinen relaatiotietokanta, on hyvä käyttöliittymä ja sitä on mielekäs käyttää. Laatuattribuutit täyttävä relaatiotietokanta ja relaatiotietokannan hallintajärjestelmä ovat siis varmasti teoriassa hyviä, mutta ne eivät välttämättä ole asiakaslähtöisiä.

Laadukkaan ja asiakaslähtöisen relaatiotietokannan toteutuksen tutkimista rajoittaa optimoiminen. Relaatiotietokantaa optimoidaan eri tavoilla riippuen siitä, mitä ominaisuutta tai ominaisuuksia halutaan priorisoida sekä miten tietokanta on rakennettu. Kun järjestelmässä muutetaan jotain, se yleensä vaikuttaa myös muualla järjestelmässä. Tämä voi ilmetä esimerkiksi ohjelmiston suoritusvirheinä tai muuten heikompana suorituskykynä. Relaatiotietokannan ollessa laaja kokonaisuus, muutoksista aiheutuvat haitat saattavat tulla vasta päivien kuluttua ilmi, kun uusi versio on jo julkaistu tuotantoon. Silloin voi

olla todella haastavaa tai jopa mahdotonta paikantaa, mistä virheet tai ei-toivotut ominaisuudet johtuvat. Lisäksi relaatiotietokannan toteutuksessa vaikuttavat ohjelmistokehittäjien ja -suunnittelijoiden omat näkemykset, koska ei ole olemassa yksiselitteistä vastausta siihen, miten relaatiotietokanta tulisi optimoida tai mitä ominaisuuksia on priorisoitava. Jatkotutkimuksena voisi selvittää tarkemmin, miten tutkielmassa esitetyt laatuattribuutit ja relaatiotietokannan tekniset ominaisuudet vaikuttavat toisiinsa – mitkä ovat toisensa poissulkevia tekijöitä ja mitkä taas tukevat toisiaan. Tämä voisi auttaa tekemään entistä suorituskykyisempiä järjestelmiä.

Lähdeluettelo

[1] Statista, 2024. url: https://www.statista.com/statistics/1131595/worldwide-popularity-database-management-systems-category/.

[2] Statista, 2024. url: https://www.statista.com/statistics/809750/worldwide-popularity-ranking-database-management-systems/.

[3] DB-engines, 2024. url: https://db-engines.com/en/ranking.

[4] G. Vial, ”Different Databases for Different Strokes”, *IEEE Software*, vol. 35, nro 2, s. 80–85, maaliskuu 2018, ISSN: 0740-7459, 1937-4194. DOI: 10.1109/MS.2018.1661308. url: https://ieeexplore.ieee.org/document/8314165/ (viitattu 06. 10. 2024).

[5] C. Delobel, ”Normalization and hierarchical dependencies in the relational data model”, *ACM Trans. Database Syst.*, vol. 3, nro 3, s. 201–222, syyskuu 1978, ISSN: 0362-5915. DOI: 10.1145/320263.320271. url: https://doi.org/10.1145/320263.320271.

[6] E. F. Codd, ”Derivability, redundancy and consistency of relations stored in large data banks”, *SIGMOD Rec.*, vol. 38, nro 1, s. 17–36, kesäkuu 2009, Association for Computing Machinery, ISSN: 0163-5808. DOI: 10.1145/1558334.1558336. url: https://doi.org/10.1145/1558334.1558336.

[7] J. L. Harrington ja J. L. Harrington, ”Relational database design and implementation : Clearly explained”, teoksessa *Relational Database Design and Implementa-*

tion : Clearly Explained, 3rd ed., Amsterdam ; Morgan Kaufmann/Elsevier, 2009, ISBN: 1-282-25841-9.

[8] B. W. Wade ja D. D. Chamberlin, ”IBM Relational Database Systems: The Early Years”, *IEEE Annals of the History of Computing*, vol. 34, nro 4, s. 38–48, 2012. DOI: 10.1109/MAHC.2012.48.

[9] J. Delplanque, A. Etien, N. Anquetil ja O. Auverlot, ”Relational Database Schema Evolution: An Industrial Case Study”, teoksessa *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2018, s. 635–644. DOI: 10.1109/ICSME.2018.00073.

[10] N. Banothu, S. Bhukya ja K. V. Sharma, ”Big-data: Acid versus base for database transactions”, teoksessa *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, 2016, s. 3704–3709. DOI: 10.1109/ICEEOT.2016.7755401.

[11] J. A. Hoxmeier, ”Typology of database quality factors”, *Software Quality Journal*, vol. 7, s. 179–193, 1998, Springer.

[12] Kotimaisten kielten keskus, *Kielitoimiston sanakirja*, 2024. url: https://www.kielitoimistonsanakirja.fi/asiakasl%C3%A4ht%C3%B6inen.

[13] T. Mikkonen, ”Formalizing design patterns”, teoksessa *Proceedings of the 20th International Conference on Software Engineering*, sarja ICSE ’98, Kyoto, Japan: IEEE Computer Society, 1998, s. 115–124, ISBN: 0818683686.

[14] F. Buschmann, R. Meunier ja H. Rohnert, *Sommerlad, P, Stal. M. A System of Patterns*, John Wiley & Sons Ltd., Pattern-Oriented Software Architecture, 1996.

[15] nngroup, 2024. url: https://www.nngroup.com/videos/jakobs-law-internet-ux/.

[16] A. Imbulpitiya, J. Whalley ja M. Senapathi, ”A Qualitative Study of Novice Experiences with Database Normalisation”, teoksessa *Proceedings of the 26th Australasian Computing Education Conference*, sarja ACE ’24, Sydney, NSW, Australia: Association for Computing Machinery, 2024, s. 94–103, ISBN: 9798400716195. DOI: 10.1145/3636243.3636254. url: https://doi.org/10.1145/3636243.3636254.

[17] T. Roman, A. Manolică ja A. R. Jelea, ”The First Meeting with a Brand. Does the First Impression Matter?”, *STRATEGICA*, s. 335,

[18] E. F. Codd, ”Relational database: a practical foundation for productivity”, teoksessa *ACM Turing Award Lectures*. New York, NY, USA: Association for Computing Machinery, 2007, s. 1981, ISBN: 9781450310499. url: https://doi.org/10.1145/1283920.1283937.

[19] E. F. Codd, ”A relational model of data for large shared data banks”, *Commun. ACM*, vol. 26, nro 1, s. 64–69, tammikuu 1983, ISSN: 0001-0782. DOI: 10.1145/357980.358007. url: https://doi.org/10.1145/357980.358007.

[20] N. Elmasri, *Fundamentals of database systems*. Pearson Education, Boston, MA, USA, 2011.

[21] Z. Li, L. Zhang ja Y. Ren, ”Database Design of Network Teaching System Based on Oracle DBMS”, teoksessa *2023 IEEE International Conference on Integrated Circuits and Communication Systems (ICICACS)*, 2023, s. 1–6. DOI: 10.1109/ICICACS57338.2023.10099756.

[22] U. Kartoun, ”A user, an interface, or none”, *Interactions*, vol. 24, nro 1, s. 20–21, joulukuu 2016, Association for Computing Machinery, ISSN: 1072-5520. DOI: 10.1145/3014046. url: https://doi.org/10.1145/3014046.

[23] A. Tang, ”Software designers, are you biased?”, teoksessa *Proceedings of the 6th International Workshop on SHAring and Reusing Architectural Knowledge*, sar-

ja SHARK '11, Waikiki, Honolulu, HI, USA: Association for Computing Machinery, 2011, s. 1–8, ISBN: 9781450305969. DOI: 10.1145/1988676.1988678. url: https://doi.org/10.1145/1988676.1988678.

[24] K. Sugihara, J. Miyao, T. Kikuno ja N. Yoshida, "A Semantic Approach to Usability in Relational Database Systems", teoksessa *1984 IEEE First International Conference on Data Engineering*, Los Angeles, CA, USA: IEEE, huhtikuu 1984, s. 203–210, ISBN: 978-0-8186-0533-8. DOI: 10.1109/ICDE.1984.7271273. url: https://ieeexplore.ieee.org/document/7271273/ (viitattu 06. 10. 2024).

[25] *Worldwide visits to amazon.com from July 2023 to December 2023*, Statista, 2023. url: statista.com/statistics/623566/web-visits-to-amazoncom/.

[26] R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. Lipson ja J. Carriere, "The architecture tradeoff analysis method", teoksessa *Proceedings. Fourth IEEE International Conference on Engineering of Complex Computer Systems (Cat. No.98EX193)*, IEEE, 1998, s. 68–78. DOI: 10.1109/ICECCS.1998.706657.