Relaatio- ja ei-relaatiotietokantojen suorituskykyerot

Turun yliopisto
Tietotekniikan laitos
TkK-tutkielma
Tietotekniikka
Joulukuu 2024
Kasimir Alamäki

TURUN YLIOPISTO
Tietotekniikan laitos

Kasimir Alamäki: Relaatio- ja ei-relaatiotietokantojen suorituskykyerot

TkK-tutkielma, 20 s.
Tietotekniikka
Joulukuu 2024

Tietojärjestelmät ja sovellukset käyttävät tietokantoja datan hallinnoimiseen. Ne voidaan jakaa kahteen pääryhmään: relaatio- ja ei-relaatiotietokantoihin. Kyseisillä tietokannoilla on merkittäviä nopeuseroja, jotka riippuvat monista eri asioista. Suorituskyky voi vaikuttaa huomattavasti järjestelmän toimintanopeuteen, joten tehokkaan tietokannan valinta on tärkeää.

Tässä tutkielmassa tutkitaan, miten tietokantatyyppien suorituskyvyt eroavat toisistaan eri tilanteissa. Kirjallisuuskatsauksessa vertaillaan tietokantojen operaatioiden nopeuksia ja tarkastellaan niihin vaikuttavia syitä. Lisäksi tutkitaan, mihin sovelluksiin tietyntyyppiset tietokannat soveltuvat ja miten relaatiotietokantojen suorituskykyä pystytään parantamaan horisontaalisella skaalauksella.

Tutkielmassa käy ilmi, että ei-relaatiotietokannat suoriutuvat lähes kaikista operaatioista nopeammin kuin relaatiotietokannat. Koostefunktioita käyttävissä operaatioissa relaatiotietokannat ovat kuitenkin nopeampia, koska ne ovat sisäänrakennettuja SQL:ssä. Muut operaatiot ovat hitaampia ACID-ominaisuuksien tarkistusten takia, jotka pitävät datan eheänä. Täten relaatiotietokannat sopivat parhaiten sovelluksiin, joissa käsitellään kriittisiä kohteita, kuten pankkikorttimaksuja. Ei-relaatiotietokannat soveltuvat sosiaalisen median palveluihin ja analyyttisiin sovelluksiin niiden skeemattomuuden ja horisontaalisen skaalautuvuuden ansiosta. Suorituskykyeroihin vaikuttavat tietokantojen toteutustavat, versiot, palvelinten komponentit, datan rakenne sekä attribuuttien ja rivien määrä.

Asiasanat: relaatiotietokanta, ei-relaatiotietokanta, suorituskyky, SQL, NoSQL

Sisällys

1 Johdanto

Tietokannat ovat erittäin tärkeä osa tietojärjestelmien ja sovellusten arkkitehtuuria. Niitä käytetään usein sovelluksissa, joissa suuria datamääriä halutaan määritellä, päivittää, hakea ja hallita helposti. Tietokannat mahdollistavat myös dataan pääsyn samanaikaisesti, joten niiden käyttö on suosittua verkkosovelluksissa.

Tietokannat voidaan jakaa kahteen päätyyppiin, relaatio- ja ei-relaatiotietokantoihin (engl. *NoSQL*). Tämä tutkielma keskittyy kyseisten tietokantatyyppien suorituskykyeroihin, sillä väärän tietokantatyypin valinta voi heikentää merkittävästi tietojärjestelmän toimintaa ja sovelluksen käyttäjäkokemusta. Lisäksi tutkielmassa käsitellään suorituskykyerojen syitä, tietokantojen parhaita käyttökohteita ja relaatiotietokantojen optimointimahdollisuuksia.

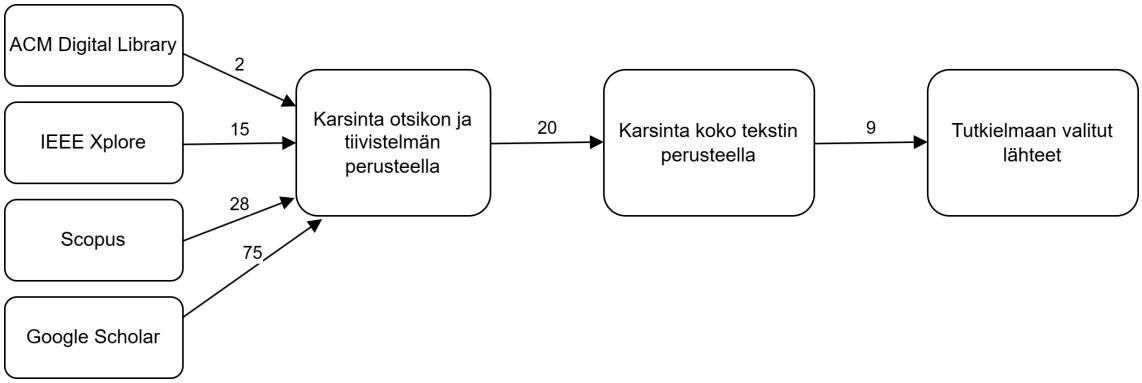
TK1: Millaisia suorituskykyeroja relaatio- ja ei-relaatiotietokantojen välillä on?

TK2: Mistä suorituskykyerot voivat johtua?

TK3: Mihin sovelluksiin relaatio- ja ei-relaatiotietokannat soveltuvat?

Kirjallisuuskatsauksen tutkimusartikkelit haettiin informaatioteknologian tunnetuista tietokannoista aikavälillä 7.–8.10.2024. Hakulause *(nosql AND sql) AND (compa* OR perf*)* muodostettiin tutkimuskysymysten perusteella. Haku suoritettiin ACM Digital Library-, IEEE Xplore-, Scopus- ja Google Scholar -tietokannoissa. Hakutulokset rajattiin julkaisuaikavälille 2013–2024, jotta voidaan huomata tietokantojen suorituskykymuutokset viimeisen 12 vuoden ajalta. Lisäksi haku rajattiin

koskemaan vain englanninkielisiä tutkimusartikkeleita. Niiden karsinta tapahtui valitsemalla mielenkiintoisimmat ensin otsikon ja tiivistelmän perusteella. Näistä artikkeleista lopullinen karsinta tapahtui lukemalla ne kokonaan läpi. Suurin osa pois karsituista artikkeleista ei keskittynyt tarpeeksi tietokantatyyppien väliseen suorituskykyvertailuun. Kuva 1.1 havainnollistaa tutkielman lähteiden valintaprosessia.



Kuva 1.1: Tutkielman lähteiden valintaprosessi.

Luvussa 2 esitellään relaatio- ja ei-relaatiotietokannat, tietokantaoperaatiot ja datan rakenne. Luvussa 3 käsitellään tutkielman kirjallisuuskatsaus, jossa analysoidaan, mitä tutkimusten suorituskykytesteissä on mitattu. Katsauksen pohjalta esitellään suorituskykytestien päätelmät sekä ristiriidat. Lisäksi pohditaan, mitkä ominaisuudet vaikuttavat tuloksiin, mihin sovelluksiin tietyt tietokannat soveltuvat ja miten relaatiotietokantojen suorituskykyä pystytään parantamaan horisontaalisella skaalauksella. Luvussa 4 on tutkielman yhteenveto.

2 Tietokannat ja tietokantatyypit

Tietokannat ovat toisiinsa liittyvän datan kokoelmia, jotka suunnitellaan ja luodaan kuvaamaan oikeaa maailmaa. Niitä voidaan käyttää esimerkiksi ostotapahtumien hallintaan, jolloin tietokanta päivittyy jokaisen ostotapahtuman yhteydessä. [1, s. 4–5] Nykymarkkinoilla olevat tietokannat voidaan jakaa kahteen päätyyppiin, relaatio- ja ei-relaatiotietokantoihin [2]. Niistä yleisin tietokantatoteutus tällä hetkellä on relationaalinen. Ei-relaatiotietokannoista on kuitenkin tullut entistä merkittävämpiä nopeasti kasvavan datamäärän tietojärjestelmissä [3], kuten sosiaalisessa mediassa ja tunnetuimmissa verkkokaupoissa. [1, s. 3]

Tietokantojen hallintaa varten käytetään tietokannanhallintajärjestelmiä (engl. *Database Management System, DBMS*). Kyseiset järjestelmät mahdollistavat tietokannan alustamisen, jakamisen käyttäjille ja sovelluksille sekä operaatioiden suorittamisen. Operaatiot, kuten datan lukeminen ja kirjoittaminen ovat erittäin merkittävä osa tietokantojen suorituskykymittauksissa [4], [5].

Tietokannoissa olevaa dataa voidaan säilöä jäsennellysti (engl. *structured*), osittain jäsennellysti (engl. *semistructured*) tai jäsentelemättömästi (engl. *unstructured*). Jäsennellyssä datassa samantyyppiset data-alkiot ovat rakenteeltaan identtisiä. [1, s. 426] Niiden tallentamista varten tietokanta tarvitsee skeeman eli tarkasti suunnitellun tietokantarakenteen, joka ei muutu usein [1, s. 34]. Relaatiotietokantoja käytetään pääasiassa jäsennellyn datan käsittelemiseen. Osittain jäsennellyssä datassa samantyyppisillä data-alkioilla voi olla samankaltainen rakenne, mutta ei

kuitenkaan identtinen. Tällöin dataa voidaan kerätä ennen kuin tiedetään, miten sitä tallennetaan ja käytetään. Jäsentelemättömässä datassa alkioiden rakenne on vain hyvin rajallisesti määritelty. [1, s. 426–428] Useimmiten ei-relaatiotietokannat käsittelevät osittain jäsenneltyä dataa, mutta ne voivat käsitellä myös jäsenneltyä ja jäsentelemätöntä dataa. [1, s. 887–896]

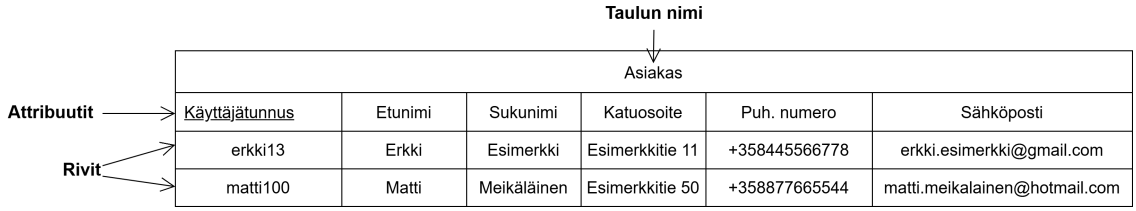
2.1 Relaatiotietokannat

Ted Codd esitteli kehittämänsä relationaalisen tietomallin vuonna 1970 ja keräsi sillä heti huomiota sen yksinkertaisuuden sekä matemaattisuuden ansiosta. Ensimmäiset kaupalliset toteutukset tulivat saataville 1980-luvun alussa. Relaatiomallia on sittemmin käytetty lukuisissa kaupallisissa ja avoimen lähdekoodin (engl. *open source*) tietokannoissa. [1, s. 149]

Suurin osa relaatiotietokannoista on vertikaalisesti skaalautuvia, eli tietokannan suorituskykyä voidaan parantaa päivittämällä sitä suorittavan palvelimen komponentteja, kuten prosessoria, RAM-muistia tai tallennustilaa [5]. Relaatiotietokantojen transaktioilla eli operaatiojoukoilla on oltava ACID-ominaisuudet (engl. *Atomicity, Consistency, Isolation, Durability*) eli atomisuus, eheyden säilyttäminen, eristyneisyys ja pysyvyys. Atomisuus tarkoittaa, että transaktio on suoritettava kokonaan tai ei ollenkaan. Eheyden säilyttäminen tarkoittaa, että tietokannan tila on eheä transaktion alussa ja lopussa. Eristyneisyydellä kuvataan, kuinka samanaikaiset transaktiot eivät häiritse toistensa suoritusta. Pysyvyys tarkoittaa, että transaktioiden muutosten on aina pysyttävä tietokannassa. Se mahdollistaa, että data ei katoa edes virhetilanteissa. [1, s. 757–758]

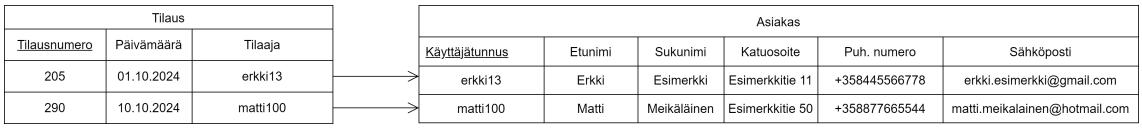
Relaatiomallissa tietokanta on kokoelma entiteettien välisiä suhteita. Entiteettiä voidaan kuvata tauluna (engl. *table*), jonka jokainen rivi kuvastaa yhtä uniikkia esiintymää (engl. *instance*). Taulun attribuuteilla kuvataan rivien erilaisia ominaisuuksia, jotka voivat olla tietotyypeiltään esimerkiksi merkkijonoja, numeroita tai

päivämääriä. Jokaisen rivin yksittäisellä attribuutilla on sama tietotyyppi. [1, s. 150–151] Kaikkien rivien on oltava uniikkeja, joten jokaisella taululla on rivin identifioiva pääavain (engl. *primary key*), joka ei voi olla `NULL`. Jotta taulujen välillä voi olla suhde, täytyy taululla olla viiteavain (engl. *foreign key*), jolla viitataan toisen taulun pääavaimeen. Relaatiomallia kuvatessa pääavain on alleviivattuna, eli kuvan 2.1 pääavain on käyttäjätunnus. [1, s. 158–163]



Kuva 2.1: Esimerkki attribuuteista ja riveistä Asiakas-taulussa.

Kuvassa 2.2 Tilaus-taulun viiteavain on tilaaja, joka viittaa Asiakas-taulun pääavaimeen eli käyttäjätunnukseen muodostaen taulujen välille suhteen. Kyseinen suhde on yhden suhde moneen, eli yhdellä asiakkaalla voi olla monta tilausta, mutta yksittäinen tilaus kuuluu vain yhdelle asiakkaalle. Tätä suhdetta merkitään 1:N. Muita taulujen välisiä suhteita ovat yhden suhde yhteen eli 1:1 ja monen suhde moneen eli M:N. [1, s. 76]



Kuva 2.2: Esimerkki taulujen välisestä suhteesta viiteavaimen avulla.

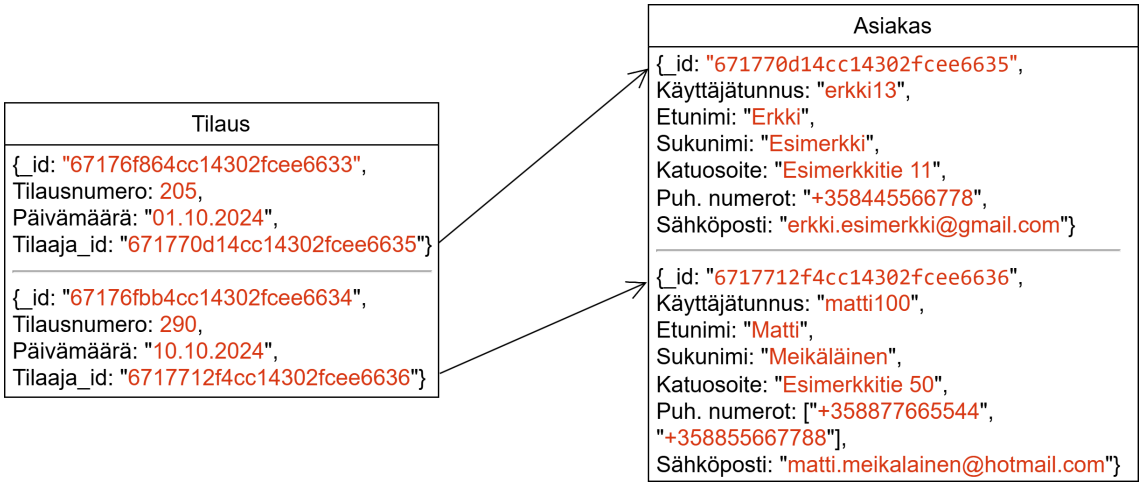
Tietokantaoperaatiot ovat olennainen osa vertailtaessa tietokantojen suorituskykyä. Dataa pystytään käsittelemään neljällä pääoperaatiolla, jotka ovat datan luominen, lukeminen, päivittäminen ja poistaminen. Kyseisistä operaatioista käytetään lyhennettä CRUD (engl. *Create, Read, Update, Delete*). [1, s. 887]

Relaatiotietokannat käyttävät kyselykielenään SQL:ää (engl. *Structured Query Language*) operaatioiden suorittamiseen. Sen muodostuminen standardikieleksi mahdollisti käyttäjien pystyvän jo 1980-luvulla vaihtamaan tietokannasta toiseen helposti opettelematta uutta kyselykieltä. Se on yksi pääsyistä, miksi kaupallisista relaatiotietokannoista tuli niin menestyksekkäitä. [1, s. 177] SQL:n uniikkeja ominaisuuksia ovat liittymisoperaatiot, joilla taulujen välille voidaan luoda suhteita niiden pää- ja vierasavainten avulla [1, s. 215].

2.2 Ei-relaatiotietokannat

Ei-relaatio- eli NoSQL -tietokantoja (engl. *Not Only SQL*) käytetään sovelluksissa, joissa relaatiomallin skeema rajoittaisi liikaa datan käsittelyä. Hyvä esimerkki tämänkaltaisesta sovelluksesta on Facebook, jossa miljoonat käyttäjät luovat julkaisuja, joissa voi tekstin lisäksi olla kuva- tai videotiedostoja. Näiden julkaisujen on tarpeen mukaan oltava näkyvissä vain julkaisijan kavereille. Käyttäjäprofiilien, julkaisujen ja julkaisijoiden välisien suhteiden säilömiseen saatetaan tarvita monia erilaisia ei-relaatiotietokantoja. Ne voidaan jakaa neljään pääryhmään: dokumentti-, avain-arvo-, sarake- ja graafitietokantoihin. [1, s. 884–888]

Dokumenttitietokannat tallentavat dataa tyypillisesti samankaltaisten dokumenttien kokoelmissa. Kokoelman dokumentit ovat siis keskenään samankaltaisia, mutta niiden attribuutit voivat kuitenkin erota toisistaan. Jokaisella dokumentilla on uniikki tunniste (engl. *ID*), jota voidaan verrata relaatiomallin pääavaimeen. Dokumenttitietokantoja käytetään pääasiassa osittain jäsennellyn datan käsittelemiseen [6]. Data tallennetaan yleensä serialisointiformaatissa, kuten JSONissa (engl. *Javascript Object Notation*) tai BSONissa (engl. *Binary JSON*). Eräs tunnettu dokumenttitietokanta on MongoDB. [1, s. 890–891] Kuva 2.3 on esimerkki dokumenttitietokannasta, jossa attribuuttien sisäistys on toteutettu listana, kun puhelinnumeroita on useampi kuin yksi.



Kuva 2.3: Esimerkki dokumenttitietokannasta, MongoDB:n dokumentaation [7] pohjalta.

Avain-arvotietokannat ovat yksinkertaisia, koska jokaista arvoa pystytään käsittelemään nopeasti uniikilla avaimella. Arvojen rakenne voi vaihdella paljon eri tietokannoissa ja sovelluksissa. Niitä voidaan tallentaa muun muassa jäsentelemättömästi [8], merkkijonoina, relaatiomallimaisesti riveinä, JSON-objekteina, kuten kuvassa 2.4 tai jonain muina itsekuvailevina datamuotoina. Eräs tunnettu avain-arvotietokanta on Amazonin kehittämä DynamoDB. [1, s. 895–896]

Avain	Arvo
671770d14cc14302fcee6635	{Käyttäjätunnus: "erkki13", Etunimi: "Erkki", Sukunimi: "Esimerkki", Katuosoite: "Esimerkkitie 11", Puh. numerot: "+358445566778", Sähköposti: "erkki.esimerkki@gmail.com"}
6717712f4cc14302fcee6636	{Käyttäjätunnus: "matti100", Etunimi: "Matti", Sukunimi: "Meikäläinen", Katuosoite: "Esimerkkitie 50", Puh. numerot: ["+358877665544", "+358855667788"], Sähköposti: "matti.meikalainen@hotmail.com"}

Kuva 2.4: Esimerkki avain-arvotietokannasta, DynamoDB:n dokumentaation [9] pohjalta.

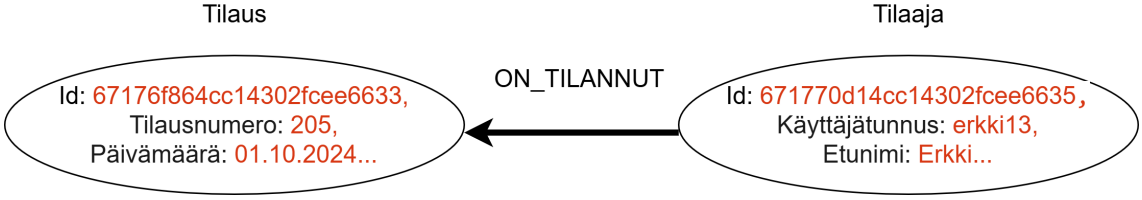
Saraketietokannat on suunniteltu analyyttiseen toimintaan, koska arvot tallennetaan yksittäisten attribuuttien sarakkeissa [10]. Lisäksi arvojen eri versiot tallennetaan aikaleimalla, joten niiden muutoksia pystytään tarkastelemaan helposti. Tyypillisesti saraketietokantoihin tallennetaan jäsenneltyä tai osittain jäsenneltyä dataa. Saraketietokannoissa avain on moniulotteinen, eli se voi koostua useista eri arvoista, kuten taulun nimestä, rivin avaimesta, sarakkeesta ja aikaleimasta. [1, s. 900] Eräs tunnettu saraketietokanta on Apache Cassandra. Kuvassa 2.5 punaiset pystyviivat havainnollistavat, kuinka saraketietokanta tallentaa dataa sarakkeissa eikä relaatiomallimaisesti riveissä.

Dataa tallennetaan ja haetaan sarakkeissa!

				Asiakas		
ID	Käyttäjätunnus	Etunimi	Sukunimi	Katuosoite	Puh. numero	Sähköposti
1	erkki13	Erkki	Esimerkki	Esimerkkitie 11	+358445566778	erkki.esimerkki@gmail.com
2	matti100	Matti	Meikäläinen	Esimerkkitie 50	+358877665544	matti.meikalainen@hotmail.com

Kuva 2.5: Esimerkki saraketietokannasta, Cassandran dokumentaation [11] pohjalta.

Graafitietokantoihin tallennetaan osittain jäsenneltyä dataa [12], jota esitetään solmuina (engl. *node*) ja niiden välisinä suhteina. Graafitietokantojen vahvuus on monimutkaisten suhteiden käsitteleminen, koska se tapahtuu dataa tallennettaessa. Muuntyyppisissä tietokannoissa suhteet luodaan tietokantaoperaatioita suoritettaessa [13]. Eräs tunnettu graafitietokanta on Neo4j. [1, s. 903] Kuva 2.6 havainnollistaa entiteettien välistä suhdetta graafitietokannassa.



Kuva 2.6: Esimerkki graafitietokannasta, Neo4j:n dokumentaation [13] pohjalta.

Ei-relaatiotietokannoille tyypillisiä ominaisuuksia ovat horisontaalinen skaalautuvuus, lopulta saavutettava eheys sekä datan jatkuva saatavuus ja kopiointi. Horisontaalisella skaalautuvuudella tarkoitetaan ei-relaatiotietokantojen suorituskyvyn parantamista lisäämällä palvelinten määrää [5]. Datan jatkuva saatavuus toteutuu kopioimalla data usealle palvelimelle, jotta se on saatavilla, vaikka yksi palvelin kaatuisi. Tämä parantaa tietokannan lukuoperaatioiden nopeutta, koska data voidaan hakea miltä tahansa palvelimelta. Toisaalta kirjoitusoperaatioiden nopeus hidastuu, koska datan arvot täytyy päivittää jokaisen palvelimen kopiolle. Lopulta saavutettava eheys tarkoittaa, että kaikilla palvelimilla olevat kopiot datasta päivittyvät lopulta samaksi, mutta eivät välittömästi niin kuin relaatiotietokannoilla. [1, s. 885] Ei-relaatiotietokannoilla ei ole universaalia kyselykieltä operaatioita varten, vaan pääasiassa jokaisella tietokannalla on omansa. Esimerkiksi graafitietokanta Neo4j käyttää kyselykielenään Cypheriä [1, s. 905]. Kyselykieli on myös mahdollista toteuttaa tietokantaa käyttävässä sovelluksessa. [1, s. 887]

3 Tietokantojen suorituskyky

Yleisin tapa vertailla relaatio- ja ei-relaatiotietokantojen suorituskykyä toisiinsa on samanlaisten operaatioiden ajan keston mittaaminen. Tyypillisesti ei-relaatiotietokantoja pidetään suorituskyvyltään nopeampana tietokantatyyppinä [4] skeemattomuuden ja ACID-ominaisuuksien puutteen takia. Kirjallisuuskatsauksessa käy kuitenkin ilmi useita tilanteita, joissa relaatiotietokanta on suorituskyvyltään ei-relaatiotietokantaa nopeampi. Taulukko 3.1 selkeyttää, mitä tietomallia ja tietokantaoperaatioita kussakin tutkimuksessa käsitellään. Avain-arvo- tai graafitietokantoja ei käsitellä tutkimuksissa, joten niiden suorituskykyä ei vertailla.

Taulukko 3.1: Kirjallisuuskatsauksessa käsitellyt ominaisuudet. Mainittujen tietokantatyyppien lisäksi jokaisessa julkaisussa tarkastellaan myös relaatiotietokantoja.

	Tietokantaoperaatio			Tietokantatyyppi			
Tutkimus	Indeksointi	Luku	Kirjoitus	Dokumentti	Avain-arvo	Sarake	Graafi
Parker ym. [3]		X	X	X			
Li ja Manoharan [4]		X	X	X		X	
Chakraborty ym. [5]		X		X		X	
Reetishwaree ja Hurbungs [14]	X	X	X	X			
Yedilkhan ym. [15]		X	X	X		X	
Zakaria [16]			X	X		X	
Hammes ym. [17]	X	X	X	X			
Mahmood [18]		X	X			X	

Taulukossa mainitulla indeksoinnilla tarkoitetaan taulun yhden tai useamman attribuutin järjestämistä. Ilman indeksointia tietokanta etsii luettavaa dataa rivi kerrallaan viimeiseen riviin asti. Indeksointi mahdollistaa lukemisen lopettamisen, kun kriteerit täyttävät rivit on löydetty. Näin lukuoperaatioita voidaan nopeuttaa kirjoitusoperaatioiden keston ja tallennustilan kustannuksella. [19]

3.1 Suorituskyvyn vertailu

Tietokantojen suorituskyvyn tutkimustuloksia vertailtaessa on tärkeää ottaa huomioon ominaisuudet, jotka vaikuttavat tuloksiin, mutta eroavat tutkimusten välillä. Kyseisiä ominaisuuksia ovat suorituskykytestien suorittamisympäristöt, kuten pilvipalvelimet tai paikallinen suorittaminen yksittäisellä tietokoneella. Pilvipalvelinten ja paikallisten tietokoneiden laitteistot voivat erota toisistaan vaikuttaen suoraan suorituskykytuloksiin. Lisäksi on otettava huomioon, onko tietokanta hajautettu (engl. *distributed*) vai keskitetty (engl. *centralized*). Horisontaalisen skaalautuvuuden ansiosta ei-relationaalinen tietomalli hyötyy hajautettavuudesta [3]. Myös tietokannoissa olevan datan määrä, rakenne ja attribuuttien määrä vaikuttavat tuloksiin. Operaatioiden tuloksia tutkiessa onkin tärkeää olla vertailematta absoluuttisia nopeuksia tutkimusten välillä. Luotettavimmat johtopäätökset saadaan, kun tutkitaan, miten tietokantojen nopeudet vertautuvat toisiinsa kunkin tutkimuksen sisällä. [4]

Parker ym. [3] vertailivat tutkimuksessaan MongoDB:tä ja relaatiotietokantaa SQL Server Expressiä (MSSQL). Tietokannoissa oleva data oli rakenteeltaan jäsenneltyä ja määrältään pientä. MSSQL:ään data tallennettiin kolmessa taulussa, kun taas MongoDB:seen se tallennettiin kolmessa dokumentissa, joissa oli viittaukset toisiin suhteiden mukaan. MSSQL oli viisi kertaa nopeampi kuin MongoDB, kun taulun attribuutin arvoa päivitettiin muun kuin pääavaimen perusteella. Kun taulua päivitettiin pääavaimen perusteella, oli MongoDB jopa 13 kertaa nopeampi. Syyksi epäiltiin sen nopeaa indeksointia pääavaimille. MongoDB oli nopeampi lähes kaikissa lukuoperaatioissa, mutta erityisesti niissä, joissa käsiteltiin entiteettien välisiä suhteita. Ainoa operaatio, jossa MSSQL päihitti MongoDB:n kaikilla datamäärillä, oli koostefunktiota (engl. *aggregate function*) käyttävä lukuoperaatio. Koostefunktiolla tarkoitetaan muun muassa keskiarvon tai summan laskemista. MongoDB oli huomattavasti hitaampi kyseisissä operaatioissa, koska sillä ei ole sisäänrakennettuja koostefunktioita toisin kuin relaatiotietokantojen kyselykielessä SQL:ssä.

Li ja Manoharan [4] vertailivat MSSQL:ää, saraketietokantoja HyperTablea ja Cassandraa sekä dokumenttitietokantoja CouchDB:tä, RavenDB:tä, MongoDB:tä ja Couchbasea. Tietokannat koostuivat yhdestä osittain jäsennellystä taulusta, jolla oli attribuutit avain ja arvo. Dokumenttitietokannat Couchbase ja MongoDB olivat nopeimpia luku- päivitys- ja poisto-operaatioissa. MSSQL oli kolmanneksi nopein luku- ja poisto-operaatioissa, mutta kolmanneksi hitain päivitysoperaatioissa. Saraketietokannat eivät suoriutuneet missään testissä hyvin.

Chakraborty ym. [5] mittasivat tutkimuksessaan Cassandran, MongoDB:n ja MySQL:n suorituskykyä. Tietokannoissa oli kaksi taulua, joissa oli yhteensä kuusi attribuuttia ja 3GB jäsentelemätöntä dataa. Cassandra oli 77 % nopeampi kuin MongoDB ja jopa 90 % nopeampi kuin MySQL dataa ladattaessa tietokantaan. Lukuoperaatioissa käytettiin neljää attribuuttia, ja tauluja yhdistettiin toisiinsa. Kyseisissä operaatioissa Cassandra oli 11 % nopeampi kuin MongoDB ja 36 % nopeampi kuin MySQL. Cassandra suoriutui operaatioista parhaiten luultavasti datan suuren määrän ansiosta.

Reetishwaree ja Hurbungs [14] vertailivat tutkimuksessaan MongoDB:tä sekä relaatiotietokantoja PostgreSQL:ää ja HyperSQL:ää. PostgreSQL on levypohjainen (engl. *on-disk*) ja HyperSQL on muistipohjainen (engl. *in-memory*) tietokanta. Muistipohjaiset tietokannat ovat teoriassa nopeampia, koska aikaa ei kulu muistin ja levyn väliseen kommunikointiin. Tosin kyseisiin tietokantoihin pystyy tallentamaan vähemmän dataa, joka myös katoaa, kun tietokanta suljetaan tai se kaatuu. Tutkimuksen tietokannoissa oli jäsennelty taulu kolmella attribuutilla, joista operaatiot kohdistettiin vain yhteen. Lisäysoperaatioissa tietokantojen välillä ei oikeastaan ollut eroa. Relaatiotietokannat olivat MongoDB:tä moninkertaisesti nopeampia lukuoperaatioissa indeksoinnin kanssa ja ilman. Tosin päivitysoperaatioissa MongoDB oli nopein. Poisto-operaatioissa ilman indeksointia HyperSQL ja MongoDB olivat lähes yhtä nopeita, mutta indeksoinnin kanssa MongoDB oli jopa 43 % nopeampi.

Yedilkhan ym. [15] vertailivat MongoDB:tä, Cassandraa ja PostgreSQL:ää. Tietokannoissa oleva data oli jäsennelty taulu, joka koostui kahdesta attribuutista. MongoDB oli nopein ja PostgreSQL hitain lisäysoperaatioissa. Lukuoperaatioissa Cassandra oli 250 kertaa nopeampi kuin MongoDB ja 70 kertaa nopeampi kuin PostgreSQL. Myös poisto-operaatioissa Cassandra oli nopein ja erot olivat vielä suuremmat. Kyseisten operaatioiden nopeus pysyi samana 20000 ja 5010000 rivin testeissä, mistä huomataan saraketietokantojen tehokkuus suurilla datamäärillä.

Zakaria [16] vertaili MySQL:ää, MongoDB:tä, Cassandraa ja relaatiotietokantaa Oraclea. Tietokantoihin tallennettiin rakenteeltaan jäsennelty taulu, jossa oli neljä attribuuttia. Oracle oli ylivoimaisesti nopein kaikissa lisäys-, päivitys- ja poisto-operaatioissa. Lisäysoperaatioissa MySQL oli jopa kymmenen kertaa hitaampi kuin toiseksi hitain tietokanta. Cassandra suoriutui päivitys- ja poisto-operaatioissa surkeasti. Syynä sille saattaa olla tutkimuksessa käytetty vähäinen datan määrä, jota oli enimmillään vain tuhat riviä. MongoDB oli noin 63 kertaa hitaampi kuin MySQL alle sadan rivin päivityksissä, mutta noin puolitoista kertaa nopeampi tuhannen rivin päivityksissä. Poisto-operaatioissa MongoDB:n ja MySQL:n erot eivät olleet johdonmukaiset eri datamäärillä.

Hammes ym. [17] tutkivat MongoDB:n ja PostgreSQL:n nopeuseroja datan ollessa jäsenneltyä ja jäsentelemätöntä. Jäsenneltyä dataa tallennettiin viidessä taulussa, joissa oli yhteensä 25 attribuuttia. Jäsentelemätöntä dataa tallennettiin kolmessa taulussa, joissa oli yhteensä 13 attribuuttia. Datan ollessa jäsenneltyä PostgreSQL oli noin 16 kertaa nopeampi koostefunktiota käyttävässä lukuoperaatiossa kuin MongoDB. Kun eräs toinen lukuoperaatio suoritettiin datan ollessa jäsentelemätöntä, oli PostgreSQL neljä kertaa nopeampi kuin MongoDB. MongoDB:ssä operaation nopeutta saatiin kuitenkin kasvatettua 30 kertaa nopeammaksi indeksoinnin avulla. Indeksoinnissa kesti yli kolme minuuttia, joka vastasi noin 39:ää indeksoimatonta lukuoperaatiota MongoDB:llä.

Mahmood [18] vertaili tutkimuksessaan MSSQL:n ja Cassandran nopeuseroja lisäys- ja lukuoperaatioissa. Molemmissa tietokannoissa oli yksi osittain jäsennelty taulu 26:lla attribuutilla ja miljoonalla rivillä. Lisäysoperaatioissa MSSQL oli lähes kymmenen kertaa nopeampi kuin Cassandra. Lukuoperaatioissa MSSQL oli sitä nopeampi verrattuna Cassandraan, mitä enemmän rivejä taulusta luettiin. Parhaimmillaan nopeusero oli seitsemänkertainen.

3.2 Vertailun päätelmät ja ristiriidat

Tutkimusten välillä huomataan useita yhtäläisyyksiä, mutta myös ristiriitoja tietokantojen suorituskyvyssä. Käy ilmi, ettei ole olemassa tietokantaa, joka suoriutuu parhaiten kaikissa tilanteissa. Operaatioiden nopeuteen vaikuttaa muun muassa datan rakenne sekä attribuuttien ja rivien määrä. Taulukko 3.2 kuvaa tiivistetysti tutkimuksissa ilmi tulleita parhaita käyttötilanteita eri tietokantatyypeille.

Taulukko 3.2: Kirjallisuuskatsauksessa käsiteltyjen tietokantojen parhaat käyttötilanteet.

Tietokantatyypit	Paras käyttötilanne	Syyt	Viitteet
Relaatio-tietokannat	Koostefunktioita käyttävät lukuoperaatiot	Sisäänrakennetut koostefunktiot SQL:ssä	[3], [17]
Dokumentti-tietokannat	Luku-, päivitys- ja poisto-operaatiot pääavaimeen tai indeksoituihin arvoihin	Nopeampi indeksointi kuin relaatiotietokannoissa	[17], [3]
Sarake-tietokannat	Poisto-, luku- ja lisäysoperaatiot indeksoimattomiin arvoihin sekä analyyttiset sovellukset suurilla datamäärillä	Vähäinen vaikutus suorituskykyyn datamäärän kasvaessa sekä attribuuttikohtainen tallennus	[15], [5], [10]

Päätelmät

Relaatiotietokannat, kuten MSSQL ja PostgreSQL ovat paras valinta koostefunktioita käyttävissä lukuoperaatioissa datan rakenteesta riippumatta. Syynä tähän on

se, että koostefunktiot ovat sisäänrakennettuja relaatiotietokantojen kyselykielessä SQL:ssä. Ei-relaatiotietokannat joutuvat toteuttamaan kyseiset funktiot sovelluksissa erillisinä prosesseina. [3], [17]

Dokumenttitietokannat, kuten MongoDB ja CouchDB ovat paras vaihtoehto luku- päivitys- ja poisto-operaatioissa datan ollessa jäsentelemätöntä tai osittain jäsenneltyä. Tietokannat suoriutuvat kyseisistä operaatioista erityisen hyvin niiden kohdistuessa pääavaimeen tai muuhun indeksoituun arvoon. [4], [17] Eräänä syynä on dokumenttitietokantojen nopeaksi optimoitu indeksointi. [3]

Saraketietokannat, kuten Cassandra tallentavat dataa attribuuttikohtaisissa sarakkeissa. Ne suoriutuvat parhaiten poisto-, luku- ja lisäysoperaatioissa, jotka kohdistuvat indeksoimattomiin arvoihin datan rakenteesta riippumatta. Kyseisissä tilanteissa datamäärän kasvattaminen vaikuttaa operaatioiden nopeuteen vain marginaalisen vähän. [15], [5] Täten saraketietokannat soveltuvat todella hyvin analyyttisiin sovelluksiin, joissa käsitellään yksittäisiä attribuutteja suuria määriä [10].

On otettava huomioon, että käsiteltyjen tutkimusten testeissä tietokantoja suoritti vain yksi palvelin. Ei-relaatiotietokannat eivät siis hyötyneet niille tyypillisistä vahvuuksista, kuten horisontaalisesta skaalautuvuudesta, lopulta saavutettavasta eheydestä tai datan jatkuvasta saatavuudesta [1, s. 885]. Myös tietokantojen versiot ja palvelinten komponentit vaihtelivat, mikä vaikutti tutkimustuloksiin.

Ristiriidat

Relaatiotietokantojen ja MongoDB:n tulokset eivät olleet johdonmukaiset päivitysoperaatioissa, jotka eivät kohdistuneet pääavaimeen Reetishwareen ja Hurbungsin [14] sekä Parkerin ym. [3] tutkimusten välillä. Vaikka molemmissa tutkimuksissa data oli jäsenneltyä, oli relaatiotietokanta toisessa tutkimuksessa nopeampi ja toisessa hitaampi kuin MongoDB. Erot tuloksissa saattavat johtua attribuuttien määrän eroista tai käytetyistä relaatiotietokannoista. Reetishwareen ja Hurbungsin tutki-

muksessa käytettiin PostgreSQL:ää sekä HyperSQL:ää, kun taas Parkerin ym. tutkimuksessa käytössä oli MSSQL.

Cassandra suoriutui tutkimuksissa vaihtelevasti. Chakrabortyn ym. [5] tutkimuksessa se oli nopein luku- ja lisäysoperaatioissa. Yedilkhanin ym. [15] tutkimuksessa lukuoperaatioiden lisäksi se oli nopein myös poisto-operaatioissa. Muissa tutkimuksissa Cassandra suoriutui kaikissa operaatioissa heikosti. Tutkimustuloksissa ei käy ilmi johdonmukaisia yhteyksiä suorituskyvyn ja datan rakenteen, määrän tai attribuuttien määrän välillä. Tulosten vaihtelevuus saattaa johtua Cassandran versiosta tai tutkimuksissa käytettyjen palvelinten tehottomista komponenteista. Osassa tutkimuksista, joissa Cassandran nopeus oli surkea, käytettiin kaksiytimistä Intel i3 -prosessoria ja kiintolevyä paljon nopeamman SSD:n sijaan [16], [18].

3.3 ACID-ominaisuuksien ja CAP-teoreeman vaikutus suorituskykyyn

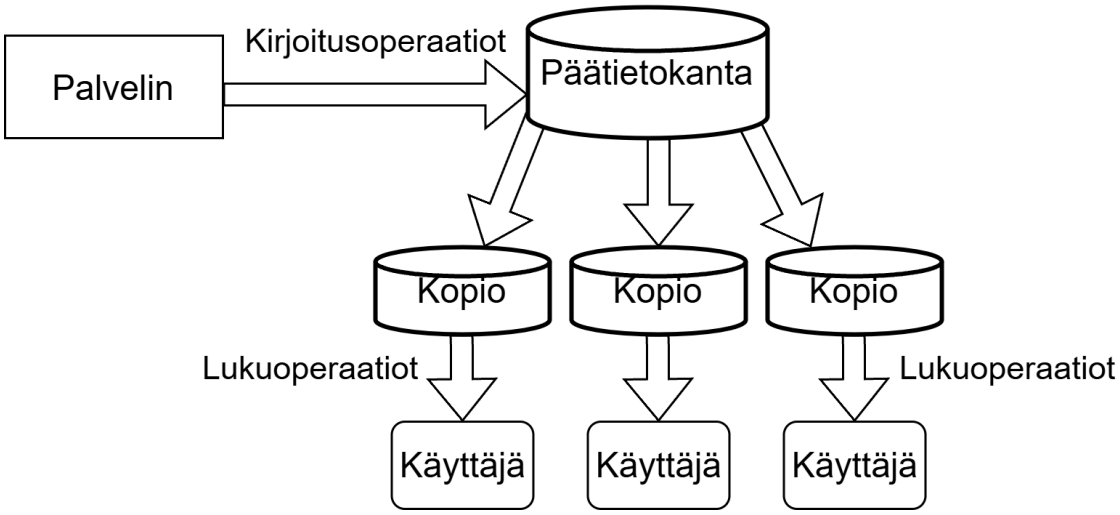
Tietokantaa valittaessa sovellukseen on tärkeää ottaa huomioon muitakin ominaisuuksia kuin pelkästään operaatioiden suoritusnopeus [15]. Relaatiotietokannat ovat välttämätön valinta datan käsittelyyn sovelluksissa, joissa käyttäjien samanaikaiset transaktiot kohdistuvat kriittisiin kohteisiin. Kriittisiä kohteita ovat muun muassa rahan siirtäminen toiselle, lentokonevarauksen tekeminen tai pankkikortilla maksaminen [17]. Relaatiotietokantojen ACID-ominaisuuksien ansiosta transaktiot ovat atomisia, ehyitä, toisiin transaktioihin vaikuttamattomia ja pysyviä. Jos transaktiota suoritettaessa tapahtuu virhetilanne, peruuntuvat jo suoritetut operaatiot eheään tilaan. Eli jos rahaa siirrettäessä tietokantapalvelin kaatuu, peruuntuu rahansiirtotransaktio ja molemmille tileille palautuu alkuperäinen määrä rahaa. [1, s. 757–758] ACID-ominaisuuksien aiheuttamien tarkistusten takia operaatioiden nopeus kuitenkin hidastuu huomattavasti, jos päivitysoperaatioita suoritetaan paljon. Siksi

ei-relaatiotietokannat ovat parempi valinta, jos sovelluksessa tarvitaan suorituskykyä, mutta ei eheää dataa. [17]

CAP-teoreema koostuu eheästä datasta, datan saatavuudesta ja partitiotoleranssista (engl. *consistency, availability, partition tolerance*). Partitiotoleranssilla tarkoitetaan tietokannan kykyä jatkaa toimimista, vaikka tietokantapalvelinten väliseen kommunikointiin käytettävässä verkossa tapahtuisi virhe. Hajautetuissa ei-relaatiotietokannoissa voidaan taata vain kaksi CAP-teoreeman ominaisuutta samanaikaisesti [17]. Yleensä datan eheydestä karsitaan käyttämällä lopulta saavutettavaa eheyttä jatkuvan eheyden sijaan. Sovelluksen käyttäjät eivät siis saa välttämättä aina päivitetyintä arvoa, jos sitä haetaan heti päivityksen jälkeen. [1, s. 888–889] Tämä tosin nopeuttaa tietokantaoperaatioita, koska tietokannan ei tarvitse odottaa aikaisempien päivitysoperaatioiden valmistumista ennen seuraavan operaation suorittamista. Sen vuoksi ei-relaatiotietokannat ovat hyvä valinta myös suurella datamäärällä, jos operaatiot eivät kohdistu kriittisiin kohteisiin. Suosittuja käyttökohteita ovat muun muassa sosiaalisen median palvelut, sähköpostijärjestelmät, analyyttiset sovellukset ja verkkokaupat. [17], [1, s. 3], [10]

3.4 Relaatiotietokantojen suorituskyvyn parantaminen

Vaikka relaatiotietokannat ovat tyypillisesti optimoitu vertikaaliselle skaalaukselle, voidaan sovelluksen palveluita jakaa manuaalisesti eri palvelimille horisontaalisella skaalauksella. Yksinkertaiset pää-kopio-tietokannat (engl. *master-slave*) ovat hyödyllinen vaihtoehto horisontaalisen skaalauksen toteuttamiseen tapauksissa, joissa operaatiot ovat pääosin lukuoperaatioita. [20] Kuva 3.1 havainnollistaa, kuinka kirjoitusoperaatiot kohdistuvat pelkästään päätietokantaan, kun taas lukuoperaatiot jakautuvat kopiotietokantojen kesken.



Kuva 3.1: Silva ym. [20] tutkimuksen pohjalta luotu havainnollistus pää-kopio-tietokannasta.

Silva ym. [20] tutkivat, miten horisontaalista skaalausta voidaan hyödyntää relaatiotietokannoissa, kun operaatiot ovat kirjoituspainotteisia. Tutkimuksessa käytettiin sovellusta, joka kirjoitti tiheästi tapahtumatietoja eli lokeja tietokantaan. Lokitapahtumien kirjoittaminen yksitellen todettiin toimimattomaksi tavaksi. Lokimäärän kasvaessa vain alle 1 % kirjoitusoperaatioista onnistui, koska lisäykset aikakatkaistiin viiden sekunnin jälkeen. Lisäksi kaikki indeksit päivitettiin jokaisella lisäyksellä, mikä on laskennallisesti raskasta. Lokien onnistunut kirjoittaminen mahdollistettiin hajauttamalla se useiksi prosesseiksi. Käyttäjien autentikointiin käytettiin erillistä henkilötietokantaa sekä autentikoidut käyttäjät tallennettiin välimuistiin. Autentikoinnin jälkeen lokit lisättiin jonoon, ja kun jonossa oli tuhat lokia, ne lisättiin lokitietokantaan kerralla. Näin tietokanta ei missään vaiheessa hidastunut ja kaikki kirjoitusoperaatiot onnistuivat.

4 Yhteenveto

Tutkielman tavoitteena oli selvittää, millaisia suorituskykyeroja relaatio- ja ei-relaatiotietokantojen välillä on. Lisäksi tavoitteena oli tutkia, mistä erot johtuvat ja mihin sovelluksiin tietyntyyppiset tietokannat soveltuvat. Vastauksia tutkimuskysymyksiin etsittiin kirjallisuuskatsauksen avulla.

Ensimmäisenä tutkimuskysymyksenä TK1 oli, millaisia suorituskykyeroja relaatio- ja ei-relaatiotietokantojen välillä on. Kävi ilmi, että relaatiotietokannat suoriutuvat parhaiten koostefunktioita käyttävissä lukuoperaatioissa, koska ne ovat sisäänrakennettuja SQL:ssä. Dokumenttitietokannat suoriutuvat parhaiten pääavaimeen tai indeksoituihin arvoihin kohdistuvissa luku-, päivitys- ja poisto-operaatioissa datan ollessa osittain jäsenneltyä tai jäsentelemätöntä. Saraketietokannat ovat nopeimpia poisto-, luku- ja lisäysoperaatioissa, jotka kohdistuvat indeksoimattomiin arvoihin datan rakenteesta riippumatta.

Toisena tutkimuskysymyksenä TK2 oli, mistä suorituskykyerot voivat johtua. Ne voivat johtua palvelimen komponenteista, suoritusympäristöstä sekä siitä, onko tietokanta toteutettu hajautetusti vai keskitetysti. Vaikka tyypillisesti ei-relaatiotietokannat hyötyvät hajautettavuudesta, voidaan horisontaalinen skaalautuvuus toteuttaa manuaalisesti myös relaatiotietokannoille. Lisäksi suorituskykyeroihin vaikuttavat tietokannassa olevan datan rakenne sekä attribuuttien ja rivien määrä.

Kolmantena tutkimuskysymyksenä TK3 oli, mihin sovelluksiin relaatio- ja ei-relaatiotietokannat soveltuvat. Relaatiotietokannat ovat välttämätön valinta sovel-

luksissa, joissa käyttäjien samanaikaiset transaktiot kohdistuvat kriittisiin kohteisiin, kuten pankkikorttimaksuihin. ACID-ominaisuudet takaavat transaktioiden eheyden sekä myös palautettavuuden virhetilanteissa. Ei-relaatiotietokannat ovat paras valinta kasvavan datamäärän sovelluksiin, joissa hyödytään skeemattomuudesta, horisontaalisesta skaalautuvuudesta ja ACID-ominaisuuksien puutteesta. Dokumenttitietokannat soveltuvat sosiaalisen median käyttäjätietojen hallintaan, kun taas saraketietokannat soveltuvat analyyttisiin sovelluksiin.

Tutkielmassa tietokantojen suorituskykyeroja vertailtiin pelkästään niiden ollessa keskitetysti toteutettuja. Jatkotutkimuksissa olisi syytä toteuttaa ei-relaatiotietokannat hajautetusti ja relaatiotietokannat keskitetysti tarpeeksi tehokkailla komponenteilla. Näin tietokantojen suorituskyvyn rajoittavat tekijät saataisiin minimoitua ja vertailutulokset kuvaisivat paremmin suorituskykyä käytännön sovelluksissa.

Lähdeluettelo

[1] R. Elmasri ja S. B. Navathe, *Fundamentals of Database Systems*, 7th. Pearson, 2015, ISBN: 978-0133970777.

[2] A. Flores, S. Ramírez, R. Toasa, J. Vargas, R. Urvina-Barrionuevo ja J. M. Lavin, "Performance Evaluation of NoSQL and SQL Queries in Response Time for the E-government", teoksessa *2018 International Conference on eDemocracy & eGovernment (ICEDEG)*, 2018, s. 257–262. DOI: 10.1109/ICEDEG.2018.8372362.

[3] Z. Parker, S. Poe ja S. V. Vrbsky, "Comparing NoSQL MongoDB to an SQL DB", teoksessa *Proceedings of the 51st Annual ACM Southeast Conference*, sarja ACMSE '13, Savannah, Georgia: Association for Computing Machinery, 2013, ISBN: 9781450319010. DOI: 10.1145/2498328.2500047.

[4] Y. Li ja S. Manoharan, "A performance comparison of SQL and NoSQL databases", teoksessa *2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, 2013, s. 15–19. DOI: 10.1109/PACRIM.2013.6625441.

[5] S. Chakraborty, S. Paul ja K. M. Azharul Hasan, "Performance Comparison for Data Retrieval from NoSQL and SQL Databases: A Case Study for COVID-19 Genome Sequence Dataset", teoksessa *2021 2nd International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*, 2021, s. 324–328. DOI: 10.1109/ICREST51555.2021.9331044.

[6] Amazon Web Services, Inc. ”What Is a Document Database?” (Ei julkaisupäivää), url: https://aws.amazon.com/nosql/document/ (viitattu 29. 10. 2024).

[7] MongoDB. ”Data Modeling”. (ei julkaisupäivää), url: https://www.mongodb.com/docs/manual/data-modeling/#std-label-manual-data-modeling-intro (viitattu 27. 11. 2024).

[8] Amazon Web Services, Inc. ”What Is a Key-Value Database?” (Ei julkaisupäivää), url: https://aws.amazon.com/nosql/key-value/ (viitattu 29. 10. 2024).

[9] Amazon Web Services, Inc. ”Data Modeling foundations in DynamoDB”. (ei julkaisupäivää), url: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/data-modeling-foundations.html (viitattu 04. 12. 2024).

[10] Atlan. ”What is a Columnar Database? Examples, Benefits, Differences & More!” (15. joulukuuta 2023), url: https://atlan.com/what-is/columnar-database/ (viitattu 29. 10. 2024).

[11] Apache Cassandra. ”Data Modeling Introduction”. (ei julkaisupäivää), url: https://cassandra.apache.org/doc/latest/cassandra/developing/data-modeling/intro.html (viitattu 04. 12. 2024).

[12] Neo4j. ”Graphs as a New Way of Thinking”. (ei julkaisupäivää), url: https://neo4j.com/news/graphs-as-a-new-way-of-thinking/ (viitattu 30. 10. 2024).

[13] Neo4j. ”Get Started with Neo4j”. (ei julkaisupäivää), url: https://neo4j.com/docs/getting-started/get-started-with-neo4j/graph-database/ (viitattu 25. 10. 2024).

[14] S. Reetishwaree ja V. Hurbungs, ”Evaluating the performance of SQL and NoSQL databases in an IoT environment”, teoksessa *2020 3rd International Conference on Emerging Trends in Electrical, Electronic and Communications Engineering (ELECOM)*, 2020, s. 229–234. DOI: 10.1109/ELECOM49001.2020.9297028.

[15] D. Yedilkhan, A. Mukasheva, D. Bissengaliyeva ja Y. Suynullayev, ”Performance Analysis of Scaling NoSQL vs SQL: A Comparative Study of MongoDB, Cassandra, and PostgreSQL”, teoksessa *2023 IEEE International Conference on Smart Information Systems and Technologies (SIST)*, 2023, s. 479–483. DOI: 10.1109/SIST58284.2023.10223568.

[16] M. ZAKARIA, ”SQL, NOSQL and NewSQL Databases: A Theoretical and Practical Comparative Survey”,

[17] D. Hammes, H. Medero ja H. Mitchell, ”Comparison of NoSQL and SQL Databases in the Cloud”, 2014.

[18] K. Mahmood, ”Performance comparison of nosql database cassandra and sql server for large databases”, *Journal of Independent Studies and Research Computing*, vol. 14, nro 2, 2016.

[19] S. Halfpap, J. Kossmann, R. Schlosser ja V. Markl, ”Looking Deeply into the Magic Mirror: An Interactive Analysis of Database Index Selection Approaches”, *Proc. VLDB Endow.*, vol. 17, nro 12, s. 4301–4304, marraskuu 2024, ISSN: 2150-8097. DOI: 10.14778/3685800.3685860.

[20] L. J. G. d. Silva, L. G. d. Vasconcelos, G. da Silva ja L. E. G. d. Vasconcelos, ”Towards Scalability in Systems with Write Operations in Relational Databases”, teoksessa *2015 12th International Conference on Information Technology - New Generations*, huhtikuu 2015, s. 267–272. DOI: 10.1109/ITNG.2015.49.