Antti Karjunen

NoSQL:n suorituskyky Big Datan näkökulmasta

Tietotekniikan kandidaatintutkielma

18. toukokuuta 2024

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Antti Karjunen

Yhteystiedot: anpeakka@student.jyu.fi

Työn nimi: NoSQL:n suorituskyky Big Datan näkökulmasta

Title in English: NoSQL performance from the perspective of Big Data

Työ: Kandidaatintutkielma

Sivumäärä: 25+0

Tiivistelmä: NoSQL-tietokantojen käyttö, erityisesti verkkosovellusten tietokantaratkaisuina, on kasvanut merkittävästi, erityisesti Big Datan ja sosiaalisen median kontekstissa. Tämän kasvun takana on muun muassa NoSQL-tietokantojen tarjoama joustavuus sovellusten tarpeisiin, kuten suurten ja monimuotoisten datamäärien hallintaan. Kun tallennettavan tiedon määrä ja tietokantakyselyiden määrä kasvavat, tietokannan suorituskyky nousee keskeiseksi tekijäksi tietokantaratkaisua valittaessa. Tässä tutkielmassa tarkastellaan NoSQL-tietokantojen suorituskykyä ja tutkitaan aiheeseen liittyviä tutkimuksia, keskittyen erityisesti Big Datan vaatimuksiin.

Avainsanat: NoSQL, Tietokanta, Suorituskyky, MongoDB, Cassandra, Big Data

Abstract: The use of NoSQL databases, especially as database solutions for web applications, has significantly increased, particularly in the context of big data and social media. Behind this growth lies the flexibility offered by NoSQL databases to meet the needs of applications, such as managing large and diverse datasets. As the amount of stored data and the number of database queries increase, database performance becomes a crucial factor in selecting a database solution. This thesis examines the performance of NoSQL databases and investigates related research, focusing specifically on the requirements of big data.

Keywords: NoSQL, Database, Performance, MongoDB, Cassandra, Big Data

Sisällys

1 Johdanto

Viime vuosina Big Data ja sen lieveilmiöt ovat päätyneet tutkijoiden keskuudesta kahvipöytäkeskusteluihin. Syitä on lukuisia, mutta erityisesti sosiaalisten medioiden valtavaksi kasvanut suosio ja useat tiedonkeräykseen liittyvät selkkaukset medioissa, ovat aiheuttaneet mielipiteitä jakavaa keskustelua Big Datasta hyödyistä, mutta myös sen eettisistä ongelmista (W. Khan ym. 2023).

Sosiaalinen media tuottaa niin valtavia määriä tietoa, että sen käsittelyyn eivät tavalliset relaatiotietokannat kykene. Vaikka NoSQL-tietokantoja, jotka tunnetaan myös ei-relaatiotietokantoina, ei luotukaan palvelemaan Big Datan vaatimuksia, on niiden yksinkertaisuus, skaalautuvuus, sekä tapa tallentaa tietoa hajautetusti syitä, miksi NoSQL-tietokantojen ja Big Datan mahdollisuudet ovat tärkeä tuntea. Yritykset, jotka haluavat kehittää palveluitaan, pyrkivät jalostamaan Big Datasta yritykselleen parempia menetelmiä. Samoin toimivat myös markkinointiosastot, sillä Big Dataa seuraamalla voidaan tunnistaa asiakkaiden ostoskäyttäytymistä.

Samanaikaisesti Big Datan roolin kasvaessa myös NoSQL-tietokantajärjestelmät ovat saaneet lisää huomiota. Nämä järjestelmät tarjoavat vaihtoehdon perinteisille SQL-tietokannoille. NoSQL-tietokannat on suunniteltu skaalautuvuus, datan monimuotoisuus ja palvelinten hajautus mielessä, ja nämä ovat juuri sitä, mitä Big Datan hallitsemiseen tarvitaan.

Tämän tutkielman tavoitteena on selvittää, millainen on NoSQL-tietokantojen suorituskyky, kun tietokantoihin on tallennettuna Big Dataa. Relaatiotietokantojen ollessa niin suosittuja ja ollessa alan standardi, tarkastellaan tutkielmassa myös suorituskykyvertailuja NoSQL:n sekä perinteisten relaatiotietokantojen välillä, ymmärtääksemme, mitkä ovat kunkin tietokantamallin vahvuudet ja heikkoudet Big Datan hallitsemisessa.

Tutkielman taustalla on havainto, että vaikka relaatiotietokannoista on tehty kattavaa tutkimusta ja niiden toimintaa tunnetaan hyvin, NoSQL-tietokannat ovat edelleen suhteellisen vähän tutkittuja, ja ei-relaatiotietokannat ovat usealle yritykselle yhä tuntematon teknologia. On kiinnostavaa selvittää, onko NoSQL-tietokantojen vähäisemmän käytön taustalla teknologisia puutteita vai onko kyseessä pelko siitä, mitä vähemmän tukitun ja dokumentoidun tietokannan käyttöönotosta voi seurata.

Tutkimusmenetelmänä tässä tutkielmassa käytetään kirjallisuuskartoitusta. Aihe soveltuu hyvin kirjallisuuskartoitukseen, koska tarkoituksena on koota ja arvioida olemassa olevaa kirjallisuutta NoSQL-tietokantojen suorituskyvystä Big Datan käsittelyssä. Tavoitteena on löytää sekä teoreettista että käytännön vertailua, josta voitaisiin vetää johtopäätöksiä siitä, kuinka hyvin NoSQL-tietokannat soveltuvat erilaisiin Big Dataa hyödyntäviin käyttötarkoituksiin.

Luvussa 2 esitellään tietokantajärjestelmien periaatteet, luvussa 3 kuvataan, mikä NoSQL-tietokanta itsessään on ja kuinka se jaotellaan eri tietokantatyyppeihin. Luvussa 4 esitellään, millaista määritelmää Big Datasta tämän tutkielman osalta käytetään ja kuinka tietokannat käsittelevät Big Dataa. Luvussa 5 yhdistetään NoSQL-tietokantatyypeistä löytynyttä suorituskykyvertailua ja Big Dataan liittyvää teoriaa. Luku 6 sisältää tutkielman kirjoittajan aiheeseen liittyvää pohdintaa.

2 Tietokantajärjestelmien periaatteet

Tässä luvussa esitellään relaatiotietokantojen ja ei-relaatiotietokantojen toimintaperiaatteet. Periaatteiden toimintaa kuvataan esimerkeillä, jotka kuvaavat pankin tilitapahtumia.

Tietokantajärjestelmät voidaan jakaa kahteen ryhmään: relaatiotietokantoihin (SQL) sekä ei-relaatiotietokantoihin (NoSQL). Tietokantojen toiminnassa keskeinen käsite on transaktio, eli tietokantaan kohdistuvien toimintojen sarja, joka suoritetaan yhtenä kokonaisuutena (Helland 2016). Transaktion onnistumisen kannalta, on sille määritelty eheyssääntöja, kuten ACID-periaate. ACID-periaate on pääasiassa relaatiotietokantojen suosiossa, mutta myös NoSQL tietokannat pyrkivät täyttämään nämä vaatimukset (Ahmed ym. 2018).

Seuraavissa osioissa periaatteita käydään läpi kronologisessa järjestyksessä, ensiksi mikä on relaatiotietokantojen noudattama ACID-periaate, sen jälkeen käymme hajautettuja tietojärjestelmiä varten kehitetyn CAP-teoreeman ja viimeiseksi siitä johdetun ei-relaatiotietokantojen suosiossa olevan BASE-periaatteen.

2.1 ACID

ACID on akronyymi sanoista atomisuus (atomicity), eheys (consistency), eristyneisyys (isolation) ja pysyvyys (durability). Seuraavaksi esitellään ACID-periaatteen käyttäen esimerkkinä tilisiirtoa.

Atomisuusvaateen mukaan transaktion kaikki tietokantatoimenpiteet suoritetaan, tai mitään ei suoriteta, ja käyttajän on myös tiedettävä, missä tilassa hänen transaktionsa on (Medjahed, Ouzzani ja Elmagarmid 2009). Tilisiirrossa sekä asiakkaan että pankin jokaisen vaiheen on tapahduttava onnistuneesti, jos jokin askelista estyy tai jää tapahtumatta, peruuntuu tilisiirto.

Eheysvaateen mukaan transaktion päättyessä odotusten mukaisesti, tietokantaan tallentuu toimenpiteiden muutokset. Tietokanta on skeemansa mukainen, eli eheä sekä ennen että jälkeen transaktiota (Medjahed, Ouzzani ja Elmagarmid 2009). Tilisiirrossa tämä tarkoittaa, että transaktion päättyessä on kummallakin tilillä transaktion seurauksena oikea määrä rahaa, eikä rahaa häviä tai synny tyhjästä.

Eristyneisyyden vuoksi kaikki transaktion tapahtumat ovat piilotettu toisilta samanaikaisilta transaktioilta, taaten eheysvaatimuksen täyttymisen (Medjahed, Ouzzani ja Elmagarmid 2009). Tilisiirrossa esimerkiksi kaksi samanaikaista tilisiirtoa ei vaikuta toisiinsa, kun maksutapahtuma aloitetaan, maksutapahtuma näkee tilin eheän tilan, eli tilan joka tilillä oli maksutapahtuman alkaessa.

Pysyvyyden takia transaktion päättyessä ja sen tuloksien tallentuessa tietokantaan, järjestelmän tulee luvata, että nämä tallennetut tiedot ovat varmasti tallentuneet tietokantaan, jos käyttäjä saa sellaisen palautteen tietokannan hallintajärjestelmältä (DBMS) ja että talletetut tiedot selviävät toimintahäiriöistä (Medjahed, Ouzzani ja Elmagarmid 2009). Tietoa ei voi lisääntyä tai poistua tietokannoista muuten, kuin käyttajän toimenpiteen seurauksena (Haerder ja Reuter 1983). Tilisiirrossa esimerkiksi jos pankin palvelin kaatuu hetkeä onnistuneeksi ilmoitetun tilisiirron jälkeen, on tilisiirron vaikutukset tietokannassa palvelimen uudelleenkäynnistäessä.

Relaatiotietokantojen käyttö Big Datan tallentamiseen, on kuitenkin johtanut ongelmiin, jotka johtuvat sekä tietojen mallintamisen puutteista, että horisontaaliseen skaalautuvuuteen liittyvistä rajoituksista, silloin kun käytössä on useita palvelimia ja suuria tietomääriä. (Moniruzzaman ja Hossain 2013; Fox ja Brewer 1999). Moniruzzamanin ja Hossainin (2013) mukaan relaatiotietokantojen tiedon määrän eksponentiaalinen kasvu, joka syntyy käyttäjien, järjestelmien ja antureiden tuottamasta tiedosta, kiihtyy entisestään suurten hajautettujen järjestelmien, kuten Amazonin, Googlen ja muiden pilvipalveluiden, keskittymisen myötä. Tiedon lisääntyvä riippuvuus ja monimutkaisuus kiihtyy internetin, Web 2.0:n, sosiaalisten medioiden sekä suuren määrän erilaisten avointen ja standardoitujen tietolähteiden vapaan pääsyn myötä (Moniruzzaman ja Hossain 2013).

2.2 CAP-Teoreema

CAP-teoreema on akronyymi sanoista eheys (consistency), saatavuus (availability) ja osituksen sietokyky (partition tolerance). CAP-teoreeman kehitti Eric Brewer vuonna 2000, tutkiessaan hajautettujen järjestelmien toimintaperiaatteita (Fox ja Brewer 1999). Siinä missä relaatiotietokannat tyypillisesti käyttävät yhden palvelimen arkkitehtuuria, hajautetut jär-

jestelmät suosivat klusterointia. Klusteri on joukko tietokoneita tai palvelimia, jotka toimivat yhdessä suuremman järjestelmän muodostamiseksi. Klusterin yksittäinen tietokone tai palvelin on nimeltänsä solmu.

Hajautettu järjestelmä voi saavuttaa vain kaksi edellä mainituista kolmesta vaatimuksesta (Fox ja Brewer 1999). Tietokantaa pidetään eheänä, kun kirjoitusoperaation jälkeen, jokainen siihen kohdistunut lukuoperaatio sisältää kirjoitusoperaation lopputuleman, riippumatta siitä, mihin tietokannan solmuun lukuoperaatio kohdistuu. Saavutettava tietokanta sallii luku- ja kirjoitusoperaatiot, vaikka osa klusterin solmuista olisi saavuttamattomissa. Onnistuneen tilisiirron päätteeksi on saldon oltava luettavissa mistä tahansa palvelimen ylläpitoon osallistuvasta tietokoneesta. Ositusta sietävä tietokanta on tietokanta, jonka luotettavuus ja tehokkuus ei kärsi merkittävästi, kun tietokannan osat ovat erotettu toisistaan esimerkiksi verkon katkeamisen vuoksi. Ilman osituksen sietoa, henkilön tehdessä tilisiirron toiminnassa olevaan solmuun, joka olisi kuitenkin verkkovirheen vuoksi erotettuna toisista solmuista, voitaisiin päätyä tilaan, jossa käyttäjä saa palautteen onnistuneesta tilisiirrosta, mutta muut solmut eivät saisikaan tietoa tilitapahtumasta, jolloin tietokanta ei olisi enää eheä.

2.3 BASE

Ei-relaatiotietokannat noudattavat BASE-periaatetta (W. Khan ym. 2023). BASE on akronyymi sanoista periaatteellisesti saatavilla oleva (basically available), pehmeä tila (soft state) ja lopulta eheä (eventually consistent). BASE-periaatteen kehitti Pat Helland (2016). Seuraavaksi esitellään BASE-periaate käyttäen jälleen tilisiirtoa esimerkkinä.

Periaatteellisesti saatavilla oleva tarkoittaa, että tietokanta pyrkii olemaan aina saatavilla, vaikkei se olisikaan täysin eheä. Osa järjestelmästä voi olla viallisia, hitaita tai kaatuneena, mutta järjestelmä silti vastaa käyttäjän pyyntöihin (Pritchett 2008). Tilisiirtoja ajatellessa pankkien hajautetut järjestelmät sallivat asiakkaiden jatkaa normaalia asiointia, vaikka osa pankin palvelimista olisikin viallisia.

Pehmeä tila sallii tietokannan joustavan eheydestä, eli esimerkiksi klusterin eri solmut saattavat antaa eri tietoa tietokannan sen hetkisestä tilanteesta (Pritchett 2008). Tilisiirrossa pehmeä tila voisi tarkoittaa, että kahden eri asiakkaan tehdessä tilisiirron kolmannelle tilille,

voi kolmannen tilin saldo olla hetken epäjohdonmukainen. Tietokanta kuitenkin saavuttaa eheyden, kun tilitapahtumat synkronoidaan.

Lopulta eheä tarkoittaa, että vaikka tietokanta ei olisi heti transaktion päättyessä eheä, kuten ACID-periaatetta seuraavat tietokannat ovat, tulee se kuitenkin saavuttamaan eheän tilan (Pritchett 2008). Kuten pehmeän tilan esimerkissä, lopulta eheä tietokanta lupaa, että tilien saldot ovat lopulta oikein, eikä tietokantaan jää solmuja, joissa tieto olisi väärin.

Seuraavassa luvussa tutustumme tarkemmin NoSQL-tietokantoihin, jotka seuraavat BASE-periaatetta (Seeger 2009).

3 NoSQL

Tässä luvussa käsitellään ei-relaatiotietokantojen historiaa, toimintaa ja miten NoSQL tietokannat jaetaan neljään eri tietokantatyyppiin.

NoSQL on vuonna 1998 Carlo Strozzin (1998) kehittämä käsite ei-relaatiotietokannoille. Vaikka tässä tutkielmassa käytetään käsitettä ei-relaatiotietokanta, esiintyy NoSQL myös osassa kirjallisuutta nimellä ei pelkästään relaatiotietokanta (engl. Not Only SQL), joka poistaa käsitteeltä SQL-vastaisuuden (Stonebraker 2010). NoSQL alkoi yleistyä vasta muutama vuosi ennen 2010-lukua, kun vuonna 2007 Google julkaisi Bigtable saraketietokannan ja vuonna 2007 Amazon julkaisi avain-arvo tietokannan, Amazon Dynamon (Fowler 2015). Oussousin (2013) mukaan tietokannat eivät kykene täyttämään ACID-periaatteen vaatimuksia isoilla datamäärillä. Tämän vuoksi NoSQL-tietokannat seuraavat BASE-periaatteita, jotka esiteltiin viime luvussa.

Fowlerin (2015) mukaan NoSQL-tietokannoilla on yleensä neljä seuraavaa piirrettä:

1. Skeeman puute: Relaatiotietokannoissa skeema määrittelee, millaista dataa tietokantaan voi tallettaa. NoSQL-tietokannat pystyvät tallentamaan järjestäytyneen datan lisäksi myös järjestäytymätöntä dataa, kuten sähköposteja, multimediaa ja sosiaalisen median sisältöä.
2. Ei-relaatioita: Siinä missä relaatiotietokannat pitävät dataa yllä eri tauluissa ja niiden välisillä yhteyksillä, voi NoSQL-tietokannoissa kaikki haluttu data olla yhdessä dokumentissa. Halutessa datan voi jakaa useampaan dokumenttiin, joihin voi viitata sitten id-arvolla.
3. Laitteiston joustavuus: Tyypillisesti relaatiotietokannat, jotka ovat vertikaalisesti skaalautuvia, vaativat tehokkaan laitteiston palvelimen ylläpitoon. NoSQL-tietokantojen horisontaalisen skaalautuvuuden vuoksi, voi palvelimen ylläpitäjä tyytyä käyttämään palvelimessaan useita vanhentuneita laitteistoja, sillä työpanos jakautuu laitteistoille.
4. Hajautettavuus: Useimmat relaatiotietokannat ovat keskitettyjä, jolloin tietokantaa palvellaan yhdellä tai muutamalla palvelinkoneella, yhdestä sijainnista. Ei-relaatiokannat ovat hajautettuja, mikä tarkoittaa sitä, että tietokannan solmut, eli palvelimet, sijaitse-

vat usein eri sijainneissa ja niitä käytetään lukuisia työpanoksen jakamiseksi.

NoSQL tietokannat jaetaan yleisesti neljään eri luokkaan: dokumenttitietokantoihin (document oriented databases), saraketietokantoihin (column-oriented databases), avain-arvo tietokantoihin (key-value databases) sekä graafitietokantoihin (graph databases) (Moniruzzaman ja Hossain 2013).

3.1 Dokumenttitietokannat

Dokumenttitietokantojen nimi tulee tyylistä, jolla dokumenttitietokannat tallentavat tietoa. Relaatiotietokannoista tutun taulun sijaan dokumenttitietokannat tallentavat tietonsa dokumenttiin. Dokumentit voivat taas kuulua suurempaan kokonaisuuteen, joita nimitetään kokoelmiksi. (Cattell 2011)

Skeemattomissa dokumenttitietokannoissa luku- ja kirjoitusoperaatioihin käytetään avainta, avain-arvo tietokannoista tuttuun tyyliin, mutta dokumenttien hakuja voidaan suorittaa dokumenttien arvojen perusteella, toisin kuin avain-arvo tietokannoissa (Győrödi ym. 2015).

Dokumenttitietokantoihin tallennettiin sen alkuaikoina XML-muotoista tietoa, mutta nykyään suosituimmaksi on noussut JSON, jota pidetään XML:n seuraajana, sekä JSON:sta tiivistettyä binäärimuotoista BSON:ia (Ahmed ym. 2018).

MongoDB ja Couchbase, jotka pitävät tietokantoja suosiota ylläpitävän verkkosivun, DB-enginesin mukaan, sijoja 5 ja 36, ovat dokumenttitietokannoista suosituimpia. (DB-Engines). MongoDB on täten myös kaikista NoSQL-tietokannoista suosituin.

3.2 Saraketietokannat

Saraketietokannat poikkeavat dokumenttitietokannoista ja muista tavallisista tietokannoista sillä, että saraketietokannoissa tieto tallennetaan sarakkeisiin, eikä riveihin. Saraketietokannat tallentavat tietonsa relaatiotietokannoista tutusti tauluihin, sillä erolla, ettei taulujen välillä ole riippuvuussuhteita. Myös relaatiotietokannoista tutuille taulukoille löytyy vastine saraketietokannoista, sarakeperheet. Sarakeperheet ovat loogisia, yhdelle fyysiselle levylle

talletettavia kokonaisuuksia, jotka sisältävät teoreettisesti loputtomasti sarakkeita. Sarakeperheitä voidaan luoda joko skeemaa suunnitellessa tai käytön aikana (Győrödi ym. 2015).

Saraketietokannat eivät ole täysin skeemattomia, kuten dokumenttitietokannat ovat. Kaikilla riveillä ei tarvitse olla jokaiselle sarakkeelle määrättyä arvoa, mutta sarakkeen arvojen tulee kuitenkin olla samaa tyyppiä (Moniruzzaman ja Hossain 2013).

Avain-arvo tietokannoista sekä dokumenttitietokannoista poiketen, saraketietokannat tallentavat riveille riviavaimia. Kun tietokantaan kohdistetaan luku-, kirjoitus- tai hakuoperaatioita, käytetään yleensä hakuun riviavainta, vaikka operaatioita voi myös monimutkaistaa hakuehdoilla. Saraketietokantoihin kohdistuvat haut ovat pääasiassa sellaisia, jotka palauttavat kokonaisia sarakkeita, eikä rivejä, kuten monessa muussa tietokantatyypissä (Cattell 2011; Seeger 2009).

Siinä missä dokumenttitietokantoihin tallennetaan tietoa, jossakin yleisemmin luettavissa olevassa tiedostomuodossa, tallennetaan saraketietokantojen tiedot yleensä binäärimuotoisena (Corbellini ym. 2017).

Saraketietokannoista suosituimpia ovat Cassandra ja HBase, jotka pitävät tietokantoja suosiota ylläpitävän verkkosivun (DB-Engines) mukaan, sijoja 12 ja 26.

3.3 Avain-arvo-tietokannat

Avain-arvo tietokannat ovat kaikista tietokantajärjestelmistä yksinkertaisimpia ja näitä tietokantoja voidaankin pitää yksinulotteisina tietokantoina (Cattell 2011). Avain-arvo tietokannoissa data tallennetaan pareina, niin että jokaisella tallennetulla arvolla on yksilöllinen avain. Avain-arvo tietokannoissa luku-, haku- ja kirjoitusoperaatioihin on käytettävä avainta, eikä tietoa voi hakea sisällön perusteella, kuten esimerkiksi dokumenttitietokannoissa pystyy. (Seeger 2009)

Avain-arvo tietokantojen perusoperaatio ovat hyvin yksinkertaisia (Seeger 2009). Tietoa laitetaan avain-arvo tietokantaan put-komennolla, haetaan get-komennolla ja poistetaan delete-komennolla. Tietokantojen hallintajärjestelmät, kuten Redis, sallivat kyllä esimerkiksi sadan poistokomennon yhdistämisen, mutta todellisuudessa kyseessä on vain putkitettu komento,

joka koostuu sadasta poistokomennosta.

Yksinkertaisuutensa vuoksi, avain-arvo tietokannat ovat optimoitu nopeaan lukemiseen ja kirjoittamiseen, ja ne soveltuvat käsittelemään suuria määriä tietoa, Big Dataa, tehokkaasti (Corbellini ym. 2017).

Redit ja Memcached, avain-arvo tietokannoista suosituimmat, ovat tutkielman kirjoitushetkellä tietokantojen suosiota tarkastelevalla sivustolla sijoilla 6 ja 32 (DB-Engines).

3.4 Graafitietokannat

Graafitietokannat ovat NoSQL-tietokantatyypeistä kaikista poikkeavampia. Tietokanta rakentuu solmuista, jotka ovat avain-arvo pareja, mutta solmujen väliset särmät ovat solmujen suhteita, eli relaatioita, jotka ovat epätyypillisiä ei-relaatiotietokannoille. Graafitietokannat eivät tue horisontaalista ositusta, toisin kuin muut NoSQL tietokantatyypit. (Gorton ja Klein 2015)

Graafitietokannoissa yleensä halutaan säilöä algoritmista tietoa, joka vastaa kysymykseen: ”kuinka monella askeleella päästään solmusta X solmuun Y”. Tällaista tietoa on esimerkiksi sukututkimukseen käytettävä tieto, sillä sukututkijaa voi kiinnostaa, kuinka monen sukupolven kautta henkilöt ovat keskenään sukua.

Graafitietokannat kärsivät yksinkertaisissa hakuoperaatioissa hitaudesta, sillä solmujen suhteet ovat usein monesta-moneen -suhteita, jolloin haettava data kasvaa eksponentiaalisesti. Graafitietokannat kuitenkin tukevat hyvin haastavia kyselyjä, jotka mitkä suoritetaan lyhyemmissä ajassa, kuin muiden tietokantatyyppien vastaavat kyselyt (Li 2018; Győrödi ym. 2015).

Verrattaessa muihin NoSQL-tietokantatyyppejen suosioon, on graafitietokantojen suosio huomattavasti heikompi (DB-Engines). Suosituimmat graafitietokannat, Neo4j ja GraphDB, pitävät suosiossa sijoja 23 ja 100.

4 Big Data

Tässä luvussa esitetään, mitä Big Data tarkoittaa tämän tutkielman näkökulmasta, sekä kerrotaan, kuinka tietokannat hallitsevat Big Dataa.

4.1 Big Datan määritelmä

Big Datalle on olemassa useita erilaisia määritelmiä, tässä tutkielmassa käytetään yleisesti hyväksyttyä kolmen V:n mallia: määrä (volume), nopeus (velocity) ja valikoima (variety) (Kacfah Emani, Cullot ja Nicolle 2015). Määrällä tarkoitetaan valtavaa datan määrää, jota luodaan ja tallennetaan jatkuvasti. Tämä määrä voi olla niin suuri, että perinteiset tietokantajärjestelmät eivät pysty käsittelemään tai tallentamaan sitä tehokkaasti. Nopeus kuvaa sitä, että Big Dataa sekä tuotetaan että pyritään käsittelemään suurella nopeudella. Tämä tarkoittaa, että datavirta on jatkuvaa ja reaaliaikaista, jolloin datan kerääminen, prosessointi ja analysointi täytyy tapahtua nopeasti päätöksenteon tueksi. Valikoima kuvaa tietojen erilaisuutta. Big Data voi olla monimuotoista, sisältäen jäsenneltyä, puolijäsenneltyä tai jäsentymätöntä dataa. Lisäksi tallennettavat tiedot voivat olla muutakin kuin tekstiä, kuten kuvia, videoita tai sensoreiden tuottamaa dataa (Hashem ym. 2015).

4.2 Big Data tietokantojen näkökulmasta

Big Dataa käsitellessä, relaatiotietokantojen hallintajärjestelmillä on kolme ongelmaa, joihin ne vastaavat heikosti: skaalautuvuus-, joustavuus- ja suorituskykyongelmat (Oussous ym. 2013).

Relaatiotietokantojen vertikaalinen skaalautuvuus tuo ongelmia Big Datan hallitsemiseen, sillä kuten aiemmin mainittu kolmen v:n mallin nopeus meille kertoo, tuotetaan dataa niin nopeasti, että palvelinten on kokoajan laajennuttava (N. Khan ym. 2023). Vertikaalinen skaalautuvuus tarkoittaisi, että jo ennalta oleviin palvelimiin lisättäisiin tehokkaampia suorittimia tai nopeampia muisteja, mutta tämä voi olla mahdotonta, jos käytössä on jo uusinta teknologiaa. Tällainen tilanne pakottaisi relaatiotietokantojen käyttäjän pyrkimään palvelinten ho-

risontaaliseen skaalautumiseen, eli lisäämään rinnakkaisia palvelimia. Ei-relaatiotietokannat ovat luotu hajautettuja järjestelemiä varten ja horisontaalinen skaalautuvuus mielessä, joten ne suoriutuvat myös paremmin Big Datan vaatimasta skaalautuvuudesta (W. Khan ym. 2023).

Toinen relaatiotietokantojen kohtaamista ongelmista on joustavuuden puute, sillä relaatiotietokannoissa on ennalta määrätty skeema. Big Datan muoto ja rakenne voi muuttua nopeasti ja skeeman muutokset ovat hyvin monimutkaisia ja ne voivat johtaa palvelinongelmiin ja suorituskykyyn heikkenemiseen, sekä usein johtaa myös lisäkustannuksiin ylläpidon osalta. Ei-relaatiotietokannat ovat skeemattomia, joten skeemattomuutensa vuoksi Big Datan muutokset eivät aiheuta muutoksia ei-relaatiotietokantojen toimivuuteen. (Oussous ym. 2013)

Kolmantena ongelmana on suorituskykyyn liittyvät puutteet. Nämä puutteet ovat vahvasti liitoksissa sekä skaalautuvuus- että joustavuusongelmiin, sillä heikompi skaalautuminen ja/tai liian tiukka skeema johtaa suorituskyvyn heikkenemiseen (Cattell 2011).

Koska ei-relaatiotietokannat on suunniteltu horisontaalinen skaalautuvuus ja joustavuus mielessä, ja suorituskyky on näistä kahdesta ominaisuudesta riippuvainen, on ei-relaatiotietokannat erinomainen valinta, kun halutaan valita tietokanta Big Datan tallentamiseksi.

Seuraavassa luvussa tarkastellaan aiemmissa luvuissa esiteltyjen ei-relaatiotietokantojen sekä relaatiotietokantojen suorituskykyä tässä luvussa kuvaillun Big Datan näkökulmasta.

5 Dokumentti- ja saraketietokantojen suorituskyky Big Datan näkökulmasta

Tässä luvussa tarkastellaan, millaisia tuloksia on dokumentti- ja saraketietokantojen suorituskykyvertailuista saavutettu. Graafitietokantojen ja avain-arvo tietokantojen suorituskykyvertailut ovat rajattu pois, puutteellisen lähdemateriaalin vuoksi.

Tietokantajärjestelmien suorituskykyä tarkastellessa on pakko tehdä rajauksia. Ei-relaatiotietokannoista dokumenttitietokanta MongoDB sekä saraketietokanta Cassandra ovat suosiossaan niin ylivoimaisia, että ne ovat valikoituneet valtaosassa kirjallisuutta suorituskyvyllisen tarkastelun kohteeksi, joko artikkelien ainoana tietokantana tai vain yhdestä muutamaan tietokantajärjestelemään verrattuna. Relaatiotietokantojen puolelta MySQL on saanut samanlaisen roolin kirjallisuudessa. Syitä näiden tietokantojen hallintajärjestelmien suosioon perustuu pääasiallisesti kummankin kattavaan dokumentointiin sekä luotettavuuteen (Győrödi ym. 2015).

Edellä mainittujen vuoksi tarkastelemme vain dokumentti- ja saraketietokantojen suorituskykyä, eli tutkielmasta on rajattu pois avain-arvo tietokannat sekä graafitietokannat. Avain-arvo tietokannoista Redis ja Riak, olisivat olleet mielenkiintoisia käsiteltäviä Gortonin ym. (2015) tutkimuksen tulosten perusteella, mutta graafitietokannoista ei tutkimuksissa ollut paljoa vertailukelpoista tietoa suorituskyvystä.

5.1 Dokumenttitietokantojen suorituskyky

Dokumenttitietokantojen suorituskykyä arvioidaan mittaamalla, kuinka kauan tietokannalta kestää suorittaa perusoperaatiot: lisääminen (insert), valitseminen (select), päivittäminen (update) ja poistaminen (delete). Koska MongoDB on sekä suosituin dokumenttitietokanta että suosituin NoSQL-tietokanta (DB-Engines), sen valitseminen dokumenttitietokantojen suorituskyvyn tarkasteluun on luonnollinen valinta.

Győrödi ym. (2015) vertailivat MongoDB:n suorituskykyä relaatiotietokantojen toiseksi suosituimpaan vaihtoehtoon, MySQL:ään. Tutkimuksessa luotiin hierarkkinen, nelitasoinen tietokanta, jossa simuloitiin keskustelupalstan toimintaa. Tietokannan tasot järjestettiin suu-

remmasta pienempään seuraavasti: keskustelupalsta, alapalsta, keskustelu ja kommentti. Toimintaperiaate oli looginen: alemman tason taulu (SQL) tai dokumentti (NoSQL), kuten kommentti, voi kuulua vain yhteen keskusteluun, joka voi puolestaan kuulua vain yhteen alapalstaan. Alapalstalla voi kuitenkin olla useita keskusteluja, joissa taas on useita kommentteja. Relaatiotietokannoissa yleisesti käytetty viiteavain on MongoDB:ssa korvattu id-kentällä, jolla voidaan viitata esimerkiksi siihen keskusteluun, johon kommentti kuuluu.

Parker ym. (2013) vertailivat MongoDB:tä ja Microsoft SQL Server Expressiä, kolmanneksi suosituinta tietokannanhallintajärjestelmää (DB-Engines). Heidän testeissään oli kolmikerroksinen hierarkia, mutta muutoin toimintaperiaate vastasi Győrödin ym. (2015) tutkimusta. Parkerin ym. (2013) tutkimuksessa käytettiin yliopiston hierarkiaa, jossa kerrokset olivat tiedekunta, projektit ja käyttäjät.

Tietokantaan lisäämistä testattaessa Győrödi ym. (2015) lisäsivät tietokantaan 10 000 käyttäjää. MongoDB:llä suoritettuna toimenpide kesti 0,29 sekuntia, kun taas MySQL:llä vastaavaan toimenpiteeseen kului 440 sekuntia. Toisessa tehtävässä luotiin 5 000 taulua tai dokumenttia kullekin hierarkian tasolle. MongoDB:llä suoritukseen kului 3,33 sekuntia ja MySQL:n 1010 sekuntia. Parkerin ym. (2013) tutkimuksessa rakennettiin lisäämistehtävää varten neljä erilaista tietokantarakennetta. Ensimmäisessä testissä luotiin hierarkian mukaiset 4 tiedekuntaa, 16 projektia ja 128 käyttäjää. Toisessa testissä vastaavat luvut olivat 4 tiedekuntaa, 64 projektia ja 256 käyttäjää. Kolmannessa testissä rakennettiin 16 tiedekuntaa, 512 projektia ja 1 024 käyttäjää, kun taas neljännessä testissä luotiin 128 tiedekuntaa, 8 192 projektia ja 4 096 käyttäjää. Parkerin ym. (2013) tutkimuksessa suoritusaikoja ei ilmoitettu tarkasti sekunneissa, vaan erot esitettiin prosentuaalisesti. MongoDB oli ensimmäisessä testissä 34 % hitaampi ja testeissä 2 ja 4 7 % hitaampi kuin Microsoft SQL Server. Microsoft SQL Server oli 10 % hitaampi kolmannessa testissä.

Valintatesteissä Győrödi ym. (2015) pyysivät tietokannalta ensiksi kaikkia keskusteluja, joihin käyttäjä osallistui. Tässä testissä MongoDB:n suorittama toimenpide kesti 0,0011 sekuntia, kun vastaava toimenpide MySQL:llä vei 0,0018 sekuntia. Toisessa testissä tiedusteltiin käyttäjän aloittamien keskustelujen määrää. Tästä testistä MongoDB suoriutui 0,0011 sekunnissa ja MySQL 0,6478 sekunnissa. Valintaa suorittaessa Parker ym. (2013) havaitsivat MongoDB:n vaikeudet selvitä monimutkaisista kyselyistä. Vaikka valtaosassa valintatestejä

MongoDB päihitti Microsoft SQL Server Expressin suoritusnopeudessa, aggregointifunktioissa, kuten SQL:n keskiarvon laskemisessa, MongoDB oli jopa 23 kertaa hitaampi.

Päivitystesteissä Győrödi ym. (2015) testasivat ensiksi kommenttien päivittämistä. Kommenttien päivittämiseen MongoDB:llä kului 0,0021 sekuntia, kun MySQL:llä vastaava toimenpide vei 0,0987 sekuntia. Toisessa testissä päivitettiin käyttäjän sähköpostiosoite, ja tämä toimenpide vei MongoDB:llä 0,0013 sekuntia ja MySQL:llä 0,0428 sekuntia. Parker ym. (2013) hyödynsivät päivitystesteissään heidän lisäämistehtävässään luotuja tietokantarakenteita. Päivitystestejä suoritettiin kolme, ja päivitykset tehtiin hyödyntäen tietokantaan tallennettuja tietueita. Ensimmäisessä testissä päivitys tehtiin käyttäjän nimeä hyödyntäen. MongoDB:ssä tämä osoittautui ongelmalliseksi, koska nimi oli indeksoimaton, toisin kuin toisissa testeissä käytetyt tunnisteet. Tämä johti jopa viisinkertaiseen hitauteen MongoDB:ssä verrattuna SQL:ään. Tiedekuntaa päivitettäessä suoritusnopeus kääntyi päinvastaiseksi, MongoDB suoriutui nopeammin kaikissa tietokantarakenteissa, ja SQL:llä kesti jopa kymmenkertaisesti enemmän aikaa MongoDB:hen verrattuna. Projektin tunnisteisiin viittaavissa päivityksissä SQL:llä kesti jopa kolmetoistakertaisesti enemmän aikaa.

Poistotestissä Győrödi ym. (2015) ensiksi poistivat kaikki käyttäjän viestit. MongoDB:llä tämä toimenpide vei 0,0028 sekuntia, kun vastaava toimenpide MySQL:llä kesti 0,3524 sekuntia. Toisessa testissä Győrödi ym. (2015) poistivat kaikki käyttäjän aloittamat keskustelupalstat, ja tämä toimenpide vei MongoDB:llä 0,0064 sekuntia ja MySQL:llä 0,8231 sekuntia.

Tuloksista, jotka saatiin Győrödin ym. (2015) suorittamista testeistä, voidaan havaita, että MongoDB suoriutui nopeammin jokaisesta testistä. Parkerin ym. (2013) tutkimuksessa esiintynyt ristiriitaisuus suorituskyvyssä, erityisesti aggregointifunktioissa, selittyy heidän mukaansa sillä, että MongoDB:n käyttäjän on luotava aggregointifunktiot itse MongoDB:n MapReduce-metodilla, toisin kuin SQL:ssä, jossa nämä funktiot ovat valmiiksi sisäänrakennettuja. Lisäksi Parker ym. (2013) selittävät MongoDB:n korkeaa suorituskykyä muissa testeissä sillä, että MongoDB tallentaa dokumentit muistiin, kun taas SQL-tietokannat tallentavat tietonsa levylle, jolle suoritetut luku- ja kirjoitustoimenpiteet ovat huomattavasti hitaampia.

5.2 Saraketietokantojen suorituskyky

Saraketietokannoista suosituin (DB-Engines), Cassandra, on valikoitunut monessa saraketietokantoja sisältävissä suorituskykyvertailuista vertailuissa testien kohteeksi. Gortonin ym. (2015) tutkimuksessa Cassandraa verrattiin dokumenttitietokanta MongoDB:hen sekä avain-arvo tietokanta Riakiin. Khanin ym. (2023) saraketietokantoja vertailevassa artikkelissa verrattiin Cassandraa ja Apache HBasea. Kummassakin edellä mainituista tutkimuksista käytettiin Yahoo Cloud Servce Benchmark (YCSB) sovellusta, jota käytetään erityisesti NoSQL tietokantojen suorituskykyvertailuissa.

Gortonin ym. (2015) testeissä luotiin terveystietoja ylläpitävä järjestelmä, jossa oli hajautetut palvelimet, joilla sijaitsi miljoona kappaletta potilastietoja sekä 10 miljoonaa laboratoriotulosta. Asiakkaita oli testeissä vaihtelevasti 1 ja 1000 kappaleen välillä. Tutkimuksessa simuloitiin kokoonpanoa, joka muistuttaisi kolmen eri datakeskuksen käyttöä, jokainen simuloitu datakeskus oli oma klusteri, joka sisälsi 3 solmua. Data oli yhdeksässä solmussa, joista kolme solmua sisälsi datan ja kuusi viimeisintä solmua oli datan replikointeja. Osituksesta vastasi Cassandran sisäänrakennettu datakeskus-tietoinen jakelu (data center aware distribution).

Gortonin ym. (2015) tutkimuksessa testattiin Cassandran suorituskykyä sekä BASE-periaatteen mukaisesta, lopulta eheä, näkökulmasta että vahvan eheyden näkökulmasta. Lopulta eheä vaatimukseen riitti, että mitatessa kirjoitus- tai lukuoperaatiot ovat jossakin solmussa päivittyneet, mutta vahvaan eheyteen vaadittiin, että jokainen solmu on päivittynyt uusimpaan tilaan. Cassandran suorituskyky parhaillaan oli 3200 operaatiota per sekunti, kun Riak 480 ja MongoDB:llä 225. Toisin kuin Riakin ja MongoDB:n kohdalla, Cassandran suorituskyky lukuoperaatioissa on sama, riippumatta siitä, onko kyseessä yhden vai yhdeksän solmun kokoonpano. Kirjoitusoperaatioissa Cassandran suorituskyky parani hieman, kun solmujen määrä kasvoi. Ilmiö selittyy sillä, että vähäisemmän solmumäärän käyttö aiheuttaa suorituskykyä hidastavan pullonkaulan, kun taas kirjoitusoperaatioiden jakaminen kolmelle solmulle kasvattaa Cassandran suorituskykyä, vaikka tämä tuottaa ylimääräistä työtä, koska data joudutaan replikoimaan kuuteen solmuun (Gorton ja Klein 2015).

Cassandran heikkoudeksi havaittiin, että esimerkiksi 32 asiakkaan vertailussa, Riakin vii-

ve oli viidesosa ja MongoDB:n neljäsosa, Cassandran viiveestä. Myös Gortonin ym. (2015) vahvan eheyden testeissä Cassandra ei suoriutunut yhtä hyvin, kuin lopulta eheä -näkökulman testeistä. Cassandran suorituskyky laski 25 % kuin verrattavan Riakin suorituskyky vain 10 %. Cassandran suorituskyky on havaittu olevan nopea kirjoitus- ja poisto-operaatioissa, mutta hidas lukuoperaatioissa (Li ja Manoharan 2013). Cassandra tallettaa tietonsa relaatiotietokantojen mukaisesti levylle, toisin kuin esimerkiksi dokumenttitietokanta MongoDB. Tämän seurauksena, jos tietokantaan kohdistettavat operaatiot voidaan hakea muisteista levyn sijaan, on MongoDB lähes väistämättä nopeampi (Antas, Silva ja Bernardino 2022).

6 Pohdinta

Pilvi- ja webpalveluiden sekä mobiilialustojen kasvaneen suosion vuoksi, kaipasivat vanhat tietokantamallit päivitystä näiden uusien vaatimusten täyttämiseen. Vanhat toimintamallit, kuten ennalta määrätyt skeemat, eivät sopineet tähän uuteen tietokantatarpeeseen. NoSQL-tietokannat ovat tarjonneet vaihtoehdon, joka tulosten perusteella täyttää nämä vaatimukset. Mikään ei myöskään näytä estävän NoSQL-tietokantojen kasvavaa läsnäoloa yritysten tietokantavalinnoissa.

Tämän tutkielmaan tavoitteena oli selvittää vastauksia seuraaviin tutkimuskysymyksiin:

1. Millainen suorituskyky NoSQL-tietokannoilla on verrattuna SQL-tietokantoihin Big Datan käsittelyssä.
2. Miten NoSQL-tietokantatyyppien suorituskyvyt vertautuvat toisiinsa Big Datan näkökulmasta.

Kattavaa kirjallisuutta liittyen eri ei-relaationaalisiin tietokantatyyppien suorituskykyyn osoittautui todella haastavaksi. Kirjallisuutta asiaan liittyen löytyi karkeasti kategorisoituna kahdenlaista: 1) kirjallisuutta, jossa verrataan tietokantatyyppejä keskenään, mutta ilman suorituskyvyn näkökulmaa. 2) kirjallisuutta, jossa suorituskykyä vertaillaan, mutta tietokantatyyppien sijaan käytetään valikoituja tietokantojen hallintajärjestelmiä, yleensä relaatiotietokannoista MySQL ja dokumenttitietokannoista MongoDB.

Ensimmäistä tutkimuskysymystä varten, jouduttiin tietokantatyyppien vertailua rajoittamaan puutteellisen kirjallisuuden vuoksi. Vastauksia kuitenkin MongoDB:n ja MySQL:n vertailuun löytyi, joka on hyvin suuntaa antava vertailu, kun vertaillaan relaatiotietokantoja ja ei-relaatiotietokantoja, sillä kyseessä on luokkansa suosituimmat tietokantojen hallintajärjestelmät. Ei-relaationaalisten tietokantojen keskinäisiä vertailuja löytyi myös useita, näissä etenkin MongoDB ja Cassandra nousivat esille.

Tässä tutkielmassa tarkasteltiin suorituskykyä Big Datan näkökulmasta, jolloin tutkimuskysymykseen 1 vastaus on, että verrattuna relaatiotietokantoihin, on NoSQL-tietokantojen suorituskyky lähes poikkeuksetta korkeampi.

Tutkimuskysymykseen 2, päädyin samanlaiseen johtopäätökseen Razun (2018) kanssa: ”Verrattaessa dokumenttitietokannoista MongoDB:tä sekä Couchbasea, saraketietokannoista Cassandraa ja HBasea sekä avain-arvo tietokantaa Redistä, datan eheyden ollessa tärkeintä, MongoDB on paras vaihtoehto. Datan ollessa järjestäytymätöntä ja käyttäjän tarvittaessa korkeaa suorituskykyä, Redis on vahvin vaihtoehto ja jos dataa on suuri määrä, Cassandra on suorituskykyisin vaihtoehto”. Tämän tutkielman keskittyessä Big Datan näkökulmaan, on Cassandra tämän tutkielman tietojen varjolla vahvin vaihtoehto Big Datan hallitsemiseen.

Jatkotutkimuksena olisi kiinnostavaa kerätä monipuolisesti erityyppistä Big Dataa, joka soveltuisi eri NoSQL-tietokantatyyppien käyttöön. Tämä sisältäisi dokumentti- ja saraketietokantojen lisäksi myös graafitietokantojen ja avain-arvo-tietokantojen käytön. Tällainen monipuolinen aineisto tarjoaisi mahdollisuuden tarkastella näiden vähemmän suosittujen NoSQL-tietokantatyyppien vahvuuksia ja soveltuvuutta Big Datan käsittelyyn.

Toinen tärkeä jatkotutkimuksen kohde olisi selvittää kattavasti, miten erilaiset tietokoneiden kokoonpanot vaikuttavat NoSQL tietokantatyyppien suorituskykyyn. Tutkielman kirjoitushetkellä, keväällä 2024 ei asiasta vielä löydy kattavaa tutkimusta. Esimerkiksi kaksi suurinta suoritinvalmistajaa: AMD ja Intel, saavat hyödynnettyä muistia eri tavalla. Tunnetusti AMD on onnistunut saamaan suorittimensa hyödyntämään korkeita muistinopeuksia paremmin kuin Intel, joka teoriassa tarkoittaisi sitä, että AMD suorituskyky voisi olla parempi. Korkeampien muistinopeuksien hyödyntäminen voi myös antaa etumatkaa MongoDB:lle verrattuna esimerkiksi Cassandraa, sillä Antas (2022) havaitsi MongoDB:n hyötyvän enemmän muisteista kuin Cassandra. Kuitenkin Intelin puolesta argumentteja voisi löytyä muun muassa paremman vakauden, suuremman suosion, sekä paremman dokumentoinnin vuoksi.

Lähteet

Ahmed, Md. Razu, Mst.Arifa Khatun, Md. Asraf Ali ja Kenneth Sundaraj. 2018. “A literature review on NoSQL database for big data processing”. *International Journal of Engineering and Technology* 7 (kesäkuu): 902–906. https://doi.org/10.14419/ijet.v7i2.12113.

Antas, João, Rodrigo Silva ja Jorge Bernardino. 2022. “Assessment of SQL and NoSQL Systems to Store and Mine COVID-19 Data”. *Computers* 11 (helmikuu): 29. https://doi.org/10.3390/computers11020029.

Cattell, Rick. 2011. “Scalable SQL and NoSQL data stores”. *SIGMOD Rec.* (New York, NY, USA) 39 (4): 12–27. ISSN: 0163-5808. https://doi.org/10.1145/1978915.1978919.

Corbellini, Alejandro, Cristian Mateos, Alejandro Zunino, Daniela Godoy ja Silvia Schiaffino. 2017. “Persisting big-data: The NoSQL landscape”. *Information Systems* 63:1–23. ISSN: 0306-4379. https://doi.org/10.1016/j.is.2016.07.009.

DB-Engines. *DB-Engines Ranking.* https://db-engines.com/en/ranking. Viitattu: 17-04-2024.

Fowler, Adam. 2015. “NoSQL For Dummies”. https://api.semanticscholar.org/CorpusID:195975329.

Fox, A. ja E.A. Brewer. 1999. “Harvest, yield, and scalable tolerant systems”, 174–178. https://doi.org/10.1109/HOTOS.1999.798396.

Gorton, Ian ja John Klein. 2015. “Performance Evaluation of NoSQL Databases: A Case Study”. Helmikuu. https://doi.org/10.1145/2694730.2694731.

Győrödi, Cornelia, Robert Gyorodi, George Pecherle ja Andrada Olah. 2015. “A Comparative Study: MongoDB vs. MySQL” (kesäkuu). https://doi.org/10.13140/RG.2.1.1226.7685.

Haerder, Theo ja Andreas Reuter. 1983. “Principles of transaction-oriented database recovery”. *ACM Comput. Surv.* (New York, NY, USA) 15 (4): 287–317. ISSN: 0360-0300. https://doi.org/10.1145/289.291.

Hashem, Ibrahim Abaker Targio, Ibrar Yaqoob, Nor Badrul Anuar, Salimah Mokhtar, Abdullah Gani ja Samee Ullah Khan. 2015. “The rise of “big data” on cloud computing: Review and open research issues”. *Information Systems* 47:98–115. ISSN: 0306-4379. https://doi.org/10.1016/j.is.2014.07.006.

Helland, Pat. 2016. “Life Beyond Distributed Transactions: An apostate’s opinion”. *Queue* (New York, NY, USA) 14 (5): 69–98. ISSN: 1542-7730. https://doi.org/10.1145/3012426.3025012.

Kacfah Emani, Cheikh, Nadine Cullot ja Christophe Nicolle. 2015. “Understandable Big Data: A survey”. *Computer Science Review* 17:70–81. https://doi.org/10.1016/j.cosrev.2015.05.002.

Khan, Nasrullah, Nijad Ahmad, Neeta Sharma ja Mohammad Niazy. 2023. “A Study of Performance Evaluation and Comparison of NoSQL Database Choosing for Big Data: HBase and Cassandra Using YCSB”. *IOSR Journal of Computer Engineering* 25 (kesäkuu): PP 01–12. https://doi.org/10.9790/0661-2503010112.

Khan, Wisal, Teerath Kumar, Cheng Zhang, Kislay Raj, Arunabha M. Roy ja Bin Luo. 2023. “SQL and NoSQL Database Software Architecture Performance Analysis and Assessments: A Systematic Literature Review”. *Big Data and Cognitive Computing* 7 (2). ISSN: 2504-2289. https://doi.org/10.3390/bdcc7020097.

Li, Yishan ja Sathiamoorthy Manoharan. 2013. “A performance comparison of SQL and NoSQL databases”. Teoksessa *2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM),* 15–19. https://doi.org/10.1109/PACRIM.2013.6625441.

Li, Ziqi. 2018. “NoSQL Databases”. *Geographic Information Science: Technology Body of Knowledge* 2018 (huhtikuu). https://doi.org/10.22224/gistbok/2018.2.4.

Medjahed, Brahim, Mourad Ouzzani ja Ahmed K. Elmagarmid. 2009. “Generalization of ACID Properties”. Teoksessa *Encyclopedia of Database Systems,* toimittanut LING LIU ja M. TAMER ÖZSU, 1221–1222. Boston, MA: Springer US. ISBN: 978-0-387-39940-9. https://doi.org/10.1007/978-0-387-39940-9_736.

Moniruzzaman, A B M ja Syed Akhter Hossain. 2013. “NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison”, eprint: 1307.0191.

Oussous, Ahmed, Fatima-Zahra Benjelloun, Ayoub Ait Lahcen ja Samir Belfkih. 2013. “Comparison and classification of nosql databases for big data”. *International Journal of Database Theory and Application* 6 (4.2013).

Parker, Zachary, Scott Poe ja S.V. Vrbsky. 2013. “Comparing nosql mongodb to an sql db”. Huhtikuu. https://doi.org/10.1145/2498328.2500047.

Pritchett, Dan. 2008. “BASE: An Acid Alternative: In partitioned databases, trading some consistency for availability can lead to dramatic improvements in scalability.” *Queue* (New York, NY, USA) 6 (3): 48–55. ISSN: 1542-7730. https://doi.org/10.1145/1394127.1394128.

Seeger, Marc. 2009. “Key-value stores: a practical overview”. *Computer Science and Media, Stuttgart.*

Stonebraker, Michael. 2010. “SQL databases v. NoSQL databases”. *Commun. ACM* 53:10–11. https://api.semanticscholar.org/CorpusID:13959501.

Strozzi, Carlo. 1998. “A relational database management system”. http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/NoSQL/Home%20Page.