Annika Tuulivuori

NewSQL-tietokantojen skaalautuvuus

Tietotekniikan kandidaatintutkielma

8. toukokuuta 2023

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Annika Tuulivuori

Yhteystiedot: annika.sk.tuulivuori@student.jyu.fi

Ohjaaja: Antti-Jussi Lakanen

Työn nimi: NewSQL-tietokantojen skaalautuvuus

Title in English: Scalability of NewSQL databases

Työ: Kandidaatintutkielma

Opintosuunta: Tietotekniikka

Sivumäärä: 19+0

Tiivistelmä: Yksi relaatiotietokantojen heikkouksista on sen skaalautuvuus. Relaatiotietokannat skaalautuvat pystysuoraan, mikä on kallista. NoSQL-tietokantojen skaalaaminen on edullisempaa, mutta sen seurauksena ACID-periaatteen mukaisista eheyden vaatimuksista joudutaan luopumaan. NewSQL-tietokantojen avulla pyritään tarjoamaan NoSQL-tietokantojen skaalautuvuus sekä perinteisten relaatiotietokantojen ACID-periaatteen mukaiset takuut transaktioille. Tämän tutkimuksen tarkoituksena on selvittää, mitä tarkoitetaan tietokannan skaalautuvuudella sekä miten se näkyy NewSQL-tietokannoissa. Lopuksi tutkitaan eri kategoriaan kuuluvien NewSQL-tietokantojen skaalautuvuutta.

Avainsanat: skaalautuvuus, relaatiotietokanta, NoSQL-tietokanta, NewSQL-tietokanta, tietokannan hallintajärjestelmä

Abstract: One of the weaknesses of relational databases is its scalability. Relational databases scale vertically, which is expensive. On the other hand, scaling NoSQL databases is more affordable, but as a result the consistency requirements have to be waived. The aim of NewSQL databases is to provide the scalability of NoSQL and ACID properties from relational databases for transactions. The purpose of this study is to find out what database scalability is and how it is reflected in NewSQL databases.

Keywords: scaling, scalability, relational, NoSQL, NewSQL, database, database management system

Kuviot

Sisällys

1 Johdanto

Perinteiset relaatiotietokantojen hallintajärjestelmät ovat olleet olemassa jo 1970-luvulta saakka. Relaatiotietokantojen synnystä puhuttaessa viitataan usein tutkijaan Edgar F. Coddiin, joka kehitti relaatiotietokantojen relaatiomallin. Perinteiset relaatiotietokantojen hallintajärjestelmät on toisaalta luotu aikana, jolloin saatavilla oleva laitteisto sekä tiedon tallennus- ja käsittelyvaatimukset olivat hyvin erilaiset kuin ne ovat nykypäivänä (Stonebraker ym. 2007).

Relaatiotietokantojen yksi suurimmista heikkouksista on niiden skaalautuvuus. Skaalautuvuudella tarkoitetaan tietokannan kykyä laajentua ilman suuria muutoksia sen arkkitehtuuriin (*structural scalability*) sekä kykyä toimia sulavasti, kun syötetty tieto lisääntyy (*load scalability*) (Bondi 2000). Relaatiotietokantojen skaalautuvuuden lisäämiseksi tietokanta tulee siirtää tehokkaammalle ja samalla kalliimmalle tietokoneelle (Leavitt 2010). Toinen vaihtoehto on kasvattaa palvelimen prosessorien tai muistin määrää (Pokorny 2011). Joka tapauksessa, jotta tietokannan hallintajärjestelmä pystyy käsittelemään kasvavaa määrää informaatiota, tulee sen kapasiteettia kasvattaa ja näin ollen kustannukset kasvavat (Oussous ym. 2013).

Oussous ym. (2013) mukaan yhtenä ratkaisuna tietokantojen hallintajärjestelmien skaalautuvuusongelmiin on nähty NoSQL-tietokannat (”No SQL” tai ”Not Only SQL”). Oussous ym. (2013) kertovat, että relaatiotietokantojen etu on niiden takuu luotettavuudesta ja ACID-periaatteiden (*atomicity, consistency, isolation, durability*) noudattamisesta. NoSQL-tietokannat eivät noudata ACID-periaatteita, sillä NoSQL kehitettiin tarjoamaan ratkaisu suuren strukturoimattoman datamäärän tallennusongelmiin. Tallennusongelmiksi Oussous ym. mainitsevat muun muassa relaatiotietokannan skaalaamisesta johtuvan kustannuksien nousun sekä sen, että relaatiotietokantoja ei ole suunniteltu käsittelemään ei-strukturoitua dataa.

NewSQL-tietokannat ovat Pavlon ja Aslettin (2016) mukaan moderni vaihtoehto relaatiotietokantojen hallintajärjestelmille. NewSQL-tietokantajärjestelmien avulla pyritään tarjoamaan sama skaalautuvuus kuin NoSQL tarjoaa ja samalla säilyttää ACID-periaatteen mukaiset takuut transaktioille eli tietokantatapahtumille. Samanaikaisesti halutaan säilyttää perinteinen relaatiomalli relaatiotietokannoista/SQL:stä. Näiden ominaisuuksien avulla voidaan käsitellä suuria määriä samanaikaisia tapahtumia.

Tämän tutkielman tarkoituksena on selvittää, mitä NewSQL-tietokantojen hallintajärjestelmistä ja erityisesti niiden skaalautuvuudesta on tutkittu. Tutkimus suoritetaan kirjallisuuskatsauksena. Kirjallisuuskartoituksen keskeisiä käsitteitä ovat NewSQL-tietokannat, relaatiotietokannat, NoSQL-tietokannat sekä tietokannan skaalautuvuus. Tässä tutkimuksessa ei oteta kantaa eri tietokantaratkaisujen käytännön toteutuksiin.

Tutkielman luvussa 2 tutkitaan perinteisiä relaatiotietokantajärjestelmiä sekä NoSQL-tietokantoja. Lisäksi pyritään selvittämään, mitkä tekijät tai puutteet edellä mainituissa tietokantaratkaisuissa loivat tarpeen kehittää uusi ratkaisu niiden tilalle. Luvussa 3 tutkitaan NewSQL-tietokantoja, ACID-periaatteiden toteutumista NewSQL-tietokannoissa sekä niiden luokittelua. Luvussa 4 tutkitaan, kuinka NewSQL-tietokannat skaalautuvat, ja eritellään niiden ositusmenetelmiä.

2 Tietokantojen skaalautuvuus ennen NewSQL-tietokantoja

2.1 Perinteiset relaatiotietokannat

Relaatiomallin kehittäjä Codd (1970, 1979) määrittelee relaatiotietokannan olevan ajallisesti muuttuva kokoelma dataa. Codd (1979) määrittelee perusrelaatioiksi (*base relations*) ne suhteet, jotka on määritelty riippumattomasti muista tietokannan relaatioista. Toisin sanoen mikään perussuhde ei ole täysin johdettavissa muista perussuhteista. Johdetut relaatiot (*derived relations*) sen sijaan voidaan johtaa kokonaan perusrelaatiosta. Coddin mukaan johdetut suhteet esittävät loppukäyttäjälle yksilöllisen näkymän tietokannasta. Kuviossa 1 olevat taulut nimeltä Kurssit ja Opettajat ovat perusrelaatioita. Kurssien_opettajat-taulu on johdettu relaatio, joka on johdettu perusrelaatioista.

Kurssit

id	kurssinNimi	opintopisteet	opettaja_id
1	Kurssi1	5	1
2	Kurssi2	3	2
3	Kurssi3	4	3
4	Kurssi3	5	4

Opettajat

id	opettajanNimi	työhuone	puhNum
1	V. Virtanen	417.1	050 111 1111
2	K. Korhonen	417.2	050 222 2222
3	M. Mäkinen	418.3	050 333 3333
4	N. Nieminen	418.4	050 444 4444

Kurssien_opettajat

opettajanNimi	kurssinNimi	opintopisteet
V. Virtanen	Kurssi1	5
K. Korhonen	Kurssi2	3
M. Mäkinen	Kurssi3	4
N. Nieminen	Kurssi3	5

Kuvio 1. Perusrelaatiot ja johdettu relaatio

Tietokantojen hallintajärjestelmien kehityksessä on pitkään luotettu pystysuoraan skaalautuvuuteen (*vertical scaling, scale-up*). Pystysuoralla skaalaamisella tarkoitetaan tietokannan siirtämistä uudemmalle, isommalle ja siten myös kalliimmalle palvelimelle (Pokorny 2011). Leavitt (2010) toteaa, että data on jaettava useammalle serverille skaalautuvuuden toteuttamiseksi, kun tietokannan skaalaaminen pystysuoraan on liian kallista. Datan jakamista useammalle palvelimelle kutsutaan vaakasuoraksi skaalautuvuudeksi (*horizontal scaling,*

scale-out) (Pokorny 2011). Cattellin (2011) mukaan perinteiset relaatiotietokannat eivät ole saavuttaneet tehokasta vaakasuoraa skaalautuvuutta. Hän kuitenkin kuvailee artikkelissaan uusia SQL-tietokantoja, joiden avulla pyritään saavuttamaan vaakasuora skaalautuvuus relaatiotietokannalle. Nämä kyseiset uudet relaatiotietokannat ovat toisaalta Pavlon ja Aslettin (2016) luokitteluiden mukaisesti NewSQL-tietokantoja ja niitä tutkitaan lisää luvussa 3.

Pokorny (2011) korostaa, että yksi relaatiotietokantojen hallintajärjestelmien perusominaisuuksista on niin kutsuttujen ACID-periaatteiden noudattaminen tapahtumien käsittelyssä. Härder ja Reuter (1983) määrittelevät periaatteiden koostuvan atomisuudesta (*atomicity*), eheydestä (*consistency*), eristyneisyydestä (*isolation*) sekä pysyvyydestä (*durability*). Atomisuudella tarkoitetaan sitä, että tapahtuma joko käsitellään kokonaan tai sitä ei käsitellä ollenkaan. Eheys taas takaa sen, että jokaisen tapahtuman jälkeen tietokanta pysyy johdonmukaisena. Eristyneisyys tarkoittaa jokaisen tapahtuman piilottamista toisilta samanaikaisilta tapahtumilta. Pysyvyydellä tarkoitetaan sitä, että tapahtuman suorittamisen ja sen tulosten siirtämisen jälkeen tietokannan on taattava, että muutokset pysyvät siellä myös mahdollisissa toimintahäiriötilanteissa.

Vaikka ACID-periaatteet ovat yksi relaatiotietokantojen perusominaisuuksista (Pokorny 2011), niistä ei puhuta tutkimuksissa samanaikaisesti relaatiotietokantojen skaalautuvuuden kanssa. Yksi syy tähän voi olla se, että järkevin tapa toteuttaa relaatiotietokannan skaalautuminen on pystysuunnassa (Pokorny 2011). Lisäksi tietokannan skaalautuvuus ja eheys ovat monesti kilpailevia ominaisuuksia. Vaakasuoraan skaalautuvassa tietokantajärjestelmässä ei välttämättä pystytä takaamaan ACID-periaatteiden mukaista atomisuutta tai eristyneisyyttä. Esimerkiksi NoSQL-tietokannoissa priorisoidaan skaalautuvuutta tietokannan eheyden sijaan, kun taas perinteisissä relaatiotietokannoissa priorisoidaan ACID-periaatteiden täyttymistä. Tästä syystä perinteisiä relaatiotietokantoja ei ole järkevää skaalata vaakasuoraan. (Cattell 2011.)

2.2 NoSQL-tietokannat

NoSQL-tietokantojen (”Not Only SQL” tai ”Not Relational”) kehityksen lähtökohdiksi mainitaan suuren strukturoimattoman informaatiomäärän tallennusongelmat sekä tarve horisontaaliselle skaalautuvuudelle (Cattell 2011; Han ym. 2011; Oussous ym. 2013). NoSQL-tietokantoja ovat kaikki ne tietokannat, joita ei voida kutsua relaatiotietokannoiksi (Leavitt 2010). NoSQL-tietokannat luokitellaan yleensä kolmeen pääkategoriaan: avain-arvo-tietokannat, dokumenttitietokannat ja saraketietokannat (Han ym. 2011). Näiden lisäksi on olemassa myös kaksi muuta tietokantatyyppiä, jotka ovat graafi- ja olio-orientoituneet tietokannat (Cattell 2011).

NoSQL-tietokannat voidaan skaalata vaakasuunnassa. Vaakasuuntaisen skaalautuvuuden pääominaisuus on datan ja kuormituksen jakaminen useammalle palvelimelle, jolloin tietokannan skaalautuvuus voidaan varmistaa tehokkaammin ja edullisemmin (Cattell 2011). Vaakasuuntaisen skaalautuvuuden mahdollistamiseksi NoSQL-tietokannoissa joudutaan kuitenkin luopumaan tiukoista tapahtumankäsittelyyn liittyvistä vaatimuksista, joita relaatiotietokantajärjestelmät noudattavat (Pokorny 2011).

NoSQL-tietokantajärjestelmissä priorisoidaan skaalautuvuutta ACID-periaatteiden noudattamisen sijaan. Tästä johtuen NoSQL-tietokantojen hallintajärjestelmät ovat lopulta johdonmukaisia, toisin kuin relaatiotietokantojen hallintajärjestelmät. Pritchett (2008) esittää vastakohdan ACID-periaatteelle, jonka lyhenne on BASE (*Basically Available, Soft state, Eventually consistent*). BASE-periaatteen mukaisesti muutokset etenevät järjestelmän läpi, kun on kulunut riittävästi aikaa (*Eventually Consistent*). Tästä syystä jotkin palvelinsolmut (*server nodes*) voivat hetkellisesti sisältää vanhentunutta tietoa (Grolinger ym. 2013). ACID-periaatteen mukaisen tietokannan eheyden (*Consistency*) luopumisen avulla NoSQL-tietokannat mahdollistavat järjestelmälle paremman suorituskyvyn sekä skaalautuvuuden. Tästä voi olla hyötyä esimerkiksi sosiaalisen median alustoilla tai sähköpostisovelluksissa, joissa data ei ole strukturoitua (Leavitt 2010).

Stonebraker (2010, 2012) kyseenalaistaa NoSQL:n kyvyn toteuttaa kasvava tarve OLTP-järjestelmille (*Online Transaction Processing*), joita esimerkiksi verkkopelit tai sosiaaliset verkostot käyttävät. Stonebraker (2010) kertoo, että OLTP-järjestelmät tarvitsevat ACID-

periaatteen mukaiset vaatimukset tietokannassa olevan datan johdonmukaisuuden varmistamiseksi. Pavlon ja Aslettin (2016) mukaan esimerkiksi Google on huomannut kehittäjien käyttävän liikaa aikaa epäjohdonmukaisen datan korjaamiseen, kun käytössä on NoSQL-tietokannan hallintajärjestelmä. Vaihtoehdoksi niille Stonebraker (2012) ehdottaa NewSQL-tietokantoja, sillä niiden avulla voidaan tarjota NoSQL-tietokantojen skaalautuvuus relaatiotietokannassa.

3 NewSQL-tietokannat

3.1 ACID-periaatteen toteutuminen

Pavlon ja Aslettin (2016) mukaan NewSQL-tietokantojen päätavoitteena on tarjota vaakasuuntainen skaalautuvuus NoSQL:n tapaan sekä säilyttää ACID-periaatteiden mukaiset takuut tietokannalle. Vaakasuuntaisen skaalautuvuuden avulla NewSQL-tietokantajärjestelmät pyrkivät vastaamaan tarpeeseen nykyaikaisesta relaatiotietokannasta, jonka avulla voidaan käsitellä suuria datamääriä. Lisäksi ACID-periaatteiden toteutuminen NewSQL-tietokannan hallintajärjestelmässä mahdollistaa samanaikaisten tapahtumien käsittelyn sekä tietokannan tilan muuttamisen SQL:n (*Structured Query Language*) avulla.

ACID-periaatteiden mukainen tietokannan atomisuus ja eristyneisyys voidaan taata NewSQL-tietokantajärjestelmissä transaktioiden samanaikaisuuksien hallinnan avulla. Melkein kaikki uusiin arkkitehtuureihin perustuvat NewSQL-tietokantaratkaisut hyödyntävät MVCC-protokollaa (*multi-version concurrency control*) samanaikaisten tapahtumien käsittelyssä. (Pavlo ja Aslett 2016.) MVCC:n mukaisesti tietokannan hallintajärjestelmä luo uuden version tietueesta (*tuple*) kun transaktio alkaa päivittämään sitä. Tietueen useiden versioiden ylläpitäminen mahdollistaa tapahtumien käsittelyn loppuun saakka, vaikka toinen tapahtuma yrittäisi päivittää samanaikaisesti samaa tietuetta. (Grolinger ym. 2013.)

ACID:n mukainen eheys ja pysyvyys mahdollistetaan puolestaan toisintamisen (*replication*) avulla. Toisintamisen perusidea on datan kopioiminen useammalle solmulle. Vahvan johdonmukaisuuden (*strong consistency*) saavuttamiseksi NewSQL-tietokantajärjestelmässä kirjoitusoperaatiot on ilmoitettava vastaanotetuiksi ja ne on asennettava kaikkiin versioihin. Vasta sen jälkeen tietokantajärjestelmä voi katsoa tapahtuman suoritetuksi eli pysyväksi. (Grolinger ym. 2013; Pavlo ja Aslett 2016.)

3.2 Luokittelu

Pavlo ja Aslett (2016) luokittelevat NewSQL-tietokannat kolmeen kategoriaan. Ensimmäinen niistä koostuu tietokannoista, jotka ovat täysin uusia ja rakennettu uutta arkkitehtuuria

käyttäen. Google Spanner, H-Store, VoltDB ja NuoDB ovat esimerkkejä näistä tietokantaratkaisuista. Nämä järjestelmät perustuvat hajautettuihin arkkitehtuureihin, jotka toimivat dynaamisesti skaalautuvilla ”shared-nothing”-solmuilla (Silva, Almeida ja Queiroz 2016) ja sisältävät useamman solmun yhtäaikaista hallintaa tukevia komponentteja. Kyseisten tietokantajärjestelmien etuna on se, että kaikki järjestelmän osat voidaan optimoida monisolmuympäristöön (*multi-node environment*). Tämän ansiosta useimmat ensimmäisen kategorian NewSQL-tietokantojen hallintajärjestelmistä pystyvät lähettämään kyselyitä suoraan solmujen välillä. (Pavlo ja Aslett 2016.)

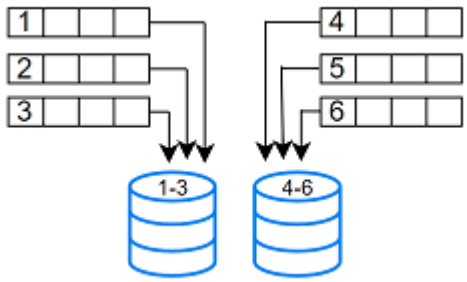
Toiseen kategoriaan kuuluvat tietokannat, jotka väliohjelmistokerroksen (*middleware layer*) avulla jakavat tietokannat useille eri solmuille automaattisesti. Tähän kategoriaan kuuluvat esimerkiksi ScaleBase, ScaleArc ja AgilData Scalable Cluster (aiemmin nimellä dbShards). Näiden järjestelmien etuna on yleensä se, että niillä on helppo korvata olemassa oleva tietokannan hallintajärjestelmä. Lisäksi kehittäjien ei tarvitse tehdä muutoksia sovellukseensa uuden tietokannan käyttöönottoa varten. (Pavlo ja Aslett 2016.)

Kolmas kategoria koostuu Pavlon ja Aslettin (2016) mukaan database-as-a-service-tuotteista (*DBaaS*), joita pilvipalveluita tarjoavat yritykset myyvät. DBaaS:ien avulla organisaation ei tarvitse ylläpitää tietokannan hallintajärjestelmää. Sen sijaan DBaaS-palveluntarjoaja on vastuussa tietokannan ylläpitoon liittyvistä toimenpiteistä. Pavlo ja Aslett painottavat, että vain ne DBaaS-tuotteet, jotka perustuvat uuteen arkkitehtuuriin, voidaan luokitella NewSQL-järjestelmiksi. Näitä tietokantojen hallintajärjestelmiä ovat Amazon Aurora sekä ClearDB.

4 NewSQL-tietokantojen ositusmenetelmät

NewSQL-tietokantajärjestelmät skaalautuvat niin, että tietokanta jaetaan epäyhtenäisiksi osajoukoiksi (Pavlo ja Aslett 2016). Näitä osajoukkoja kutsutaan joko osioiksi (*partitions*) tai sirpaleiksi (*shards*). Pavlon ja Aslettin mukaan tietokannan taulut on jaettu vaakasuunnassa useaan osaan, joiden rajat perustuvat yhden tai useamman taulun sarakkeen arvoihin. Tietokannan vaakasuuntainen osittaminen voidaan tehdä joko alueosituksen (*range partitioning*) tai hajautusosituksen (*hash partitioning*) avulla.

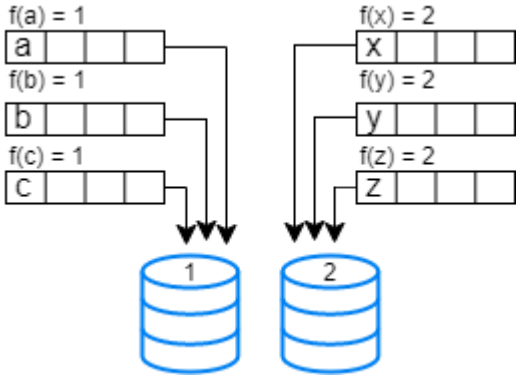
Alueosituksessa data jaetaan ositusavaimen (*partition key*) perusteella. Ositusavain valitaan taulun attribuuteista. Avaimen valinnan jälkeen tiedot jaetaan tietokantajärjestelmän solmujen kesken siten, että jokainen osio sisältää ne rivit, joilla on sama ositusavain. (Grolinger ym. 2013; Costa, Maia, Carlos ym. 2015.) Kuviossa 2 tauluun on tallennettu verkkokaupan ostoksia. Ostoksista tallennetaan ostoksen päivämäärä, asiakkaan tunnus ja nimi sekä ostoksen summa. Ositusavaimeksi valitaan päivämäärä, jota kuvastaa jokaisen ositetun rivin numero. Tammikuuta kuvaa numero 1, helmikuuta numero 2 ja niin edelleen. Näin taulu voidaan jakaa tietokannan palvelimien välille esimerkiksi kolmen kuukauden mittaisina jaksoina, joten ensimmäisellä palvelimella on tammikuusta maaliskuuhun ja toisella huhtikuusta kesäkuuhun sijoittuvien ostosten tiedot. NewSQL-tietokantajärjestelmistä esimerkiksi CockroachDB sekä Google Spanner käyttävät alueositusta.



Kuvio 2. Alueositus (*range partitioning*)

Hajautusosituksen tavoitteena sen sijaan on Costa, Maia, Carlos ym. (2015) mukaan jakaa tietokantaan tallennettu data mahdollisimman tasaisesti solmujen kesken. Hajausosituksessa, kuten alueosituksessakin, valitaan ositusavain, jonka perusteella tiedon jakaminen suo-

ritetaan. Ositus tehdään käyttämällä hajautusalgoritmia *f*, jolle annetaan parametrina valittu ositusavain. Kuvio 3 mallintaa hajautusositusta. Siinä valittu hajautusalgoritmi jakaa taulun rivit ositusavaimen perusteella kahdelle eri palvelimelle. NewSQL-tietokantojen hallintajärjestelmistä esimerkiksi CockroachDB käyttää hajautusositusta.

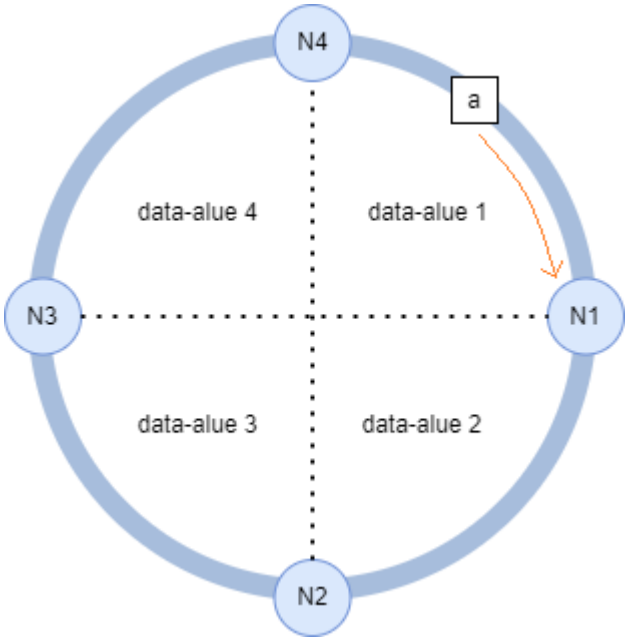


Kuvio 3. Hajautusositus (*hash partitioning)*

Pavlo ja Aslett (2016) painottavat, että kaikki muut luvussa 3 mainittujen kategorioiden tietokantaratkaisut voivat käyttää alueositusta sekä hajautusositusta, paitsi database-as-a-service-ratkaisut (*DBaaS*). Sekä Amazon Aurora (AmazonAurora 2023) että ClearDB (ClearDB 2023) käyttävät MySQL-moottoria, joka ei Pavlon ja Aslettin artikkelin kirjoittamisen aikaan tukenut tietokannan ositusta. Kuitenkin vuonna 2018 julkaistussa MySQL 8.0 -versiossa tähän on tullut muutos ja nykyään se tukee molempia aikaisemmin mainittuja ositustapoja (MySQL 2023). Vaikka MySQL on relaatiotietokantajärjestelmä, voi sitä käyttää NewSQL-tietokannan hallintajärjestelmässä, jos sen päälle lisätään väliohjelmistokerros. Toinen vaihtoehto MySQL:n hyödyntämiselle on käyttää sitä perusmoottorina ja rakentaa sitä hyödyntäen uuteen arkkitehtuuriin perustuva DBaaS-ratkaisu. (Pavlo ja Aslett 2016.)

Grolinger ym. (2013) mainitsevat myös kolmannen ositustavan, jota kutsutaan nimellä johdonmukainen hajautus (*consistent hashing*). Tämä hajautustekniikka käyttää hajautusfunktiota, jonka avulla data jaetaan valitun avaimen mukaisesti suoraan tietokantajärjestelmän solmuihin. Solmut ikään kuin muodostavat renkaan ja jokainen renkaalla oleva solmu vastaa tietyn alueen avaimista. Kuviossa 4 N1:stä N4:een ovat tietokantajärjestelmän solmuja ja jokainen solmu vastaa sitä edeltävästä data-alueesta. Näin ollen esimerkiksi solmu N1 vastaa ensimmäisestä data-alueesta. Järjestelmä käyttää hajautusfunktiota määrittelemään olion

a sijainnin. Hajautusfunktio määrää oliolle *a* paikan ja sen jälkeen rengasta kuljetaan myötäpäivään, kunnes ensimmäinen solmu tulee vastaan. Kuvio mukaisessa tilanteessa olio *a* määrätään lopulta solmuun N1. Grolingerin ym. (2013) mukaan johdonmukainen hajautus on nopea ja luotettava tapa datan jakamiseen. NewSQL-tietokannoista esimerkiksi VoltDB ja Clusterix käyttävät kyseistä hajautusta.



Kuvio 4. Johdonmukainen hajautus (*consistent hashing*)

5 Yhteenveto

Tämän kandidaatintutkielman aiheena on NewSQL-tietokantojen skaalautuvuus. Tutkimus suoritettiin kirjallisuuskartoituksena, jonka tarkoituksena oli selvittää, mitä NewSQL-tietokantojen skaalautuvuudesta on tutkittu. Tutkimuksessa selvitettiin onnistuneesti, miten NewSQL-tietokantajärjestelmät voidaan skaalata ilman, että tarvitsee luopua ACID-periaatteiden mukaisista takuista.

Tutkimuksessa todettiin, että tietokannan skaalautuvuudella tarkoitetaan tietokannan kykyä laajentua ilman suuria muutoksia arkkitehtuuriin sekä kykyä toimia syötetyn datamäärän suurentuessa. Tietokantaa voidaan skaalata joko pystysuunnassa tai vaakasuunnassa. NewSQL-tietokantajärjestelmissä vaakasuuntainen skaalautuminen toteutetaan siten, että tietokanta jaetaan osajoukkoihin alueosituksen, hajautusosituksen tai johdonmukaisen hajautuksen avulla. Lisäksi NewSQL-tietokantojen hallintajärjestelmät pyrkivät takaamaan ACID-periaatteiden mukaiset takuut transaktioille toisintamisen sekä tapahtumien samanaikaisuuden hallinnan avulla.

Tutkielman laatimisessa ensimmäiseksi ongelmaksi osoittautui NewSQL-tietokannoista tehtyjen tutkimusten vähäisyys. NewSQL-termiä käytettiin ensimmäisen kerran vasta vuonna 2011, joten aiheena se on suhteellisen tuore. Kuitenkin monet aihetta käsittelevät tutkimusartikkelit ovat maksullisia, joten niitä ei voitu ottaa huomioon tämän tutkielman tutkimuksessa. Suurimmassa osassa tutkimuksista, joita voitiin hyödyntää, NewSQL-tietokantoja verrataan NoSQL-tietokantoihin. Tutkimuksissa ei kuitenkaan verrata NewSQL-tietokantajärjestelmien luotettavuutta aikaisempiin tietokantojen hallintajärjestelmiin. Monessa tutkimuksessa myös viitataan yhteen NewSQL-termin kehittäjien kirjoittamaan artikkeliin, joten lähteissä ei ole variaatiota. Lisäksi NewSQL-tietokantojen skaalautuvuudesta kerrotaan useassa tutkimuksessa vain, että ne skaalautuvat vaakasuoraan, mutta skaalautuvuutta ei selitetä syvemmin.

Toiseksi ongelmaksi osoittautui relaatiotietokantojen tutkiminen. Relaatiomalli kehitettiin 1970-luvulla, minkä vuoksi niitä koskevissa tutkimuksissa on tulkinnanvaraisuuksia. Monessa tutkimuksessa puhutaan vain tietokannoista, sillä ennen NoSQL-tietokantojen yleistymis-

tä oli vain relaatiomalliin tehtyjä tietokantoja. Näin ollen relaatiotietokannoista oli haastavaa löytää ajankohtaisempia tutkimuksia, sillä valtaosa tietokantoihin liittyvästä viimeaikaisesta tutkimuksesta keskittyy uudempiin aihealueisiin.

Lähteet

AmazonAurora. 2023. “Oracle to Aurora MySQL Migration Playbook”. Viitattu 23. huhtikuuta 2023. https://docs.aws.amazon.com/dms/latest/oracle-to-aurora-mysql-migration-playbook/chap-oracle-aurora-mysql.html.

Bondi, André B. 2000. “Characteristics of scalability and their impact on performance”. Teoksessa *Proceedings of the 2nd international workshop on Software and performance,* 195–203.

Cattell, Rick. 2011. “Scalable SQL and NoSQL data stores”. *Acm Sigmod Record* 39 (4): 12–27.

ClearDB. 2023. Viitattu 23. huhtikuuta 2023. https://www.cleardb.com/.

Codd, Edgar F. 1970. “A relational model of data for large shared data banks”. *Communications of the ACM* 13 (6): 377–387.

———. 1979. “Extending the database relational model to capture more meaning”. *ACM Transactions on Database Systems (TODS)* 4 (4): 397–434.

Costa, Caio H, PHM Maia, F Carlos ym. 2015. “Sharding by hash partitioning”. Teoksessa *Proceedings of the 17th International Conference on Enterprise Information Systems,* 1:313–320.

Grolinger, Katarina, Wilson A Higashino, Abhinav Tiwari ja Miriam AM Capretz. 2013. “Data management in cloud environments: NoSQL and NewSQL data stores”. *Journal of Cloud Computing: advances, systems and applications* 2:1–24.

Han, Jing, Ee Haihong, Guan Le ja Jian Du. 2011. “Survey on NoSQL database”. Teoksessa *2011 6th international conference on pervasive computing and applications,* 363–366. IEEE.

Härder, Theo, ja Andreas Reuter. 1983. “Principles of transaction-oriented database recovery”. *ACM computing surveys (CSUR)* 15 (4): 287–317.

Leavitt, Neal. 2010. “Will NoSQL databases live up to their promise?” *Computer* 43 (2): 12–14.

MySQL. 2023. “MySQL 8.0 Reference Manual; 24.2 Partitioning Types”. Viitattu 23. huhtikuuta 2023. https://dev.mysql.com/doc/refman/8.0/en/partitioning-types.html.

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).

Pavlo, Andrew, ja Matthew Aslett. 2016. “What’s really new with NewSQL?” *ACM Sigmod Record* 45 (2): 45–55.

Pokorny, Jaroslav. 2011. “NoSQL databases: a step to database scalability in web environment”. Teoksessa *Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services,* 278–283.

Pritchett, Dan. 2008. “BASE: An Acid Alternative: In partitioned databases, trading some consistency for availability can lead to dramatic improvements in scalability.” *Queue* 6 (3): 48–55.

Silva, Yasin N, Isadora Almeida ja Michell Queiroz. 2016. “SQL: From traditional databases to big data”. Teoksessa *Proceedings of the 47th ACM Technical Symposium on Computing Science Education,* 413–418.

Stonebraker, Michael. 2010. “SQL databases v. NoSQL databases”. *Communications of the ACM* 53 (4): 10–11.

———. 2012. “New opportunities for new sql”. *Communications of the ACM* 55 (11): 10–11.

Stonebraker, Michael, Samuel Madden, Daniel J Abadi, Stavros Harizopoulos, Nabil Hachem ja Pat Helland. 2007. “The end of an architectural era: It’s time for a complete rewrite”. Teoksessa *Making Databases Work: the Pragmatic Wisdom of Michael Stonebraker,* 463–489.