hyväksymispäivä arvosana

arvostelija

Citrusleaf-tietokanta: Voiko NoSQL-tietokanta tarjota transaktioiden ACID-ominaisuudet?

Mikko Kangasaho

Helsinki 7.3.2012

Seminaarityö

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Matemaattis-luonnontieteellinen tiedekunta	Tietojenkäsittelytieteen laitos

Tekijä — Författare — Author
Mikko Kangasaho

Työn nimi — Arbetets titel — Title

Citrusleaf-tietokanta: Voiko NoSQL-tietokanta tarjota transaktioiden ACID-ominaisuudet?

Oppiaine — Läroämne — Subject
Tietojenkäsittelytiede

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Seminaarityö	7.3.2012	12 sivua

Tiivistelmä — Referat — Abstract

Suuret nykyaikaiset Web-sovellukset vaativat tietokannan suorituskyvyltä paljon. Pystysuuntainen skaalaaminen on kallista eikä se tarjoa aina riittävästi suorituskykyä. Vaakasuuntainen skaalaaminen on halvempaa ja mahdollistaa paremman suorituskyvyn. Relaatiotietokannat eivät skaalaudu vakasuunnassa hyvin, sillä ACID-ominaisuuksien toteuttaminen hajautetuissa järjestelmissä on haastavaa. CAP-teoreeman mukaan hajautetut tietokantojen täytyy valita oikeellisuudesta, saatavuudesta ja osioiden sietokyvystä kaksi ominaisuutta. Hajautetuissa tietokannoissa osioiden sietokyky on pakko valita, joten todellinen valinta tehdään saatavuuden ja oikeellisuuden välillä. Relaatiotietokannat valitsevat oikeellisuuden, ja NoSQL-tietokannat valitsevat yleensä saatavuuden. NoSQL-tietokantojen suorituskyky on parempi, sillä ne höllentävät ACID-vaatimuksia ja tarjoavat niiden asemesta BASE-ominaisuuksia. Citrusleaf-tietokannan väitetään olevan NoSQL-tietokanta joka tukee ACID-ominaisuuksia. Citrusleaf-tietokannasta tehtyjä väitteitä tarkastelemalla havaitaan, että sen väite ACID-ominaisuuksien tukemisesta on voimassa vain rajoitetummassa mielessä kuin mitä tietokannan kehittäjät antavat ymmärtää. Citrusleaf-tietokanta on kuitenkin mielenkiintoinen tietokanta, joka antaa sovelluskehittäjän valita haluaako hän tietokannan olevan oikeellinen vai saatavilla.

ACM Computing Classification System (CCS):
H2.4 [Information Systems—Distributed Databases]
H2.4 [Information Systems—Transaction Processing]
C2.4 Computer Systems Organization—Distributed Databases]

Avainsanat — Nyckelord — Keywords
NoSQL, BASE, ACID, relaatiotietokanta, Citrusleaf

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Sisältö

1 Johdanto

Nykyaikaiset sovellukset saattavat käsitellä valtavia määriä dataa ja tietokantaoperaatioita. Esimerkiksi Facebook, Twitter ja Google ovat globaaleja palveluita joiden tietokantojen täytyy kestää suurta kuormaa. Tietokantoja on aikaisemmin skaalattu pystysuunnassa, mutta *pystysuuntainen skaalautuminen* on kallista [Pok11] eikä sillä päästä yhtä hyvään suorituskykyyn kuin *vaakasuuntaisella skaalaamisella* [Pri08].

Vaakasuuntainen skaalaaminen tekee ACID-ominaisuuksien ylläpidosta vaikeaa. *NoSQL-tietokannat* ovat luopuneet ACID-ominaisuuksista, jotta ne skaalautuisivat paremmin vaakasuunnassa [Lea10, Pok11]. NoSQL-tietokantojen ACID-ominaisuuksia lievempiä oikeellisuusvaatimuksia kutsutaan BASE-ominaisuuksiksi [Pri08]. BASE-ominaisuudet määrittävät tietokannan olevan *lopulta eheä* (eventually consistent), eli että jollekin tietokannan solmulle tehdyt operaatiot replikoituvat muille solmuille asynkronisesti.

Brewerin [Bre00] CAP-teoreema määrää, että hajautettu järjestelmä voi valita kaksi seuraavista kolmesta ominaisuudesta: *saatavuus*, *eheys* ja *osituksen sieto* [GL02]. Ei-hajautetuissa järjestelmissä ei tapahdu solmujen ositusta erillisiin osioihin, joten niissä voidaan valita saatavuus ja eheys. Hajautetuissa järjestelmissä solmujen jakautuminen erillisiin osioihin on mahdollista, joten järjestelmän on oltava joko saatava tai eheä [Vog09].

ACID-ominaisuudet vaativat järjestelmältä CAP-ominaisuuksista eheyttä. Monet NoSQL-tietokannat valitsevat CAP-ominaisuuksista saatavuuden ja osituksen siedon, joten ne eivät voi täyttää ACID-ominaisuuksia. Yleensä NoSQL-tietokantojen etuna pidetään juuri sitä, että ne tarjoavat hyvän suorituskyvyn tinkimällä ACID-ominaisuuksista [Cat11].

Srinivasan ja Bulkowksi [SB11] kertovat Citrusleaf-tietokannan olevan hajautettu NoSQL-tietokanta, joka tarjoaa ACID-ominaisuudet. Citrusleaf-tietokantaa mainostetaan myös välittömästi eheänä, mutta samalla erittäin suorituskykyisenä tietokantana. Srinivasan ja Bukowskin mukaan Citrusleaf-tietokanta yhdistää relaatiotietokantojen eheysominaisuudet NoSQL-tietokantojen vaakasuuntaisen skaalautuvuuden kanssa.

Luku 2 esittelee lyhyesti NoSQL-tietokannoille esiteltyjä määritelmiä. Luvussa 3 käsitellään tietokannoilta vaadittavia ACID- ja BASE-ominaisuuksia, sekä niiden suhdetta CAP-teoreemaan. Luku 4 esittelee Citrusleaf-tietokantaa ja arvioi sen tarjoamia ACID-ominaisuuksia. Citrusleaf-tietokannasta ei ole saatavilla muita lähteitä

kuin Srinivasan ja Bulkowskin artikkeli [SB11], joten sitä tarkastellaan ainoastaan tätä artikkelia vasten. Srinivasan on Citrusleaf-tietokantaa kehittävän Citrusleaf Inc. -yrityksen teknologiajohtaja ja Bulkowski sen toimitusjohtaja [Cit12]. Luvussa 5 on yhteenveto seminaarityöstä ja arviointia seminaarityön rajoituksista.

2 NoSQL-tietokannat

NoSQL-tietokanta on laaja termi, jolla tarkoitetaan tietokantoja, jotka eivät noudata perinteistä relaatiomallia [Pok11, Lea10]. Useimmat NoSQL-tietokannat eivät käytä SQL-kyselykieltä, mutta on olemassa myös NoSQL-tietokantoja joissa kyselyitä voi tehdä SQL-kielellä. Tästä syystä akronyymin NoSQL on määritetty tarkoittavan "Not Only SQL" [Cat11, Ore10, Pok11], eli "ei vain SQL:ää".

Leavitt [Lea10] määrittelee NoSQL-tietokannat relaatiotietokantojen negaatioksi. Hänen mukaansa erityyppisiä NoSQL tietokantoja on paljon, mutta niitä yhdistää se että ne eivät ole relationaalisia. Leavitt mainitsee NoSQL-tietokantojen tärkeimmäksi eduksi sen, että ne pystyvät käsittelemään hyvin rakenteetonta tietoa.

Cattellin [Cat11] mukaan NoSQL-tietokannoilla on tyypillisesti kuusi keskeistä piirrettä:

1. Vaakasuuntainen skaalautuvuus (horizontal scaling)
2. Tiedon tallentaminen hajautetusti eri solmuille
3. Yksinkertainen komentorajapinta jonka avulla tietokantaan voidaan liittyä
4. ACID-ominaisuuksia heikompi transaktioiden hallinta
5. Hajautetut hakemistot ja muistin käyttö tiedon tallentamiseen
6. Mahdollisuus lisätä uusia attribuutteja dataan dynaamisesti

Orend [Ore10] pitää NoSQL-tietokantojen tunnuspiirteenä sitä, että ne eivät tue ACID-ominaisuuksia. Lisäksi hän huomauttaa että NoSQL-tietokannat yleensä ovat hajautettuja tietokantoja, kaavattomia (schema-free) ja niillä on oma kyselykielensä joka ei ole SQL.

Monet NoSQL-tietokannat eivät tue liitos-operaatioita (join) ja tietueita pystyy hakemaan ainoastaan niiden pääavaimen perusteella. Tarvittavat liitosoperaatiot täytyy tehdä sovelluskerroksessa [Pok11].

NoSQL-tietokannat ovat yleensä suunniteltu siten, että ne skaalautuvat vaakasuunnassa. Tällä tarkoitetaan sitä, että järjestelmän suorituskykyä on mahdollista parantaa lisäämällä solmujen määrää. Tällöin sekä data että järjestelmään kohdistuva kuorma jakautuvat usealle solmulle. Lisäksi vaaditaan, että nämä solmut eivät saa jakaa keskenään muistia tai kovalevytilaa (shared nothing architecture) [Lea10, Cat11]. Pystysuuntaisella skaalautuvuudella (vertical scaling) tarkoitetaan järjestelmän suorituskyvyn parantamista yksittäisen solmun suoritustehon nostamisella. Relaatiotietokannat skaalautuvat hyvin pystysuunnassa, mutta huonosti vaakasuunnassa [Cat11].

NoSQL-tietokannat ovat yleensä suorituskyvyltään parempia kuin relaatiotietokannat. Yksi syy niiden paremmalle suorituskyvylle on se, etteivät ne tyypillisesti noudata transaktioiden ACID-periaatteita [Lea10]. Orendtin [Ore10] mukaan ACID-ominaisuuksien puute mahdollistaa NoSQL-tietokantojen vaakasuuntaisen skaalautuvuuden.

Kolme yleistä NoSQL-tietokantatyyppiä ovat avain-arvo -tietokannat (key-value stores), saraketietokannat (column-oriented databases) ja dokumenttitietokannat (document-based stores). Tunnettuja esimerkkejä edellä mainituista ovat Amazonin kehittämä SimpleDB (avain-arvo -tietokanta), Facebookin kehittämä Cassandra (saraketietokanta) ja avoimen lähdekoodin MongoDB (dokumenttitietokanta) [Lea10]. Tässä seminaarityössä esiteltävä Citrusleaf-tietokanta on avain-arvo -tietokanta [SB11].

3 ACID, CAP-teoreema ja BASE

ACID-ominaisuudet ovat transaktioille asetettavia vaatimuksia. Akronyymi ACID tulee sanoista *atomisuus* (atomicity), *oikeellisuus* (consistency), *eristyvyys* (isolation), ja *pysyvyys* (durability). ACID-ominaisuudet tarjoavan tietokannan transaktioilta vaaditaan seuraavaa [HR83]:

1. Atomisuus: Transaktion sitouduttua kaikki sen operaatiot jäävät voimaan tai mikään niistä ei jää voimaan.

2. Oikeellisuus: Transaktio säilyttää sitouduttuaan tietokannan eheän tilan.

3. Eristyvyys: Transaktion sisällä tapahtuvat operaatiot eivät näy muille transaktioille. Eristyvyys vaaditaan jotta eristyvyysanomalioita (likainen luku, likainen kirjoitus, toistokelvoton luku) ei tapahtuisi.

4. Pysyvyys: Sitoutuneen transaktion tekemät muutokset jäävät tietokantaan pysyvästi kunnes joku toinen transaktio poistaa ne.

CAP-teoreeman mukaan hajautettu tietokanta voi tarjota kaksi seuraavista kolmesta ominaisuudesta [GL02]:

1. Oikeellisuus (consistency): Hajautettuun tietokantaan tehtyjen operaatioiden on näytettävä siltä, kuin ne tehtäisiin yhdestä solmusta koostuvaan tietokantaan. Toisin sanottuna kirjoitusoperaation on oltava luettavissa tietokannasta sen onnistuttua.

2. Saatavuus (availability): Jos toiminnassa oleva solmu saa palvelupyynnön, sen on vastattava siihen. Tämä määritelmä sallii solmujen kaatumisen, sillä kaatunut solmu ei ole enää toiminnassa.

3. Osituksen sietokyky (partition tolerance): Jos tietokantarypäs jakautuu verkon toimimattomuuden takia useampaan osioon, täytyy järjestelmän silti toimia oikein. Tietokantaan tulisi pystyä kirjoittamaan ja siitä tulisi voida lukea vaikka solmut olisivat osittuneet.

On syytä huomata, että ACID-ominaisuuksien oikeellisuus ja CAP-ominaisuuksien oikeellisuus ovat eri asioita. ACID-ominaisuuksien oikeellisuudella tarkoitetaan tietokannan eheysrajoitteiden täyttymistä transaktion sitouduttua [HR83], kun taas CAP-teoreeman oikeellisuudella tarkoitetaan sitä, että transaktion sitouduttua sen tekemät operaatiot näkyvät kaikille sitä seuraaville transaktioille [Vog09].

Osituksen sietokykyä ei tarvita jos tietokanta ei ole hajautettu. Tällöin järjestelmä pystyy takaamaan saatavuuden ja oikeellisuuden. Saatavuus on taattu, sillä järjestelmä koostuu yhdestä solmusta, ja mikäli solmu on toiminnassa se vastaa palvelupyyntöön. Oikeellisuus pystytään takaamaan transaktioprotokollilla [Vog09]. Suuret nykyaikaiset Web-sovellukset vaativat vaakasuuntaista skaalautuvuutta. Hajautetuissa järjestelmissä on aina mahdollista syntyä osioita, joten CAP-teoreema pakottaa sovelluskehittäjät valitsemaan saatavuuden ja oikeellisuuden välillä [Pri08].

ACID-ominaisuudet tarjoavan tietokannan täytyy valita CAP-teoreeman ominaisuuksista oikeellisuus [Pri08]. Atomisuus, eristyvyys ja pysyvyys takaavat, että transaktion sitouduttua oikea data on muiden transaktioiden luettavissa. Pokorny [Pok11] pitää ACID-ominaisuuksia ja CAP-teoreeman oikeellisuutta toisiaan vastaavina vaatimuksina. ACID-ominaisuudet takaavan tietokannan on oltava välittömästi eheä

(immediately consistent), eli yhdelle solmulle tehty operaatio on replikoitava muille solmuille synkronisesti. Muutoin olisi mahdollista, että yhdelle solmulle kirjoittaneen transaktion sitouduttua toinen transaktio lukisi toiselta solmulta vanhentunutta dataa. Oikeellisuus ja välitön eheys tarkoittavat, että tietokantaryppään osittuessa solmut ovat toiminnassa, mutta eivät pysty vastaamaan palvelupyyntöön ennen kuin tietokantaryppään osittuminen päättyy ja data saadaan replikoitua muille solmuille [Vog09]. Oletetaan että esimerkkitietokanta koostuu toiminnassa olevista solmuista N1 ja N2. Oletetaan lisäksi järjestelmän olevan osittunut, eli N1:n ja N2:n välillä ei ole mahdollista siirtää dataa. Jos järjestelmältä vaaditaan oikeellisuutta, niin N1:lle tehty kirjoitus on replikoitava synkronisesti N2:lle. Koska N1 ja N2 välillä ei ole yhteyttä, joudutaan odottamaan sen palaamista. Tietokanta ei näin ollen voi toteuttaa kirjoitusoperaatiota, joten se ei ole saatavilla.

Monissa web-sovelluksissa saatavuus on kriittistä. Jos järjestelmän pitää olla saatavilla solmujen osittuessa, täytyy oikeellisuusvaatimuksesta luopua. Käytännössä täytyy luopua välittömästä eheydestä, sillä solmulle tehtäviä muutoksia ei voida replikoida synkronisesti muille solmuille. Tällöin toiminnassa olevat solmut voivat vastata niille tulleisiin palvelupyyntöihin, mutta ei voida taata että kirjoitusoperaation jälkeinen samaan alkioon kohdistuva luku ei palauttaisi juuri kirjoitettua vanhempaa versiota datasta [Vog09]. Jos oletetaan toiminnassa olevat solmut N1 ja N2 saatavassa hajautetussa tietokannassa, niin N1:n kohdistuva kirjoitus replikoituu tietyn aikaikkunan sisällä N2:lle. Tällöin jos N2:lle tehdään luku ennen kuin N1:lle tehty muutos on päätynyt N2:lle, niin N2 palauttaa vanhentunutta dataa. Tällöin ACID-ominaisuudet ole enää mahdollisia, sillä ACID-ominaisuudet vaativat, että transaktion sitouduttua sen tekemien muutosten täytyisi näkyä muille transaktioille. Relaatiotietokannat tarjoavat yleensä ACID-ominaisuudet ja siten CAP-teoreeman oikeellisuuden, mutta eivät saatavuutta.

NoSQL-tietokantojen ACID-ominaisuuksia heikompia eheysvaatimuksia kutsutaan BASE-ominaisuuksiksi (basically available, soft state, eventually consistent) [Cat11]. Lopulta eheä -vaatimus tarkoittaa, että solmujen arvot päivittyvät epäsynkronisesti. Ehkä tunnetuin lopulta eheä -järjestelmä on Internetin nimipalvelujärjestelmä DNS [Vog09]. Pehmeällä tilalla tarkoitetaan sitä, että järjestelmän data ei ole aina oikeellista [Pok11]. Järjestelmä takaa, että jos alkioon ei kohdistu jatkuvasti uusia päivityksiä, niin lopulta kaikilla solmuilla on sama versio datasta [Vog09]. Tietyn aikaikkunan sisällä on mahdollista, että tietokanta palauttaa vanhentunutta dataa. Järjestelmän kaikki toiminnassa olevat solmut voivat kuitenkin vastata palvelupyyntöihin, joten järjestelmä on saatavilla. BASE-ominaisuudet toteuttavat

järjestelmät ovat yksinkertaisempia toteuttaa kuin ACID-ominaisuudet täyttävät, ja ne mahdollistavat paremman suorituskyvyn ja saatavuuden hajautetuissa järjestelmissä [FGC+97].

CAP-teoreeman aiheuttama valinta tietokannan ominaisuuksien välillä pakottaa sovelluskehittäjän ottamaan järjestelmän rajoitukset huomioon. Oikeellisuuden valitsevassa järjestelmässä sovelluksen kehittäjän on käsiteltävä tilanne, jossa esimerkiksi kirjoitusoperaatio ei onnistu ryppään osittumisen vuoksi. Saatavuuden valitsevassa järjestelmässä kehittäjän täytyy ottaa huomioon, että sovelluksesta luettava data voi olla vanhentunutta [Vog09]. Valinta CAP-teoreeman ominaisuuksien välillä täytyy tehdä sovelluksen ongelmakentän mukaan. Liiketoimintavaatimukset ratkaisevat onko sovellukselle tärkeämpää saatavuus vai oikeellisuus. Esimerkiksi pankkisovellukselle oikeellisuus on tärkeämpää, mutta vaikkapa internetfoorumilla ei välttämättä haittaa vaikka käyttäjä joskus lukisi vanhentunutta dataa.

4 Citrusleaf

Citrusleaf [Cit12] on hajautettu ACID-ominaisuudet takaava avain-arvo NoSQL-tietokanta [SB11]. Srinivasan ja Bulkowski [SB11] kertovat Citrusleaf-tietokannan erottuvan muista NoSQL-tietokannoista siten, että se takaa ACID-ominaisuudet ja täyttää CAP-teoreeman oikeellisuusominaisuuden, mutta on samalla erittäin tehokas ja skaalautuva. He mainitsevat lisäksi Citrusleaf-tietokannan tarjoavan korkeaa saatavuutta (high availability). Srinivasan ja Bulkowski jättävät termin kuitenkin määrittelemättä, mutta tulkintani mukaan he eivät tarkoita sillä samaa kuin CAP-teoreeman saatavuudella tarkoitetaan. Toisin kuin monet muut NoSQL-tietokannat, Citrusleaf-tietokantaa ei ole julkaistu avoimen lähdekoodin lisenssillä [Cit12].

4.1 Arkkitehtuuri

Korkealla tasolla Citrusleaf-tietokannassa on palvelinrypäs ja asiakaskerros (client layer). Asiakas sijaitsee tyypillisesti samalla solmulla kuin varsinainen tietokantaa käyttävä sovellus [SB11].

4.1.1 Tietokantarypäs

Citrusleaf-tietokannan tietokantarypäs koostuu hajautetuista solmuista, jotka eivät jaa keskenään muistia tai kovalevytilaa. Solmut kommunikoivat TCP/IP -protokollan yli. Solmut ilmoittavat itsestään väliajoin lähetettävällä signaalilla, jonka muut solmut ottavat vastaan. Jos jokin solmu havaitsee itselleen uuden solmun, tai ei vastaanota itselleen tutun solmun lähettämää signaalia, se aloittaa Paxos-algoritmiin [Lam01] perustuvan päättelyn muiden solmujen kanssa. Paxos-algoritmin lopputuloksena solmut tietävät mitä muita solmuja ryppääseen kuuluu. Signaalien lisäksi solmut käyttävät tietokantaoperaatioiden toteuttamisen yhteydessä saamiaan vastauksia muilta solmuilta päätelläkseen mitkä solmut ovat toiminnassa [SB11].

4.1.2 Asiakaskerros

Asiakaskerros tarjoaa sovellukselle rajapinnan tietokantaan. Tämän lisäksi se huolehtii tietokantaryppään asetuksista ja transaktioista tietokantaryppään ja sovelluksen välillä. Kuormantasaus (load balancing) hoidetaan asiakaskerroksessa, eikä palvelinpäässä. Kun sovellus pyytää asiakaskerrosta tekemään jonkin tietokantakyselyn, niin asiakaskerros lähettää kyselyn solmulle, jonka se luulee sisältävän haetun datan. Asiakaskerros tallentaa tiedon solmusta, joka on viimeksi sisältänyt sen haluaman datan [SB11].

4.2 Tietomalli

Citrusleaf-tietokannan data voidaan jakaa *nimiavaruuksiin* (namespaces). Nimiavaruudet ovat lähes kuin relaatiotietokantajärjestelmien tietokannat. Nimiavaruuksien sisällä on *joukkoja* (sets). Joukkojen vastine relaatiotietokannoissa on relaatio eli taulu. Joukkojen sisällä on myös *tietueita* (records). Tietueiden vastine relaatiotietokannassa on rivi. Jokaisella tietueella on avain ja yksi tai useampia nimettyjä *säiliöitä* (bins). Avaimen tulee olla uniikki joukon sisällä. Säiliöt muistuttavat relaatiotietokannan attribuutteja. Säiliöt sisältävät *arvoja*. Arvot ovat vahvasti tyypitettyjä. Vaikka arvot ovat tyypitettyjä, niin säiliöt eivät ole. Kahden eri tietueen saman säiliön arvot voivat olla eri tyyppisiä [SB11].

Citrusleaf-tietokanta voi käyttää fyysisenä tallennusmediana muistia (DRAM), kovalevyä tai SSD-levyä. Tallennusmedia voidaan valita nimiavaruuskohtaisesti, jolloin esimerkiksi usein käsiteltävän pienen nimiavaruuden voi säilyttää muistissa. Avaimis-

ta luodaan hakemisto (index) nopeaa hakua varten. Hakemistot tallennetaan muistiin. Koska avaimet voivat minkä kokoisia tahansa, tallennetaan niistä hakemistoon vain rajatun mittainen tiiviste [SB11].

Citrusleaf-tietokanta muistuttaa tietomalliltaan relaatiotietokantaa, mutta eroaa siitä siten että Citrusleaf-tietokannalla ei ole kaavaa (schema). Joukkoja ja säiliöitä ei tarvitse määritellä etukäteen, vaan niitä voi lisätä ja poistaa dynaamisesti [SB11].

4.3 ACID

Citrusleaf-tietokanta takaa, että jokainen yhtä tietuetta käsittelevä operaatio on atominen. Lisäksi kun yhtä tietuetta koskeva kirjoitusoperaatio on tehty, taataan että kaikki seuraavat lukuoperaatiot saavat ajan tasalla olevaa dataa. Montaa tietuetta käsittelevät transaktiot toteutetaan hakemalla joko koko joukon data tai osa siitä, ja iteroimalla tietueet läpi. [SB11]. Artikkelissa jäi epäselväksi miten useaa tietuetta käsittelevät transaktiot toimivat, ja voidaanko taata että useaa tietuetta koskevat operaatiot ovat atomisia.

Srinivasan ja Bulkowski [SB11] eivät käsittele kovin selvästi ACID-ominaisuuksien oikeellisuutta. Citrusleaf-tietokannan tietomallissa ei ole samanlaisia eheysvaatimuksia kuin relaatiotietokannoissa, joten ACID-ominaisuuksien eheysvaatimuksen täyttäminen ei ole vaikeaa. Tietueiden avainten täytyy olla uniikkeja joukon sisällä, mutta artikkelissa ei kerrottu miten tätä vaatimusta valvotaan ja kenen vastuulla sen säilyttäminen on.

Citrusleaf-tietokanta käyttää lyhytkestoisia lukulukkoja ja salpausmekanismia taatakseen transaktioiden eristyvyyden. Samaan tietoalkioon kohdistuvat luku- ja kirjoitusoperaatiot sarjallistetaan, mutta niiden järjestys on määrittelemätön. Moniin tietueisiin operoivien transaktioiden eristyvyyden takaamiseksi käytetään optimistista rinnakkaisuuden hallintaa (optimistic concurrency control). Tällöin transaktiot tekevät muutoksia olettaen, että toiset transaktiot eivät muuta dataa. Ennen transaktion sitoutumista tarkistetaan onko mikään muu transaktio muokannut samoja tietoalkioita [SB11].

Pysyvyys taataan replikoimalla data usealle solmulle. Jokainen päivitys kirjoitetaan usealle solmulle ennen kuin asiakaskerrokseen palautetaan tieto operaation onnistumisesta. Data kirjoitetaan tyypillisesti sekä muistiin että levylle [SB11]. Srinivasan ja Bulkowskin artikkeli ei kerro miten data viedään muistista levylle.

Srinivasan ja Bulkowskin [SB11] artikkelin perusteella vaikuttaa siltä, että Citrusleaf-

tietokanta takaa ACID-ominaisuudet vain yhtä tietuetta käsitteleville operaatioille. Jos transaktio käsittelee useaa tietuetta, niin tietokanta ei takaa enää atomisuutta eikä eristyvyyttä. Tällöin sovelluskehittäjän on itse toteutettava sovellustasolla haluamansa ACID-ominaisuudet, sillä tietokanta voi taata ACID-ominaisuudet vain yhtä tietuetta käsitteleville operaatioille.

4.4 CAP-teoreema

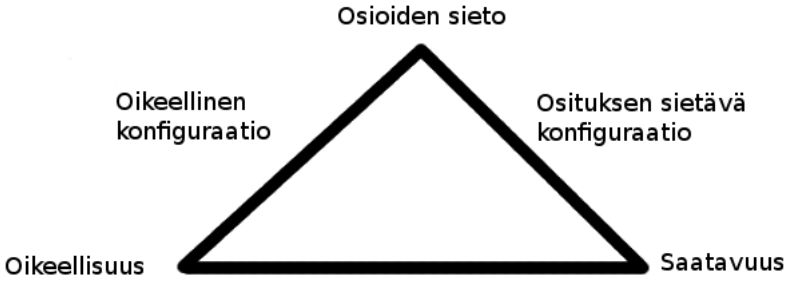
Citrusleaf-tietokannassa yhtä tietuetta käsittelevät operaatiot ovat oikeellisia, sillä ne replikoidaan muille solmuille synkronisesti. Useampaa tietuetta käsittelevät transaktiot toteutetaan sarjana yhtä avainta käsitteleviä operaatioita. Useaa tietuetta käsittelevät transaktiot tarvitsevat lukulukon tietokantaan, jolloin ne lukevat tarvitsemansa datan [SB11].

Citrusleaf-tietokannan voi konfiguroida kahteen tilaan: ositusta sietävään tai oikeelliseen. Osituksen sietävällä konfiguraatiolla osiot jatkavat toimintaansa tietokantaryppään osittuessa, vaikka ne eivät voi kommunikoida keskenään. Tällöin on mahdollista, että joku osio ei sisällä kaikkea dataa mitä tietokannassa on. Tällöin tietokanta voi vastata dataa lukevalle sovellukselle ettei dataa löytynyt. Kaikki osiot voivat myöskin ottaa vastaan kirjoitusoperaatioita ja on mahdollista, että kirjoituksen jälkeen tehty lukuoperaatio ei palauta uusinta dataa. Jos useampi osio saa samaan alkioon kohdistuvia kirjoituksia, niin osituksen päättyessä Citrusleaf-tietokanta yrittää joko ratkaista konfliktin itse aikaleimojen avulla tai säilyttää molemmat versiot alkiosta. Jos molemmat versiot säilytetään, niin alkioon kohdistuvat lukuoperaatiot palauttavat molemmat versiot ja sovelluksen on ratkaistava konflikti [SB11].

Oikeellisella konfiguraatiolla tietokantaryppään osittuessa vähemmän päätösvaltainen (minority quorum) osio lopettaa toimintansa. Kukaan ei saa vanhaa dataa, mutta tietokannan saatavuus on huono [SB11]. Artikkelista ei selviä miten päätellään mikä on vähemmän päätösvaltainen osio. Lisäksi artikkeli vaikuttaa olettavan, että rypäs ei voi osittua useampaan kuin kahteen osaan.

Ositusta sietävä konfiguraatio ei selvästikään takaa ACID-ominaisuuksia. ACID-ominaisuudet eivät voi täyttyä, mikäli on mahdollista että sitoutuneen transaktion muutokset eivät ole sitä seuraavien transaktioiden saatavilla. Citrusleaf-tietokanta takaa ACID-ominaisuudet vain jos se ei takaa CAP-teoreeman osituksen sietokykyä. CAP-teoreema pakottaa tekemään valinnan saatavuuden, oikeellisuuden ja osituksen sietokyvyn välillä. Citrusleaf-tietokanta tarjoaa sovelluskehittäjälle mahdolli-

suuden valita joko saatavuuden tai oikeellisuuden. Kuvassa 1 on kuvattu Citrusleaf-tietokannan tarjoamat konfiguraatiovaihtoehdot suhteessa CAP-teoreeman ominaisuuksiin. Osituksen sietävä konfiguraatio ei tarjoa oikeellisuutta, ja oikeellinen konfiguraatio ei tarjoa saatavuutta.



Kuva 1: Citrusleaf-tietokannan tarjoamat CAP-teoreeman ominaisuudet eri konfiguraatioilla.

5 Yhteenveto

Relaatiotietokannat tarjoavat transaktioille ACID-ominaisuudet, mutta eivät skaalaudu hyvin vaakasuunnassa. NoSQL-tietokannat tyypillisesti skaalautuvat hyvin, mutta joutuvat luopumaan ACID-ominaisuuksista skaalautuvuuden saavuttamiseksi. NoSQL-tietokantojen tarjoamaa lievennettyä oikeellisuusmallia kutsutaan BASE-ominaisuuksiksi. BASE-ominaisuudet takaavat, että järjestelmään tehdyt operaatiot näkyvät ennen pitkää kaikilla tietokantaryppään solmuilla. BASE-ominaisuudet takaavassa tietokannassa on mahdollista, että jokin operaatio kirjoittaa johonkin tietokannan alkioon, mutta myöhemmin samaan alkioon tehtävässä lukuoperaatiossa palautetaan vanhentunutta dataa.

CAP-teoreeman mukaan järjestelmän täytyy valita seuraavista kolmesta ominaisuudesta kaksi: oikeellisuus, saatavuus ja osioiden sietäminen. ACID-ominaisuudet sopivat yhteen oikeellisuuden ja saatavuuden kanssa, mutta eivät osioiden sietämisen.

BASE-ominaisuudet sopivat yhteen saatavuuden ja osioiden sietämisen kanssa, mutta eivät oikeellisuuden kanssa. Relaatiotietokannat yleensä täyttävät oikeellisuuden ja saatavuuden. NoSQL-tietokannat valitsevat tyypillisesti saatavuuden ja osioiden sietämisen, vaikkakin jotkut NoSQL-tietokannat valitsevat osioiden sietämisen ja oikeellisuuden. On syytä huomata, että hajautetussa tietokannassa ei käytännössä voi välttää osioiden syntymistä, joten hajautettujen tietokantojen on pakko valita osioiden sietäminen yhdeksi CAP-teoreeman ominaisuuksista.

Citrusleaf-tietokannan väitetään olevan skaalautuva hajautettu NoSQL-tietokanta, joka takaa transaktioiden ACID-ominaisuudet. Srinivasan ja Bulkowskin [SB11] artikkelista käy ilmi, että Citrusleaf-tietokanta tarjoaa ACID-ominaisuudet vain tiettyjen ehtojen täyttyessä: tietokantarypäs ei saa osittua ja transaktiot saavat tehdä operaatioita vain yhteen tietueeseen. Tämä on paljon rajoitetumpi tuki ACID-ominaisuuksille kuin Srinivasan ja Bulkowskin artikkelin otsikko ja tiivistelmä antavat ymmärtää.

Koska hajautetussa järjestelmässä tietokantarypäs voi osittua, Citrusleaf-tietokanta tarjoaa käyttäjälle mahdollisuuden valita oikeellisuuden ja saatavuuden välillä. Jos valitaan saatavuus, niin ACID-ominaisuudet eivät enää voi täyttyä. Jos käyttäjä valitsee osituksen sietokyvyn, niin tietokanta ei voi enää olla saatavilla. Valinnan tarjoaminen käyttäjälle on kiinnostava ominaisuus, sillä sama tietokantajärjestelmä voi näin ollen sopia hyvin eri tyylisiin käyttötapauksiin. Jos tietokantarypäs osittuu hyvin harvoin, niin oikeellinen konfiguraatio voi olla liiketoiminnallisesti järkevä valinta. Toisaalta mitä enemmän tietokantaryppäässä on solmuja, niin sitä suuremmaksi osittumisen todennäköisyys kasvaa.

Citrusleaf-tietokannasta ei ole julkaistu kovin paljoa tutkimusta. Tämä seminaarityö käsitteli Citrusleaf-tietokantaa pelkästään Srinivasan ja Bulkowskin [SB11] artikkelin pohjalta. Artikkeli jätti joitain osia tietokannan toiminnasta avoimeksi. Epäselväksi jäi miten useaa tietuetta käsittelevät transaktiot toimivat. Toivottavasti aiheesta julkaistaan lisää artikkeleja, jotta sovelluskehittäjillä olisi parempi mahdollisuus valita tarpeisiinsa sopiva tietokantaratkaisu.

Lähteet

Bre00 Brewer, E., Towards robust distributed systems. *Proceedings of the Annual ACM Symposium on Principles of Distributed Computing*, osa 19, 2000, sivut 7–10.

Cat11 Cattell, R., Scalable sql and nosql data stores. *ACM SIGMOD Record*, 39,4(2011), sivut 12–27.

Cit12 Citrusleaf Inc., Yrityksen www-sivut, Helmikuu 2012. `http://citrusleaf.com/`.

FGC+97 Fox, A., Gribble, S., Chawathe, Y., Brewer, E. ja Gauthier, P., Cluster-based scalable network services. *ACM SIGOPS Operating Systems Review*, osa 31. ACM, 1997, sivut 78–91.

GL02 Gilbert, S. ja Lynch, N., Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *ACM SIGACT News*, 33,2(2002), sivut 51–59.

HR83 Haerder, T. ja Reuter, A., Principles of transaction-oriented database recovery. *ACM Computing Surveys (CSUR)*, 15,4(1983), sivut 287–317.

Lam01 Lamport, L., Paxos made simple. *ACM SIGACT News*, 32,4(2001), sivut 18–25.

Lea10 Leavitt, N., Will nosql databases live up to their promise? *Computer*, 43,2(2010), sivut 12–14.

Ore10 Orend, K., Analysis and classification of nosql databases and evaluation of their ability to replace an object-relational persistence layer, 2010.

Pok11 Pokorny, J., Nosql databases: a step to database scalability in web environment. *Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services*. ACM, 2011, sivut 278–283.

Pri08 Pritchett, D., Base: An acid alternative. *Queue*, 6,3(2008), sivut 48–55.

SB11 Srinivasan, V. ja Bulkowski, B., Citrusleaf: A real-time nosql db which preserves acid. *Proceedings of the VLDB Endowment*, 4,12(2011).

Vog09 Vogels, W., Eventually consistent. *Communications of the ACM*, 52,1(2009), sivut 40–44.