Henri Haverinen

SQL- ja NoSQL-tietokantojen suorituskykyerot

Tietotekniikan kandidaatintutkielma

17. huhtikuuta 2016

Jyväskylän yliopisto

Tietotekniikan laitos

Tekijä: Henri Haverinen

Yhteystiedot: henri.s.haverinen@student.jyu.fi

Työn nimi: SQL- ja NoSQL-tietokantojen suorituskykyerot

Title in English: Performance differences between SQL and NoSQL databases

Työ: Kandidaatintutkielma

Sivumäärä: 22+0

Tiivistelmä: SQL-tietokantoja on käytetty lähes tietokantojen olemassaolon alusta asti. Nykyään dataa on yhä enemmän kuin ennen, se on monipuolisempaa, ja sen prosessointi on työläämpää. NoSQL-tietokannat ovatkin nousseet perinteisten SQL-tietokantojen rinnalle tarjoamaan vaihtoehtoisia ratkaisuja nykypäivän haasteiden ratkaisemiseksi. Tutkielmassa tarkastellaan tietokantojen suorituskykyeroja käymällä läpi jo tehtyjä tutkimuksia aihealueesta, sekä pohditaan ovatko NoSQL-tietokannat valmiita käytettäviksi.

Avainsanat: Tietokanta, SQL, NoSQL, suorituskyky

Abstract: SQL databases has been used almost since databases were invented. Nowadays there is a lot more data than before, it is more diverse, and processing it requires more work. NoSQL databases have risen to great favour as they try to offer alternative solutions for challenges we face today in the IT field. In this thesis performance differences between SQL and NoSQL databases are studied basing on existing researches of the topic. Also some reasoning is done whether NoSQL databases are ready enough for usage.

Keywords: Database, SQL, NoSQL, performance

Kuviot

Taulukot

Sisältö

1 Johdanto

Tietokantoja on käytetty tiedonhallinnassa jo 1960-luvulta lähtien (Oxford English Dictionary 2016). Tietokanta on kokoelma dataa, joka on organisoitu erityisesti datan nopeaa etsimistä ja noutamista varten. Yleensä tietokantoihin tallennettavaa dataa on paljon, ja siksi tietokannat ovat suunniteltuja organisoimaan dataa edellä mainittuja ominaisuuksia suosien (“Database | Definition of Database by Merriam-Webster” 2016).

Ensimmäiset tietokannat olivat linkitetyn listan tyylisiä, ja hyvin pian niissä todettiinkin olevan monia puutteita. Codd (1970) esitteli artikkelissaan mullistavan relaatiomallin tietokannoille vuonna 1970. Relaatiomalli on teoreettinen malli tietokantojen loogiselle rakenteelle, jossa data jaetaan pienempiin osiin, joita kutsutaan relaatioiksi. Tarvittaessa relaatioita pystytään yhdistämään toisiinsa, jolloin kerralla on mahdollista noutaa tietokannasta enemmän dataa.

Muutaman vuoden kuluttua Coddin artikkelin julkaisusta IBM kehitti tänäkin päivänä tunnetun ja paljon käytetyn standardoidun kielen SQL:n (Structured Query Language), jota käytetään relaatiotietokannassa olevan datan hallintaan (Chamberlin ja Boyce 1974). Siitä lähtien SQL-tietokantojen suosio alkoi kasvaa, ja ne ovat edelleen käytetyimpiä tietokantoja.

2000-luvulla aloitettiin puhumaan NoSQL-tietokannoista. NoSQL-tietokannalla tarkoitetaan tietokantaa, jonka malli poikkeaa perinteisestä SQL-tietokannan relaatiomallista. Vaikka NoSQL-tietokantoja on ollut olemassa tietokantojen olemassaolon alusta asti, termi NoSQL tuli yleiseen käyttöön vasta 2000-luvulla, kun suuret yritykset, kuten Facebook, Google, ja Amazon, ilmaisivat tarpeensa NoSQL-tietokannoille (Leavitt 2010).

Tänä päivänä datan määrä on valtavaa, ja sitä tuotetaan koko ajan suurempia määriä. Internet of Thingsin 3.5 myötä kaikki fyysiset laitteet alkavat olla verkossa ja lähettävät sekä vastaanottavat dataa. Web 2.0:n myötä käyttäjien on ollut helpompi tuottaa ja jakaa dataa verkossa ja niin sanottua bigd ataa kertyy yhä enemmän. Palveluita tarjotaan paljon suoraan verkon kautta, esimerkiksi pilvipalveluina. Kaikki tämä data on lisäksi myös hyvin vaihtelevaa ja monipuolista, eikä vanhat mallit välttämättä enää sovi nykyajan tarpeisiin. Onkin siis tullut yhä aiheellisemmaksi panostaa suorituskykyyn, jotta nykyajan tietoa voidaan käsitellä ja palveluita tarjota tehokkaasti.

SQL- ja NoSQL-tietokannoista sekä niiden suorituskyvyistä on tehty paljon tutkimuksia, ja niitä kehitetään myös koko ajan paremmiksi ja tehokkaammiksi. Silti perinteiset relaatiotietokannat eivät välttämättä enää riitä nykypäivän tarpeille, ja rinnalle ovatkin nousseet kovaa vauhtia NoSQL-tietokannat tarjoamaan uusia

vaihtoehtoja.

Tämän tutkielman tarkoituksena on selvittää SQL- ja NoSQL-tietokantojen suorituskykyeroja pohjautuen jo tehtyihin tutkimuksiin ja testeihin. NoSQL-tietokantoja on kehitetty vastaamaan tarpeisiin, joihin SQL-tietokannat eivät välttämättä pysty tai sovellu. Kuitenkin NoSQL-tietokannat ovat vielä suhteellisen uusia, ja onkin tärkeää selvittää, ovatko ne vielä tarpeeksiä kypsiä toimimaan SQL-tietokantojen korvaajana.

Luvussa 2 käsitellään lyhyesti SQL- ja NoSQL-tietokantoja sekä niiden keskeisiä piirteitä. Luvussa 3 esitellään jo tehtyjä tutkimuksia ja niiden koeasetelmia, joita on käytetty tämän kandidaatintutkielman lähteinä. Luvussa 4 käydään läpi tutkimusten tuloksia, ja luku 5 sisältää johtopäätökset ja pohdintaa.

2 SQL- ja NoSQL-tietokannat

Termi tietokanta viittaa kokoelmaan toisiinsa liittyvää dataa ja tapaan kuinka se on organisoitu. Tietokannassa olevaan dataan pääsyyn ja sen hallinnointiin käytetään yleensä tietokannan hallintajärjestelmää. Tietokannan hallintajärjestelmät koostuvat joukosta ohjelmistoja, joiden avulla käyttäjä pystyy olemaan vuorovaikutuksessa tietokantojen kanssa. Tyypillisesti hallintajärjestelmät tarjoavat useita eri toimintoja käyttäjälle, kuten datan hakemisesta sen tallentamiseen, muokkaamiseen ja poistamiseen. Koska itse tietokanta ja tietokannan hallintajärjestelmä ovat hyvin lähellä toisiaan, on arkikielessä sana tietokanta vakiintunutkin tarkoittamaan molemmat sisältävää kokonaisuutta.

Tietokantamalli on datamallin yksi tyyppi. Tietokantamalli määrittelee tietokannan loogisen rakenteen sekä pääperiaatteet, kuinka dataa tietokantaan voi tallentaa, järjestää ja muokata. Erilaisia tietokantamalleja on useita, joista yleisin ja suosituin on relaatiomalli (DB Engines 2016).

Sekä tietokanta että tietokannan hallintajärjestelmä noudattavat tietyn tietokantamallin edellä mainittuja periaatteita. Tietokantajärjestelmällä viitataan kollektiivisesti tietokantamalliin, hallintajärjestelmään ja tietokantaan.

2.1 SQL-tietokannat

SQL on lyhenne sanoista Structured Query Language (Encyclopædia Britannica Online 2016) ja se on suunniteltu relaatiotietokantojen datan hallintaan. SQL on ANSIn (American National Standards Institute) ja ISOn (Internation Organization for Standardization) standardoima kieli.

Relaatiomallisessa tietokannassa data esitetään monikkoina (tuple). Monikko kuuluu relaatioon ja sisältää relaatiossa määritellyt attribuutit (Codd 1970). Monikko siis kuvaa yhtä tietuetta relaatiossa. Relaatiossa on myös määritelty jokaiselle monikolle uniikki avain, josta monikon tunnistaa yksiselitteisesti. Uniikkien avainten avulla monikkoja voidaan yhdistää toisten relaatioiden monikoihin.

Arkikieleen on muodostunut edellä mainituille termeille omat ’SQL-termit’, joita yleensä käytetään relaatiomallin termien sijaan. Taulukko 1 selventää näiden termien yhteneväisyyden, ja kuva 1 esittää esimerkin relaatiotietokannan datasta. Myös termi SQL-tietokanta on yleistynyt arkikieleen relaatiotietokannan synonyymiksi, vaikka todellisuudessa SQL-standardin implementoivat tietokannat poikkeavat osittain Coddin alkuperäisestä relaatiomallista (Codd 1990).

Taulukko 1. SQL- ja relaatiotietokantojen termien väliset yhteydet

SQL-termi	Relaatiotietokantatermi	Kuvaus
Rivi	Monikko	Kuvaa yhtä tietuetta
Sarake	Attribuutti	Monikon sisältämä elementti
Pöytä / taulu	Relaatio	Sisältää monikot ja attribuutit

2.2 NoSQL-tietokannat

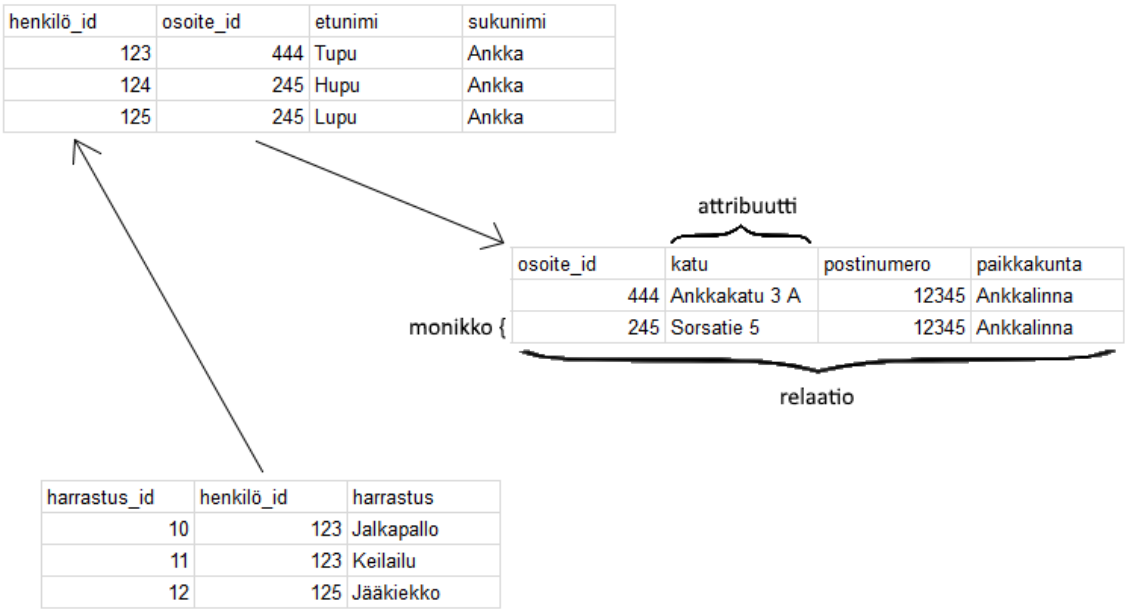
NoSQL tarkoittaa ’non SQL’ tai ’Not Only SQL’. Termi tarkoittaa tietokantoja jotka noudattavat jotain muuta kuin relaatiomallia. NoSQL-tietokannat käyttävätkin paljon erilaisia malleja valmistajasta riippuen. Suosituin NoSQL-tietokantamalli on dokumenttisuuntautunut tietokanta (DB Engines 2016). Muita NoSQL-tietokantamalleja ovat esimerkiksi avain-arvo-, graafi- ja oliotietokannat. Seuraavaksi esittelen lyhyesti kaksi suosituinta NoSQL-tietokantamallia: dokumentti- sekä avain-arvo-tietokannat.

Dokumenttisuuntautunut tietokanta tallentaa tietoa dokumentteihin, joiden formaatti on ennalta määritelty. Dokumenttien sisältävää dataa ei sen sijaan määritetä ennalta, vaan jokainen dokumentti sisältää sekä itse datan, että metadatan, joka kertoo dokumentin sisällön merkityksen. Tästä johtuen dokumenttisuuntautuneet tietokannat ovat hyvin skaalautuvia verrattuna relaatiotietokantoihin (Bhogal ja Choksi 2015). Yleisesti käytettyjä dataformaatteja ovat XML, YAML ja JSON, joskin ne eivät rajoitu edellä mainittuihin. Esimerkki JSON-mallisesta datasta on kuvassa 2. Suosittuja dokumenttisuuntautuneita tietokantoja ovat MongoDB, CouchBase ja Clusterpoint DBMS.

Avain-arvo-tietokanta tallentaa kaikessa yksinkertaisuudessaan avain-arvo-pareja. Avain toimii tunnisteena, jonka avulla siihen liitetty arvo voidaan noutaa tietokannasta. Avain-arvo-parin arvon ei tarvitse olla tismalleen yksi arvo, vaan se voi olla esimerkiksi lista tai objekti, joka sisältää useamman eri arvon (Bhogal ja Choksi 2015). Avain-arvo-tietokanta muistuttaakin käytännössä sanakirjaa (dictionary) tai hajautuskarttaa (hashmap). Kuten useat muutkin NoSQL-tietokannat, ei avain-arvo-tietokantakaan vaadi etukäteen määriteltyä skeemaa 2.3 (Bhogal ja Choksi 2015). Suosittuja avain-arvo-tietokantoja ovat esimerkiksi Redis ja Oracle NoSQL.

2.3 Miksi NoSQL-tietokanta?

NoSQL-tietokantojen NoSQL database (2016) sivulta löytyvään määritelmään on myös sisällytetty tyypillisiä ominaisuuksia, joita NoSQL-tietokannat pyrkivät tarjoamaan. NoSQL-tietokannat pyrkivät tarjoamaan yksinkertaisempia, skaalautu-



Kuvio 1. Esimerkki relaatiomallista

vampia, tehokkaampia ja edullisempia ratkaisuja SQL-tietokantojen tilalle. NoSQL-tietokannat ovat myös lähes aina avointa lähdekoodia, joka helpottaa kehitystä. SQL-tietokannoissa täytyy määritellä skeema ennen datan lisäämistä, toisin sanoen tietokannalle on kerrottava etukäteen, millaista dataa sinne halutaan tallentaa. NoSQL-tietokannat ovat sen sijaan skeemattomia, jonka ansiosta dataa voidaan lisätä ilman sen tarkempaa määrittelemistä etukäteen (Bhogal ja Choksi 2015). Skeemattomuuden ansiosta kehitys on ketterämpää, sekä tietokanta on hyvin skaalautuva datan suhteen.

NoSQL-tietokannat ovat yleensä suunniteltuja skaalautumaan horisontaalisesti (Bhogal ja Choksi 2015), millä tarkoitetaan siis tietokannan jakautumista usealle eri palvelimelle. Tämä on yleensä edullisempaa kuin skaalaaminen vertikaalisesti, eli nostamalla yhden palvelimen tehoa esimerkiksi lisäämällä uusi prosessori. Tietokannan kyky skaalautua horisontaalisesti on suuri etu, sillä nykyään yhä useampi palvelu tarjotaan pilvestä tai vastaavasta klusteroidusta ympäristöstä. Lisäksi horisontaalisella skaalautumisella ei teoriassa ole rajoja verrattuna vertikaaliseen skaalautumiseen. Uusia palvelimia voi aina lisätä, mutta loputtomasti

```
{
  "id": 1,
  "nimi": {
    "etu": "Erkki",
    "suku": "Esimerkki"
    },
  "osoite": {
    "katu": "Osoite 123 A",
    "postinumero": 40100,
    "Paikkakunta": "Jyvaskyla"
    "Maa": "Suomi"
  },
  "harrastukset": [ "Esimerkin nayttaminen" ]
}
```

Kuvio 2. Esimerkki JSON-mallisesta datasta

uusia prosessoreita ei.

Lisäksi NoSQL-tietokantoja mainostetaan suorituskyvyltään tehokkaampina kuin perinteisiä SQL-tietokantoja (“NoSQL Databases Explained | MongoDB” 2016). NoSQL-tietokantojen pitäisi pystyä käsittelemään valtavia määriä dataa nopeasti, mutta tämä voi vaihdella paljon tietokantatyypistä ja datan rakenteesta riippuen.

Kaikella on kuitenkin hintansa, ja NoSQL-tietokannat vaihtavatkin osan luotettavuudestaan edellä mainittuihin ominaisuuksiin (Bhogal ja Choksi 2015). SQL-tietokannat tarjoavat ACID-transaktiot, kun NoSQL-tietokannat yleensä uhraavat nimenomaan tämän ominaisuuden korkeaan suorituskykyyn ja saatavuuteen. NoSQL-tietokannat kuitenkin pyrkivät pitämään ACID-ominaisuutta muistuttavan, mutta heikomman BASE-ominaisuuden.

ACID on lyhenne sanoista jakamattomuus (Atomicity), eheys (Consistency), eristyneisyys (Isolation) ja pysyvyys (Durability) (Haerder ja Reuter 1983). ACID-ominaisuuden tarkoitus on taata, että kaikki tietokantatransaktiot suoritetaan luotettavasti. Transaktio on suoritettava kokonaan, tai ei ollenkaan (jakamattomuus). Transaktio ei ’riko’ mitään (eheys), vaan transaktion jälkeen tietokanta on edelleen eheä. Transaktion ollessa käynnissä muut eivät saa käyttää käsiteltävää dataa (eristyneisyys), ja tämän lisäksi vahvistetun transaktion on säilyttävä tietokannassa (pysyvyys).

BASE on lyhenne englannin kielen sanoista ’Basically available’, ’Soft state’ ja

’Eventually consistent’ (Pritchett 2008). Tietokanta on siis aina tavoitettavissa (basically available), vaikka pyydetty data ei olisikaan ajantasaista. Data saattaa olla myös virheellistä, vaikka kirjoitusoperaatioita ei tapahtuisikaan kyseisellä hetkellä. Tämä voi johtua tietokannan automaattisesti suorittamista datan prosessoinneista, kuten replikoinnista (soft state). Kuitenkin tietokanta päätyy lopulta tilaan, jossa data on yhdenmukaista kaikkialla, mikäli uusia kirjoitusoperaatioita ei tehdä (eventually consistent).

Strauch, Sites ja Kriha (2011) listaavat useita syitä NoSQL-tietokantojen suosioon nousulle. Yksi suurimmista syistä lienee kuitenkin ajassa. 30 vuotta sitten tietokannat oli suunniteltu sen aikaisille ’supertietokoneille’. Data oli rakenteeltaan hyvin strukturoitua, eikä muutoksia olemassa oleviin skeemoihin juuri tullut. Nykyään tietokantoja käytetään kaikkialla, puhelimista sosiaaliseen mediaan. Data on strukturoimattomampaa ja sitä tuotetaan suuria määriä. Lisäksi yhä useampia palveluita tarjotaan pilvestä, jota ajatellen NoSQL:ää onkin kehitetty.

3 Tietokantojen suorituskyvyn testaaminen tutkimuksissa

Lähes aina suorituskyky on tärkeä tekijä ohjelmistoissa. Käytännössä kaikki ohjelmistot myös lukevat ja kirjoittavat dataa, ja tähän tarkoitukseen ohjelmiston alla piilee usein tietokanta. Tietokannan valinnalla voi olla siis yksinään suuri vaikutus ohjelmiston suorituskykyyn, ja siitä syystä aiheeseen liittyvä testaus ja tutkimus on tärkeää.

Suorituskykyä on testattu paljon yksinkertaisella datalla, kuten myös paremmin todellisessa käytössä olevaa dataa kuvaavalla testidatalla. Jälkimmäisenä mainituissa tutkimuksissa on käytetty yleensä oikeaa dataa, tai rakenteeltaan sitä vastaavaa. Data voi olla esimerkiksi yritysdokumentteja tai Internet of Things dataa. Tällainen data voi poiketa paljon geneerisestä datasta, mikä saattaa vaikuttaa tietokantojen suorituskykyyn. Tässä tutkielmassa on käyty läpi muutamia tietokantojen suorituskykyyn liittyviä tutkimuksia. Seuraavissa luvuissa on perehdytty tarkemmin erilaisiin tietokantojen käyttötilanteisiin, sekä niiden parissa tehtyihin suorituskykytutkimuksiin.

3.1 Yksinkertaiseen dataan liittyviä testejä

Boicea, Radulescu ja Agapin (2012) vertailivat Oracle SQL-tietokantaa ja MongoDB NoSQL-tietokantaa. Suorituskykyä testattiin kolmella perusoperaatiolla: insert-, update- ja delete-operaatioilla. Insert-operaatio lisää kantaan uuden tietueen. Heidän tutkimuksessaan tietokantaan lisättiin yksinkertainen tietue, joka sisälsi vain kolme arvoa: id, etunimi, sukunimi. Update-operaatiolla päivitettiin jo olemassa olevien tietueiden sukunimi-arvoa. Lopulta edellisissä operaatioissa luodut tietueet poistettiin delete-operaatiolla. Jokaista operaatiota testattiin joukoilla, joiden koot olivat 10, 100, 1000, 10 000, 100 000 ja 1 000 000.

Li ja Manoharan (2013) tutkivat kuuden eri NoSQL-tietokannan, sekä yhden SQL-tietokannan, MS SQL Express, suorituskykyä avain-arvo -parien tallentamisessa. Tutkimuksessa käytettiin neljää perusoperaatiota; tietokannan luontia, avain-arvo -parin lukemista, kirjoittamista sekä poistamista. Tietokannan luonti tehtiin viisi kertaa, joihin kuluneista ajoista otettiin keskiarvo. Tietueita koskevissa operaatioissa suorituskykyä testattiin joukoilla, joiden koot olivat 10, 50, 100, 1000, 10 000 ja 100 000. Lisäksi tutkimuksessa testattiin kaikkien avainten hakemista tietokannasta edellä mainitun kokoisilla joukoilla.

3.2 YCSB ja APM

YCSB (Yahoo! Cloud Serving Benchmark) on suosittu avoimen lähdekoodin ohjelmistokehys erilaisten avain-arvo-, sekä pilvestä tarjottavien tietokantojen suorituskyvyn testaamiseksi. YSCB tarjoaa valmiin asiakkaan työmäärien generoimiseksi, sekä rajapinnan sen laajentamiseksi. YCSB sisältää myös valmiita työmääräskenaarioita (workload scenarios) ajettavaksi. YCSB tukee useita eri tietokantoja, mikä on eräs syy sen suosioon. Sillä on helppo testata usean eri tietokantojen suorituskykyä samalla alustalla ja samanlaisilla työmäärillä. YCSB voikin auttaa valitsemaan sopivan tietokannan projektiin (Cooper 2016).

Rabl ym. (2012) käyttivät YCSB:tä tutkimuksessaan. He testasivat kuuden avoimen lähdekoodin tietokantaa yrityksen Application Performance Management:in (APM) tuottaman datan hallintaan. APM tarkoittaa yrityksen applikaatioiden/-sovellusten suorituskyvyn monitorointia ja hallintaa. APM yleensä muuntaa 'IT-metriset' arvot bisnesmerkityksellisiksi, esimerkiksi 'tietyn sivun latausaika sen ollessa suurimmalla rasituksella'.

APM-data on tavallisesti melko yksinkertaista. Se koostuu yleensä nimestä, arvosta sekä aikaleimasta. Tietueen rakenne on siis usein kiinteä. Testidata koostuikin avain-arvo -pareista. Testeissä käytettiin viittä eri työmääräskenaariota (workload scenario), jotka koostuivat insert-, read- ja scan-operaatioiden kombinaatioista. Insert- ja read-operaatiot kirjoittavat tai lukevat arvon tietokannasta. Scan-operaatiolla voidaan tarkistaa systeemin terveyden tila tai laskea kokonaisarvoja tietyltä aikaväliltä, esimerkiksi usean operaation suorittamiseen kulunut aika viimeisen tunnin ajalta.

3.3 ECM

Enterprise content management (ECM) tarkoittaa teknologioiden, työkalujen ja metodien käyttöä sisällön hallintaan ja jakamiseen yrityksen sisällä. Tällainen data on suurimmaksi osaksi strukturoimatonta kuten sähköposteja, tekstidokumentteja sekä kuvatiedostoja. Tällaisen tiedon taltiointi ja jakaminen on työlästä, ja siksi yritykset ovatkin alkaneet etsiä ratkaisuja NoSQL-tietokannoista.

Perinteisesti ECM:n taltiointiin on käytetty SQL-tietokantoja, vaikkeivat ne sovikaan hyvin strukturoimattoman datan taltiointiin (Rats ja Ernestsons 2013). Usein taltiointi onkin toteutettu tallentamalla tietokantaan vain dokumentteihin liittyvä metadata, kuten kirjoittaja, dokumenttityyppi ja avainsanat. Itse dokumentit ovat taltioitu omaan säilytyspaikkaan. SQL-tietokanta tarvitsee metadataa etenkin hakufunktion toimintaa varten. Haku tapahtuu kuitenkin vain metadatasta, jolloin sisällön hakeminen ja löytäminen voi olla vaikeaa.

NoSQL-tietokannat tarjoavat uusia mahdollisuuksia ECM:n taltiointiin. Dokumenttisuuntautuneet tietokannat pystyvät tallentamaan dokumentit kokonaisina, jolloin sisältöä ei tarvitse pilkkoa dataan ja metadataan. Rats ja Ernestsons (2013) tekivät tutkimusta NoSQL-tietokannan soveltuvuudesta ECM:n taltiointiin perinteisen SQL-tietokannan sijasta. Tietokantojen suorituskykyä he testasivat dokumenttien lataukseen, etsintään sekä indeksointiin kuluneilla ajoilla. Testidatana he käyttivät kopiota englannin kielisestä Wikipediasta, yhteensä 13 miljoona dokumenttia. SQL-tietokantana tutkimuksessa käytettiin Namejs-tietokantaa, joka on MS SQL:ään pohjautuva ja ECM:n taltiointiin suunniteltu tietokanta. NoSQL-tietokantana käytettiin Clusterpoint-tietokantaa, joka on dokumenttisuuntautunut NoSQL-tietokanta.

3.4 Sensoridata

Sensoreiden käyttö on yleistynyt paljon viimeisen vuosikymmenen aikana. Sensorit ovat pienentyneet ja halventuneet, sekä niiden käyttö on helpottunut. Täten myös sensorien tuottaman datan määrä on lisääntynyt, ja sitä myöten myös tarve sen tehokkaalle taltioinnille ja prosesoinnille.

Tyypillisesti sensoridatan käsittelyyn liittyy kaksi keskeistä operaatioita: sensori tallentaa yhden uuden arvon tietokantaan, ja analysaattori lukee useita arvoja tietokannasta sekä käsittelee niitä (Veen, Waaij ja Meijer 2012). Sensoridata ja siihen liittyvät työmäärät voivatkin poiketa paljon valmiiden benchmark-ohjelmien synteettisistä työmääristä.

Veen, Waaij ja Meijer (2012) testasivat tietokantojen suorituskykyä sensoridatan lukemisessa ja kirjoittamisessa, niin fyysisellä kuin virtuaalisella alustalla. Testioperaatioina he käyttivät yhden sekä usean asiakkaan yhtä sekä useaa yhtäaikaista kirjoitus- ja lukuoperaatiota.

3.5 IoT

Internet of Thingsillä tarkoitetaan fyysisten esineiden, niin sanottujen älylaitteiden, yhdistämistä verkkoon. Laitteet pystyvät siis kommunikoimaan keskenään verkon yli, ja tuottavat aivan uusia mahdollisuuksia tietotekniikan saralla. Tyypillisiä esimerkkejä IoT-laitteista ovat esimerkiksi sensorit, RFID-tunnisteet, tietokoneet sekä älypuhelimet.

Vaikka IoT tuo mukanaan uusia mahdollisuuksia, tuo se myös mukanaan uusia haasteita. IoT-laitteet tuottavat dataa ja kommunikoivat keskenään, jonka seurauksena on suuri tarve datan tehokkaalle käsittelylle. Tässäkin tapauksessa data on hyvin monipuolista, sitä on erilaisissa formaateissa ja se on rakenteeltaan

hyvin vaihtelevaa. Lisäksi tarve käsitellä ja siirtää dataa verkon yli reaaliajassa nostaa tietokantojen suorituskyvyn avainasemaan.

Mai, Nurminen ja Di Francesco (2014) tutkivat tietokantojen kykyä vastata IoT:n vaatimuksiin. He testasivat neljää avoimen lähdekoodin tietokantaa: MySQL, MongoDB, CouchDB ja Redis, näistä kolme jälkimmäistä ovat NoSQL-tietokantoja. Suorituskykyä testattiin kahdella datatyypillä, jotka ovat hyvin tyypillisiä IoT:n parissa: sensoridatalla sekä multimediadatalla. Testeissä tietokanta-asiakkaat suorittivat luku- sekä kirjoitusoperaatioita tietokantoihin, ja näihin operaatioihin kuluneista ajoista otettiin keskiarvo.

Mai, Nurminen ja Di Francesco (2014) simuloivat sensoreita usealla sensorisolmulla (node), jotka lähettivät tasaisin väliajoin generoitua dataa tietokantaan. Multimediadatan tapauksessa asiakas lähetti koko ajan multimediadataa tietokantaan, jonka jälkeen muut asiakkaat pystyivät hakemaan sinne tallennettuja tiedostoja. MySQL-tietokannan kanssa käytettiin blob (Binary Large Object) datatyyppiä isojen tiedostojen tallentamiseen, kun taas MongoDB:n tapauksessa käytettiin GridFS-ominaisuutta. GridFS jakaa tietokannan kahteen osaan: kokoelmaan, jossa säilytetään tiedostojen metadata, sekä toiseen kokoelmaan jossa itse data säilytetään pieninä paloina.

4 Tulokset

Tässä luvussa on käsitelty edellisessä luvussa esiteltyjen tutkimuksien tuloksia. Aluksi on käsitelty tuloksia tutkimuskohtaisesti, ja viimeisessä alaluvussa on yhteenveto suorituskykytesteistä.

4.1 Yleiset testit

Boicea, Radulescu ja Agapin (2012) saivat tutkimuksessaan varsin odotettavia tuloksia. MongoDB oli kaikissa heidän tekemissään testeissä huomattavasti Oraclea nopeampi. MongoDB säilytti hyvin nopeutensa tietueiden määrästä riippumatta. Nopeus pysyi update ja delete -operaatioissa lähes samana. Oraclen tapauksessa operaatioiden kesto kasvoi valtavasti 100 000 tietueen kokoisilla, ja sitä suuremmilla joukoilla.

NoSQL-tietokantojen välillä on myös paljon eroja. Tutkijoiden Li ja Manoharan (2013) saamissa tuloksissa oli paljon eroavaisuuksia eri valmistajien NoSQL-tietokantojen, sekä operaatioiden välillä. NoSQL-tietokannoista parhaiten pärjäsivät MongoDB sekä Couchbase. SQL Express suoriutui myös hyvin testeissä. Ainoastaan tietokannan luonnissa SQL Express oli hitain. Huomionarvoista on, että usein NoSQL-tietokannat ovat optimoituja avain-arvo -parien tallentamiseen kun taas SQL-tietokannat eivät. Siitä huolimatta SQL Express oli kokonaisuudessa selvästi tehokkaampi kuin usea NoSQL-tietokanta, ja SQL Express olikin esimerkiksi nopein kaikkien avainten noutamisessa tietuejoukon koosta riippumatta.

4.2 YCSB

Tutkijoiden Rabl ym. (2012) tutkimuksessa saaduissa tuloksissa NoSQL-tietokanta Cassandralla oli suurin suoritusteho kaikissa testeissä, joskin samasta syystä Cassandran kirjoitus- ja lukuoperaatioiden latenssi oli suhteellisen suuri. MySQL-tietokannan suorituskyky oli lähes yhtä hyvä kuin Cassandralla kirjoitus- ja lukuoperaatioiden kohdalla. Scan-operaatioiden suoritus oli MySQL-tietokannalla huomattavasti hitaampaa, mikä johtunee osaksi operaatioiden huonosta implementoinnista YCSB:ssä.

Myös muut NoSQL-tietokannat pärjäsivät testeissä hyvin. HBase:lla oli alhaisin latenssi kirjoitusoperaatiossa, mutta suurin lukuoperaatioissa. Project Voldemort tarjosi myös hyvin tasaisen latenssin operaatiosta riippumatta. Mielenkiintoista on myös huomata, että kolmen erityisesti skaalautuvuuteen pohjautuvan NoSQL-tietokannan - Cassandra, Voldemort ja HBase - suorituskyky kasvoi lineaarisesti

solmujen määrän suhteen (Rabl ym. 2012).

4.3 ECM

Dokumenttien prosessointi ei juuri hidastunut NoSQL-tietokantojen kohdalla dokumenttien määrän kasvaessa (Rats ja Ernestsons 2013). SQL-tietokanta sen sijaan alkoi hidastua huomattavasti, kun käsiteltävänä oli 100 000 dokumentin kokoinen otos. Haku 13 miljoonasta dokumentista kesti SQL-tietokannasta yli 40 minuuttia, kun taas NoSQL-tietokanta suoriutui samasta tehtävästä alle sekunnissa. Dokumenttien latausajat olivat NoSQL-tietokannassa paremmat, ja indeksointi oli noin kahdeksan kertaa nopeampaa. Testitulokset ovat hyvin selitettävissä NoSQL-tietokannan kyvyllä säilöä dokumentit kokonaisina. SQL-tietokanta joutuu pilkkomaan datan useaan tauluun, jolloin operaatiot kestävät luonnollisesti huomattavasti kauemmin.

4.4 Sensoridata

Sensoridatan tapauksessa tulokset vaihtelivat hurjasti operaatiosta ja yhtäaikaisista asiakkaiden määrästä riippuen (Veen, Waaij ja Meijer 2012). Myös alustan virtualisointi vaikutti suorituskykyyn, joskin useimmissa tapauksissa negatiivisella tavalla.

Yksikään tietokanta ei ollut tutkijaryhmän Veen, Waaij ja Meijer (2012) tekemissä testeissä kokonaisuudeltaan selvästi muita parempi. NoSQL-tietokanta MongoDB oli huomattavasti muita tietokantoja tehokkaampi yksittäisissä kirjoitusoperaatioissa, kun taas SQL-tietokanta PostreSQL oli tehokkain useiden lukuoperaatioiden tekemisessä. NoSQL-tietokanta Cassandra pärjäsi muuten hyvin useiden asiakkaiden yhtäaikaisissa operaatioissa, mutta kärsi eniten virtualisoinnin vaikutuksesta suorituskykyyn. MongoDB ja PostreSQL kokivat vähemmän vaikutusta virtualisoinnista, joskin pientä eroa oli kuitenkin havaittavissa. Muutamassa tapauksessa virtualisointi nosti suorituskykyä, mikä johtunee virtuaalialustan välimuistittamisesta.

4.5 IoT

Sensoridataa koskevien suorituskykytestien tulokset olivat hyvin samanlaisia kuin tutkijakolmikon Veen, Waaij ja Meijer (2012) tukimuksessa (Mai, Nurminen ja Di Francesco 2014). Tietokannoista MongoDB koki vähiten latenssia suurtenkin määrien kirjoittamisessa tietokantaan, kun taas MySQL pärjäsi MongoDB:tä paremmin tiedon hakemisessa. Tiedon noutamisessa tietokannasta parhaimman tuloksen kuitenkin sai Redis, joskin MySQL:n tulokset olivat liki samat. CouchDB:n

tulokset olivat kohtuullisen huonot. Lisäksi CouchDB:n tekemä tietokanta oli kooltaan monta kertaa suurempi muiden tekemiin tietokantoihin verrattuna.

Multimediadatan kirjoittamis- ja lukutestejä tehtiin vain MongoDB ja MySQL tietokannoilla (Mai, Nurminen ja Di Francesco 2014). Kirjoittamiseen liittyvässä latenssissa MySQL pärjäsi paremmin. Kirjoitettavan tiedoston koko ja asiakkaiden määrä vaikutti luonnollisesti viiveeseen, ja huomattavaa eroa alkoikin syntyä kirjoitettaessa yli 2MiB kokoisia tiedostoja. Lukuoperaatiossa erot olivat jokseenkin häilyviä, eikä selvää eroa tietokantojen välille tullut. Tulokset olivat yllättäviä, sillä NoSQL-tietokantoja käytetään usein suuren strukturoimattoman datan tallentamiseen, ja MongoDB:n olisikin voinut odottaa pärjäävän paremmin.

4.6 Tuloksien yhteenveto

Keskimäärin NoSQL-tietokannat siis pärjäsivät hyvin suorituskykytesteissä, päihittäen SQL-tietokannat useassa tehtävässä. NoSQL-tietokantojen välillä on kuitenkin suuria eroja, ja joissakin tutkimuksissa osa NoSQL-tietokannoista pärjäsi selvästi huonommin kuin perinteiset SQL-tietokannat. Lisäksi osa saattoi olla tehokkaita vain tietyissä operaatioissa, tai tietynlaista dataa käsiteltäessä.

Testeissä käytetty data vaikutti selvästi tietokantojen suorituskykyyn. Sensoridataa, joka on rakenteeltaan useasti hyvin yksinkertaista, kuten yksittäisiä lukuja, oli nopeampaa kirjoittaa NoSQL-tietokannalla. Kyseisen datan lukeminen oli kuitenkin nopeampaa SQL-tietokannalla (Veen, Waaij ja Meijer 2012; Mai, Nurminen ja Di Francesco 2014). SQL-tietokannoista datan hakeminen onkin helppoa, myös useista eri tauluista yhtäaikaa, ja tämä näkyy myös suorituskyvyssä.

Oletuksena voi pitää, että NoSQL-tietokannat pystyvät prosessoimaan strukturoimatonta dataa paremmin. Kyky tallentaa suurta dataa kokonaisina, esimerkiksi dokumentteja, nopeuttaa prosessointia huomattavasti. ECM-dataa käsiteltäessä NoSQL-tietokanta Clusterpoint suoritui prosessoinnista selvästi paremmin kuin SQL-tietokanta Namejs (Rats ja Ernestsons 2013). SQL-tietokanta ei pystynyt edes toimimaan kohtuullisessa ajassa. Kuitenkin multimediadataa, joka on myös rakenteeltaan strukturoimatonta, pystyi SQL-tietokanta käsittelemään tehokkaammin (Mai, Nurminen ja Di Francesco 2014). Vaikka NoSQL-tietokannat ovatkin usein suunniteltuja strukturoimattoman datan säilömiseen, ei niiden suorituskyky kaikissa tapauksissa päihitä SQL-tietokantoja.

NoSQL-tietokannat säilyttävät myös keskimäärin paremmin tehokkuutensa datamäärien kasvaessa. Datamäärän kasvaessa NoSQL-tietokantojen suorituskyky pysyi suhteellisen samana, kun taas SQL-tietokantojen suoritusnopeus hidastui huomattavasti enemmän vastaavissa tilanteissa (Rats ja Ernestsons 2013). Tässäkin on toki poikkeuksia, ja SQL-tietokannoille suotuisalla, kuten stukturoidulla

datalla, voi SQL-tietokannat pärjätäkin paremmin.

Myös testeissä käytetyillä operaatioilla oli paljon vaikutusta suoritusnopeuteen (Li ja Manoharan 2013). Lähes kaikissa tutkimuksissa käytettiin yksinkertaisia operaatioita, kuten tietueen luonti, lukeminen, päivittäminen ja poistaminen. Kompleksisemmat operaatiot saattaisivat antaa hyvinkin erilaisia tuloksia. NoSQL-tietokannat eivät luonnostaan tue monimutkaisia kyselyjä, ja niiden implementointi voi olla hyvinkin työlästä (Leavitt 2010), mikä lienee yksi syy monimutkaisten kyselyiden vähäiselle käytölle testeissä.

Tietokantojen iällä on varmasti myös vaikutusta. SQL-tietokantojen suorituskykyä on ehditty hioa kymmenien vuosien käytön aikana. NoSQL-tietokantojen sen sijaan ollessa vielä suhteellisen uusia, käyvät ne läpi paljon muutoksia. Nämä muutokset vaikuttavat varmasti suorituskykyyn, ja tulevaisuudessa samat tutkimukset saattaisivat antaa erilaisia tuloksia. Lisäksi suorituskykyjen testaaminen ja tutkiminen auttavat kehittäjiä keskittymään ongelmakohtiin, kuten tiettyjen operaatioiden nopeuden parantamiseen.

5 Pohdinta

Tässä kandidaatintutkielmassa käytiin läpi SQL- ja NoSQL-tietokantojen keskeisiä eroja, sekä erityisesti vertailtiin tietokantojen suorituskykyä. Suorituskykyyn liittyviä tutkimuksia on tehtyä useita, ja tässä tutkielmassa käytiin läpi muutamia yleisiä sekä tarkempia ’tosi elämään’ liittyviä tutkimuksia.

Vaikka NoSQL-tietokantoja mainostetaan suorituskyvyllisesti tehokkaampina kuin SQL-tietokantoja, useat tutkimukset osoittivat toista. Yleensä vain muutama NoSQL-tietokanta päihitti SQL-tietokannat, joissakin tapauksissa ei yksikään. Testidatalla oli myös suuri vaikutus suorituskykyyn. NoSQL-tietokannat olivat yleensä parempia strukturoimattoman datan käsittelyssä. Strukturoidun datan tapauksessa SQL-tietokanta oli usein suorituskyvyltään parempi kuin NoSQL-tietokanta.

Oleellista SQL- ja NoSQL-tietokantojen väliltä valitsemisessa ei ole pelkästään suorituskyky, vaan myös käyttötarkoitus. Molemmat ovat suunniteltuja tekemään hieman eri asioita paremmin kuin toinen. Mikäli datan eheys ja tietokannan luotettavuus ovat korkeimmalla prioriteetilla, silloin valinta on todennäköisesti SQL-tietokanta. Jos taas dataa on valtavasti, se on pääasiassa strukturoimatonta ja käyttökohteen kannalta on mahdollista vaihtaa osa luotettavuudestaan suorituskykyyn, silloin NoSQL-tietokanta on todennäköisesti parempi valita.

NoSQL-tietokantojen ollessa vielä suhteellisen uusia, liittyy niihin huomattavasti suurempi oppimiskäyrä kuin SQL-tietokantojen käyttöön. Usealle IT-alan työntekijälle SQL-tietokannat ovatkin jo entuudestaan tuttuja, jolloin kynnys valita ne on pieni.

NoSQL-tietokannat tuskin tulevat syrjäyttämään perinteisiä SQL-tietokantoja, eivät ainakaan vuosiin. Molemmissa tietokantatyypeissä on omat vahvuutensa ja heikkoutensa. Viimeaikoina on myös puhuttu NewSQL-tietokannoista, jotka ovat edellä mainittujen hybridejä. Pyrkimyksenä on yhdistää SQL-tietokantojen varmuus NoSQL-tietokantojen skaalautuvuudella ja suorituskyvyllä.

Lähteet

Bhogal, J., ja I. Choksi. 2015. “Handling Big Data Using NoSQL”. Teoksessa *Advanced Information Networking and Applications Workshops (WAINA), 2015 IEEE 29th International Conference on,* 393–398. Maaliskuu. doi:10.1109/WAINA.2015.19.

Boicea, A., F. Radulescu ja L. I. Agapin. 2012. “MongoDB vs Oracle – Database Comparison”. Teoksessa *Emerging Intelligent Data and Web Technologies (EIDWT), 2012 Third International Conference on,* 330–335. Syyskuu. doi:10.1109/EIDWT.2012.32.

Chamberlin, Donald D., ja Raymond F. Boyce. 1974. “SEQUEL: A Structured English Query Language”. Teoksessa *Proceedings of the 1974 ACM SIGFIDET (Now SIGMOD) Workshop on Data Description, Access and Control,* 249–264. SIGFIDET ’74. Ann Arbor, Michigan: ACM. doi:10.1145/800296.811515.

Codd, E. F. 1970. “A Relational Model of Data for Large Shared Data Banks”. *Commun. ACM* (New York, NY, USA) 13, numero 6 (kesäkuu): 377–387. ISSN: 0001-0782. doi:10.1145/362384.362685.

———. 1990. *The Relational Model for Database Management: Version 2.* 371–388. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. ISBN: 0-201-14192-2.

Cooper, B. F. 2016. “Yahoo! Cloud Serving Benchmark (YCSB)”. Viitattu 6. maaliskuuta. https://github.com/brianfrankcooper/YCSB/wiki.

“Database | Definition of Database by Merriam-Webster”. 2016. Viitattu 12. maaliskuuta. http://www.merriam-webster.com/dictionary/database.

DB Engines. 2016. “DB-Engines Ranking per database model category”. Viitattu 28. helmikuuta. http://db-engines.com/en/ranking_categories.

Encyclopædia Britannica Online. 2016. “SQL”. Viitattu 27. helmikuuta. http://global.britannica.com/technology/SQL.

Haerder, Theo, ja Andreas Reuter. 1983. “Principles of Transaction-oriented Database Recovery”. *ACM Comput. Surv.* (New York, NY, USA) 15, numero 4 (joulukuu): 287–317. ISSN: 0360-0300. doi:10.1145/289.291.

Leavitt, Neal. 2010. “Will NoSQL Databases Live Up to Their Promise?” *Computer* (Los Alamitos, CA, USA) 43, numero 2 (helmikuu): 12–14. ISSN: 0018-9162. doi:10.1109/MC.2010.58.

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

Mai, Phan Thi Anh, J.K. Nurminen ja M. Di Francesco. 2014. “Cloud Databases for Internet-of-Things Data”. Teoksessa *Internet of Things (iThings), 2014 IEEE International Conference on, and Green Computing and Communications (GreenCom), IEEE and Cyber, Physical and Social Computing(CPSCom), IEEE,* 117–124. Syyskuu. doi:10.1109/iThings.2014.26.

NoSQL database. 2016. “NoSQL Databases”. Viitattu 31. maaliskuuta. http://nosql-database.org.

“NoSQL Databases Explained | MongoDB”. 2016. Viitattu 17. huhtikuuta. https://www.mongodb.com/nosql-explained.

Oxford English Dictionary. 2016. “database, n.” Viitattu 30. maaliskuuta. http://www.oed.com/view/Entry/47411.

Pritchett, Dan. 2008. “BASE: An Acid Alternative”. *Queue* (New York, NY, USA) 6, numero 3 (toukokuu): 48–55. ISSN: 1542-7730. doi:10.1145/1394127.1394128.

Rabl, Tilmann, Sergio Gómez-Villamor, Mohammad Sadoghi, Victor Muntés-Mulero, Hans-Arno Jacobsen ja Serge Mankovskii. 2012. “Solving Big Data Challenges for Enterprise Application Performance Management”. *Proc. VLDB Endow.* 5, numero 12 (elokuu): 1724–1735. ISSN: 2150-8097. doi:10.14778/2367502.2367512.

Rats, J., ja G. Ernestsons. 2013. “Using of cloud computing, clustering and document-oriented database for enterprise content management”. Teoksessa *Informatics and Applications (ICIA),2013 Second International Conference on,* 72–76. Syyskuu. doi:10.1109/ICoIA.2013.6650232.

Strauch, Christof, Ultra-Large S. Sites ja Walter Kriha. 2011. “NoSQL databases”. http://www.christof-strauch.de/nosqldbs.pdf.

Veen, J.S. van der, B. van der Waaij ja R.J. Meijer. 2012. “Sensor Data Storage Performance: SQL or NoSQL, Physical or Virtual”. Teoksessa *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on,* 431–438. Kesäkuu. doi:10.1109/CLOUD.2012.18.