Tuomas-Matti Soikkeli

NoSQL-tietokannat: vertailu relaatiotietokantoihin ja luokittelu tietomallin sekä käyttökohteiden mukaan

JYVÄSKYLÄN YLIOPISTO
TIETOJENKÄSITITTELYTIETEIDEN LAITOS
2013

TIIVISTELMÄ

Soikkeli, Tuomas-Matti
Kandidaatintutkielma
Jyväskylä: Jyväskylän yliopisto, 2014, 28 s.
Tietojärjestelmätiede
Ohjaaja: Leppänen, Mauri

Tässä tutkielmassa tutkitaan kirjallisuuskatsauksen avulla miten NoSQL-tietokantoja voidaan luokitella ja mitkä ovat niiden käyttökohteet. Toiseksi tutkitaan miten NoSQL-tietokannat ovat kehittyneet ja minkälaisia ominaispiirteiltä niillä on. Samalla tutustutaan miten NoSQ-tietokannat eroavat perinteisistä relaatiotietokannoista ja voidaanko relaatiotietokannat korvata NoSQL-tietokannoilla. Lisäksi tutkitaan NoSQL-tietokantoihin kohdistunutta krittiikiä. Lopuksi tässä tutkielmassa NoSQL-tietokannat luokitellaan neljään ryhmään: avain-arvopari-varastot, dokumenttivarastot, sarakepohjaiset tietokannat ja verkkotietokannat. Jokaisen tietomallin kohdalla esitellään tarkemmin tietomallin piirteet ja käyttökohteet sekä havainnollistetaan tietomallia esimerkein.

Asiasanat: NoSQL, tietokanta, ei-relaationaalinen

ABSTRACT

Soikkeli, Tuomas-Matti
Candidate's thesis
Jyväskylä: University of Jyväskylä, 2014, 28 p.
Information Systems, Candidates Thesis
Supervisor(s): Leppänen, Mauri

In this candidate's thesis NoSQL-databases are researched to categorize and find the most common uses cases for them. Secondly we research how NoSQL-databases are evolved and what are their main properties. At the same time, we study how NoSQL-databases differs from ordinary relational database management systems and are these relational databases replaceable with NoSQL-databases. Moreover we study the critque that the NoSQL faces. Finally, NoSQL-databases are divided into four groups: key-value stores, document stores, column based databases and graph databases. In each data model we study the common properties of particular data model and we explore the use cases for these data models. This is done by providing real-world examples.

Keywords: NoSQL, database, non-relational

KUVIOT

TAULUKOT

Sisältö

1 JOHDANTO

Useat suuret organisaatiot ja yritykset kuten esimerkiksi Google, Amazon ja LinkedIn tallentavat mittavia määriä dataa jatkokäyttöä ja analyysiä varten. Tyypillisesti tiedon tallentamiseen on käytetty SQL-tietokantoja, joihin tieto tallennetaan rakenteelliseen muotoon relaatiomallin mukaan. Rakenteellisen tiedon tallennus ja hakeminen ovat kalliita ja aikaavieviä prosesseja. Siten datamäärien kasvaessa eksponentiaalisesti on siirrytty käyttämään ei-relaationaalisia tietokantoja. Nämä järjestelmät eivät ole kuitenkaan uusi keksintö.

Viime vuosien aikana on ilmestynyt uusia, hyvin erilaisiin tietomalleihin perustuvia tietokantajärjestelmiä. Näitä tietomalleja ovat esimerkiksi avain-arvo-parivarastot, dokumenttivarastot, sarakepohjaiset tietokannat ja verkkotietokannat. Yleisesti näitä kutsutaan NoSQL-tietokannoiksi, koska ne eivät perustu relaatiomallin SQL-kieleen.

Tämä tutkielma selvittää kirjallisuuskatsauksen avulla aluksi NoSQL-tietokantojen kehitykseen ja historiaan liittyviä seikkoja. Sen jälkeen tutkitaan miksi NoSQL-tietokannat ovat kasvattaneet suosiotaan, miten NoSQL-tietokantoja voidaan luokitella sekä mitä käyttökohteita NoSQL-tietokannoilla on. Samalla selvitetään miten nämä tietokannat eroavat relaatiomallista ja sen tiedon manipulointiin liittyvästä SQL-kielestä.

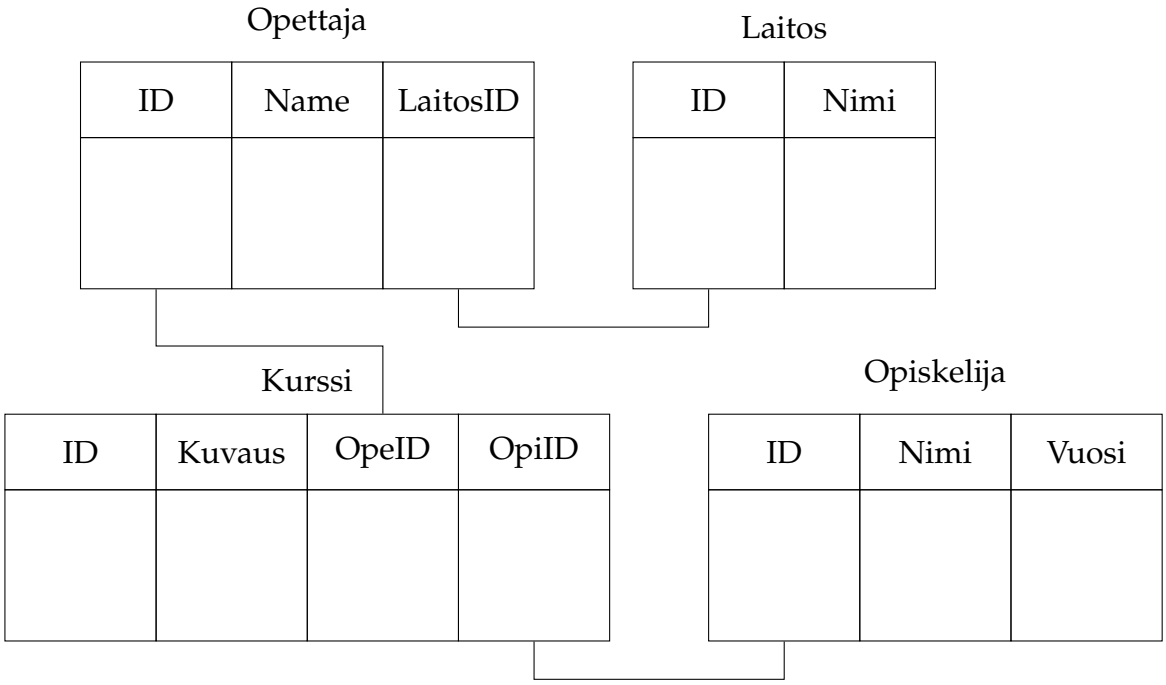
2 NOSQL-TIETOKANTOJEN SYNTY JA KEHITYS

Tässä kappaleessa kerrotaan NoSQL-tietokantojen kehittämiseen johtaneita syitä ja sitä edeltäviä ajanjaksoja sekä tapahtumia. Aluksi tutkitaan SQL-tietokantojen kehitystä ja relaatiomallin vaikutusta tiedonkäsittelyyn. Sen jälkeen kerrotaan miten ACID-ominaisuudet ovat olleet tärkeässä asemassa relaatiotietokantoja. Tämän jälkeen kerrotaan kuinka tiedon valtava massa verkkoliikenteessä on muuttanut tiedonkäsittelyn vaatimuksia ja näinollen tietokannan vaatimuksia suhteessa käsiteltävän tiedon määrään. Lopuksi esitellään CAP-teoreema ja sen pohjalta kehitetty oikeellisuusmalli BASE, joka mahdollistaa tiedon tehokkaan käsittelyn hajautetussa ympäristössä.

2.1 SQL-tietokannat ja relaatiomalli

SQL-kieleen perustuvat relaatiotietokannat ovat tämän hetken suosituimpia tietokannanahallintajärjestelmiä. Edgar Codd (1970) esitti tietomallin jolla voidaan relaatiomallia (verkkomalli ja hierarkkinen malli) yksinkertaisemmalla sekä laitteistoriippumattomalla tavalla jäsentää tallennettava tieto. Tämä malli sai nopeasti suosiota ja johti useiden tutkimusprojektien (Chamberlin & Boyce, 1974) kautta SQL-hakukielen (engl. Structured Query Language) syntyyn (Melton, 1993, s. 8-10). SQL-kieltä käyttävät tietokannat saavuttivat laajamittaisen suosion. Tämä tapahtui koska SQL-kieli käsittelee dataa luonnolisen kielen tavalla. SQL-kieli muistutti siis englantia, eikä tietämystä konekielestä enää tarvittu. (Melton, 1998) SQL-kielen standardoinnin (Melton, 1993, s. 3) jälkeen sitä onkin käytetty laajamittaisesti tiedon tallentamiseen.

Relaatiomallissa (kuvio 1) tieto tallennetaan tauluihin, joka kostuu riveistä ja riveillä olevista arvoista. Taulut voidaan linkittää toisiinsa moninaisten avainarvojen avulla.



Kuvio 1: Tauluihin ja riveihin perustuva esimerkki relaatiomallista

2.2 ACID

Useiden tutkimusten (Gray ym., 1981; Division ym., 1979) pohjalta Andreas Reuter ja Theo Härder julkaisivat artikkelin "Principles of Transaction-Oriented Database Recovery"(Härder & Reuter, 1983), jossa määriteltiin termi ACID.

Lyhennys ACID (engl. Atomicity, Consistency, Isolation, Durability) koostuu suomennettuna sanoista atomisuus, eheys, eristyneisyys ja pysyvyys. ACID tarkoittaa joukkoa ominaisuuksia jotka takaavat tietokannan transaktioiden suorituksen luotettavuuden, niin että jokaisen transaktion suorituksen jälkeen pystytään varmistumaan, että halutut operaatiot ovat todella suoritettu. Esimerkkinä voidaan mainita pankkijärjestelmien toimivuus. Siirrettäessä rahaa toisen käyttäjän tilille on varmistettava, että transaktio on todella tapahtunut alusta loppuun. Mikäli transaktion suorituksen aikana tapahtuu järjestelmän kaatuminen, on palattava alkuperäiseen tilaan, jolloin transaktio alkoi.

- **Atomisuus:** Transaktion aikana suoritetaan joko kaikki suorituksen osat tai ei ainuttakaan. Täten jos yksi osa transaktiosta epäonnistuu, koko transaktio epäonnistuu.
- **Eheys:** Transaktion tulee jatkuvasti täyttää kaikki säännöt ja protokollat, jotka on määritelty järjestelmässä. Transaktio ei riko näitä sääntöjä, joten tietokanta pysyy yhtenäisessä tilassa transaktion suorituksen aikana.

- **Eristyneisyys:** Transaktiot eivät voi hallita toisia transaktioita, jotka ovat kesken. Näinollen jokainen transaktio on itsenäinen.
- **Pysyvyys:** Transaktion valmistuessa sitä ei voida peruuttaa, sen tila on pysyvä. Transaktio selviää järjestelmän sammuttamisen.

Seuraavassa alakappaleessa esitellään tiedon määrän kasvaminen ja sen aiheuttamat kustannukset ACID-ominaisuudet täyttävässä järjestelmässä.

2.3 Tiedon valtava massa

Internetin kehityksen myötä tietokantojen vaatimukset ovat kuitenkin muuttuneet. McKinsey & Company (2011) esitti tutkimuksessaan, että alle tuhannen työntekijän yrityksillä on keskimäärin 3,8 petatavua dataa tallennettuna ja vuotuinen datan määrän kasvusuhde on neljäkymmentä prosenttia. Tätä tiedon jatkuvasti kasvavaa tulvaa kutsutaan tiedon valtavaksi massaksi (engl. Big Data).

Näin ollen tarvitaan erittäin tehokkaita järjestelmiä, jotka pystyvät prosessoimaan tietoa petatavu -luokassa (Han, Haihong, Le, & Du, 2011). Näihin vaatimuksiin perinteiset SQL-tietokannat eivät pysty enää vastaamaan kunnolla. SQL-tietokantojen käsittelee dataa ACID-ominaisuuksia noudattaen(Pokorny, 2013). Tästä johtuen ACID-ominaisuudet aiheuttavat tallennettavalle tiedolle kiinteitä kustannuksia koska jokaisen yksinkertaisen solun ja rivin tallentamisen yhteydessä pitää varmistua tietokannan transaktioiden luotettavuudesta. Tämä hidastaa merkittävästi SQL-tietokantojen nopeutta kun dataa on runsaasti. Tuhansien yhtäaikaisten transaktoiden luotettava käsittely ei ole enää mahdollista, sillä tiedon tallentamisen ja käsittelyn kustannukset alkavat kasvaa eksponentiaalisesti tietyn rajapyykin jälkeen.

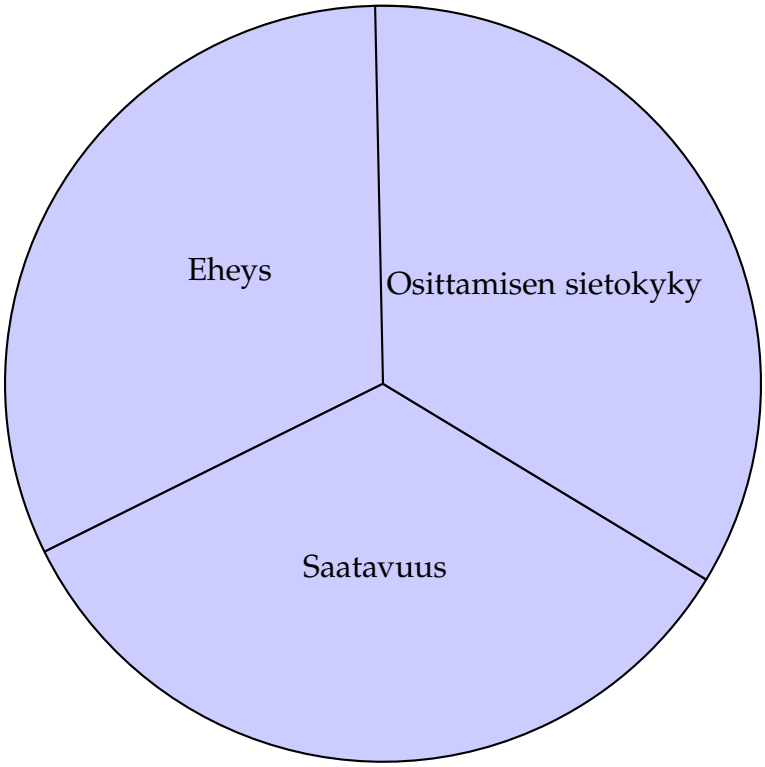
Tietokantajärjestelmille on muodostunut uusi haaste. Tietoa on yksikertaisesti liikaa, jotta sitä voitaisiin käsitellä tietokannassa käyttäen ACID-ominaisuuksia. Seuraavassa kappaleessa esitellään teoreema, joka kuvaa miten järjestelmän ominaisuuksia pitää muokata, erityisesti heikentämällä transaktioiden luotettavuutta, jotta voidaan kustannustehokkaasti tallentaa ja käsitellä tiedon valtaa massaa.

2.4 CAP-teoreema

Konferenssissa ACM Symposium on the Principles of Distributed Computing Eric Brewer (2000) esitteli CAP-teoreeman jotka kutsutaan myös Brewerin teoreemaksi (kuva 2).

CAP-teoreema esittää, että on olemassa kolme tärkeintä vaatimusta hajautetun järjestelmän suunnitteluun, toteutukseen ja käyttönottoon. Nämä ovat eheys, saatavuus ja osittamisen sietokyky eli CAP (engl. consistency, availability, partition tolerance):

- **Eheys:** Jokainen verkon solmu saa yhtäaikaiseen hakuun saman tuloksen.
- **Saatavuus:** Asiakasohjelmien on aina pystyttavä kirjoittamaan ja lukemaan tietoa.
- **Osittamisen sietokyky:** Järjestelmä toimii, vaikka verkkoyhteys joidenkin solmujen välillä katkeaa



Kuvio 2: Hajautetun järjestelmän kolme vaatimusta: CAP-teoreema (Brewerin teoreema)

Brewerin mukaan ei ole mahdollista taata jokaisen kolmen vaatimuksen toimivuutta. Tämä tulee ottaa huomioon järjestelmän suunnittelussa. Jos näin ei tehdä, voi lopputuloksena olla, että jokainen esitetty ominaisuus järjestelmässä rikkoutuu yhtä aikaa. Tämä voi johtaa katastrofaaliseen lopputulokseen järjestelmän sekä liiketoiminnan kannalta. Gilbert ja Lynch (2002) todistivat Brewerin olleen oikeassa ja tätä pidetäänkin yleisesti hyväksyttävänä lähestymistapana tietokantojen tarkasteluun (mm. Vogels, 2009; Cattell, 2011; Pokorny, 2013).

2.5 BASE

CAP-teoreeman mukaiset hajautetut järjestelmät eivät voi enää toteuttaa kaikkia ACID-ominaisuuksia. On mahdollista kuitenkin tehdä järjestelmä, jossa transaktioiden reliabiliteetti tuotetaan jonkin ajan kuluttua, ei välittömästi. Vogels (2009) esitteli lopulta oikeellisen (engl. evetual consistency) oikeellisuusmallin, jonka avulla Pritchett (2008) johti BASE oikeellisuusmallin. BASE (engl. Basically Available, Soft state, Eventual consistency) on lyhennelmä suomen kielen sanoista: periaatteessa saatavilla, pehmeä tila ja lopulta oikea. Toisin sanottuna BASE on oikeellisuusmalli transaktioille, joka ei takaa oikeellisuutta heti. Sen sijaan oikeellisuus taataan tietyn ajan kuluttua, missä aika rippuu järjestelmän muista ominaisuuksista tai muuttujista kuten senhetkisen kuorman määrästä.

BASE ominaisuudet tarkoittavat käytännössä, että tietokantajärjestelmä saattaa palauttaa vanhentuneen arvon, jonka tulisi jo olla päivittynyt. Esimerkiksi sosiaalisen median data ei tarvitse olla reaaliaikaisesti saatavilla, vaan riittää että tieto leviää servereiden välillä kohtuullisen ajan kuluessa. Tietokantaa suuniteltaessa ACID-ominaisuuksien sijasta BASE oikeellisuusmallin valitseminen antaa mahdollisuuden kehittää erittäin tehokkaita tietokantajärjestelmiä, koska tiedon oikeellisuudesta ei tarvitse varmistua välittömästi (Bailis & Ghodsi, 2013).

3 NoSQL

Tässä kappaleessa esitellään NoSQL-tietokannat, jotka pyrkivät ratkaisemaan tiedonkäsittelyn volyymin jatkuvasti uusiutuvan kertaluokan luomat haasteet. Eri NoSQL-tietokannat eroavat huomattavasti toisistaan, mutta kappaleessa esitellään yleisimmät ominaispiirteet, joiden katsotaan määrittelevän NoSQL-tietokantoja. Aluksi esitellään NoSQL:n määritelmä, jonka jälkeen luokitellaan ne neljään eri kategoriaan ja tutkitaan niiden käyttötarkoituksia erityisesti suhteessa tiedon valtavaan massaan.

3.1 Määritelmä

NoSQL-käsitettä käytettiin ensimmäistä kertaa vuonna 1998, kun Strozzi (1998) julkaisi relationaalisen tietokannanhallintajärjestelmän, joka ei käyttänyt SQL-kieltä tiedon tallennukseen tai hakemiseen. Suurempaan tietoisuuteen NoSQL-tietokannat levisivät kuitenkin vasta 2009 kun Last.fm yhteisöpalvelun kehittäjä Jon Oskarsson kutsui koolle kehittäjiä kuulemaan NoSQL-tietokannoista (Evans, 2009).

Ei-relaatiotietokantoja on ollut olemassa jo 60-luvun lopulta asti, mutta uudemmat NoSQL-nimitystä kantavat tietokannat ovat 2000-luvulla kehitettyjä ei-relaationaalisia tietokantoja (Leavitt, 2010). NoSQL-tietkokannat ovat ei-relaationaalisia tietokantoja, joissa ei käytetä SQL-kieltä tiedon tallennukseen tai hakemiseen. Eri tavoilla toimivilla NoSQL-tietokannoilla on yhteisenä tekijänä se, että ne eivät tallenna dataa SQL-kielen avulla. NoSQL (engl. Not only SQL tai Not Relational) nimitys valittiin kuvastamaan tätä kahtiajaottelua SQL- ja NoSQL-tietokantojen välillä. Nimityksen osalta ei olla saavutettu yhtenäistä konsensusta (Cattell, 2011), mutta NoSQL-tietokantojen ominaisuuksista löytyy useita yhteisiä tekijöitä, jotka esitellään seuraavissa alaluvuissa.

3.2 Skaalautuvuus

Yksi tärkein NoSQL-tietokantojen ominaisuus on skaalautuvuus. Sillä tarkoitetaan mahdollisuutta lisätä järjestelmään resursseja, niin että suoritusteho parantuu. Tällä

skaalamisella halutaan vastata käsiteltävän tietomäärän kasvamiseen. (Tiwari, 2011)

Järjestelmiä voidaan skaalata kahdessa suunnassa: horisontaalisesti ja vertikaalisesti. Vertikaalisesti skaalatessa yksittäisen laitteiston tehokkuutta kasvatetaan. Päivitetään tietokone tai vaihdetaan sen komponentteja tehokkaampiin. Tässä tapauksessa käytetään usein supertietokoneita, jotka ovat erittäin kalliita. Horisontaalisesti skaalatessa hajautetaan laskentateho useiden tietokoneiden välille. Tässä tapauksessa voidaan käyttää halpoja komponentteja, jotka voidaan liittää järjestelmään vaivattomasti. Suoritustehoa on näin helppo lisätä tarpeen vaatiessa, eikä suuria kertainvestointeja tarvitse tehdä.

SQL-tietokantojen ACID-ominaisuudet vaikeuttavat horisontaalista skaalattavuutta, sillä tiedon eheyden vuoksi on varmistuttava, että eri laitteilla sijaitseva data on luotettavasti tallennettu ja verkon muiden solmujen saatavilla. NoSQL-tietokannat ottavat tässä suhteessa askeleen poispäin tiedon eheyden ja transaktioiden oikeellisuuden takaavista ACID-ominaisuuksista kohti BASE-ominaisuuksia parantaakseen suoritustehoa (Tiwari, 2011).

3.3 Hajautettavuus

Valtavan tietomassan on aiheuttanut tarpeen prosessoida dataa tehokkaammin. NoSQL-tietokantojen eräs tärkeimmistä ominaisuuksista onkin kyky hajauttaa laskenta useille eri laitteille. Tällöin voidaan hyödyntää rinnakkainlaskentaa suurten datamäärien käsittelyyn. Rinnakkaislaskenta on ideaali tapa hajauttaa laskenta horisontaalisesti skaalatulle järjestelmälle, joka koostuu useasta erillisestä laitteesta. Nämä laitteistot ovat tyypillisesti rakennettu edullisilla kuluttajakomponenteilla. (Silberschatz, Korth, & Sudarshan, 2002).

Pilvilaskentaa voidaan siis hyödyntää käyttäen NoSQL-tietokantoja. Ne ovat luonteeltaan helposti skaalattavissa, joten pilvilaskennan mahdollistama suuri laskentakapasiteetti on relaatiotietokantoihin verrattuna helposti valjastettavissa tehokkaaseen tiedon prosessointiin.

Hajautetussa järjestelmässä tieto voidaan replikoida verkon solmujen välillä. Tämän onkin tyypillisesti NoSQL-tietokantojen tapa varmistua tiedon eheydestä, kun taas relaatiotietokannat varmistavat tiedon eheyden vain transaktioiden suhteen, eikä relaatiotietokannoilla ole helppoa tapaa tiedon replikointiin. Tämä johtu siitä, että verkon solmujen välillä ACID-ominaisuuksien takaaminen on vaikea ja kallis operaatio.

3.3.1 MapReduce

MapReduce on Googlen kehittämä ohjelmointiparadigma joka mahdollistaa massiivisen tietomäärän käsittelyn käyttäen yksinkertaisia tekniikkaa, jonka ansiosta työ on helppo hajauttaa rinnakkainlaskettavaksi. Näin saavutetaan tehokas tapa prosessoida valtavia määriä dataa. Suuret Internetin toimijat kuten Amazon, Google

sekä Facebook ovat rakentaneet omat datan analyysi- ja hakujärjestelmänsä käyttäen MapReduce -ohjelmia laajamittaisesti. (Dean & Ghemawat, 2008)

MapReduce onkin siis ohjelma tai algoritmi joka käyttää NoSQL-tietokantoja. Joihinkin tietokantoihin se on sisäänrakennettu, mutta sen ei tarvitse olla osa NoSQL-tietokantaa. MapReduce käyttää NoSQL-tietokannan luontaista ominaisuutta, yksinkertaista ja helppoa hajautettavuutta, hyväkseen tiedon prosessoinnissa. Erityisesti isot tietomäärät on tehokasta hajauttaa laskettavaksi MapReducen avulla.

MapReduce on kaksivaiheinen algoritmi, jonka ensimmäinen vaihe on *map()*-funktio joka hyväksyy parametrikseen arvoja ja sen palautusarvona on avain-arvo-pareja. Nämä toimivat syötteenä *reduce()* funktiolle joka laskee yhteen aiemmin lasketun arvojen summan. Funktioiden suoritus voidaan hajauttaa järjestelmän solmuille, jolloin laskenta suoritetaan rinnakkain. Pääsolmu jakaa tehtävän pienempiin osiin ja lähettää tehtävän eteenpäin verkossa. Seuraavat verkon solmut voivat tehdä tämän tehtävän uudestaan, kunnes tehtävä on riittävän pieninä tehtävinä tai verkon kaikki solmut ovat käytössä. Tulokset palautetaan sen lähettäneelle solmulle, joka suorittaa summaten tuloksia ja lähettää sen eteenpäin kunnes kaikki tulokset ovat laskettu.

MapReducea voidaan käyttää esimerkiksi sanojen esiintymiseen dokumenteissa, joka lienee yleisin käyttötapaus Internetin hakukoneiden toimintaan liittyen. Aluksi *map()* -funktio laskee yhteen jokaisessa dokumentissa esiintyvän sanan ja lisää sen tuloksen siihen liittyvään sanaan. Seuraavaksi *reduce()* -funktio summaa sanat ja sen esiintymien summan josta vastaukseksi saadaan kaikkien sanojen esiintymien summat 3.

```
map(String avain, String arvo):
   // avain: dokumentin nimi
   // arvo: dokumentti
   int tulos = 0;
   for each sana s in arvo: tulos++;
   lisaa(s, tulos);

reduce(String avain, String[] arvot):
   // avain: sana
   // arvot: taulukko summia
   int tulos = 0;
   for each arvo in arvot: tulos += arvo;
   summaa(result);
```

Kuvio 3: MapReduce algoritmi. Suomennettu Chu ym. (2006) mukaan.

3.4 Yksinkertaisuus

Yksi merkittävimmistä eroista relaatiotietokantoihin on NoSQL-tietokantojen yksinkertainen käyttö. Yksinkertaisuudella tarkoitetaan tässä tapauksessa monimutkaisen hakukielen ja skeeman puuttumista. Ohjelmistokehitys voidaan aloittaa ilman mittavaa tietokannan määrittelyprosessia, mikä vähentää kehityskustannuksia. Toisaalta ACID-ominaisuuksia haluttaessa joudutaan tekemään enemmän töitä sovelluskerroksella. Monimutkaisten hakujen toteuttaminen ilman SQL-kielen kaltaista, hyväksi havaittua, hakujärjestelmää on erittäin vaikeaa (Leavitt, 2010).

3.5 Rajapinta ja hakukielet

Varsin näkyvä ominaisuus NoSQL-tietokannoissa on ohjelmointirajapinta, joka voi olla SQL-tyyppinen kieli, pelkkää HTTP-protokollaa käyttävä rajapinta, suora matalan tason ohjelmointikielen rajapinta tai mikä tahansa kombinaatio edellisistä. SQL-kieltä ei kuitenkaan käytetä kuin muutamissa poikkeuksissa.

Useat, mutta eivät kaikki, NoSQL-tietokannat tarjoavat REST-rajapinnan (engl. Representational State Transfer) kommunikointiin, joka on HTTP-protokollaa käyttävä tekniikka tiedon kuljettamiseen Internetissä. REST-rajapinnan avulla päästään eroon laitteisto- ja kieliriippuvaisuuksista. Nämä heterogeeniset asiakkaat voivat kommunikoida tietokannan kanssa jolloin HTTP-kutsukuorma voidaan tasata ja tulokset voidaan tallentaa välimuistiin. (Hecht, 2011)

3.6 Tehokkuus

NoSQL-tietokannat ovat tehokkaampia prosessoimaan dataa kuin relaatiotietokannat. Skeemattomuus antaa mahdollisuuden aloittaa ohjelmistokehitys ilman laajamittaista tietokannan määrittelyprosessia. Esimerkiksi miljoonalla perusoperaatiolla aikaa mitattaessa tyypillinen relaatiotietokanta jää nopeudessa tyypillisen NoSQL-tietokannan varjoon (kuva 1). Tehokuus saavutetaan siis tilanteissa, missä tietokannan vaatimukset keskittyvät vain-luku tai eniten lukuja -tilanteisiin.

	MySQL	MongoDB
INSERT	882078	5781
UPDATE	27782	3
DELETE	38079	1

Taulukko 1: Perusoperaatioiden suoritusaika millisekunteina, miljoona operaatiota (Boicea ym., 2012).

3.7 Kritiikki

NoSQL on käytössä useimmilla suurimmilla ICT-alan toimijoilla kuten Google, Amazon, LinkedIn, Digg, mutta sille löytyy myös vastustajansa. Useimmat vastustajat argumentoivat, että uusilla teknologiolla on aina kehittymisvaihe, jolloin sitä ei kannata käyttää epävakaudesta ja teknologian nuoresta iästä johtuen. Tässä vaiheessa uskotaan myös NoSQL-tietokantojen olevan. Kattavaa määrää käyttötapauksia ei ole käyty läpi tai ohjelmistosta saattaa löytyä vielä ohjelmistovirheitä, jotka tuotantoympäristössä aiheuttavat merkittävää haittaa liiketoiminnalle, kuten järjestelmän seisonta-aikaa. (Tiwari, 2011; Strauch, Sites, & Kriha, 2011)

NoSQL-tietokantojen hyödyllisyydestä ja käyttötarkoituksista on myös paljon epäselvyyttä. Selkeää konsensusta ei ole nähtävillä. Tieteellisestä kirjallisuudesta vasta-argumentteja NoSQL-tietokannoille ei juuri löydy, mutta Internetin keskustelupalstoilla on huomattavissa trendi, jossa yritykset ovat lakanneet käyttämästä NoSQL-tietokantoja. Esimerkiksi Sarah Mei (2013) blogissaan selvittää miksi Diaspora, yhteisörahoitettu hajautettu sosiaalinen verkko, joutui ongelmiin valitesaan MongoDB:n relaatiotietokannan sijaan. Hylätessään relaationaalisen tiedonkäsittelyn Diasporan kehittävät joutuivat uudelleen kehittämään osan relaatioalgebran ominaisuuksista ja toteuttamaan ne sovelluskerroksella. Yhdeksän kuukauden kehittämisen jälkeen he tekivät päätöksen kehittää koko tietokantajärjestelmän uudelleen käyttän MySQL relaatiotietokantaa. Sarah Mei väittää, että NoSQL-tietokannat eivät juurikaan eroa tyypillisestä välimuistista. Suurin ero oli se, että kehittäjillä ei ollut mahdollisuutta palauttaa järjestelmän tilaa tietokannan perusteella. Väitettä ei kuitenkaan ole perusteltu tutkimustiedolla, mutta herää kysymys siitä, onko NoSQL-tietokantojen jatkokehittäminen vain matkaa kohti relaatioalgebran uudelleenkeksimistä.

Boicea ym. (2012) väittävätkin että NoSQL-tietokantoja ei kannata käyttää mikäli suoritusteho on kriittistä ja dataa on erittäin paljon. Tästä voidaan päätellä se että, NoSQL-tietokannoilla ei ole kuin rajalliset käyttötarkoitukset.

Uudemmat SQL:n perustuvat teknologiat, jotka ovat horisontaalisesti skaalattavissa, ovat myös kehittyneet. NoSQL-tietokantojen suurin kehittävä voima onkin ollut CAP-teoreeman mukainen argumentti, että ACID-ominaisuuksia ei voida saavuttaa verkkoon hajautetussa järjestelmässä. Kuitenkin esimerkiksi MySQL Cluster, VoltDB ja Clustrix ovat uuden sukupolven relaatiotietokantajärjestelmiä, jotka ovat keskittyneet parantamaan skaalautuvuuden ominaisuuksia . Näitä tietokantoja kutsutaan nimellä NewSQL. NoSQL:n vastustajat ovat ilmaiseet, että NewSQL-tietokannat ratkaisevat relaatiotietokantojen puutteet ja, että siltikin on mahdollista taata ACID-ominaisuudet. (Pokorny, 2013; Stonebraker, 2010).

Cattell (2011) esittelee artikkelissaan "Scalable SQL and NoSQL data stores"skaalautuvia relaatiotietokannanhallintajärjestelmiä. Voidaankin esittää kysymys ovatko NoSQL-tietokannat vain kehittymättömiä tietokantoja, jotka ovat suurten yritysten ratkaisuja erityisiin ongelmiin? Esimerkiksi Googlen BigTable ratkaisee Internetin

suosituimman hakukoneen tiedonkäsittelyn vaatimukset. Se ei ole siis yleishyödyllinen ja tyypillisen ohjelmistoarkkitehtuurin käytössä sopiva tietokanta.

Myös NoSQL nimitys saa kritiikkiä. Stonebraker (2010) mukaan NoSQL-tietokantojen nopeudella ei ole mitään merkitystä sen kanssa, että ne eivät käytä SQL-kieltä. Hänen mukaansa helposti replikoitavia relaatiotietokantajärjestelmiä, tietyillä ominaisuuksilla, tiettyyn käyttöön, on olemassa ja ne kehittyvät jatkuvasti nopeudessa.

4 LUOKITTELU JA KÄYTTÖKOHTEET

Aluksi tässä kappaleessa luokitellaan NoSQL-tietokannat tietokannan perusteella, jonka jälkeen esitellään jokainen tietomalli omana kokonaisuutena, sekä esitellään esimerkkejä tietomalleihin perustuvista tietokantatuotteista.

4.1 Luokittelu

NoSQL-tietokantojen luokittelusta ei olla saavutettu yhtenevää luokittelutapaa johtuen NoSQL-tietokantojen nuoresta iästä ja niiden ominaisuuksien monimuotoisuudesta. Luokiteltavat järjestelmät vaihtelet suuresti tietomallin ja muiden ominaisuuksien perusteella. Useat näistä ovat myös hybridiversioita, eli tietokantoja, joissa käytetään useampaa tietomallia.

Tässä tutkielmassa tutkittavat tietokannat luokitellaan neljään tyyppiin tietomallin perusteella: avain-arvopari-varasto (key-value store), saraketietokanta (column based databases), dokumenttitietokanta (document store) ja verkkotietokanta. (graph database). Tietokannat on koottu myös taulukkoon 2. Tietokantojen luokitteluun on käytetty `http://nosql-database.org/` -sivustoa, joka on kerännyt yhteen eri valmistajien NoSQL-tuotteita vuodesta 2009.

Seuraavaksi esitellään luokitellut tietomallit omassa alakappaleissaa ja jokaisen tietomallin kohdalla käydään läpi tyypilliset ominaisuudet ja soveltuvuus eri käyttökohteisiin. Näiden lisäksi esitellään tietomalleja esimerkein.

NoSQL-tietokantojen luokittelu tietomallin mukaan		
Tietokanta	Tietomalli	Käyttökohteet
Redis Memcached SimpleDB DynamoDB Riak	Avain-arvo-pari	välimuisti tiedon hajatus jonot reaaliaikaiset palvelut
MongoDB Amazon CouchDB	Dokumentti	yleiskäyttö
Cassandra Hadoop Hypertable Base SimpleDB	Sarake	tietovarastointi asiakkuudenhallinta analytiikka
Neo4j Infinite Graph TITAN AllegroGraph	Verkko	sosiaalinen media tieteellinen tilastointi sisällönhallinta suosittelujärjestelmät

Taulukko 2: NoSQL-tietokantojen luokittelu tietomallin ja käyttökohteiden mukaan

4.2 Avain-arvo-parivarasto (key-value store)

Yksinkertaisin NoSQL-tietomalli tiedon tallentamiseen on käyttää avain-arvo-pareja (kuva 4). Kaikki data tallennetaan avaimen avulla, joka toimii samalla yksinkertaisena osoittimena (engl. index). Yksinkertaiset avain-arvo-parit voidaan esittää esimerkiksi JSON-notaation (engl. JavaScript Object Notation) avulla. Samalla osoittimella tehty haku palauttaa siihen liittyvän datan. (Cattell, 2011; Seeger & Ultra-Large-Sites, 2009)

avain	arvo
etunimi	Tuomas-Matti
sukunimi	Soikkeli
email	tsoikkeli@gmail.com

Kuvio 4: Avain-arvo-parit

Avain-arvo-paritietokantojen käyttäminen on yleensä hyvin yksinkertaista (kuva 5), sillä erillistä hakukieltä, kuten SQL-kieltä ei käytetä. Hakuja voidaan tehdä

vain avaimen avulla (joissakin tapauksissa arvo voi olla numeerinen arvo, jonka perusteella voidaan tehdä hakuja). Koska arvoja vasten ei voida tehdä hakuja, tietomallilla ei ole tiedossa olevaa rakennetta. Näin saavutetaan erittäin tehokkaat perusoperaatiot *put()* ja *get()*.

```
1  // Tallennus
2  put("etunimi","Tuomas-Matti");
3  put("sukunimi","Soikkeli");
4  put("email","tsoikkeli@gmail.com");
5
6  // Haku
7  var arvo1 = get("etunimi");
8  var arvo2 = get("sukunimi");
9  var arvo3 = get("email");
10
11 // Tulostaa Tuomas-Matti Soikkeli tsoikkeli@gmail.com
12 tulosta(etunimi + " " + sukunimi + " " + email);
```

Kuvio 5: Avain-arvo-parivaraston käyttö, pseudokieli

4.2.1 Dynamo

Dynamo on Amazonin suunnittelama ja toteuttava avain-arvo-paritietokanta. Amazon kehitti dynamoa omiin tarpeisiinsa ja lopulta siitä julkaistiin artikkeli "Dynamo: Amazon's Highly Available Key-value Store"(DeCandia ym., 2007), joka onkin pohjana useille uusille NoSQL-tietokannoille.

Dynamolla on tähtitieteelliset vaatimukset. Esimerkiksi Amazonin verkkokaupassa oleva ostoskärry -palvelu käsittelee yli kolme miljoonaa ostosta päivässä ja yhtäaikaisia asiakkaita tai tiloja on käytössä satoja tuhansia.

Lopputuloksena on erittäin tehokas, saatavilla oleva, järjstelmän kaatumisesta selviävä, skaalattava datavarasto, joka onnistuneesti täyttää halutut vaatimukset.

4.3 Dokumenttivarasto

Dokumenttitietokannat tallentavat datan dokumentteihin, jotka on järjestetty avain-arvopari-varaston tapaan osoittimen avulla. Dokumenttitietokannat ovat kuten avain-arvo-parivarastoja, mutta arvona voidaan tallentaa toisisijaisia osoittimia, listoja tai muita monimutkaisempia tietorakenteita. Toisinsanotta tämä tarkoittaa, että dokumenttivarastot voivat tallentaa vapaamuotoista, ei-relaationaalista dataa, dokumentteja (kuva 6). Dokumenttivarastot eivät keskity korkeisiin luku- ja kirjoitusnopeuksiin vaan tarjoavat kattavan tavan tallentaa tiedon valtavaa massaa. Hakuomi-

naisuudet ovat usein monipuoliset, koska avain-arvo-parivarastoista eroten dataa voidaan hakea myös avaimen lisäksi arvoista.

```
{
    "blogikirjoitus": {
        "otsikko" : "NoSQL-tietokannat",
        "sisalto" : "NoSQL-tietokannat ovat uusi teknologia...",
        "kirjoittaja" : "Tuomas-Matti Soikkeli",
        "tagit" : [
            "tagi" : "NoSQL",
            "tagi" : "ei-relaationaalinen"
            ],
        "kommentit" : [
            "kommentti" : {
                "kirjoittaja" : "Taneli Koivisto",
                "sisalto" : "Kiitos kirjoituksestasi!"

            },
            "kommentti" : {
                "kirjoittaja" : "Mauno",
                "sisalto" : "En pitanyt tasta kirjoituksesta."
            }
        ]
    }
}
```

Kuvio 6: Dokumentti, esimerkki JSON-formaatissa

Dokumenttivarastojen kannattajat väittävät, että tieto on semanttisesti hyvin samankaltaista, mutta syntaksi voi olla hyvin erilaista. Esimerkkinä reaalimaailman dokumentista on käyntikortti. Toisella on käyntikortissaa faksi, mutta se ei tarkoita että kaikkien käyntikorttien omistajien tulisi kirjoittaa käyntikorttiinsa: "Faksi : ei". Tätä dokumenttiparadigmaa käyttäen Internetissä liikkuva tieto on kokonaisia, reaalimaailmaan liittyviä dokumentteja, ei niinkään relaatioita, tauluja tai rivejä.

Dokumenttivarastot ovat siis NoSQL-aatteen yleiskäyttöisiä tietovarastoja. Ne ovat jokseenkin tehokkaita, mutta niillä on myös piirteitä relaatiotietokannan ominaisuuksista. Dokumenttivarastoilla voi olla esimerkiksi monipuolinen hakukieli.

Seuraavaksi esitellään kaksi esimerkkiä dokumenttivarastoista: MongoDB ja CouchDB. Niistä esitellään perusominaisuudet ja tärkeimmät käyttökohteet.

4.3.1 MongoDB

MongoDB on 2009 julkaistu dokumenttivarasto ja sitä pidetäänkin yhtenä tärkeimpänä NoSQL-liikkeen puolestapuhujana. MongoDB:n tallentaa tietoa skeemavapaassa, ei-relaationaalisessa muodossa käyttän JSON-formaattia. MongoDB on laajasti käytössä internetin suurilla toimijoilla kuten MTV, Disqus, Craiglist, Disney , Sourceforge,The Guardian, Forbes, The New York Times, bit.ly, GitHub, FourSquare ja niin edelleen (Boicea ym., 2012).

MongoDB:n tärkeimmät ominaisuudet ovat kokea tehokkuus, korkea saatavuus ja automaattinen skaalautuvuus. MongoDB käyttää transaktoiden prosessointiin BASE-ominaisuuksia, jolloin se pystyykin tarjoamaan nopeampaa tiedon käsittelyä verrattuna relaatiotietokantoihin. Hakurajapintana toimii suora ohjelmointikielen API, REST-rajapinta, sekä JavaScript-komentokehote.

MongoDB:n katsotaan soveltuvan tietokannaksi järjestelmiin, missä tietoa on erittäin paljon, mutta tiedon skeema vaihtelee. MongoDB:ä suositellaan kätettäväksi operationaalisen älykkyyden hallintaan, jossa reaaliaikaista dataa koostetaan näkyväksi tiedoksi käyttämällä koostamisfunktioita, MapReducea. Toisaalta MongoDB:n sopii myös tavanomaisen verkkokaupan tuotekatalogin tietovarastona, sillä sen joustava tietomalli antaa mahdollisuuden käyttää sitä myös perinteisissä käyttökohteissa, missä tyypillisesti käytetään relaatiotietokantaa (MongoDB, 2014).

4.3.2 CouchDB

CouchDB on dokumenttivarasto, joka tallettaa datan puolirakenteelliseen muotoon, dokumentiksi. CouchDB on toteutettu funktionaalisella Erlang -ohjelmointikielellä, joka keskittyy rinnakkaisuuteen. Sen avulla CouchDB lupaa täyttävänsä ACID-ominaisuudet (CouchDB, 2014), jota pidetään CouchDB:n ertyisominaisuutena. Vaikkakin ACID-ominaisuudet täyttyvät vain yhden dokumentin kontekstissa.

4.4 Sarakepohjaiset tietokannat (column)

Sarakepohjaiset tietokannat tallentavat dataa relaatiomallin tavalla tauluihin, mutta sarakepohjaisesti. Relaatiomallissa data tallennetaan rivien mukaan. Sarakepohjaisesti dataa tallennettaessa saavutetaan tehokkaita ominaisuuksia erityisiin käyttötarkoituksiin kuten tietovarastointi, CRM-järjestelmät (egl. content relations management) ja metatiedon tallentaminen kuten esimerkiksi kirjastokorttien katalogit (Stonebraker ym., 2005).

Saraketietokannassa samaan sarakkeeseen kuuluva data tallennetaan samaan sarakkeeseen. Näin sarakemallia käyttävä tietokanta on luontaisesti valmiina tarjoamaan tietoa. Esimerkiksi taulukossa 3 olevasta skeemasta pystytään hakemaan dataa, niin että tiedetään montako asiakasta asuu tietyssä kaupungissa tai tietyn postinumeron alueella. Hakuoperaatio on todella tehokas verrattuna relaatiotieto-

kantoihin, jossa koko taulun data tulee käydä läpi. Sarakepohjainen tietokanta saa suoraan haetun tiedon ajassa *O(1)*. Tämä antaa käytännöllisiä työkaluja suurten tietomäärien analyysiin. Tämän analyysin avulla voidaan jalostaa liiketoiminnalle hyödyllistä tietoa tai tieteellisessä tilastoinnissa tarvittavaa dataa (Stonebraker ym., 2005).

Rivimalli

ID	etunimi	sukunimi	katuosoite	postinumero	kaupunki
1	Tuomas	Soikkeli	Kauppakatu 5	40100	Jyväskylä
2	John	Doe	Laajavuorentie 10	40740	Jyväskylä
3	Mary	Doe	Laajavuorentie 10	40740	Jyväskylä
4	Matti	Meikäläinen	Pengerkatu 2	00500	Helsinki

Sarakemalli

ID	etunimi	sukunimi	katuosoite	postinumero	kaupunki
1	Tuomas	Soikkeli	Kauppakatu 5	40100	Jyväskylä
2	John	Doe	Laajavuorentie 10	40740	Jyväskylä
3	Mary	Doe	Laajavuorentie 10	40740	Jyväskylä
4	Matti	Meikäläinen	Pengerkatu 2	00500	Helsinki

Taulukko 3: Sarakepohjainen tietomalli verrattuna rivipohjaiseen tietomalliin

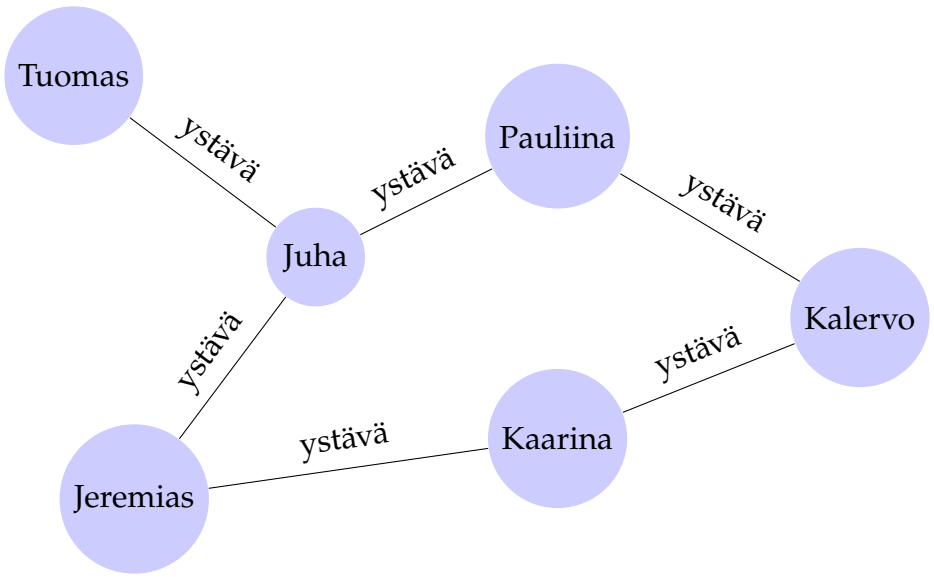
4.4.1 BigTable

Googlen käyttöön kehitetty BigTable käyttää tietomallinnaan moniulotteisesta, järjestettyä sanakirja -tietorakennetta, johon tallennetaan avain-arvo-pareja, jotka ovat indeksoitu käyttäen rivejä, sekä sarakkeita. Tietomalli mahdollistaa tehokkaan hakemisen sarakepohjaisesti tietovaraston tapaan tai perinteisesti rivin avulla. Näin saadaan erityisiä ominaisuuksia datan hakemiseen ja huomattavia tehonlisäyksiä verrattuna perinteiseen taulumalliin tässä käyttötarkoituksessa.(Padhy, Patra, & Satapathy, 2011)

BigTable on onnistunut tietokantatuote Googlen käyttöön ja sitä käyttää myös useat muut suuret yritykset ja organisaatiot. Internetin hakukoneiden indeksointi tallennetaan lähes yksinomaan käyttäen tehokasta BigTable:a. BigTable on helposti skaalattavissa, se prosessoi tehokkaasti dataa ja sen yksinkertainen ja joustava tietomalli ovat houkutteleva yhdistelmä (Chang ym., 2008).

4.5 Verkkotietokannat (graph)

NoSQL-tietokantojen neljäs malli on verkkotietokannat. Verkkotietokannoissa tieto tallennetaan verkkomallia käyttäen. Verkkomalli esittää datan verkon solmuina ja niiden välisinä suhteina. (Vicknair ym., 2010) Verkko eli graafi on tietojenkäsittelytieteissä tietomalli, joka koostuu joukosta yksittäisiä solmuja sekä niiden välillä olevista kaarista. Nämä kaaret kuvaavat verkon solmujen välisiä suhteita (kuva 7).



Kuvio 7: Esimerkki verkkoa käyttävästä tietomallista sosiaalisen median tietokantajärjestelmässä

Verkkotietokannat sopivat käyttötarkoituksiin missä tieto on useassa relaatiossa. Sosiaalinen media, tieteellinen laskenta, sisällönhallinta, suosittelujärjestelmät. Relaatiotietokannan useat JOIN-operaatiot ovat raskaita operaatioita, verkkotietokannassa ne ovat tavallinen operaatio johtuen tietomallista. Useita relaatioita, eli verkon kaaria sisältävä skeema on myös helposti hajautettavissa niin myöskin koostettavissa MapReducen avulla.

4.5.1 Neo4J

Neo4j on Neo Technology:n valmistama avoimen lähdekoodin verkkotietokanta, jonka tietomallina toimii luonnollisesti verkko. Pääominaisuutena Neo4j:ssä katsotaan olevan ACID-ominaisuuksien noudattaminen. Neo4j tallettaa massivisen määrn dataa ollessaan samalla horisontaalisesti skaalattavissa ja tiedon ollessa jatkuvasti kuitenkin saatavilla. Neo4j tarjoaa oman verkkomalli-hakukielen, jonka mainostetaan olevan tehokas ja luonnollista kieltä muistuttava kuten SQL-keili. Hakuja on mahdollista tehdä myös REST-rajapinnan yli tai suoralla Java API:lla.

5 YHTEENVETO

NoSQL-tietokannat eroavat suuresti relaatiotietokannoista ja eivät noudata relaatiomallia. NoSQL-tietokannoista puuttuu tarkoituksellisesti tiettyjä relaatiotietokantojen ominaisuuksia, jotta saavutetaan tehokkuutta ja skaalautuvuutta. NoSQL-tietokannat kuuluvat useisiin eri kategorioihin, joita on vaikea luokitella. Eri kategoiroiden NoSQL-tietokannoilla on moninaisia käyttötapauksia ja rajoitteita.

Avain-arvo-parivarastot keskittyvät tiedon tehokkaaseen käsittelyyn ja järjestelmän skaalautuvuuteen, ja usein ne tarjoavat vain rajallisen hakukielen tai -rajapinnan. Dokumenttivarastot ovat yleiskäyttöisempiä ja ne käyttävät hieman monimutkaisempia hakujärjestelmiä. Sarakepohjaiset tietokannat ovat suosittuja järjestelmiä tietovarastointiin ja analytiikkaan. Verkkotietokannat ovat verkkomallia käyttäviä tietokantoja, joihin tietoa voidaan tallentaa verkkoon. Tämä on luonnollinen malli tallentaa esimerkiksi sosiaalisen median dataa.

NoSQL-tietokantojen ominaispiirteitä ovat hajautettavuus, tehokkuus, yksinkertaisuus ja SQL-hakukielen puuttuminen. Näiden lisäksi NoSQL-tietokantojen oikeellisuumalli katsotaan olevan lähempänä BASE-ominaisuuksia kuin ACID-ominaisuuksia, mutta poikkeuksia on runsaasti.

Hajautettavuus mahdollistaa tiedon käsittelyn kuluttajatason tietokoneissa, joten järjestelmää on helppo laajentaa ilman suuria kustannuksia. *Tehokkuus* NoSQL-tietokannoissa johtuu käytettävästä oikeellisuusmallista, lopulta oikeellinen, joka mahdollistaa ACID-ominaisuuksista luopumisen. ACID-ominaisuudet aiheuttavat suuria lisäkustannuksia tiedonkäsittelylle. NoSQL tietokantojen *Yksinkertaisuus* johtuu siitä, että SQL-kielen katsotaan olevan monimutkainen hakukieli ja siitä luovuttaessa saavutetaan helpompia ja lyhyempiä, halvempia, järjestelmäkehityksen vaiheita. Toisaalta kehittäjät joutuvat toteuttamaan joitakin relaatiotietokantojen ominaisuuksia sovellustasolla, mikäli niitä tarvitaan. *Hakurajapintana* käytetään tyypillisesti REST-rajapintaa HTTP-protokollan avulla. Tämän lisäksi yleisiä ovat ohjelmointikielen oma API, jolloin myöskin saavutetaan yksinkertaisuutta ja tehokkuutta, kun hakukielen kaltaisia välikäsiä ei ole.

NoSQL-tietokannat ovat saaneet osakseen huomattavan määrän kritiikkiä. Tutkittua tietoa on vähän johtuen teknologian nuoresta iästä, joten osa käyttäjäsegmen-

tistä jättäytyy käyttämättä näitä teknologoita. Useat tutkijat ovat myös huomanneet, että NoSQL-tietokannat eivät ole vastaus kaikkiin tilanteisiin, vaan tietokanta tulee valita huolellisesti ja ei-relaatiotietokannat ovat yksi vaihtoehto muiden joukossa. Suuret yritykset kuten Google ja Amazon ovat valmistaneet omat järjestelmänsä käyttäen NoSQL-teknologoita, mutta se ei kerro, että kaikkien kehittäjien tulisi käyttää näitä tuotteita. Myöskän tehokkuus ei ole yksimielisesti NoSQL-tietokantojen ansiota. On argumentoitu ettei tehokkuus kärsi ACID-ominaisiuuksien toteuttamisesta vaan muista ominaisuuksista kuten esimerkiksi kirjanpidossa ja säikeiden hallinnassa. Uudet NewSQL-tietokannat tarjoavat relaatiotietokantojen ominaisuudet skaalattavassa ympäristössä, joten niiden kehittyessä esitetäänkin kysymys miksi enää käyttää NoSQL-tietokantoja.

LÄHTEET

Bailis, P., & Ghodsi, A. (2013, maaliskuuta). Eventual consistency today: Limitations, extensions, and beyond. *Queue*, *11*(3), 20:20–20:32. Lainattu saatavilla `http://doi.acm.org/10.1145/2460276.2462076` doi: 10.1145/2460276.2462076

Boicea, A., Radulescu, F., & Agapin, L. I. (2012). Mongodb vs oracle-database comparison. Teoksessa *Eidwt* (s. 330–335).

Brewer, E. (2000). *Towards Robust Distributed Systems.* (Puhe konferessissa ACM Symposium on the Principles of Distributed Computing, Portland, Oregon)

Cattell, R. (2011). Scalable SQL and NoSQL data stores. *ACM SIGMOD Record*, *39*(4), 12–27.

Chamberlin, D. D., & Boyce, R. F. (1974). SEQUEL: A structured English query language. Teoksessa *Proceedings of the 1974 ACM SIGFIDET workshop on Data description, access and control* (s. 249–264).

Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., ... Gruber, R. E. (2008). Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, *26*(2), 4.

Chu, C.-T., Kim, S. K., Lin, Y.-A., Yu, Y., Bradski, G., Ng, A. Y., & Olukotun, K. (2006). Map-reduce for machine learning on multicore. Teoksessa *Nips* (osa 6, s. 281–288).

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

CouchDB. (2014). Apache CouchDB 1.6 Documentation [Ohjelmiston käsikirja].

Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. *Communications of the ACM*, *51*(1), 107–113.

DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., ... Vogels, W. (2007). Dynamo: amazon's highly available key-value store. Teoksessa *Acm sigops operating systems review* (osa 41, s. 205–220).

Division, I. B. M. C. R., Lindsay, B., Selinger, P., Galtieri, C., Gray, J., Lorie, R., ... Wade, B. (1979). *Notes on distributed databases*.

Evans, E. (2009). *Eric Evans's Weblog.* Lainattu 4.4.2014, saatavilla `http://blog.sym-link.com/2009/05/12/nosql_2009.html`

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

Gray, J., ym. (1981). The transaction concept: virtues and limitations. Teoksessa *Vldb* (osa 81, s. 144–154).

Han, J., Haihong, E., Le, G., & Du, J. (2011). Survey on NoSQL database. Teoksessa *Proceedings of 2011 6th international conference on pervasive computing and applications (ICPCA)* (s. 363–366). IEEE Computer Society.

Hecht, R. (2011). NoSQL Evaluation. *International Conference on Cloud and Service Computing*.

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

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

McKinsey & Company. (2011). Big data: The next frontier for innovation, competition, and productivity.

Melton, J. (1993). *Understanding the new SQL: a complete guide*. The Morgan Kaufmann Series in Data Management Systems.

Melton, J. (1998). Database language SQL. Teoksessa P. Bernus, K. Mertins, & G. Schmidt (toim.), *Handbook on architectures of information systems* (s. 103-128). Springer Berlin Heidelberg.

MongoDB. (2014). MongoDB documentation (release 2.4.9) [Ohjelmiston käsikirja].

Padhy, R. P., Patra, M. R., & Satapathy, S. C. (2011). RDBMS to NoSQL: Reviewing some next-generation non-relational databases. *International Journal of Advanced Engineering Science and Technologies*, *11*(1), 15–30.

Pokorny, J. (2013). NoSQL databases: a step to database scalability in web environment. *International Journal of Web Information Systems*, *9*(1), 69–82.

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

Sarah, M. (2013). *Why you should never use MongoDB.* Lainattu saatavilla `http://www.sarahmei.com/blog/2013/11/11/why-you-should-never-use-mongodb/`

Seeger, M., & Ultra-Large-Sites, S. (2009). Key-value stores: a practical overview.

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2002). *Database system concepts* (osa 4). McGraw-Hill New York.

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

Stonebraker, M., Abadi, D. J., Batkin, A., Chen, X., Cherniack, M., Ferreira, M., ... others (2005). C-store: a column-oriented dbms. Teoksessa *Proceedings of the 31st international conference on very large data bases* (s. 553–564).

Strauch, C., Sites, U.-L. S., & Kriha, W. (2011). *NoSQL databases.* Lainattu saatavilla `http://www.christof-strauch.de/nosqldbs.pdf`

Strozzi, C. (1998). *NoSQL – A relational database management system.* Lainattu 5.4.2014, saatavilla `www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/nosql/Home%20Page`

Tiwari, S. (2011). *Professional NoSQL*. John Wiley & Sons.

Vicknair, C., Macias, M., Zhao, Z., Nan, X., Chen, Y., & Wilkins, D. (2010). A comparison of a graph database and a relational database: a data provenance perspective. Teoksessa *Proceedings of the 48th annual southeast regional conference* (s. 42).

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