Tuomas-Matti Soikkeli

NoSQL-tietokannat: perusominaisuudet ja luokittelu

JYVÄSKYLÄN YLIOPISTO
TIETOJENKÄSITITTELYTIETEIDEN LAITOS
2015

TIIVISTELMÄ

Soikkeli, Tuomas-Matti
NoSQL-tietokannat: perusominaisuudet ja luokittelu
Jyväskylä: Jyväskylän yliopisto, 2015, 31 s.
Tietojärjestelmätiede
Ohjaaja: Leppänen, Mauri

NoSQL-tietokannat ovat olleet eräitä 2000-luvun kiistellyimmistä teknologioista. Niiden käyttötarkoituksista tai hyödyllisyydestä ei ole yleisesti hyväksyttyä tutkimustietoa, johtuen NoSQL-tietokantojen kehittymättömyydestä ja nuoresta iästä. Tämän tutkielman tarkoituksena on selvittää, miten NoSQL-tietokantoja voidaan luokitella ja mitkä ovat niiden käyttökohteet. Tätä varten kerrotaan ensin, miten NoSQL-tietokannat ovat kehittyneet ja minkälaisia ominaispiirteiltä niillä on. Sitten kuvataan, miten NoSQL- tietokannat eroavat perinteisistä relaatiotietokannoista ja voidaanko relaatiotieto- kannat korvata NoSQL-tietokannoilla. Lopuksi esitetään NoSQL-tietokantoihin kohdistunutta kritiikkiä. Tutkimuksessa luokitellaan NoSQL-tietokannat luokitellaan neljään ryhmään: avain-arvopari-varastot, dokumenttivarastot, sarakepohjaiset tietokannat ja verkkotietokannat. Jokaisen tietomallin osalta esitellään tarkemmin tietomallin piirteet ja käyttökohteet sekä havainnollistetaan tietomallia esimerkein. Tutkimuksen tuloksia voidaan hyödyntää esimerkiksi yritysten teknologiavalintoihin, sillä ne vaikuttavat projektin onnistumiseen. Toisaalta tämä tutkimus toimii perehdytyksenä aihealueeseen jatkotutkimusta tehdessä.

Asiasanat: NoSQL, tietokanta, ei-relaationaalinen

ABSTRACT

Soikkeli, Tuomas-Matti
NoSQL-databases: basic features and classification
Jyväskylä: University of Jyväskylä, 2014, 31 p.
Information Systems, Candidates Thesis
Supervisor(s): Leppänen, Mauri

NoSQL-datastores have been one of the most controversial technologies of 20th century. Currently we have no eminent evidence that NoSQL-databases are useful and how they should be used. In this candidate's thesis NoSQL-databases are researched to categorize and identify the most common use cases for them. Secondly the author researches how NoSQL-databases have evolved and what are their main properties. At the same time, NoSQL-databases are compared how they differ from ordinary relational database management systems and are these relational databases replaceable with NoSQL-databases. Moreover, we study the critique 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. Enterprises can use this study to aid them with selection of technology. This can lead to successful software project. On the other hand researches can use this study to familiarize themselves to the area of knowledge.

Keywords: NoSQL, database, non-relational

KUVIOT

TAULUKOT

Sisältö

ESIPUHE

Tämä tutkielma on tehty LaTeX-ladontajärjestelmällä, jota suosittelen lämpimästi kaikille tutkielman kirjoittajille. Tutkimussuunnitelman kirjoittamisen aikaan Jyväskylän Yliopiston Tietojärjestelmätieteiden opiskelijoille oli tarjolla vain Microsoft Word -dokumenttipohja, jonka moninaiset ongelmat saivat minut harkitsemaan muita vaihtoehtoja. Tuloksena syntyi tutkielma, jonka olen julkaissut avoimena lähdekoodina. Varoituksen sanana voin sanoa, että siirtymä tavanomaisista tekstinkäsittelyohjelmista ei ole helppo. Opettelu vaatii runsaasti aikaa, kuten tämän tutkielman tekeminen osoitti.

Päädyin käyttämään tekstin ja koodin muokkaamiseen web-pohjaista, sekä ilmaista ShareLatex (http://www.sharelatex.com) -palvelua. ShareLatex-editori ja -kääntäjä suoraviivaistaa monia LaTeX:n monimutkaisuuksia. Lähdekoodi on saatavilla kokonaisuudessaan osoitteesta https://github.com/t-my/tuomas-soikkeli-candidates-thesis

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 aikaa vieviä operaatioita. Siten datamäärien kasvaessa eksponentiaalisesti on kehitetty ei-relaationaalisia tietokantajärjestelmiä. Näitä järjestelmiä kutsutaan NoSQL -tietokannoiksi. (Leavitt, 2010; Pokorny, 2013) 2000-luvulta alkaen on kehitetty NoSQL-tietokantoja perinteisten relaatiotietokantojen rinnalle. Näitä tietomalleja ovat esimerkiksi avain-arvo-parivarastot, dokumenttivarastot, sarakepohjaiset tietokannat ja verkkotietokannat. Myös muita tietomalleja käyttäviä järjestelmiä on runsaasti, mutta tässä tutkielmassa keskitytään näiden neljän tietomallin tutkimiseen. (Cattell, 2011)

NoSQL-tietokantojen nuoren iän myötä NoSQL-tietokannoista ei ole kattavaa tutkimustietoa. Tässä tutkielmassa tutkitaan näitä epäselvyyksiä seuraavien tutkimuskysymysten avulla:

- Mitä ovat NoSQL-tietokantojen perusominaisuudet?
- Miten NoSQL-tietokantoja voidaan luokitella?
- Miten NoSQL-tietokannat eroavat relaatiomalliin perustuvista tietokannoista?
- Mihin käyttötarkoitukseen NoSQL-tietokantoja voidaan käyttää?

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.

Tämän tutkimuksen avulla lukijaa saa selvän kuvan tietokantojen teoriapohjasta, NoSQL-tietokantojen tyypillisistä käyttökohteista sekä tavanomaisista ominaisuuksista.

Tämä tutkimus jakaantuu kolmeen sisältölukuun. Ensimmäisessä sisältöluvussa tutustutaan NoSQL-tietokantojen kehittymiseen johtaneisiin seikkoihin eli historiaan. Tämän lisäksi ensimmäisessä kappaleessa käydään läpi relaatiotietokantojen ja NoSQL-tietokantojen toiminnan teoriaa ja ominaisuuksia. Toisessa kappaleessa määritelmän jälkeen käydään läpi NoSQL-tietokantojen keskeisimmät ominaisuudet ja yleisimmät määrittelevät tekijät. Toisen kappaleen lopussa käsitellään myös kritiikkiä aiheeseen liittyen. Kolmannessa kappaleessa esitellään aluksi luokittelu, jonka jälkeen käydään läpi tietokantasovelluksia tietomalleittain.

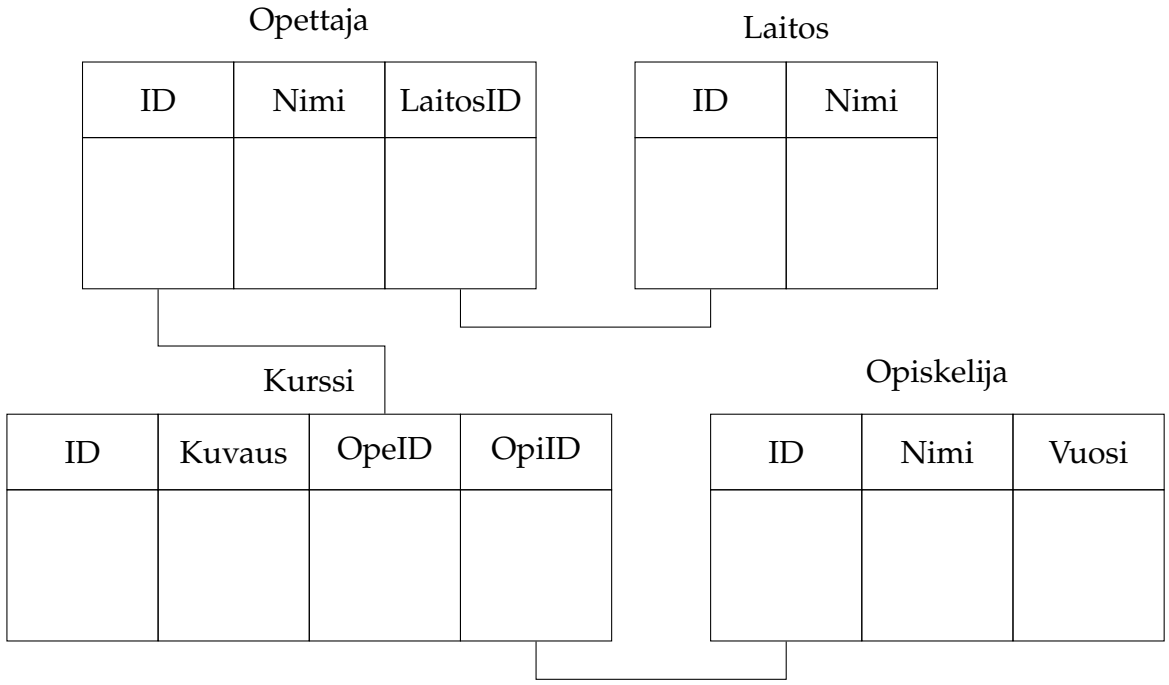
2 NOSQL-TIETOKANTOJEN TAUSTA

Tässä luvussa kuvataan NoSQL-tietokantojen kehittämiseen johtaneita syitä. Aluksi kerrotaan lyhyesti relaatiotietokannoista ja SQL:stä. Toiseksi käsitellään relaatiotietokannoille olennaisia ACID-ominaisuuksia. Kolmanneksi kerrotaan, kuinka tiedon valtava massa verkkoliikenteessä on muuttanut tiedonkäsittelyn ja tallentamisen vaatimuksia. Lopuksi esitellään CAP-teoreema ja sen pohjalta kehitetty BASE-oikeellisuusmalli, joka mahdollistaa tiedon tehokkaan tallentamisen ja käsittelyn hajautetussa ympäristössä.

2.1 Relaatiotietokannat ja SQL

Edgar Codd (1970) esitti tietomallin, jolla voidaan (silloisia valtamalleja, verkkomallia ja hierarkkista mallia) yksinkertaisemmalla ja fyysistä rakenteista riippumattomalla tavalla jäsentää tallentavaa tietoa. Tämä malli sai nopeasti suosiota ja johti useiden tutkimusprojektien (Chamberlin & Boyce, 1974) kautta SQL-kyselykielen (engl. Structured Query Language) syntyyn (Melton, 1993, s. 8-10). SQL-kielen standardoinnin (Melton, 1993, s. 3) jälkeen sitä onkin käytetty laajamittaisesti tiedon tallentamiseen. SQL:ään perustuvat relaatiotietokannat ovat tämän hetken suosituinpia tietokannan hallintajärjestelmiä.

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

Tietokanta sisältää suuren määrän tietoja, joita monet ohjelmat voivat samaan aikaa hakea ja päivittää. Jotta voidaan varmistaa se, etteivät samaan tietoon kohdistuvat operaatiot samaan aikaan suoritettuina tuota ongelmia, tarvitaan jokin samanaikaisten tapahtumien hallintamekanismi. Andreas Reuter ja Theo Härder julkaisivat vuonna 1983 artikkelin "Principles of Transaction-Oriented Databasse Recovery", joissa he aiempiin tutkimuksiin pohjautuen (Gray ym., 1981; Division ym., 1979) määrittelivät termin ACID, joka on lyhenne sanoista Atomicity, Consistency, Isolation ja Durability. Vastaavat suomenkieliset termit ovat atomisuus, eheys, eristyneisyys ja pysyvyys.

ACID tarkoittaa joukkoa ominaisuuksia, jotka takaavat tietokannan tapahtumien suorituksen luotettavuuden niin, että jokaisen tapahtuman suorituksen jälkeen pystytään varmistumaan, että halutut operaatiot ovat oikein suoritettu. Esimerkiksi siirrettäessä rahaa toisen käyttäjän tilille on varmistettava, että tapahtuma, jolla nostetaan rahaa toiselta tililtä ja pannaan vastaava summa toiselle tilille, on todella tapahtunut alusta loppuun. Mikäli tapahtuman suorituksen aikana sattuu järjestelmän kaatuminen, on palattava alkuperäiseen järjestelmän tilaan. Seuraavaksi annettaan edellä mainituille termeille lyhyet määritelmät (Härder & Reuter, 1983):

- **Atomisuus:** Tapahtuman aikana suoritetaan joko kaikki tietokantaoperaatiot tai ei ainuttakaan. Täten jos yksi osa tapahtumasta epäonnistuu, koko transaktio epäonnistuu.
- **Eheys:** Tapahtuman tulee jatkuvasti täyttää kaikki eheyssäännöt. Tapahtuma ei riko näitä sääntöjä, joten tietokanta pysyy yhtenäisessä tilassa tapahtuman suorituksen aikana.
- **Eristyneisyys:** Tapahtuma tulee voida suorittaan ikään kuin muita samanaikaisia tapahtumia ei olisikaan. Toisin sanoen muiden tapahtumien ei tule vaikuttaa sen suorittamiseen.
- **Pysyvyys:** Vahvistetun (commited) tapahtuman tekemät muutokset tietokannassa ovat lopullisia. Niihin eivät vaikuta mahdolliset virheet tai häiriöt.

ACID-ominaisuuksien varaan on relaatiotietokannoille kehitetty mekanismeja (esim. lukitusprotokollat, toipuminen), joilla pyritään turvaamaan tietokannan eheys kaikissa olosuhteissa. Seuraavassa alaluvussa esitellään tiedon määrän kasvaminen ja sen aiheuttamat kustannukset ACID-ominaisuudet täyttävässä järjestelmässä.

2.3 Tiedon valtava massa

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

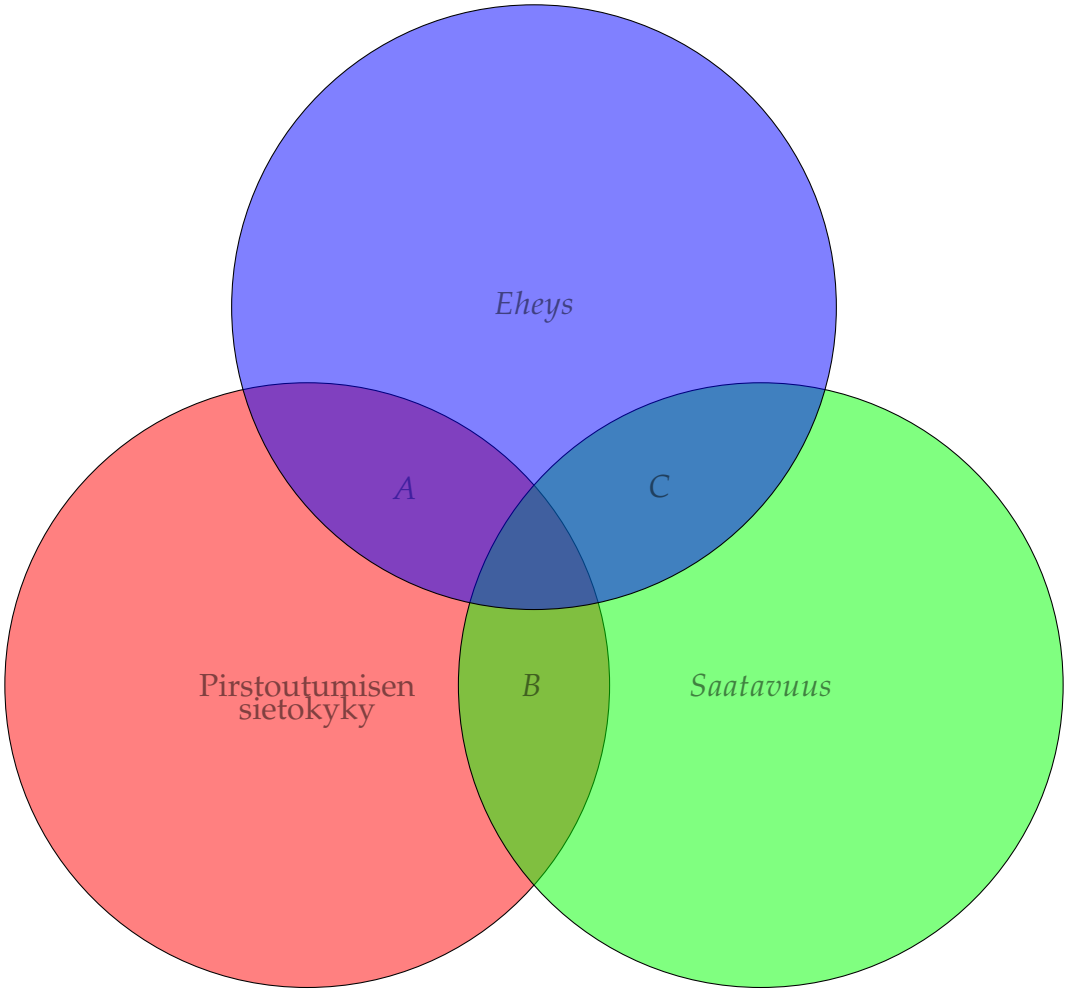
Näin ollen tarvitaan erittäin tehokkaita järjestelmiä, jotka pystyvät prosessoimaan suuria tietomääriä (Han, Haihong, Le, & Du, 2011). Näihin vaatimuksiin perinteiset SQL-tietokannat eivät pysty enää vastaamaan kunnolla. Koska SQL-tietokannat käsittelevät dataa ACID-ominaisuuksia noudattaen(Pokorny, 2013), jokaisen yksinkertaisen solun ja rivin tallentamisen yhteydessä pitää varmistua tietokannan tapahtumien luotettavuudesta. Tämä hidastaa merkittävästi käsittelyn nopeutta, kun dataa on runsaasti. Yhtäaikaisten tapahtumien luotettava käsittely ei ole enää mahdollista, sillä tiedon tallentamisen ja käsittelyn kustannukset alkavat kasvaa eksponentiaalisesti käsiteltävän tietomäärän kasvaessa suureksi.

Tietokantajärjestelmille on muodostunut uusi haaste. Tietoa on yksikertaisesti liikaa, jotta sitä voitaisiin käsitellä käyttäen ACID-ominaisuuksia. Seuraavaksi esitellään teoreema, joka auttaa väljentämään tietokannalle asetettuja vaatimuksia.

2.4 CAP-teoreema ja BASE-oikeellisuusmalli

Eric Brewer esitti vuonna 2000 (Brewer, s.a.) kuviossa 2 esitetyn CAP-teoreeman, jota kutsutaan myös Brewerin teoreemaksi. CAP-teoreeman mukaan, on olemassa kolme ominaisuutta, jotka vaikuttavat hajautetun järjestelmän suunnittelun, toteutukseen ja käyttöönottoon. Nämä ominaisuudet ovat eheys, saatavuus ja pirstoutumisen sietokyky eli CAP (engl. consistency, availability, partition tolerance).

Eheys (consistency) tarkoittaa, että tietokannan tosinteiden (replicates) tulee olla eheitä, jolloin hajautetun järjestelmän solmut näkevät kunkin tiedon samanlaisena. Saatavuus (availability) tarkoittaa, että palvelu (tietyn tiedon lukeminen tai kirjoittaminen) on saatavissa silloin, kun sitä tarvitaan. Pirstoutumisen sietokyky (partition tolerance) tarkoittaa, että verkon pirstoutuessa tiedonkäsittelyn tulee jatkua edelleen solmuissa (Celko, 2013, s. 10).



Kuvio 2: Hajautetun järjestelmän kolme ominaisuutta: 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. Gilbert ja Lynch (2002) todistivat Brewerin olleen oikeassa ja tätä pidetäänkin yleisesti hyväksyttävänä lähestymistapana tietokantojen tarkasteluun (mm. Vogels, 2009; Cattell, 2011; Pokorny, 2013).

Tietokannan valinnassa tulee ottaa huomioon CAP-teoreemassa esitellyt kolme ominaisuutta. Yhdestä niistä on luovuttava. Saatavuuden ja eheyden omaavat tietokannat (C) ovat usein perinteisiä relaatiotietokantoja. Data on aina saatavilla ja tiedon eheys taataan erilaisten tekniikoiden, kuten viite-eheyden avulla. Näillä tietokannoilla on kuitenkin hankaluuksia verkon pirstaloituessa. Eheyttä ja pirstoutumisen sietokykyä tarjoavat (A) tietokannat puolestaan eivät voi taata tiedon saatavuutta. Saatavuutta ja pirstoutumisen sietokyvyn tarjoavat tietokannat (B) saavuttavat lopulta oikeellisen tiedon. (Hurst, 2012)

BASE-oikeellisusmallin avulla pyritään takaamaan hajautetuille järjestelmille mahdollisimman korkea saatavuus ja tehokkuus. BASE-mallin esitti Vogels (2009), ja se nimi on lyhenne sanoista Basically Available, Soft state ja Eventual Consistency. Vastaavat suomenkieliset termit ovat: periaatteessa käytettävissä, ei aina oikeellinen ja lopulta oikeellinen. Periaatteessa käytettävissä tarkoittaa, että tiedon halutaan olevan käytettävissä aina vaikkei se olisikaan ajan tasalla. Ei aina oikeellinen tarkoittaa, että haettu tieto ei ole välttämättä oikea johtuen eri solmujen eri tiloista. Lopulta oikeellinen tarkoittaa, että vaikka tiedot eivät ole joka hetki eheitä, järjestelmä pitää huolen, että ne tulevat eheiksi jollakin aikavälillä. Se minkä ajan kulutta näin tapahtuu, riippuu järjestelmän muista ominaisuuksista kuten senhetkisen kuorman määrästä. (Celko, 2013, s.11)

BASE-ominaisuudet tarkoittavat käytännössä, että tietokantajärjestelmä saattaa palauttaa vanhentuneen arvon, jonka tulisi jo olla päivittynyt. Tämä ei aina ole kuitenkaan ongelmallista. Esimerkiksi sosiaalisen median datan ei tarvitse olla reaaliaikaisesti saatavilla, vaan riittää että tieto leviää palvelimien 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-TIETOKANNAT

Tässä luvussa esitellään NoSQL-tietokantoja, jotka pyrkivät ratkaisemaan jatkuvasti kasvavan tiedon määrän luomat haasteet. Aluksi kerrotaan NoSQL-käsitteen taustasta. Toiseksi esitellään NoSQL-tietokantojen keskeisiä ominaisuuksia. Kolmanneksi kerrotaan NoSQL-tietokantoja kohtaan esitetystä kritiikistä.

3.1 Tausta

NoSQL-käsitettä käytettiin ensimmäistä kertaa jo 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 1960-luvun lopulta asti, mutta uudemmat NoSQL-nimitystä kantavat tietokannat ovat 2000-luvulla kehitettyjä ei-relaationaalisia tietokantoja (Leavitt, 2010). NoSQL-tietokannat 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. Kuitenkin poikkeuksia löytyy runsaasti. Joissakin NoSQL-tietokannoissa käytetään joitakin SQL-kielen ominaisuuksia. NoSQL (engl. Not only SQL tai Not Relational) nimitys valittiin kuvastamaan tätä kahtia jaottelua SQL- ja NoSQL-tietokantojen välillä. Nimityksen osalta ei ole olemassa täydellistä konsensusta (Cattell, 2011), mutta NoSQL-tietokantojen ominaisuuksista löytyy useita yhteisiä tekijöitä, jotka esitellään seuraavaksi.

3.2 Skaalautuvuus

Yksi tärkeimmistä NoSQL-tietokantojen ominaisuuksista on skaalautuvuus. Sillä tarkoitetaan mahdollisuutta lisätä järjestelmään resursseja niin, että suoritusteho parantuu. Tällä skaalaamisella halutaan vastata käsiteltävän tietomäärän kasvamiseen. (Tiwari, 2011)

Järjestelmiä voidaan skaalata kahdessa suunnassa: horisontaalisesti ja vertikaalisesti. Vertikaalisesti skaalatessa yksittäisen laitteiston (esim. palvelimen) tehokkuutta kasvatetaan. Tämä voi tarkoittaa tietokonetta päivittämällä tai komponenttien vaihtamista. Tässä tapauksessa käytetään usein supertietokoneita, jotka ovat erittäin suuria investointeja. 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 tapahtumine oikeellisuuden takaavista ACID-ominaisuuksista kohti BASE-ominaisuuksia parantaakseen suoritustehoa (Tiwari, 2011).

3.3 Hajautettavuus

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

Myös pilvilaskentaa voidaan siis hyödyntää NoSQL-tietokantojen yhteydessä. Ne ovat helposti skaalattavissa, joten pilvilaskennan mahdollistama suuri laskentakapasiteetti on relaatiotietokantoihin verrattuna helposti valjastettavissa tehokkaaseen tiedon prosessointiin.

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

MapReduce (kuvio 3) on Googlen kehittämä ohjelmointiparadigma, joka mahdollistaa massadatan käsittelyn käyttäen yksinkertaisia tekniikkaa, jonka ansiosta työ on helppo hajauttaa. 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. MapReduce on ohjelma tai algoritmi joka käyttää NoSQL-tietokantoja. Joihinkin tietokantoihin se on sisäänrakennettu, mutta sen ei tarvitse olla osa NoSQL-tietokantaa. MapReduce käyttää hyväkseen NoSQL-tietokannan

helppoa hajautettavuutta tiedon prosessoinnissa. Iso tietomäärä on tehokasta hajauttaa laskettavaksi MapReducen avulla. Satojen tai tuhansien tietokoneiden verkko käsittelee tietoa rinnakain, jolloin suuri määrä tietoa saadaan käsitellyksi järkevässä ajassa. MapReduce on kaksivaiheinen algoritmi. Sen ensimmäinen vaihe on *map()*- funktio, joka hyväksyy parametrikseen arvoja ja antaa palautusarvona on avain-arvopareja. Nämä toimivat syötteenä *reduce()* -funktiolle, joka laskee yhteen tai muulla tavalla koostaa 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 yhdistää tuloksia ja lähettää sen eteenpäin kunnes kaikki tulokset ovat laskettu (Dean & Ghemawat, 2008).

MapReducea voidaan käyttää esimerkiksi sanojen esiintymisen selvittämiseen 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 (Dean & Ghemawat, 2008).

```
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 Skeemattomuus

Skeemattomuudella tarkoitetaan tiedon kopiointia moneen eri tauluun tai dokumenttiin. Jos tieto sijoitetaan yhteen dokumenttiin relaation sijasta, saadaan aikaan tehokas hakuoperaatio, sillä tietoa ei tarvitse hakea monesta eri paikasta. Toisaal-

ta tietoa joudutaan monistamaan. Näin tiedon kokonaismäärä kasvaa ja levytilaa käytetään enemmän. Tiedon eheyden ylläpitäminen skeemattomassa tietomallissa on vaikeaa. Eheyden ylläpitämisestä tulee raskas operaatio, sillä päivityksen myötä joudutaan käymään useita eri dokumentteja läpi yhden sijasta. Usein NoSQL-tietokannoissa ei taata tiedon eheyttä, vaan monessa eri dokumentissa olevaa tietoa joudutaan ylläpitämään ohjelmatasolla (Cattell, 2011).

Relaatiotietokannoista poiketen skeemattomassa tietokannassa tietoa denormalisoimaan, jotta saavutetaan nopeampia hakuja. Denormalisoinnilla tarkoitetaan, että tieto tallennetaan samaan dokumenttiin/tauluun. Relaatiotietokannassa tietoa tallennetaan skeeman määräämällä tavalla. Tämän lupauksen puuttuessa, voidaan tietoa palauttaa ilman rajoitteita tiedon muotoon liittyen (Vajk, Feher, Fekete, & Charaf, 2013).

Skeemattomuudesta ja denormalisoinnista johtuen liitos (JOIN) -operaatioita ei tyypillisesti tueta NoSQL-tietokannoissa. Liitos -operaatioita kuitenkin tarvitaan esimerkiksi monesta moneen relaatiossa. Tällöin liitos toteutetaan tietokantajärjestelmän sijaan ohjelmatasolla. (Pokorny, 2013).

Skeemattoman tietokannan joustavuus on NoSQL-tietokantojen yksi keskeisimmistä ominaisuuksista. Kehittäminen on helppo aloittaa ja nähdään nopeasti tuloksia ilman pitkäkestoista tietokannan määrittelyä. Tehokkaiden hakujen kehittäminen on suoraviivaista, mutta tiedon eheys joudutaan kehittämään itse ohjelmatasolla, mikäli se on ohjelman vaatimuksena. Toisaalta monimutkaisten hakujen toteuttaminen on vaikeaa ilman SQL-hakukieltä (Leavitt, 2010).

3.5 Rajapinta ja kyselykielet

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 asiakasohjelmat kommunikoivat tietokannan kanssa, jolloin HTTP-kutsukuorma voidaan tasata ja tulokset voidaan tallentaa välimuistiin. (Hecht, 2011)

3.6 Kritiikki

NoSQL-tietokantoja on käytössä useimmilla suurimmilla ICT-alan toimijoilla kuten Googlella, Amazonilla ja LinkedIn:llä, mutta sille löytyy myös vastustajansa. Vastustajat argumentoivat, että uusilla teknologiolla on aina kehittymisvaihe, jolloin sitä ei kannata käyttää epävakaudesta ja teknologian nuoresta iästä johtuen. NoSQL-

tietokantojen uskotaan olevan juuri tässä vaiheessa. Kattavaa määrää käyttötapauksia ei ole käyty läpi tai ohjelmistosta saattaa löytyä vielä virheitä, jotka tuotantoympäristössä aiheuttavat merkittävää haittaa liiketoiminnalle. (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) selvittää blogissaan miksi Diaspora, yhteisörahoitettu hajautettu sosiaalinen verkko, joutui ongelmiin valittuaan MongoDB:n relaatiotietokannan sijaan. Hylätessään relaationaalisen tiedonkäsittelyn Diasporan kehittäjät joutuivat uudelleen kehittämään osan relaatioalgebran ominaisuuksista ja toteuttamaan ne sovelluskerroksella. Yhdeksän kuukauden kehittämisen jälkeen he tekivät päätöksen rakentaa koko tietokantajärjestelmän uudelleen käyttäen 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, onko NoSQL-tietokantojen päälle sovelluksen rakentaminen vain matkaa kohti relaatioalgebran uudelleenkeksimistä.

Boicea, Radulescu, ja Agapin (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.

Viime vuosina on kehitetty uusia SQL-kieleen perustuvia järjestelmiä, jotka ovat horisontaalisesti skaalattavissa. NoSQL-tietokantojen suurin kehittävä voima on 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 skaalautuvuutta. Näitä tietokantoja kutsutaan yhteisellä nimellä NewSQL-tietokannat. NoSQL:n vastustajat ovat ilmaisseet, että NewSQL-tietokannat ratkaisevat relaatiotietokantojen puutteet ja niillä on siltikin mahdollista taata ACID-ominaisuudet. (Pokorny, 2013; Stonebraker, 2010).

Voidaankin esittää kysymys ovatko NoSQL-tietokannat vain kehittymättömiä tietokantoja, jotka ovat suurten yritysten ratkaisuja erityisiin ongelmiin? Esimerkiksi Googlen BigTable tyydyttää internetin suosituimman hakukoneen tietojenkäsittelyn vaatimukset. Se ei ole siis yleishyödyllinen ja tyypillisen ohjelmistoarkkitehtuurin käyttöön sopiva tietokanta. (Cattell, 2011)

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

4 NOSQL-TIETOKANTOJEN LUOKITTELU JA ESIMERKKEJÄ

Tässä luvussa kuvataan NoSQL-tietokantoja. Aluksi esitellään yksinkertainen tietomallin mukainen luokittelu. Toiseksi käsitellään kutakin luokkaa hieman tarkemmin ja kuvataan kustakin luokasta 1-2 esimerkki.

4.1 Luokittelu

NoSQL-tietokannoille on esitetty hyvin monenlaisia luokitteluja johtuen niiden nuoresta iästä ja ominaisuuksien monimuotoisuudesta. Tudorica ja Bucur Tudorica ja Bucur (2011) esittivät, että NoSQL-tietokannat voidaan luokitella kvalitatiivin tai kvantitatiivisin perustein. Kvalitatiivinen luokittelu keskittyy mittaamaan lähinnä suorituskykyä eri tuotteiden välillä, kun taas kvalitatiivinen luokittelu pyrkii erottelemaan eri NoSQL-tietokannat vaatimusten tai ominaisuuksien perusteella. Tässä tutkielmassa ollaan kiinnostuneita tietomalliin perustuvasta luokittelusta. Useat NoSQL-tietokannat ovat hybridejä, eli niissä on useamman luokan ominaispiirteitä.

Tässä tutkielmassa tutkittavat tietokannat luokitellaan kvalitatiivisesti neljään tyyppiin tietomallin perusteella: avain-arvo-varasto (key-value store), sarakevarasto (column based databases tai column family databases), dokumenttivarasto (document store) ja verkkotietokanta (graph database). Taulukossa 1 on esitetty NoSQL-tuotteita luokiteltuna neljään luokkaan. Taulukossa on esitetty myös tyypillisiä käyttökohteita. Tietokantojen luokitteluun on käytetty `http://nosql-database.org/` -sivustoa, jonne on kerätty eri valmistajien NoSQL-tuotteita vuodesta 2009 lähtien.

Seuraavaksi esitellään luokitellut tietomallit omissa alaluvuissaan 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 hajautus 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 1: NoSQL-tietokantojen luokittelu tietomallin ja käyttökohteiden mukaan

4.2 Avain-arvo-varasto

Yksinkertaisin NoSQL-tietomalli käyttää tiedon tallentamiseen avain-arvo-pareja (kuvio 4). Kaikki data tallennetaan ja haetaan avaimen avulla. Yksinkertaiset avain-arvo-parit voidaan esittää esimerkiksi JSON-notaation (engl. JavaScript Object Notation) avulla. Samalla osoittimella tehty haku palauttaa aina 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-varastojen käyttäminen on yleensä hyvin yksinkertaista (kuvio 5), sillä erillistä hakukieltä, kuten SQL-kieltä ei käytetä. Hakuja voidaan tehdä vain avaimen avulla. Joissakin tapauksissa arvo voi olla numeerinen, jonka perusteella

hakuja tehdään. Koska arvoihin perustuen 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 etunimi = get("etunimi");
8  var sukunimi = get("sukunimi");
9  var email = get("email");
10
11 // Tulostaa Tuomas-Matti Soikkeli tsoikkeli@gmail.com
12 tulosta(etunimi + " " + sukunimi + " " + email);
```

Kuvio 5: Avain-arvo-varaston käyttö, pseudokieli

Kuviossa 5 tehdään aluksi kolme kirjoitus, eli put() operaatiota. Ensimmäinen parametri on avain ja toinen arvo. Seuraavaksi sama tieto haetaan get() operaatiolla, jolloin funktiolle annetaan parametrina avain. Lopuksi tulostetaan muuttujien avulla merkkijono.

Dynamo on Amazonin suunnittelema ja toteuttama avain-arvo-varastoon perustuva tietokanta. Amazon kehitti Dynamoa omiin tarpeisiinsa ja lopulta siitä julkaistiin artikkeli Dynamo: Amazon's Highly Available Key-value Store (DeCandia ym., 2007). Dynamo on ollut pohjana useille uusille NoSQL-tietokannoille. Dynamolla voidaan sanoa olevan valtavat 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, järjestelmän kaatumisesta selviävä, skaalattava tietokanta, joka täyttää halutut vaatimukset.

4.3 Dokumenttivarasto

Dokumenttivaraston mukaiseen tietomalliin perustuvat tietokannat tallentavat datan dokumentteihin, jotka on järjestetty avain-arvopari-varaston tapaan osoittimen avulla. Dokumenttitietokannat ovat kuten avain-arvo-varastoja, mutta arvona voidaan tallentaa toissijaisia osoittimia, listoja tai muita monimutkaisempia tietorakenteita. Toisin sanoen dokumenttivarastot voivat tallentaa vapaamuotoista, ei-relaationaalista dataa, dokumentteja (kuvio 6). Dokumenttivarastot eivät keskity korkeisiin

luku- ja kirjoitusnopeuksiin vaan tarjoavat kattavan tavan tallentaa tiedon valtavaa massaa, ilman ennalta määriteltyä rakennetta eli skeemaa. Hakuominaisuudet ovat usein monipuoliset, koska toisin kuin avain-arvo-varastoissa, dataa voidaan hakea avaimen lisäksi arvojen perusteella.

```
{
    "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

Kuviossa 6 on esimerkki dokumentista, joka on esitetty JSON-formaatissa. Dokumentti on esitys blogimerkinnästä, joka koostuu yksinkertaisista avain-arvo-pareista, taulukoista (array) ja hakurakenteista (map). Eri tietotyyppien arvona voi olla lisää tietotyyppejä. Dokumentin rakennetta ei ole rajattu eksplisiittisesti skeeman avulla, vaan rakenne on avoin. Rakennetta voidaan muokata myös ajon aikana. Usein kuitenkin dokumentin rakenne määritellään sovelluskerroksella. Dokumenttiin liittyvä tieto kirjoitetaan suoraan dokumenttiin, eikä esimerkiksi toiseen dokumenttiin osoittimen avulla.

Dokumenttivarastojen kannattajat väittävät, että tieto on semanttisesti hyvin samankaltaista, mutta syntaksi voi olla hyvin erilaista. Esimerkkinä reaalimaailman dokumentista on käyntikortti. Toisella on käyntikortissaan faksi, mutta se ei tarkoi-

ta 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 yleiskäyttöisiä tietovarastoja. Ne ovat jokseenkin tehokkaita, mutta niillä on myös piirteitä relaatiotietokannan ominaisuuksista. Dokumenttivarastoilla voi olla myös monipuolinen kyselykieli. Tämän lisäksi on olemassa dokumenttivarastoja jotka toteuttava ACID-ominaisuudet. Yksi näistä on CouchDB, joka esitellään myöhemmin tässä luvussa.

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

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

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

MongoDB:n katsotaan soveltuvan tietokannaksi järjestelmiin, jossa tietoa on erittäin paljon, mutta tiedon skeema vaihtelee. MongoDB:ä suositellaan käytettäväksi operationaalisen älykkyyden hallintaan. Se tarkoittaa tiedon koostamista suurista tietovirroista reaaliaikaisesti. Tietovirroista koostetaan näkymiä, jotka antavat yritykselle tietoa päätöksenteon tueksi. Reaaliaikaista dataa koostetaan näkyväksi tiedoksi käyttämällä MapReduce:ia. 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).

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

4.4 Sarakevarasto

Sarakepohjaiset tietokannat tallentavat dataa relaatiomallin tavalla tauluihin, mutta sarakepohjaisesti. Relaatiomallissa data tallennetaan rivien mukaan. Sarakepohjaisesti dataa tallennettaessa parannetaan tehokkuutta, joka on tärkeää tietovarastoinnissa, CRM-järjestelmissä (egl. customer relationships maangement), ja metatiedon

tallentamisessa (esimerkiksi kirjastokorttien katalogit). (Stonebraker ym., 2005)

Sarakepohjaisessa tietokannassa samaan sarakkeeseen kuuluva data tallennetaan samaan sarakkeeseen. Näin sarakemallia käyttävä tietokanta on luontaisesti valmiina tarjoamaan tietoa. Esimerkiksi kuviossa 7 olevasta skeemasta pystytään hakemaan dataa niin, että tiedetään, montako asiakasta asuu tietyssä kaupungissa tai tietyn postinumeron alueella. Hakuoperaatio on tehokas verrattuna relaatiotietokantoihin, jossa koko taulun data tulee käydä läpi. Tämä antaa käytännöllisiä työkaluja suurten tietomäärien analyysiin. Tämän analyysin avulla voidaan jalostaa liiketoiminnalle hyödyllistä tietoa tai tieteellisessä tilastoinnissa. (Stonebraker ym., 2005)

Rivimalli

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

Sarakemalli

ID
1
2
3
4

etunimi
Tuomas
John
Mary
Matti

sukunimi
Soikkeli
Doe
Doe
Meikäläinen

katuosoite
Kauppakatu 5
Laajavuorentie 10
Laajavuorentie 10
Pengerkatu 2

postinumero
40100
40740
40740
00500

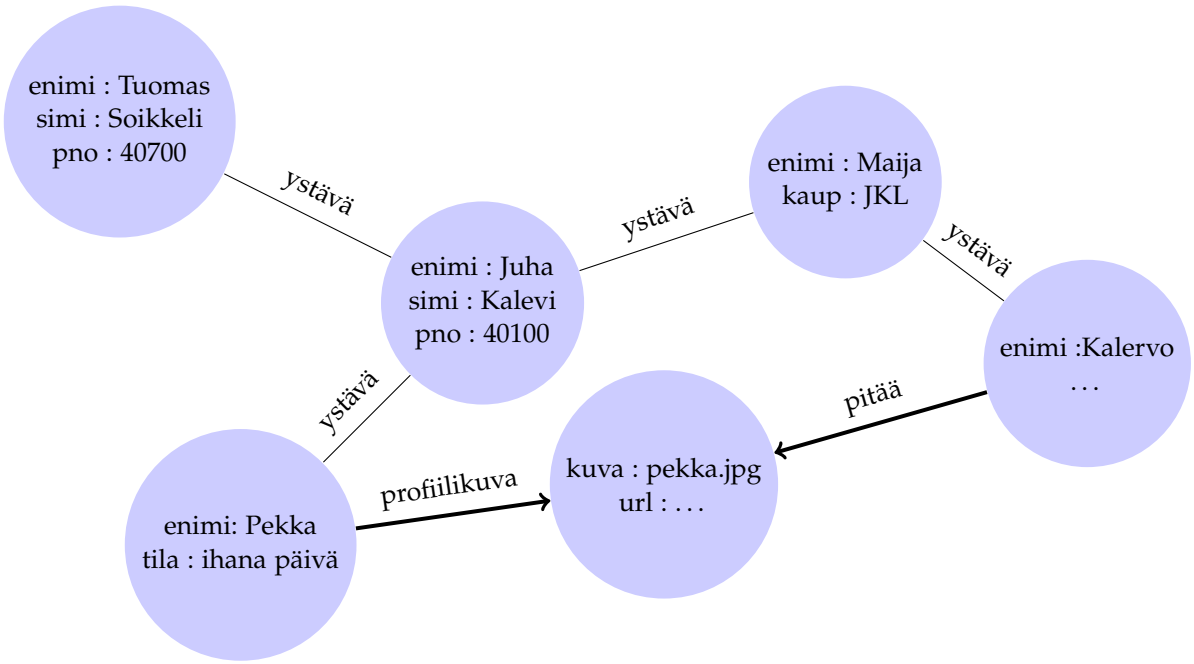
Kuvio 7: Sarakepohjainen tietomalli verrattuna rivipohjaiseen tietomalliin

Googlen käyttöön kehitetty BigTable käyttää tietomallinnaan moniulotteisesta, järjestettyä sanakirja -tietorakennetta, johon tallennetaan avain-arvo-pareja, jotka on 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.(Padhy, Patra, & Satapathy, 2011)

BigTable-tietokantaa käyttää Googlen lisäksi myös useat muut suuret yritykset ja organisaatiot. Internetin hakukoneiden indeksointi tallennetaan lähes yksinomaan käyttäen BigTable-tietokantaa. BigTable on helposti skaalattavissa, se käsittelee tehokkaasti dataa ja sen yksinkertainen ja joustava tietomalli ovat houkutteleva yhdistelmä (Chang ym., 2008).

4.5 Verkkotietokanta

Neljäs NoSQL-tietokantojen tietomalliluokka on verkkotietokanta. Verkkotietokannoissa tieto tallennetaan verkkomallia käyttäen. Verkkomalli (kuva 8). esittää datan verkon solmuissa ja niiden välisissä suhteissa. Verkko on tietojenkäsittelytieteissä tietomalli, joka koostuu joukosta yksittäisiä solmuja sekä niiden välillä olevista kaarista. Nämä kaaret kuvaavat verkon solmujen välisiä suhteita. (Vicknair ym., 2010)



Kuvio 8: Esimerkki verkkoa käyttävästä tietomallista sosiaalisen median tietokantajärjestelmässä

Verkkotietokannat sopivat sellaisiin tilanteisiin missä tieto on verkottunutta eli tieto paljolti relaatiossa. Tämä tarkoittaa että tietoalkiot liittyvät toisiinsa verkon kaaren avulla. Esimerkkejä verkottuneesta tiedosta on sosiaalinen media, sisällönhallinta, suosittelujärjestelmät. Verkkotietokannat ovat hyödyllisiä myös tieteellisessä laskennassa. Jos tietokannan rakenteeseen kuuluu paljon monesta-moneen relaatioita, relaatiotietokannassa käytetään liitostauluja. Liitostaulut liitetään yhteen viiteavaimen avulla. Relaatiotietokannan useat liitokset (join) ovat raskaita, joita pyritään nopeuttamaan denormalisoimalla skeemaa. Verkkotietokannassa liitos on tavallinen operaatio. Jokainen verkon solu sisältää listan suhteistaan toisiin verkon soluihin. Nämä suhteet ovat järjestetty tyypin suunnan ja tyypin mukaan. Liitosta vastaava operaatio hakee tämän solun suhdetaulukosta lista, jolloin sillä on suora yhteys siihen liittyviin alkioihin. Tällöin tarve usean eri taulun läpikäyntiin häviää. Useita suhteita, eli verkon kaaria sisältävä skeema on myös helposti hajautettavissa ja koostettavissa MapReducen avulla.

Neo4j on Neo Technology -nimisen yrityksen valmistama avoimen lähdekoodin verkkotietokanta. Pääominaisuutena Neo4j:ssä katsotaan olevan ACID-ominaisuuksien noudattaminen. Neo4j tallettaa massiivisen määrän dataa, mutta samalla takaa horisontaalisesti skaalattavissa ja tiedon jatkuvan saatavilla olon. Neo4j tarjoaa oman kyselykielen, jonka mainostetaan olevan tehokas ja luonnollista kieltä muistuttava kuten SQL-kieli. Hakuja on mahdollista tehdä myös REST-rajapinnan yli tai suoralla Java API:lla.

5 YHTEENVETO

Tämän tutkielman tarkoitus on NoSQL-tietokantojen luokittelu ja perusominaisuuksien kuvaus. Tutkimuskysymykset olivat:

- Mitä ovat NoSQL-tietokantojen perusominaisuudet?
- Miten NoSQL-tietokantoja voidaan luokitella?
- Miten NoSQL-tietokannat eroavat relaatiomalliin perustuvista tietokannoista?
- Mihin käyttötarkoitukseen NoSQL-tietokantoja voidaan käyttää?

NoSQL-tietokantojen ominaispiirteitä ovat skaalautuvuus, hajautettavuus, skeemattomuus sekä SQL-kielen puuttuminen. Skaalautuvuus on ominaisuus, jonka avulla järjestelmään voidaan lisätä resursseja, jotta voidaan käsitellä tietoa tehokkaammin. Tietomäärä kasvaa jatkuvasti, jolloin tarvitaan yksinkertainen tapa lisätä järjestelmään resursseja. NoSQL-tietokannat skaalautuvat usein horisontaalisesti, jolloin voidaan käyttää tavanomaisia ja halpoja komponentteja. 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.

NoSQL-tietokannat voidaan luokitella usealla eri tavalla. Tässä tutkielmassa ne luokitellaan tietomallin perusteella neljään eri luokkaan: avain-arvo-varastot, dokumenttivarastot, sarakevarastot ja verkkotietokannat. Avain-arvo-varastot keskittyvät tiedon tehokkaaseen käsittelyyn ja järjestelmän skaalautuvuuteen, ja usein ne

tarjoavat vain rajallisen hakukielen tai -rajapinnan. Usein avain-arvo-varastot ovat välimuistina toimivia, erittäin nopeita, muistinvaraisia ja yksinkertaisia tietovarastoja. Dokumenttivarastot ovat yleiskäyttöisempiä ja ne käyttävät hieman monimutkaisempia hakujärjestelmiä. Dokumenttivarastoissa nähdään myös usein relaatiotietokannasta tuttuja ominaisuuksia, kuten liitosoperaatioita. Jopa SQL-hakukieltä on käytetty. Myös tiedon oikeellisuutta ylläpitäviä ominaisuuksia on tuettu. Sarakepohjaiset tietokannat ovat suosittuja järjestelmiä suuren tietomäärän varastointiin ja analytiikkaan. Verkkotietokannoissa tieto varastoidaan nimensä mukaisesti sarakkeisiin, joka mahdollistaa rinnakkainlaskennan sekä tiedon koostamisen tehokkaasti esimerkiksi Map-Reduce -ohjelmointiparadigmalla. Verkkotietokannat ovat verkkomallia käyttäviä tietokantoja, joihin tietoa voidaan tallentaa verkkoon. Tietomallissa tieto linkittyy toisiinsa verkon kaarien avulla. Tämä on luonnollinen malli tallentaa esimerkiksi sosiaalisen median dataa, jossa linkittynyttä tietoa on paljon.

NoSQL-tietokannat ovat saaneet osakseen huomattavan määrän kritiikkiä. Tutkittua tietoa on vähän johtuen teknologian nuoresta iästä, joten osa käyttäjäsegmentistä jättäytyy käyttämättä näitä teknologiota. 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-teknologioita, mutta se ei kerro, että kaikkien kehittäjien tulisi käyttää näitä tuotteita. Myöskään tehokkuus ei ole yksimielisesti NoSQL-tietokantojen ansiota. On argumentoitu, ettei tehokkuus kärsi ACID-ominaisuuksien 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.

Avain-arvo-varastoja voidaan käyttää välimuistina, tiedon hajautukseen, viestijonoina eli avain-arvo-varastot ovat hyödyllisiä tehokkuutta ja pientä vasteaikaa vaativissa tehtävissä. Tällaisia ovat esimerkiksi reaaliaikaiset palvelut, jossa tieto välitetään perille mahdollisimman nopeasti. Dokumenttivarastot ovat yleiskäyttöisiä NoSQL-tietokantoja, ja niiden kerrotaan korvaavan relaatiotietokannat. Dokumenttivarastoja käytetään yksinkertaisten sovellusten tietokannaksi, mutta ACID-ominaisuuksien puuttumisen vuoksi niitä ei kannata käyttää kriittisten palveluiden perustana. Sarakevarastot ovat vakiinnuttaneet asemansa tietovarastoinnin ja analytiikan alalla. Suuret yritykset käyttävät lähes yksinomaan sarakavarastoja suurten tietomassojen tallentamiseen. Tallentamisen jälkeen tietoa voidaan tarkasti analysoida koostamalla suuresta tietomäärästä tärkeää tietoa yrityksen päätöksenteon avuksi.

Tätä tutkimusta voidaan hyödyntää tehdessä teknologiavalintoja uusiin ohjelmistoprojekteihin. Myöskin vanhan teknologian päivittämisen yhteydessä kannattaa ajatella NoSQL-tekonologoita. Tätä tutkimusta voidaan käyttää myös tutkimuksen perustana ja perustiedon saantiin.

NoSQL-tietokantojen kehitys on nopeaa, sillä jo tämän tutkielman kirjoittamisen aikana on syntynyt useita uusia tietokantoja, jotka voivat vaikuttaa tämän tutkielman tutkimustuloksiin. Tässä tutkimuksessa tehtiin ainoastaan tietomallin perusteella luokittelu. Jatkotutkimusta tarvitaan, miten NoSQL-tietokantoja voidaan luokitella muilla tavoin. Yksi tärkeä jatkotutkimusaihe onkin, tarvitaanko todella ACID-ominaisuuksia ja pystytäänkö tietoa käsittelemään nopeasti samalla taaten ACID-ominaisuudet. Uudet toimijat alalla mainostavat pystyvänsä tähän.

LÄHTEET

Bailis, P., & Ghodsi, A. (2013). Eventual consistency today: Limitations, extensions, and beyond. *Queue*, *11*(3), 20–32.

Boicea, A., Radulescu, F., & Agapin, L. I. (2012). MongoDB vs Oracle–Database Comparison. Teoksessa *Proceedings of the 2012 Third International Conference on Emerging Intelligent Data and Web Technologies* (s. 330–335). IEEE Computer Society.

Brewer, E. (s.a.). *Towards Robust Distributed Systems.* Lainattu 14.6.2015, saatavilla `https://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf`

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

Celko, J. (2013). *Joe Celko's complete guide to NoSQL: What every SQL professional needs to know about non-relational databases*. Newnes.

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*, *26*(2), 4–26.

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).

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*.

Hurst, N. (2012). *Visual guide to NoSQL systems.* Lainattu 14.6.2015, saatavilla `http://blog.nahurst.com/visual-guide-to-nosql-systems`

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.

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.

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.

Tudorica, B. G., & Bucur, C. (2011). A comparison between several nosql databases with comments and notes. Teoksessa *Roedunet international conference (roedunet), 2011 10th* (s. 1–5).

Vajk, T., Feher, P., Fekete, K., & Charaf, H. (2013). Denormalizing data into schema-free databases. Teoksessa *Cognitive infocommunications (coginfocom), 2013 ieee 4th international conference on* (s. 747–752).

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.