hyväksymispäivä arvosana

arvostelija

NoSQL-keskusmuistitietokantajärjestelmät

Tuomo Jokimies

Helsinki 09.03.2012

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

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

Tekijä – Författare – Author

Tuomo Jokimies

Työn nimi – Arbetets titel – Title

NoSQL-keskusmuistitietokantajärjestelmät

Oppiaine – Läroämne – Subject

Tietojenkäsittelytiede

Työn laji – Arbetets art – Level	Aika – Datum – Month and year	Sivumäärä – Sidoantal – Number of pages
Seminaarityö	09.03.2012	10 sivua + 2 liitesivua

Tiivistelmä – Referat – Abstract

Web 2.0 ja sosiaalisen median palveluiden suuri suosio ovat luoneet tarpeen uudenlaisille NoSQL-tietokantajärjestelmille. NoSQL-tietokantajärjestelmät eivät pohjaudu perinteiseen relaatiotietokantamalliin, vaan ne voivat olla esimerkiksi avain-arvo- tai dokuementtipohjaisia.

Keskusmuistissa toimivat NoSQL-tietokantajärjestelmät ovat nopeita ja yksinkertaisia käyttää, mutta ne voivat tukea myös monimutkaisempia tietorakenteita ja tietotyyppejä. NoSQL-keskusmuistitietokantajärjestelmiä onkin käytetty menestyksekkäästi muun muassa suurissa sosiaalisen median palveluissa.

Käyttämällä NoSQL-keskusmuistitietokantajärjestelmiä oikeissa tilanteissa voidaan saavuttaa merkittävästi tehokkaampia ja skaalautuvampia sovelluksia.

ACM Computing Classification System (CCS):
H.2 [Database Management]
H.3 [Information Storage and Retrieval]

Avainsanat – Nyckelord – Keywords

NoSQL, keskusmuisti, tietokantajärjestelmät, Memcached, Project Voldemort, Redis, Terrastore

Säilytyspaikka – Förvaringställe – Where deposited

Muita tietoja – Övriga uppgifter – Additional information

Sisältö

1 Johdanto

Web 2.0 ja sosiaalisen median palveluiden suuri suosio ovat luoneet tarpeen uudenlaisille tietokantajärjestelmille [Päi10]. Suuria tietomääriä on mahdotonta säilyttää ja käsitellä yhdellä palvelimella, vaan säilytys ja laskenta pitää hajauttaa [RuB11].

Uusien tietokantajärjestelmien tarpeiden myötä myös hyvän tietokantajärjestelmän määrittelevät periaatteet ovat muuttuneet. BASE-periaatteet ovat korvanneet perinteisesti hyvän tietokantajärjestelmän määrittelevät ACID-periaatteet [Päi10]. Luvussa 2 käsitellään tutkielman kannalta olennaisia käsitteitä ja periaatteita.

Perinteisten relaatiotietokantamalliin pohjaavien tietokantajärjestelmien rinnalle kehittyneitä uudenlaisia tietokantajärjestelmiä kutsutaan yhteisnimellä NoSQL [Cat11]. NoSQL-tietokantajärjestelmät eivät perustu relaatiotietokantamalliin, vaan ne voivat olla esimerkiksi avain-arvo- tai dokumenttipohjaisia [Cat11]. Luvussa 3 esitellään eri tyyppisiä keskusmuistissa toimivia NoSQL-tietokantajärjestelmiä.

2 Käsitteistö

Tässä luvussa käsittellään tutkielman kannalta oleellisia käsitteitä. Luvussa 2.1 määritellään NoSQL-termi. Luvussa 2.2 käsitellään ACID-periaatetta, joka havaitaan ongelmalliseksi keskusmuistitietokantajärjestelmien kannalta. ACID-periaatteen vastakohtana pidettyä skaalautuville järjestelmille sopivampaa BASE-periaatetta käsitellään luvussa 2.3. Luvussa 2.4 käydään läpi CAP-teoreema.

2.1 NoSQL

NoSQL on yhteisnimitys tietokantajärjestelmille, jotka ovat luopuneet perinteistä relaatiotietokantamallista eivätkä täten myöskään käytä niille suunniteltua SQL-kyselykieltä [Bar10]. NoSQL-termi on kuitenkin ongelmallinen, sillä se kuvaa ainoastaan mitä kyseinen tietokantajärjestelmä ei ole, eikä sitä, mitä se todellisuudessa on [Ler10]. Uusille kehittyneemmille tietokantajärjestelmille onkin keksitty muita nimityksiä, kuten Beyond SQL [FCP12].

NoSQL-tietokantajärjestelmiä on useita erilaisia, kuten sarake- ja verkko-orientoitu-

neita tietokantajärjestelmiä [Cat11]. Tässä tutkielmassa keskitytään kuitenkin tarkastelemaan avain-arvo- ja dokumenttipohjaisia NoSQL-keskusmuistitietokantajärjestelmiä.

2.1.1 Avain-arvo-tietokantajärjestelmät

Avain-arvo-tietokantajärjestelmät (*key-value*) koostuvat avain-arvo-pareista, joissa jokaista yksilöivää avainta vastaa tasan yksi arvo [Cat11]. Arvoja käsitellään siis avaimen perusteella. Arvo voi olla yksinkertainen merkkijono tai monimutkaisempi tietorakenne [Ler10]. Avain-arvo-tietokantajärjestelmät tarjoavat yleensä tiedon haku-, lisäys- ja poisto-operaatiot [Cat11].

2.1.2 Dokumenttitietokantajärjestelmät

Dokumenttitietokantajärjestelmät tukevat jäsennetyn tiedon tallentamista [WeD12]. Dokumenttitietokantajärjestelmät koostuvat dokumenteista, jotka sisältävät arvoja. Nämä arvot voivat olla toisia dokumentteja, listoja tai tavallisia skalaariarvoja [Cat11].

2.2 ACID-periaate

ACID-periaatetta on pidetty perinteisesti hyvän tietokantajärjestelmän määritelmänä [Bar10]. ACID on lyhenne sanoista jakamattomuus (*atomicity*), eheys (*consistency*), eristyneisyys (*isolation*) ja pysyvyys (*durability*) ja ne tarkoittavat seuraavaa [Bar10]:

- **Jakamattomuus** tarkoittaa, että tietokantaoperaatio joko suoritetaan tai se epäonnistuu. Operaation epäonnistuessa operaation tulee epäonnistua kokonaan ja tietokanta tulee palata tilaan, jossa se oli ennen operaation alkua.
- **Eheys** määrää, että tietokanta on hyväksyttävässä tilassa ennen operaatioita sekä niiden jälkeen.
- **Eristyneisyys** edellyttää, että tietokantaoperaatiot ovat itsenäisiä toisistaan – samaa tietoa muutettaessa, toinen operaatio joutuu odottamaan toisen valmistumista.
- **Pysyvyys** merkitsee, että tietokantaan tehdyt muutokset säilyvät jopa virheellisestä järjestelmätoiminnosta huolimatta.

Cattellin mukaan [Cat11] keskusmuistitietokantajärjestelmät täyttävät useimmiten kolme ensimmäistä ACID-periaatetta, mutteivät pysty takaamaan tietojen pysyvyyttä, sillä tallennus tapahtuu haihtuvaan muistiin. Hänen mukaan kuitenkin jotkut NoSQL-tietokantajärjestelmät varmuuskopioivat säännöllisesti tietoja keskusmuistista levylle, mutta tämä ei välttämättä tapahdu jokaisen päivityksen yhteydessä.

2.3 BASE-periaate

BASE (*basically available, soft-state, eventual consistency*) -periaatetta voidaan pitää ACID-periaatteen vastakohtana [Bar10]. Prichett kirjoittaa, että [Pri08] BASE-tietokannasta saatava tieto ei välttämättä ole ajantasainen, mutta saatavilla oleva tieto on aina luettavissa ja päivitettävissä. Hänen mukaan lopullinen eheys (*eventual consistency*) takaa, että tietokanta kuitenkin eheytyy tietyn ajan kuluessa.

BASE-tietokannat skaalautuvat helposti lukuisille palvelimille [Pri08] ja ne ovatkin korvanneet ACID-tietokannat useissa pilvipalveluissa [Päi10]. BASE-periaatteiden idea onkin, että luopumalla ACID-periaatteista saavutetaan paljon korkeampi suorituskyky ja parempi skaalautuvuus [Cat11].

2.4 CAP-teoreema

CAP lyhenne tulee sanoista eheys (*consistency*), saatavuus (*availability*) ja osituksen sietokyky (*partition-tolerance*) [Sto10]. Alun perin Eric Brewerin esittelemän teoreeman mukaan on mahdotonta taata kaikkia kolmea tekijää samanaikaisesti [Bar10]. CAP-teoreeman tekijät tarkoittavat seuraavaa [Sto10]:

- **Eheys** edellyttää, että jokainen solmu näkee samat tiedot samaan aikaan.
- **Saatavuus** tarkoittaa, että on pystyttävä aina lukemaan ja kirjoittamaan.
- **Osituksen sietokyky** olettaa, että järjestelmä toimii vaikka yhteys joidenkin solmujen välillä katkeaa.

Bartholomew kirjoittaa, että [Bar10] tavanomaiset SQL-tietokantajärjestelmät ovat keskittyneet eheyden ja ACID-sääntöjen noudattamisen tavoitteluun. Hänen mukaan NoSQL-tietokantajärjestelmät keskittyvät joko eheyden ja saatavuuden tai saatavuuden ja osituksen sietokyvyn tavoitteluun. Eheyttä ja saatavuutta tavoittelevat tietokantajärjestelmät ovat hänen mukaansa suunniteltu käytettäväksi yhdessä palve-

linsalissa ja niillä voidaan käsitellä suuria määriä tietoa helposti. Saatavuutta ja osituksen sietokykyä tavoittelevat tietokantajärjestelmät sopivat puolestaan nopeaa vasteaikaa tarvitseville web-sovelluksille.

3 NoSQL-keskusmuistitietokannat

Tässä luvussa tarkastellaan tarkemmin neljää eri NoSQL-keskusmuistitietokantajärjestelmää. Luvussa 3.1 käsitellään yksinkertaista avain-arvo-pohjaista Memcached-tietokantajärjestelmää. Luvussa 3.2 käsitellään Memcached-tietokantajärjestelmää monipuolisempaa avain-arvo-pohjaista Redis-tietokantajärjestelmää. Helposti skaalautuvaa suurien tietomassojen käsittelyyn kehitettyä Project Voldemort -tietokantajärjestelmää käsittellään luvussa 3.3. Dokumenttipohjaista Terrastore-tietokantajärjestelmää käsitellään luvussa 3.4. Lopuksi luvussa 3.5 tehdään lyhyt katsaus muutamaan muun NoSQL-keskusmuistitietokannan tärkeimpiin ominaisuuksiin.

3.1 Memcached

Memcached on avoimen lähdekoodin NoSQL-keskusmuistitietokantajärjestelmä, johon tieto tallennetaan avain-arvo-pareina. Tiedon tallentaminen on toteutettu hajautustauluna ja Memcached toteuttaa operaatiot avain-arvo-parin asettamiselle, arvon hakemiselle avaimella sekä avaimen poistolle [Ler08].

Esimerkissä 1 käsitellään yksinkertaista Memcached-tietokantaa käyttävää Ruby-ohjelmaa. Ohjelman alussa määritellään käytettävät kirjastot ja rivillä kolme luodaan asiakasohjelman välityksellä yhteys portissa 11211 palvelevaan Memcached-palvelimeen. Seuraavalla rivillä asetetaan *vihannes*-nimisen avaimen arvoksi *porkkana*. Operaatiolle voidaan myös antaa parametreina tiedon voimassaoloaika – mikäli parametria ei anneta, on tieto voimassa ikuisesti [Ler08]. Rivillä viisi haetaan samaiselta Memcached-palvelimelta avainta *vihannes* vastaava arvo joka tulostetaan. Ohjelma siis tulostaa *Arvo on 'porkkana'*.

Lerner kirjoittaa [Ler08] Memcached-tietokantajärjestelmän olevan suunniteltu käytettäväksi muualta haetun tiedon väliaikaiseen säilytykseen ja olevan täten hyvin rajoitettu operaatioltaan vailla sisäänrakennettua tukea monimutkaisille tietotyypeille

```
1: require 'rubygems'
2: require 'memcache'

3: CACHE = MemCache.new 'localhost:11211'
4: CACHE.set('vihannes', 'porkkana')

5: value = CACHE.get('vihannes')
6: puts "Arvo on '#{value}'"
```

Esimerkki 1: Memcached-tietokantajärjestelmää käyttävä Ruby-ohjelma.

– joskin hänen mukaan tuki niille voidaan toteuttaa asiakasohjelman puolella. Hänen mukaan Memcached jättää myös hajautuksen toteuttamisen asiakasohjelman vastuulle, jonka tulee päättää mille Memcached-palvelimelle tieto tallennetaan ja vastaavasti miltä sitä haetaan. Asiakasohjelmia on saatavilla useille eri ohjelmointikielille ja hänen mukaan Memcached onkin tärkeä osa skaalautuvien web-sovellusten rakentamisessa – sitä käyttämällä voidaan nopeuttaa sivuston vasteaikaa ja vapauttaa resursseja muuhun käyttöön, esimerkiksi relaatiotietokannoilta.

Membrain ja Membase ovat Memcached-tietokantajärjestelmän kanssa yhteensopivia keskusmuistitietokantajärjestelmiä, jotka laajentavat Memcached-tietokantajärjestelmän toiminnallisuutta tuomalla tiedon pysyvyyden (*persistence*), toisintamisen (*replication*), korkean käytettävyyden (*high availability*) ja varmuuskopioinnin [Cat11]. Cattellin mukaan [Cat11] ilman tiedon pysyvyyttä ja toisintamista Memcached-tietokantajärjestelmää ei voida pitää varsinaisena tietovarastona, toisin kuin Membrain ja Membase ovat.

3.2 Redis

Redis on yhden henkilön projektista liikkeelle lähtenyt avoimen lähdekoodin avain-arvo-pohjainen NoSQL-keskusmuistitietokantajärjestelmä, joka on kirjoitettu C-ohjelmointikielellä [Cat11]. Tavanomaisista avain-arvo-tietokantajärjestelmistä poiketen Redis pystyy käsittelemään myös listoja, joukkoja, järjestettyjä joukkoja ja hajautustauluja [Ler10]. Toisin kuin edellisessä luvussa käsitelty Memcached, Redis voi tallentaa tiedon keskusmuistin lisäksi säännöllisesti myös levylle tiedon varmuuskopiointia varten [Cat11]. Redis voidaan asettaa kirjoittamaan myös kaikki muutokset suoraan levylle, joka toisaalta alentaa huomattavasti suorituskykyä [RuB11].

Muiden avain-arvo-tietokantajärjestelmien tapaan Redis toteuttaa lisäys-, haku- ja

poisto-operaatiot [Cat11]. Redis sisältää myös operaatioita listojen ja joukkojen käsittelyyn sekä muita operaatioita [Ler10]. Luvun esimerkit perustuvat Redis-tietokantajärjestelmän dokumentaation tietotyyppien esittelyoppaaseen [RInt]. Esimerkissä 2 havainnollistetaan tietokantajärjestelmän mukana tulevaa komentoriviohjelmaa käyttäen laskurin kasvattamista ja vähentämistä sen omilla operaatioilla.

```
redis> SET laskuri 1
OK
redis> INCR laskuri
(integer) 2
redis> DECR laskuri
(integer) 1
```

Esimerkki 2: Laskurin kasvattaminen Redis-tietokantajärjestelmässä.

Esimerkissä 3 lisätään listan *luvut* perään kolme alkiota, jotka seuraavaksi haetaan listasta; antamalla lopetusindeksiksi -1 saadaan koko lista. Viimeisenä haetaan ja samalla poistetaan listan ensimmäinen alkio.

```
redis> RPUSH luvut eka toka kolmas
(integer) 3
redis> LRANGE luvut 0 -1
1) "eka"
2) "toka"
3) "kolmas"
redis> LPOP luvut
"eka"
```

Esimerkki 3: Listan käyttö Redis-tietokantajärjestelmässä.

Esimerkissä 4 lisätään joukkoon *vihannekset* kolme alkioita. Seuraavaksi joukon alkiot listataan ja yritetään lisätä samaan joukkoon yhtä alkiota, joka löytyy jo joukosta. Tämä ei onnistu, sillä kaikkien joukon alkioiden tulee olla uniikkeja [Ler10]. Lopuksi vielä varmistetaan, ettei lisäys onnistunut, kysymällä joukon mahtavuutta, eli sen alkioiden lukumäärää. Esimerkeissä esiteltyjen operaatioiden lisäksi Redis sisältää myös joukko-operaatiot muun muassa unionille ja leikkaukselle sekä tuen järjestetyille joukoille [Ler10].

Ruflinin ja Burkhart kirjoittavat [RuB11] Redis-tietokantajärjestelmän skaalautuvan horisontaalisesti lukuoperaatioiden suhteen – tietokanta voidaan toisintaa useammalle solmulle isäntä-renki-periaatteella, missä järjestelmässä on useita renkejä, joilla on vain lukuoikeudet tietokantaan ja isäntä, johon varsinaiset muutokset tehdään. Hei-

```
redis> SADD vihannekset kurkku tomaatti porkkana
(integer) 3
redis> SMEMBERS vihannekset
1) "porkkana"
2) "tomaatti"
3) "kurkku"
redis> SADD vihannekset kurkku
(integer) 0
redis> SCARD vihannekset
(integer) 3
```

Esimerkki 4: Joukon käyttö Redis-tietokantajärjestelmässä.

dän mukaansa kirjoitusoperaatioiden suhteen skaalautuminen on myös mahdollista, mutta se vaatii asiakasohjelmalta tuen jatkuvalle hajauttamiselle. Lernerin mukaan [Ler10] Redis soveltuukin tilanteisiin, joissa tarvitaan nopeaa tietokantajärjestelmää, jolta vaadittavat tietorakenteet löytyvät Redis-tietokantajärjestelmän tarjoamista tietorakenteista eikä mahdollinen viimeisimmän tiedon menettäminen ole haitallista.

3.3 Project Voldemort

Project Voldemort on helposti skaalautuva ja suurien tietomassojen käsittelyyn suunniteltu avain-arvo-pohjainen NoSQL-keskusmuistitietokantajärjestelmä [SKG10]. Voldemort on avointa lähdekoodia ja se on kirjoitettu Java-ohjelmointikielellä [Cat11]. Sen tunnetuimpia käyttäjiä ja kehittäjiä on sosiaalisen median LinkendIn-palvelu, jossa sitä käytetään erityisesti suuria tietomassoja käsittelevien toiminnallisuuksien toteutuksessa [SKG10].

Muiden avain-arvo-pohjaisten NoSQL-keskusmuistitietokantajärjestelmien tapaan Voldemort tukee haku-, lisäys- ja poisto-operaatioita [Päi10]. Aiemmissa luvuissa esitellyistä NoSQL-keskusmuistitietokantajärjestelmistä poiketen Voldemort käyttää moniversioivaa samanaikaisuuden hallintaa päivitysoperaatioissa (*multi-version concurrency control – MVCC*) [Cat11]. Moniversioiva samanaikaisuuden hallinta liittyy tietokantamalliin, jossa tietoalkiosta on saatavilla nykyversion lisäksi myös vanhempia versioita – alkion päivitysoperaatio siis ainoastaan luo uuden alkion eikä välttämättä poista vanhaa alkiota [BeG83].

Esimerkissä 5 havainnollistetaan Voldemort-tietokantajärjestelmän moniversioivan samanaikaisuuden hallinnan toimintaa sen omaa komentoriviohjelmaa käyttäen. Esimerkki perustuu tietokantajärjestelmän dokumentaation pikaoppaaseen [VQck].

Esimerkissä tallennetaan avainta *juoma* vastaava arvo *olut* tietokantaan ja heti perään haetaan samaisella avaimella samainen arvo. Seuraavaksi tallennetaan avainta *juoma* vastaava arvo *vesi* ja haetaan se tietokannasta kuten edellä, jolloin huomataan versionumeron muuttuneen.

```
> PUT "juoma" "olut"
> GET "juoma"
version(0:1): "olut"
> PUT "juoma" "vesi"
> GET "juoma"
version(0:2): "vesi"
```

Esimerkki 5: Project Voldemort-tietokantajärjestelmän moniversioiva samanaikaisuuden hallinta.

Sumbalyn ja kumppaneiden mukaan [SKG10] Voldemort skaalautuu horisontaalisesti sekä luku- että kirjoitusoperaatioiden suhteen. Cattell kirjoittaa [Cat11] tietokantajärjestelmän huolehtivan automaattisesti myös osien partitionnista – yhtenäinäistä hajautusta (*consistent hashing*) käyttämällä tieto jaetaan koko solmujen muodostaman renkaan kesken. Voldemort tukee myös tiedon tallentamista levylle ja sen mukana tuleekin BerkleyDB ja MySQL toteutukset [Päi10].

3.4 Terrastore

Terrastore on dokumenttipohjainen NoSQL-keskusmuistitietokantajärjestelmä, jota käytetään HTTP-protokollan välityksellä tapahtuvien haku- ja tallennusoperaatioiden kautta [Cat11]. Tietokannan dokumentit ovat JSON (*JavaScript Object Notation*) -muodossa [WeD12].

Luvun esimerkit perustuvat Terrastore HTTP Client API -dokumentaatioon [TAPI]. Esimerkissä 6 tallennetaan HTTP-protokollan PUT-päivitysoperaatiolla JSON-objekti avaimella *auto* tietokannan säiliöön (*bucket*) *parkkihalli*. Esimerkissä 7 haetaan GET-pyynnöllä säiliöstä *parkkihalli* avainta auto vastaava dokumentti ja saamme vastaukseksi saman JSON-objektin, jonka tallensimme tietokantaan edellisessä esimerkissä.

```
curl -X PUT -H "Content-Type: application/json"
  -d "{\"merkki\" : \"Ferrari\"}"
  http://localhost:8000/parkkihalli/auto
```

Esimerkki 6: Dokumentin lisääminen Terrastore-tietokantaan.

```
curl -X GET -H "Content-Type: application/json"
  http://localhost:8000/parkkihalli/auto

{"merkki" : "Ferrari"}
```

Esimerkki 7: Dokumentin hakeminen Terrastore-tietokannasta.

Esimerkissä 8 haetaan kokonainen säiliö sen sisältämine dokumentteineen. Pyyntö muistuttaa esimerkin 6 pyyntöä, mutta siitä on jätetty pois dokumentin avain. Esimerkissä 9 poistamme esimerkissä 6 lisäämämme dokumentin. Mikäli halutaan poistaa koko säiliö, tarvitsee esimerkin 9 pyynnöstä jättää pois esimerkin 8 tapaan dokumen - tin avain.

```
curl -X GET -H "Content-Type: application/json"
  http://localhost:8000/parkkihalli/auto

{"auto":{"tyyppi" : "Ferrari"}}
```

Esimerkki 8: Säiliön hakeminen Terrastore-tietokannasta.

```
curl -X DELETE -H "Content-Type: application/json"
  http://localhost:8000/parkkihalli/auto
```

Esimerkki 9: Dokumentin poistaminen Terrastore-tietokannasta.

Cattellin mukaan [Cat11] kyselyille voidaan antaa myös ehtolauseita sekä rajoja ja se tukee map-reduce-periaatetta, jossa tehtävä hajautetaan suurelle määrälle solmuja (*map*), minkä jälkeen tulokset kootaan yhdeksi vastaukseksi (*reduce*). Lisäksi hänen mukaansa Terrastore-tietokantajärjestelmästä on myös saatavilla Java- ja Python-asiakasohjelmatoteutukset.

3.5 Muut

3.5.1 Scalaris

Scalaris on Redis-tietokantajärjestelmän kaltainen avoimen lähdekoodin avain-arvo-pohjainen NoSQL-keskusmuistitietokantajärjestelmä, joka on kirjoitettu Erlang-ohjelmointikielellä [Cat11]. Cattellin mukaan [Cat11] Scalaris takaa tiedon olevan eheää ja tukevan ACID-transaktioita. Hänen mukaan Scalaris osaa myös hajauttaa tiedon useammalle solmulle ja toisintaa synkronisesti.

3.5.2 Tokyo Cabinet

Tokyo Cabinet on C-ohjelmointikielellä kirjoitettu NoSQL-keskusmuistitietokanta [Cat11]. Cattellin mukaan [Cat11] Tokyo Cabinet tarjoaa useita eri tietokantamoottorivaihtoehtoja. Hänen mukaansa niiden valinnasta riippuu myös tietokantajärjestelmän tarjoamat operaatiot – joskin kaikki tarjoavat yleiset haku-, lisäys- ja poisto-operaatiot.

4 Yhteenveto

Luvussa 3.1 havaittiin, että NoSQL-keskusmuistitietokantajärjestelmillä, kuten Memcachded, on tärkeä rooli perinteisten relaatiotietokantajärjestelmien rinnalla nykyaikaisissa web-sovelluksissa [Ler08]. Luvussa 3.2 esiteltiin ominaisuuksiltaan jo hieman monipuolisempi tietokantajärjestelmä Redis, joka tukee useampia tietorakenteita, kuten listoja ja joukkoja [Ler10].

Luvussa 3.3 esiteltiin Project Voldemort, joka skaalautuu helposti ja kykenee käsittelemään suuria määriä tietoa nopeasti [SKG10]. Sumbalyn ja kumppaneiden mukaan [SKG10] Project Voldemort on myös yksi kulmakivistä maailmanlaajuiselle sosiaalisen median LinkedIn-palvelulle, jossa sitä on käytetty menestyksekkäästi jo useiden vuosien ajan.

Luvussa 3.4 tutkimme dokumenttipohjaista Terrastore NoSQL-keskusmustitietokantajärjestelmää. Terrastore poikkeaa suuresti aikaisemmissa luvuissa esitellyistä tietokantajärjestelmistä, sillä se pohjautuu dokumentteihin avain-arvo-parien sijaan ja käyttää HTTP-protokollaa tietokantaoperaatiossa [Cat11]. Lopuksi käytiin läpi lyhyesti muutama NoSQL-keskusmuistitietokantajärjestelmä, joista yksi oli myös kaupallinen [FCP12].

NoSQL-tietokantojen soveltuvuus riippuu pitkälti sovelluksen käyttötarpeesta [Ler10], mutta käyttämällä niitä oikeissa tilanteissa voidaan saavuttaa merkittävästi tehokkaampia ja skaalautuvampia sovelluksia [Ler08, SKG12].

Lähteet

Bar10 D. Bartholomew. SQL vs. NoSQL. *Linux Journal*, 2010,195 (2010). [Myös: http://www.linuxjournal.com/article/10770].

BeG83 P.A. Bernstein, N. Goodman. Multiversion concurrency control—theory and algorithms. *ACM Transactions on Database Systems*, 8,4 (1983), sivut 465–483.

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

FCP12 F. Färber, S.K. Cha, J. Primsch, C. Bornhövd, S. Sigg, W. Lehner. SAP HANA database: data management for modern business applications. *ACM SIGMOD Record*, 40,4 (2012), sivut 45–51.

Ler08 R.M. Lerner. At the forge: memcached. *Linux Journal*, 2008,176 (2008). [Myös: http://www.linuxjournal.com/magazine/forge-memcached].

Ler10 R.M. Lerner. At the forge: Redis. *Linux Journal*, 2010,197 (2010). [Myös: http://www.linuxjournal.com/article/10836].

Pri08 D. Pritchett. BASE: An ACID alternative. *ACM Queue*, 6,3 (2008), sivut 48–55.

Päi10 R. Päivärinta. Applicability of NoSQL Databases to Mobile Networks. M.Sc. Thesis, Faculty of Information and Natural Sciences, School of Science and Technology, Aalto University, 2010.

RInt A fifteen minute introduction to Redis data types. http://redis.io/topics/data-types-intro [29.2.2012].

RuB11 N. Ruflin, H. Burkhart. Social-data storage-systems. *DBSocial '11 Proc. of the First ACM SIGMOD Workshop on Databases and Social Networks*, 2011, sivut 7–12.

SKG12 R. Sumbaly, J. Kreps, L. Gao, A. Feinberg, C. Soman, S. Shah. Serving large-scale batch computed data with Project Voldemort. *FAST '12: Proc. of the 10th USENIX Conference on File and Storage Technologies*, 2012.

Sto10 M. Stonebraker. In search of database consistency. *Communications of the ACM*, 53,10 (2010), sivut 8–9.

TAPI Terrastore HTTP Client API. http://code.google.com/p/terrastore/wiki/HTTP_Client_API [29.2.2012].

VQck Project Voldemort Quickstart. http://project-voldemort.com/quickstart.php [29.2.2012].

WeD12 C. von der Weth, A. Datta. Multiterm keyword search in NoSQL systems. *IEEE Internet Computing*, 16,1 (2012), sivut 34–42.