Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

SISSI KOIVUNEN

NoSql-tietokannan suunnittelu ja toteutus Apache Cassandralla

Case: Headai

TIETOJENKÄSITTELYN TUTKINTO-OHJELMA
2022

Tekijä(t)
Koivunen, Sissi

Julkaisun laji
Opinnäytetyö, AMK

Päivämäärä
Marraskuu 2022

Sivumäärä
35

Julkaisun kieli
Suomi

Julkaisun nimi
NoSql-tietokannan suunnittelu ja toteutus Apache Cassandralla

Tutkinto-ohjelma
Tietojenkäsittelyn tutkinto-ohjelma

Tiivistelmä
Tämän opinnäytetyön tarkoituksena oli päästä testaamaan yleistä mitattavuutta, energiankulutusta ja tehokkuutta. Tätä varten tavoitteena oli rakentaa NoSql-tietokanta, joka pystyisi käsittelemään massadataa. Opinnäytetyön toimeksiantajana toimii yritys nimeltä Headai Oy. Se erikoistuu luonnollisen kielen analysointiin kognitiivisen tekoälyn avulla.

Opinnäytetyö lähtee liikkeelle teoriaosuudella, jossa syvennyttiin Apache Cassandran tietokannan hallintajärjestelmään sekä massadataan. Suunnitteluvaiheessa tietokantaa mallinnettiin useiden datamallien avulla. Valmiita datamalleja arvioitiin erilaisten laskentojen perusteella. Tietokannan rakentamisessa edettiin Debian -pakkauksen asentamisella, jonka jälkeen tehtiin tietokannan määrittelyä. Tarkkaa määrittelyä seurasi skeeman siirto palvelimelle. Datan sijoittaminen tietokantaan hoidettiin Java ohjelmointirajapinnan avulla. Java-ohjelmointirajapinta toteutettiin Eclipse IDE - koodieditorilla, ja sen luonnissa käytettiin Java ohjelmointikieltä. Optimoinnin toteuttamista esitettiin esimerkkien ja ohjeiden avulla.

Avainsanat
NoSql-tietokanta, Apache Cassandra, massadata, Java

Author(s)
Last name, First name
Koivunen, Sissi

Type of Publication
Bachelor's thesis

Date
November 2022

Number of pages
35

Language of publication:
Finnish

Title of publication
Modeling and building of NoSql database using Apache Cassandra

Degree programme
Degree Programme in Information Technology

Abstract
The meaning of this thesis was to be able to test general measurability, energy consumption and efficiency. Building a NoSql database that would handle big data, made it possible. Supervisor of this thesis was a company called Headai Ltd. They specialize in analyzing natural language with cognitive artificial intelligence.

The thesis starts with theory part. In this part, Apache Cassandra and big data was delved into. In designing the database, it was modeled using variety of data models. The data models were then evaluated with different calculations. A Debian package was installed in the building process. After installation, configuration was done for the database. Database schema was then brought to use in the server. Importing data into database was done with Java Application Programming Interface, or in short API. The API was executed with Eclipse IDE code editor and the used programming language was Java. Implementing optimization was presented using examples and through general guidance.

Keywords
NoSql database, Apache Cassandra, big data, Java,

SISÄLLYS

SYMBOLI- JA LYHENNELUETTELO

NoSql	Lyhenne engalnnin sanoista Not only SQL, eli ei ainoastaan SQL. Tällä termillä kuvataan tietokantoja, jotka poikkeavat perinteisestä relaatiomallista.
CQL	Apache Cassandran kyselykieli
OpenJDK	Open Java Development Kit on ilmainen, avoimen lähdekoodin ohjelmistokehityspaketti.
OracleJDK	Oraclen kehittämä ohjelmistokehityspaketti.
Python	Monipuolinen ohjelmointikieli
API	Application Programming Interface eli ohjelmointirajapinta, jonka kautta eri sovellukset kommunikoivat keskenään.

1. JOHDANTO

Tämä työ syntyi tarpeesta tietokannalle, joka voisi säilöä suuren määrän dataa. Massadataksi kutsutaan suuria määriä dataa, jotka ovat vaikeasti hallittavissa. Tämänkaltaisen datan hallitsemiseen ei kelpaa mikä tahansa tietokanta.

Tässä työssä käydään läpi Apache Cassandra tietokannan hallintajärjestelmän luonti alusta loppuun asti. Luonti tapahtuu suunnittelusta aina optimointiin asti. Kerron Apache Cassandrasta ja massadatasta tarkemmin. Tämän jälkeen keskityn tietokannan suunnitteluvaiheeseen, jota seuraa rakennusvaihe ja lopulta optimointi.

Opinnäytetyölle on olemassa kohdeyritys nimeltä Headai Oy, jota varten tämä työ toteutetaan. Headai on aloittanut toimintansa vuonna 2015, ja se erikoistuu luonnollisen kielen analysointiin kognitiivisen tekoälyn avulla. Yrityksen liikevaihto oli 514 000 euroa vuonna 2021, ja se työllisti kahdeksan henkilöä. Opinnäytetyön tarkoituksena on testata yleistä mitattavuutta, energiankulutusta ja tehokkuutta. Tavoitteena on tietokanta, joka pystyy mahdollisimman optimaalisesti käsittelemään massadataa.

2. APACHE CASSANDRA

2.1. Yleistä

Apache Cassandra on Apachen tarjoama yhteisötuote ja se on avoimen lähdekoodin tietokanta, tehden siitä kaikille vapaan käytettäväksi. Cassandra on tyypiltään NoSql-tietokanta. Nämä tietokannat ovat erittäin skaalautuvia ja joustavia. Apache Cassandra kehitettiin Facebookissa ja sitä käytetään tietojen hakemiseen ja tallentamiseen, sekä tiedon käsittelyyn. Sille on ominaista käsitellä valtavia määriä tietoa ja se pystyy myös

käsittelemään jäsentämätöntä dataa. Cassandra on loistava valinta massadatalle. (Education-wikin www-sivut 2021.)

Cassandra omaa korkean suorituskyvyn, joka ei heikenny datan määrästä huolimatta. Se on hyvä yrityksille, joilla ei ole varaa datan kadottamiseen tai järjestelmän kaatumiseen. Yksi keskeisimmistä eduista, joka erottaa Cassandran toisista NoSql-tietokannoista, on sen kyky käsitellä tehokkaasti valtavaa tietomäärää. Se pystyy kirjoittamaan dataa todella nopeasti, vaikuttamatta lukemistehokkuuteen. Tämän nopeus ja tarkkuus eivät muutu datan määrästä huolimatta. Yritykset suosivat Cassandraa myös sen horisontaalisen skaalautuvuuden ansiosta. Tämä mahdollistaa käyttäjiä yksinkertaisesti lisäämään laitteistoa uusien asiakkaiden ja datan tullessa. Kun ilmestyy yllättäviä muutoksia tarpeessa, on Cassandraa helppoa skaalata ilman sammutuksia tai isoja säätöjä. Cassandralla on paljon ominaisuuksia, joita muut NoSql-tietokannat eivät pysty tarjoamaan. Cassandra on myös monien suurten yritysten käytössä.

2.2. Historia

Cassandra kehitettiin alun perin Facebookin Inbox Search -ominaisuuden tehostamiseksi. Avinash Lakshman ja Prashant Malik olivat kehittämässä Cassandraa ja myöhemmin se julkaistiinkin avoimen lähdekoodin projektina Google code -palvelussa heinäkuussa 2008. Vajaa vuosi myöhemmin, maaliskuussa 2009 Cassandra otettiin Apache Incubator -projektiksi ja vuonna 2010, 17. helmikuuta se valmistui huipputason projektiin. Siitä lähtien Cassandrasta on ilmestynyt uusia versiojulkaisuja, jokainen tuoden uusia ominaisuuksia tämän päivän Cassandraan. (Gitbook www-sivut n.d.)

2.3. Apache Cassandran arkkitehtuuri

Cassandran toimivuus ei ole kiinni yksittäisen kohdan vikaantumisesta. Tämän mahdollistaa sen arkkitehtuuri, joka on keskinäisesti jaettua. Tarkoittaen, että Cassandran järjestelmä itsessään jakautuu useille eri solmuille klusterilla. Tämä saa aikaan kuvan, että Cassandra toimisi itsenäisesti kullakin solmulla, mutta toiminta on kuitenkin keskitettyä. Jokainen solmu jakaa omasta tilastaan, ja myös muiden solmujen tilasta, tietoa kaikkien muiden solmujen kesken. Klusteri on kokoelma solmuja. Klustereita

voidaan myös maantieteellisesti hajauttaa monille palvelinkeskuksille ympäri maailmaa. Solmu sisältää dataa, ja esittää yhtä ilmentymää Cassandrasta. Cassandran arkkitehtuuri on myös isännätöntä, joten mikä tahansa solmu pystyy täysin samaan toimintaan kuin kaikki muutkin solmut. (Apache Cassandran www-sivut, 2022, Cassandra Basics; ScyllaDB www-sivut, 2022, Cassandra Cluster.)

3. MASSADATA

3.1. Yleistä

Massadata ei ole yksiselitteistä, mutta yleisesti se tarkoittaa vaikeasti hallittavia suuria datamääriä. Tätä suurta datamäärää pyritään silti hyödyntämään. Tietoliikenteen jatkuvassa kasvussa löydetään koko ajan uusia tapoja hallita ja hyödyntää dataa. (SAS www-sivut n.d.)

Massadatan tärkeys ei tule sen isosta määrästä, tai siitä kuinka paljon dataa meillä on. Sen arvo tulee esiin siinä, kuinka dataa käytetään ja mitä sillä tehdään. Data ja analytiikka yhdistettynä edistävät kehitystä, joka voi muuttaa maailmaa. Massadata tarjoaa paljon mahdollisuuksia, ja sitä voidaan hyödyntää jokaisella alalla. Sen avulla voidaan arvioida erilaisia päätöksiä ja myös simuloida niitä. Tuottavuus parantuu ja turhia kustannuksia pystytään leikkaamaan. Niinkin yksinkertainen asia, kun tietojen keräys ja analysointi, voi avata uusia näkökulmia ja ratkaisuja.

3.2. Haasteet

Massadatan avulla voimme saavuttaa paljon, mutta siihen liittyy myös omia haasteita. Vähäinen tietämys massadatasta voi koitua kohtaloksi. Esimerkiksi yrityksissä massadatan käyttöönotto on suuri projekti. Jos tietämys aiheesta puuttuu, voi koko

käyttöönotto epäonnistua pahasti. Usein juuri se vähäinen tietämys on se, mikä tuottaa hankaluuksia.

Myös yksi ehkä suurimmista ja puhutuimmista haasteista on massadataan liittyvä yksityisyys. Tämän takaaminen on ensisijaisen tärkeää, sillä data on todella arvokasta ja voi sisältää hyvinkin arkaluontoista tietoa. Miskellyn haastatteleman Andreas Freundin mukaan yksityisyydestä tulee yhä enemmän menneisyyttä digitaalisen jalanjälkemme kasvaessa, ellemme siirry hajautettuun maailmaan (Miskelly, 2015, Big Data: Challenges, Risk and Solutions). Hajautettu maailma perustuu lohkoketjuun, jossa on sisäänrakennettu vahva salaus.

4. SUUNNITTELU

4.1. Tietokantasuunnitelma

Suunniteltava tietokanta tulee varastoimaan massadataa. Eli todella paljon dataa ja paljon erilaista dataa. Tietokannan käyttötarkoituksena on testata yleistä mitattavuutta, energiankulutusta ja tehokkuutta. Tälle ei ole kuitenkaan olemassa oman ymmärrykseni mukaan esimerkiksi järjestelmää tai applikaatiota. Kohdeyritys Headai on luonut kognitiivisen tekoälyn avulla työkaluja, joilla analysoidaan luonnollista kieltä. Tämän tietämyksen perusteella tietokantaa tullaan todennäköisesti analysoimaan tekoälyn ja muiden vastaavien työkalujen tai menetelmien avulla.

4.2. Tietokannan mallinnus

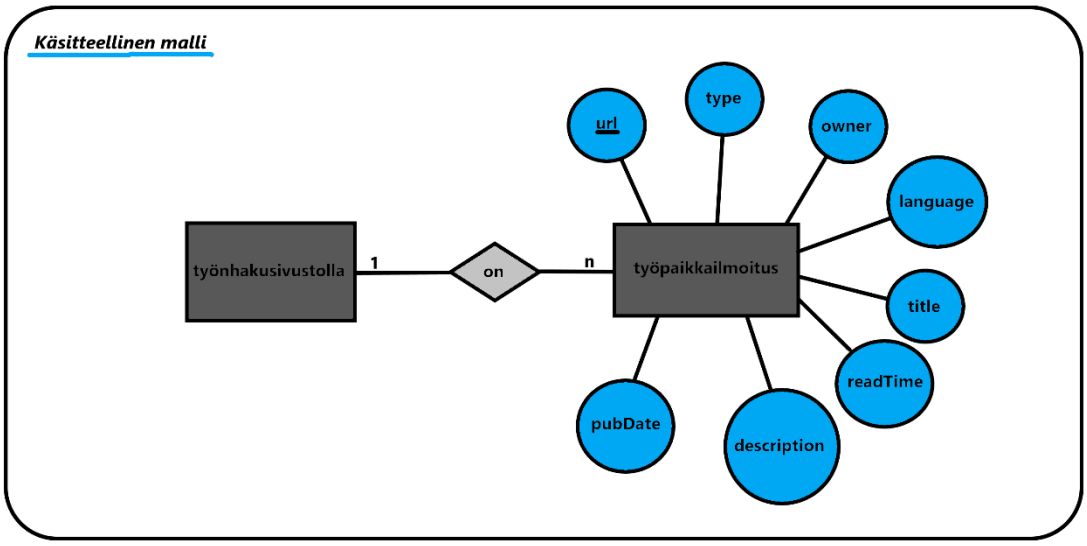
Tietokannan mallinnus on erittäin tärkeä osa tietokannan luontia. Hyvän ja tarkan mallin avulla tietokannan rakentaminen on sujuvaa, ja vältetään ikäviä yllätyksiä. Cassandran mallinnus on nimenomaan kyselyihin perustuvaa. ”You need to start with queries. Yes, you should know all important access patterns to data beforehand” (Matloka, 2019, 7 mistakes when using Apache Cassandra). Kyselyt ovat pyyntöjä, jotka noutavat tai muuttavat tietoa tietokannassa. Näistä kyselyistä alkaa mallinnus, joiden

ympärille rakennetaan itse tietokanta ja vasta viimeisenä lisätään data. Mallinnuksen suunnittelussa tulee olla selkeä kuva siitä mitä kyselyjä tietokanta tulee tekemään. Tätä tietokantaa ei voi rakentaa pelkästään datan pohjalta, kuten esimerkiksi relaatiotietokantaa.

Aloitin kyselyiden hahmottamisesta. Tutkiessani tietokantaan tulevaa dataa, potentiaalisia kyselyitä nousi esille. Täysin tarkkojen kyselyiden muodostaminen oli haastavaa, ilman tietämystä tarkasta kuvaillusta tarpeesta tietokannalle. Minulla on kuitenkin yleiskuva siitä, että tietokantaa tullaan analysoimaan työkaluilla, jotta esimerkiksi mittauksen testaus voidaan toteuttaa. Pyrin siis pitämään kyselyt yksinkertaisina, koska täysin tarkkaa kuvaa tietokannan käytöstä ei ole. Liian monen kyselyn muodostaminen voi myös lisätä kustannuksia ja tuottaa viivettä niissä. Yksinkertaisuus oli avainsana suunnittelussa.

4.2.1. Käsitteellinen malli

Hetken pyöriteltyäni potentiaalisia kyselyjä, tuli selkeäksi kaksi kyselyä, joiden kanssa jatkan suunnittelussa. Ensimmäinen kysely on: ”Löydä työpaikkailmoituksia url-osoitteen avulla” ja toinen kysely on: ”Löydä yksityiskohtia työpaikkailmoituksesta”. Suunnittelussa etenin Apache Cassandran virallisen dokumentaation avulla. Muodostin ensimmäiseksi käsitteellisen mallin tietokannasta. Käsitteellinen malli on yksinkertainen, mutta tarpeeksi monitasoinen selventääkseen tietokannan datarakenteita ja suunnittelumalleja. Käsitteellinen malli on pohja muille tuleville malleille. (Apache Cassandran www-sivut, 2022, Conceptual Data Modeling.)

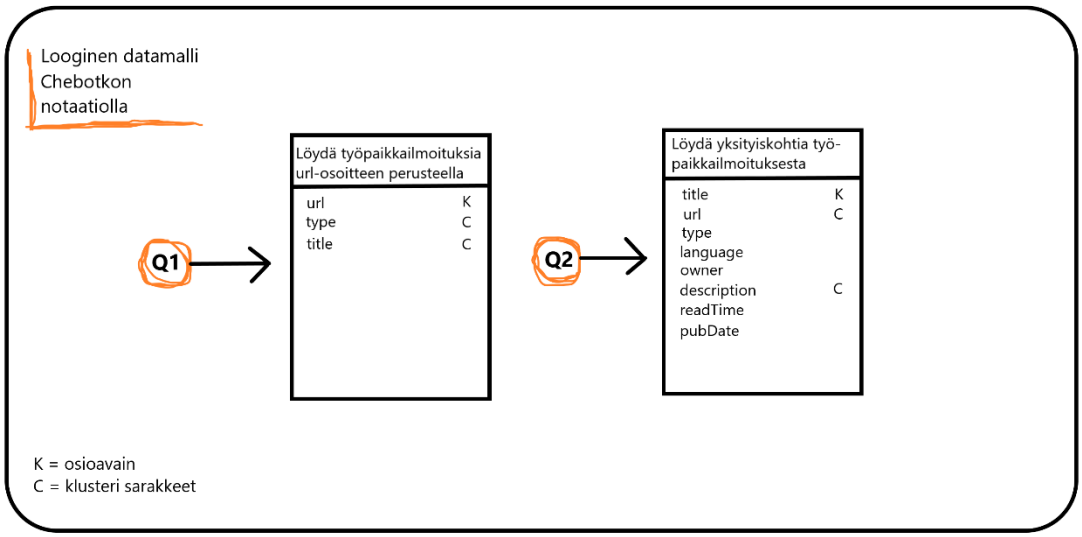


Kuva 1. Käsitteellinen malli tietokannasta.

Käsitteellinen malli perustuu minulle annettuun dataan, sekä kahteen muodostamaani kyselyyn. Mallin mukaan työhakusivustolla on työpaikkailmoitus ja näiden suhde on yhden suhde moneen. Työpaikkailmoituksella on ominaisuuksia, jotka tulevat tietokannassa olemaan taulukon sarakkeita. Mallissa ’url’ on alleviivattu. Määritin sen osioavaimeksi. Loput ominaisuudet eli sarakkeet ovat klusterisarakkeita. (Kuva 1.)

4.2.2. Looginen datamalli

Kyselyt määriteltyäni, on aika kehittää taulujen mallia paremmin. Loogisessa datamallissa jokainen kysely sisältää yhden taulun. Käsitteellinen malli tuo mukanaan suhteet ja entiteetit tähän malliin. Looginen datamalli käyttää Chebotkon notaatiota. Tämä Artem Chebotkon suosittu notaatio on yksinkertainen, mutta valaiseva tapa visualisoida kyselyjen ja taulujen välisiä suhteita. (Apache Cassandran www-sivut, 2022, Logical Data Modeling.)

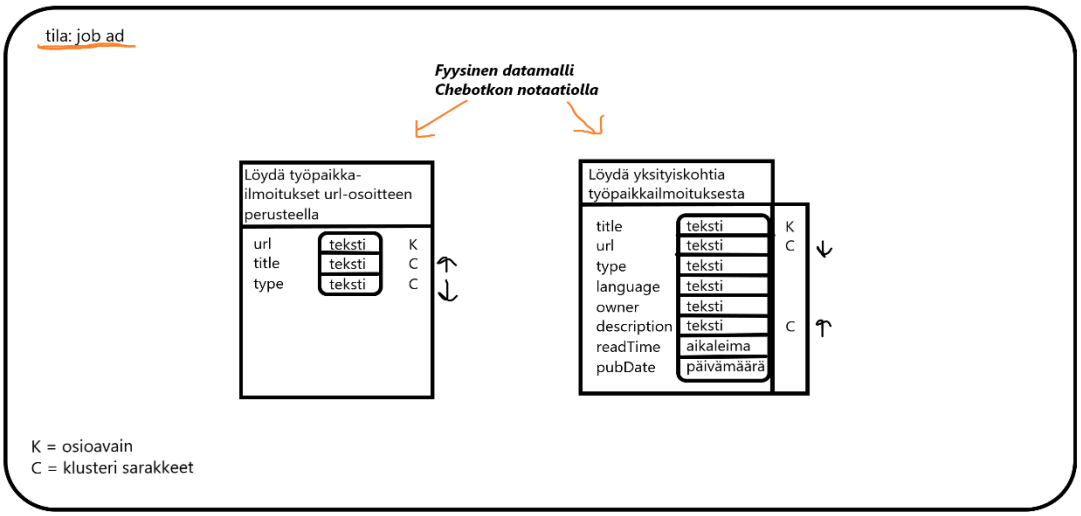


Kuva 2. Looginen datamalli Chebotkon notaatiolla.

Mallissa on kaksi taulua kahdelle määrittämälleni kyselylle. Kuvassa kyselyiden otsikot säilyvät taulukoiden otsikoina selkeyden säilyttämiseksi. Otsikoiden alta löytyy sarakkeet. Osioavain sarakkeet on merkitty K kirjaimella, ja klusterisarakkeet C kirjaimella. Pyrin tietokannan mallinnuksessa pitämään yksinkertaisuuden mielessä. (Kuva 2.)

4.2.3. Fyysinen datamalli

Fyysisen datamallin luonti on suhteellisen yksinkertaista. Mallin luomiseksi kävin läpi loogisen datamallin taulut, tyypittäen jokaisen sarakkeen. Tässä kohtaa pitää tietää minkä tyyppistä dataa jokaiseen sarakkeeseen on tulossa. Tyypiksi voi antaa minkä tahansa pätevän CQL datatyypin. CQL on Cassandran oma kyselykieli. Tämä datamalli käyttää myös Chebotkon notaatiota. (Apache Cassandran www-sivut, 2022, Physical Data Modeling.)



Kuva 3. Fyysinen data malli Chebotkon notaatiolla.

Kuvassa on edelleen kaksi taulua, niiden otsikot ja sarakkeet. Lisäsin sarakkeisiin niiden datatyypin. Dataa tarkastelemalla, päättelin sopivan datatyypin jokaiselle sarakkeelle. Suurin osa sarakkeista on tekstimuotoista dataa. Joukosta kuitenkin löytyy myös päivämäärä- ja aikaleimamuotoista dataa. Tietyt klusterisarakkeet saivat vielä ominaisuudeksi nousevan tai laskevan järjestyksen. Tämä järjestys auttaa klusteria jakamaan dataa tehokkaammin, jotta esiin tulisivat paremmin kaikista keskeisimmät ominaisuudet kyselyä tehdessä. (Kuva 3.)

4.3. Datamallien arviointi ja mahdollinen kehittäminen

Datamallien ollessa valmiina on hyvä arvioida taulujen mallia, jotta saadaan optimaalisin suorituskyky. Ensiksi tarkastelin taulujen osioiden kokoa ja varmistin ettei ne ole liian suuret. Tämän lisäksi tein laskelmaa levytilasta, eli laskin kuinka paljon kukin taulukko tulee viemään tilaa levyltä.

4.3.1. Osion koon laskeminen

Osion koon selvittäminen onnistuu laskemalla osion solujen määrän (Apache Cassandran www-sivut, 2022, Evaluating and Refining Data Models: Calculating Partition size). Solujen määrä on siis yhtä kuin osion koko. Laskemiseen käytetään Apache Cassandran virallisesta dokumentaatiosta löytyvää kaavaa.

[N_v = N_r (N_c - N_{pk} - N_s) + N_s\]

N_v	osioiden määrä
N_r	rivien määrä
N_c	sarakkeiden määrä
N_s	staattisten sarakkeiden määrä
N_{pk}	pääavain sarakkeiden määrä (sis. Osioavain sarakkeet ja klusteri sarakkeet)

Kaava 1. Osion laskennan kaava.

Kaavan mukaan osioiden määrä on yhtä kuin staattisten sarakkeiden määrä plus rivien määrä kertaa arvojen määrä riviä kohti. Arvojen määrä riviä kohti on pääavain sarakkeiden ja staattisten sarakkeiden vähennys sarakkeiden kokonaismäärästä (Kaava 1). Minulla on kaksi taulua, joille molemmille laskin osioiden määrän erikseen. Ensimmäisessä taulussa on yhteensä kolme saraketta, joista kolme on pääavain sarakkeita ja nolla staattisia sarakkeita. Täten arvojen määrä riviä kohti on nolla. Ensimmäisessä taulussa siis rivien määrä on yhtä kuin osioiden määrä, koska arvojen määrä riviä kohti on nolla ja staattisia rivejä myös on nolla. Toisessa taulussa sarakkeita on yhteensä kahdeksan, joista kolme on pääavain sarakkeita ja staattisia sarakkeita nolla. Vähennyksen lopputuloksena arvojen määrä riviä kohti on neljä. Tämän taulun osioiden määrä on yhtä kuin rivien määrä kertaa neljä.

Tähän asti tehtyjen laskelmien perusteella, jää vielä selvittämättä rivien määrä molemmissa tauluissa. Miettiessä rivien määrää, oli vaikeuksia käsittää sitä, koska määrä lasketaan applikaation mallin perusteella. Tiedossani ei ole applikaatiota, jota varten tämä tietokanta tulisi, joten piti miettiä taas aivan toisesta näkökulmasta. Tietokannalla tullaan testaamaan yleistä mitattavuutta, energiankulutusta ja tehokkuutta. Tämän kaiken uskon liittyvän datan määrään ja parhaaseen mahdolliseen optimointiin, jotta näitä voitaisiin testata. Aloin miettimään rivien määrää siltä kannalta, että kuinka monta työpaikkailmoitusta on yhtä datatiedostoa kohden. Arvioni oli seitsemästä sadasta kolmeen tuhanteen työpaikkailmoitusta. Laskennassa pelasin varman päälle, ottaen huomioon aina pahimman mahdollisen tilanteen. Datatiedostoa on tavoitteena vielä monistaa, jotta aikaiseksi saadaan massadataa. En ollut varma, kuinka monta kertaa tiedostoa kuuluisi monistaa. Tein silti arvion, jos sitä monistetaan kymmenen kertaa, sata kertaa tai jopa tuhat kertaa. Asetin ylärajaksi tuhat kertaa. Tämän ja aikaisempien

laskentojen perusteella, ensimmäisen taulun osion määrä on noin puolestatoista kahteen miljoona solua. Toisen taulun osion määrä on noin kaksitoista miljoonaa solua. Yhden osion yläraja on kaksi miljardia solua. Arvioni osioista kaksitoista miljoonaa solua ja noin kaksi miljoonaa solua kuulostavat suurelta. Täytyy silti ottaa huomioon, että tämä on arvio pahimmalle mahdolliselle tilanteelle.

4.3.2. Levytilan laskeminen

Osion määrän laskemisen lisäksi, tein myös laskelmaa levytilasta. Eli kuinka paljon kukin taulu tulee viemään tilaa levyltä. Tämän laskeminen oli enemmän haastavaa ja siihen kuului monta vaihetta. Tähän käytetty kaava oli niin monitasoinen, että en kaavaa tässä avaa tarkemmin. Käyn silti laskennan vaiheet läpi. Vaiheissa lasketaan sarakkeiden tavumääriä ja metadatan määrää (Apache Cassandran www-sivut, 2022, Evaluating and Refining Data Models: Calculating size on Disk). Huomiona vielä, että alla olevissa vaiheissa lasken molemmat taulut kerrallaan, enkä erikseen.

Ensimmäisessä vaiheessa lasketaan osioavaimien summa (Apache Cassandran www-sivut, 2022, Evaluating and Refining Data Models: Calculating size on Disk). Tauluista löytyy molemmista yksi osioavain, ‘url’ sekä ‘title’. Nämä osioavaimet ovat tekstityyppistä dataa, joten tavumäärä on yksinkertaisesta yksi tavu per merkki (DataStax Documentation www-sivut, 2022, CQL data types). Tavumäärä osioavaimista oli yhteensä keskimäärin 235 tavua.

Toisessa vaiheessa lasketaan staattisten sarakkeiden summa (Apache Cassandran www-sivut, 2022, Evaluating and Refining Data Models: Calculating size on Disk). Staattisia sarakkeita ei löydy kummastakaan taulusta. Täten tavumäärän summa on nolla.

Kolmannessa vaiheessa lasketaan yhteen klusteri sarakkeiden ja muiden sarakkeiden tavumäärä (Apache Cassandran www-sivut, 2022, Evaluating and Refining Data Models: Calculating size on Disk). Sarakkeissa on tekstityyppistä dataa, kuten myös päivämäärä- ja aikaleimatyyppistä dataa. Tyypiltään päivämäärä sarake on neljä tavua ja aikaleimatyyppinen sarake korkeintaan kolmetoista tavua (DataStax Documentation

www-sivut, 2022, CQL data types). Tauluista saadut tavumäärät, täytyi vielä kertoa aiemmin lasketun osion määrällä. Ensimmäisestä taulusta tulokseksi tuli 192,6 megatavua, ja toisesta taulusta tuli 62 839,5 megatavua.

Neljännessä vaiheessa lasketaan vielä metadatan määrä (Apache Cassandran www-sivut, 2022, Evaluating and Refining Data Models: Calculating size on Disk). Metadatan määrä lasketaan kertomalla joka osion määrä kertaa kahdeksan. Tauluista tuli metadatan määräksi 15,3 megatavua ja 92 megatavua. Kaiken tämän jälkeen nämä kaikki tavumäärät lasketaan vielä kerran yhteen. Ensimmäinen taulu vie arviolta 208 megatavua ja toinen taulu 62 932 megatavua tilaa.

Ensimmäinen taulu ei vie edes yhtä gigatavua tilaa, mutta toisen taulun suuruus on huomattava. Cassandra pyörii vielä hyvin 500 gigatavun ja yhden teratavun välillä, mutta ehdoton yläraja on viisi teratavua dataa yhdellä solmulla (DataStax Documentation www-sivut, 2022, About Apache Cassandra: Selecting hardware for enterprise implementations). Kokonaisuudessaan siis tuo toisen taulun suuri määrä ei ole vielä lähelläkään ylärajaa, mutta optimoinnin kannalta, sitä kannattaisi ehkä jakaa pienempiin palasiin. Täytyy myös ottaa huomioon, että laskut on tehty pahimman mahdollisen tilanteen varalle.

4.4. Tietokannan skeema

Laaja ja tarkka suunnitelmavaihe päättyy valmiiseen skeemaan. Skeema kuvastaa tietokannan rakennetta. Se kertoo myös sen, että kuinka dataa tulisi sijoittaa tauluihin. Tämä skeema on toteutettu CQL-kielellä. CQL eli Cassandra kyselykieltä pääsääntöisesti käytetään kommunikoinnissa Apache Cassandra tietokannan kanssa. Tulen jatkossa puhumaan Cassandra kyselykielestä nimellä CQL. CQL on hyvin samankaltainen SQL-kielen kanssa, joten sen käyttöön oli helppo sopeutua. Kyselykieliä käytetään yksinkertaisuudessaan muodostamaan kyselyitä tietokantaan.

Database schema:

```
CREATE KEYSPACE job_ads WITH replication =
   {'class': 'SimpleStrategy', 'replication_factor' : 1};


CREATE TABLE job_ads_by_url(
   url text,
   title text,
   type text,
   PRIMARY KEY((url), title, type))
   WITH comment = 'Q1. Löydä työpaikkailmoituksia url-osoitteen perusteella'
   AND CLUSTERING ORDER BY (title ASC);


CREATE TABLE job_ad_details(
   title text,
   description text,
   url text,
   type text,
   language text,
   owner text,
   readTime timestamp,
   pubDate date,
   PRIMARY KEY((title), description, url))
   WITH comment = 'Q2. Tuo lisätietoja työpaikkailmoituksesta'
   AND CLUSTERING ORDER BY (description ASC);
```

Kuva 4. Tietokannan skeema.

Skeemassa luodaan tila ’job_ads’ replikaatio luokalla ’SimpleStrategy’, jossa replikaatio kerroin on yksi. Replikaatiot ovat kopioita klusterin riveiltä. Luokalla tarkoitetaan tapaa, miten replikaatiot jaetaan klusterin solmujen kesken. Replikaatio kerroin kertoo, kuinka monta replikaatiota säilytetään. Kun käytössä on yksi palvelinkeskus, käytetään valitsemaani ’SimpleStrategy’ luokkaa. On olemassa toinenkin luokka, jota käytetään, jos klusteri sijoittuu useammille palvelinkeskuksille. Sille ei kuitenkaan ollut tarvetta tässä työssä. Kertoimena on yksi, koska minulla on vain yksi solmu klusterilla käytössä. Muussa tapauksessa, useammat replikaatiot jakautuisivat kellon myötäisesti muille solmuille. (Apache Cassandran www-sivut, 2022, Data Definition: Common Definitions; Kuva 4.)

Seuraavaksi luodaan taulut ’job_ads_by_url’ ja ’job_ad_details’. Tauluun sijoittuu jo aikaisemmista datamalleista tutut sarakkeet, sekä niiden datatyypit. Skeemassa vielä määritellään osioavain ja klusterisarakkeet. Kuvassa 4, osioavain ja klusterisarakkeet ilmoitetaan sulkujen sisällä. Sisimmäisten sulkujen sisällä on osioavain ja ulommaisten sulkujen sisällä klusterisarakkeet. Klusterisarakkeille myös määritellään nouseva järjestys. Selkeyden vuoksi, lisäsin vielä molempiin tauluihin kommentiksi, mitä kyselyä kyseinen taulu palvelee. (Kuva 4.)

5. TIETOKANNAN RAKENNUS

5.1. Cassandran asentamisen esivaatimukset

Apache Cassandran latаukselle on tiettyjä esivaatimuksia, joiden pitää täyttyä ennen latausta. Javan ohjelmistokehityspaketti täytyy ladata, jos sellaista ei vielä koneesta löydy. Cassandran virallisessa dokumentoinnissa kerrotaan, että ohjelmistokehityspaketti OpenJDK tai OracleJDK kumpikin käy (Apache Cassandran www-sivut, 2022, Installing Cassandra: Prerequisites). Minulla on käytössä OpenJDK versio 1.8.0. Kuvassa 5 on esitetty komento Java version tarkistukseen.

Kuva 5. Komento Java version tarkistukseen.

Käyttääkseen CQL komentokehotetta (cqlsh) tulee olla asennettuna Python 3.6, tai siitä seuraavat versiot, tai Python 2.7. Komentokehote on Python-pohjainen, joten se tarvitsee sitä toimiakseen (PyPi www-sviut, 2022, cqlsh 6.0.0). Minulla on käytössä Python 3.8.10. Pythonin version saa selville komennolla, joka on esitetty kuvassa 6.

```
root@cassandra-oppari:~# python3
Python 3.8.10 (default, Jun 22 2022, 20:18:18)
```

Kuva 6. Komento Python version selvittämiseen.

5.2. Asennuksen eteneminen

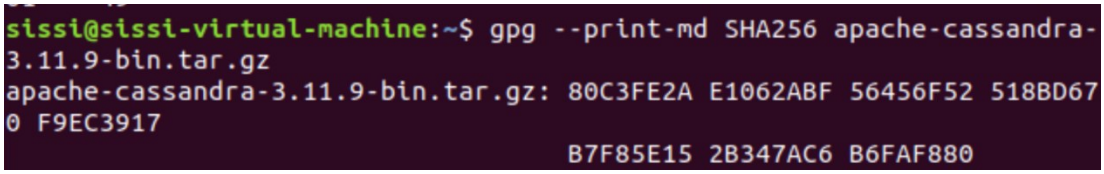
Apache Cassandran esivaatimusten täyttämisen jälkeen voidaan siirtyä itse asennukseen. Asensin Cassandran version 3.11.9. Asentaessa Apache Cassandraa halusin asentaa sen ensin virtuaalikoneelle, ja sitten vasta palvelimelle. Kohdeyritys Headai on antanut käyttööni palvelimen, johon lopullinen työ tulee käyttöön. Palvelin on asennusvaiheessa ollut Linux Ubuntu versio 18.04, joten virtuaalikone on Linux Ubuntu 18.04. Palvelin on myöhemmin päivitetty versioon 20.04. Virtuaalikone oli hyvä lisä asennuksessa. Halusin testata komentoja asennuksen yhteydessä ensin testausympäristössä, ja vasta sitten asentaa palvelimelle. Virtuaalikone toimii VMware vSphere virtualisointialustalla. Virtuaalikoneella toimii Javasta versio 11.0.4. ja Pythonista versio 2.7.17.

5.2.1. Tarball-asennus

Cassandran voi asentaa eri tavoilla, ja aluksi päätin ladata sen tar-tiedostomuodossa. Tar on tiedostomuoto, joka yhdistää useita tiedostoja (Fisher, 2021, What is a TAR File?). Seurasin asennuksessa Apache Cassandran virallisia ohjeita. Apache Cassandran lataussivustolta ladataan tiedosto peilistä. Peilisivusto on kokoelma tiedostoja, jotka on kopioitu toiselta palvelimelta, jotta tiedostot olisivat saatavissa useammasta eri sijainnista. Tiedostot ovat siis täydellisiä kopioita alkuperäiseltä palvelimelta. Alla näkyy komento, jonka avulla saadaan tiedosto ladattua. (Geekboots www-sivut, 2019, What is a mirror download and how does it works?)

$ curl -OL http://apache.mirror.digitalpacific.com.au/cassandra/3.11.9/apache-cassandra-3.11.9-bin.tar.gz

Seuraavaksi varmistin tiedoston koskemattomuuden. Varmistan tiedoston tarkisteen käyttämällä GPG:tä. GPG on julkisen avaimen salauksen toteutus, jota voidaan käyttää varmistamaan tiedoston alkuperän aitoutta. Tämän jälkeen vertasin vielä tiedoston allekirjoitusta SHA256 tiedostoon. Kuvissa 7 ja 8 näkyy tiedoston varmistamisen ja allekirjoituksen vertaamiseen liittyvät komennot ja tulokset. (Apache Cassandran www-sivut, 2022, Installing Cassandra: Installing the binary tarball; GnuGP www-sivut, 2019, Integrity Check.)



Kuva 7. Tiedoston varmentamiseen käytetty komento

Kuva 8. Tiedoston allekirjoituksen vertailuun käytetty komento

Seuraavaksi piti purkaa tiedosto, mutta sitä yrittäessä tuli virhe. Löysin ratkaisun, jolla tiedoston voi purkaa, mutta se saattaisi johtaa siihen, että joitakin hakemistoja tulee puuttumaan tiedostosta. Yrittäessä purkaa tiedostoa käy ilmi, että tiedosto ei ole gzip-muodossa. Tiedoston kuuluisi olla gzip-muodossa, mutta sen sijaan se on HTML, ASCII text -muodossa. Todennäköisesti se jollakin tapaa ei ladannut itse tiedostoa, vaan listan peilejä. Tässä kohtaa päätin koittaa erilaista asennustapaa.

5.2.2. Debian -pakkauksen asentaminen

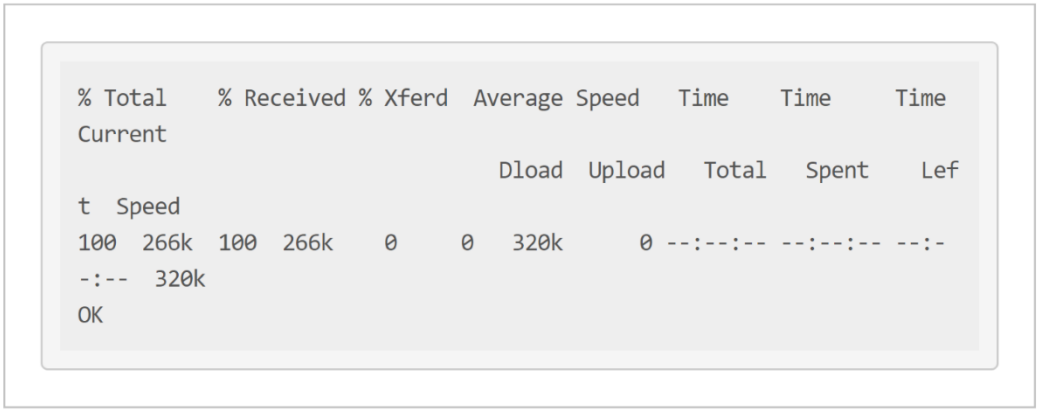
Tällä kertaa latasin Cassandrasta version 4.0. Latasinn Cassandran Apache repon eli arkiston (repository) tiedostoon nimeltä ’cassandra.sources.list’ (Apache Cassandran www-sivut, 2022, Installing Cassandra: Installing the Debian packages). Lataus suoriutuu alla olevilla komennoilla.

$ echo ”deb http://www.apache.org/dist/cassandra/debian 40x main” | sudo tee -a /etc/apt/sources.list.d/cassandra.sources.list

$ sudo cat /etc/apt/sources.list.d/cassandra.sources.list

Yllä olevista komennoista vain ensimmäinen kuuluu Cassandran viralliseen ohjeistukseen. Minulla oli ongelmia saada avaimet lisättyä vain ensimmäisen komennon jälkeen. Toinen komento ratkaisi ongelman. Latauksen jälkeen lisään Apache Cassandra arkiston avaimet palvelimen luotetuiden avaimien listaan (Apache Cassandran www-sivut, 2022, Installing Cassandra: Installing the Debian packages). Avaimien lisääminen onnistuu alhaalla olevalla komennolla. Komennon tuloksena pitäisi tulla kuvan 9 mukainen tulos.

$ curl -L https://www.apache.org/dist/cassandra/KEYS | sudo apt-key add -



Kuva 9. Avainten lisäykseen käytetyn komennon tulos

Tämän jälkeen päivitin pakkauksen osoittimen/indeksin (index). Osoitin nimensä mukaisesti osoittaa/näyttää datan sijainnin. Päivityksen myötä, täytyy enää asentaa itse Cassandra. Asensin Cassandran seuraavalla komennolla. (Apache Cassandran www-sivut, 2022, Installing Cassandra: Installing the Debian packages.)

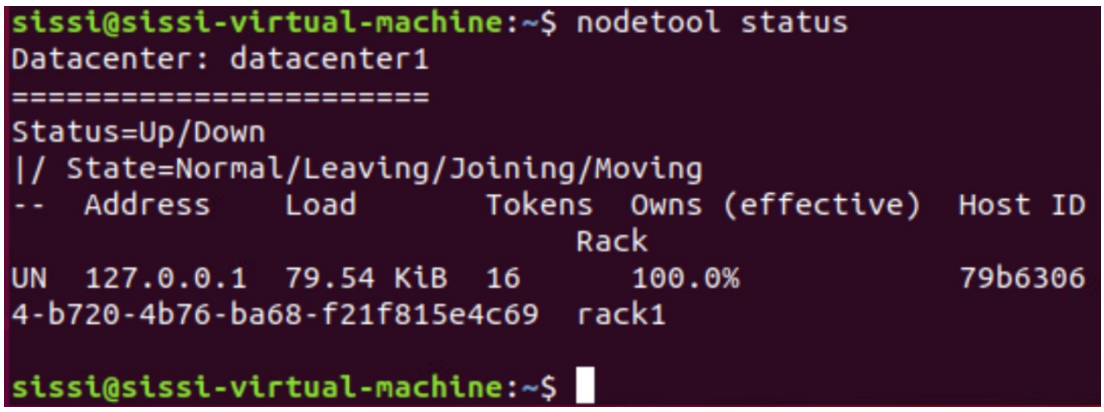
$ sudo apt-get install cassandra

Kun asennus oli suoriutunut loppuun asti, tarkistin vielä solmun tilan kuvassa 10 käytetyllä komennolla. Joskus asennuksen jälkeen solmun tilassa voi näkyä jotain virheellistä. Tämä voi johtua siitä, että joskus Cassandralla voi kestää hetki käynnistyä. Jos aikaa on kuitenkin kulunut jo huomattavasti, ja yhteys solmuun on epäonnistunut, on ehkä seuraava toimenpide tarpeellinen. Cassandran uudelleenkäynnistys yleensä korjaa tällaiset ongelmat. Alhaalla näkyvä komento suorittaa uudelleenkäynnistämisen.

$ sudo service cassandra restart

5.3. Määrittely

Varmistan aina solmun tilan, ennen CQL komentokehotteen avaamista. Avaan CQL komentokehotteen käyttämällä komentoa ’cqlsh’. Komentokehotteessa voin muun muassa määritellä Cassandraa tai luoda taulun tietokantaan. Solmun tilan voi tarkistaa alla kuvassa näkyvällä komennolla. Solmun tila on ’UN’ eli Up/Normal, ylhäällä ja normaali. Tämä kertoo, että solmun tila on niin kuin kuuluisikin. (Kuva 10.)



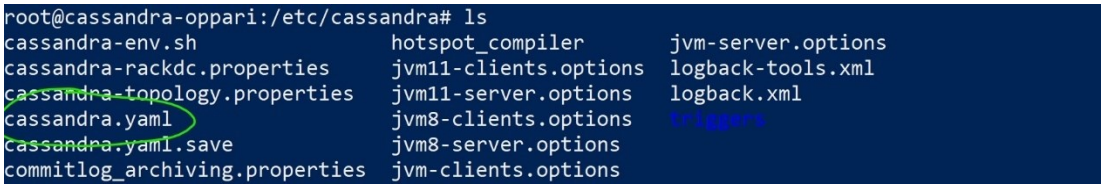
Kuva 10. Solmun tilan tarkastaminen

Kun Apache Cassandra on saatu asennettua, sitä on hyvä määritellä. On tärkeää määritellä se oikealla tavalla, vastaamaan sen käyttötarkoitusta. Apache Cassandran määritys onnistuu sen oletus konfiguraatio tiedostossa. Konfiguraatio tiedosto on nimeltään ’cassandra.yaml’. Tämän tiedoston määrittely riittää hyvin yhden solmun klusterille. Jos klusterilla olisi useampi solmu, tulisi määrittelyä laajentaa muihin määrittely

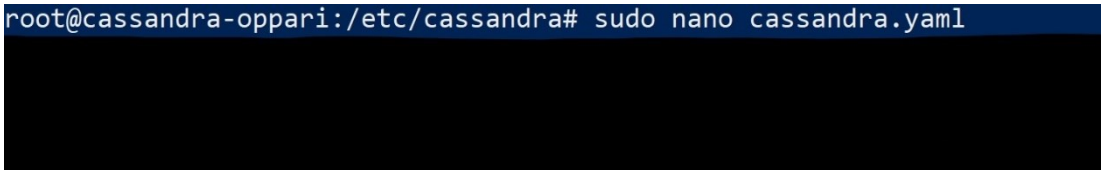
tiedostoihin. Tiedosto ’cassandra.yaml’ löytyy /etc/cassandra hakemistosta. (Apache Cassandran www-sivut, 2022, Configuring Cassandra)

Siirryn ensimmäisenä edellä mainittuun hakemistoon alla näkyvällä komennolla. Hakemistossa näkyy konfiguraatio tiedosto (Kuva 11). Seuraavaksi avaan tiedoston muokattavaksi (Kuva 12).

$ cd /etc/cassandra



Kuva 11. Hakemistossa vihreällä ympyröity konfiguraatio tiedosto



Kuva 12. Komento tiedoston muokkaamiseen

Konfiguraatio tiedostossa on monia ominaisuuksia muokattavissa. Jokaisen muokkauksen jälkeen, solmu täytyy uudelleen käynnistää. Klusterin nimi on oletuksella ”Test Cluster”. Nimi on yleensä ensimmäinen ominaisuus, jota halutaan muokata. Yrittäessäni vaihtaa nimeä, ja uudelleen käynnistää solmua, törmäsin ongelmiin. Cassandran status oli virheellinen. Normaalisti status kuuluisi olla seuraavanlainen ’active: running’. Status oli kuitenkin ’active: exited’. Yritin kaikenlaista, mutta status ei muuttunut. Vaihdoin klusterin nimen takaisin oletusnimeen, ja ongelma korjaantui.

Tämän jälkeen olin varovainen määrittelyn etenemisessä. Tein seuraavia määrittelyjä tiedostoon. Ominaisuus ’listen_address’ on solmun IP-osoite. Tämä on oletuksena ’localhost’, ja vaihdoin sen palvelimen IP-osoitteeseen. Ominaisuus ’seeds’ on lista klusterin siemensolmuja. Se sisältää siemensolmujen IP-osoitteet. Siemensolmut ovat solmuja, jotka kertovat mahdollisille uusille solmuille klusterin kehän topologian. Solmut käyttävät juoru protokollaa solmujen väliseen kommunikointiin (Datastax

Documentation www-sivut, 2022, About Apache Cassandra: Internode communications). Sen avulla ne ovat ajan tasalla toisten solmujen sijainnista ja tilasta. Minulla listalla on vain yksi IP-osoite, koska klusterillakin on vain yksi solmu kokonaisuudessaan. IP-osoite oli listalla oletuksella ’127.0.0.1:7000’, ja vaihdoin tämän myös palvelimen IP-osoitteeksi.

Määrittelin vielä näiden lisäksi kaksi muuta ominaisuutta tiedostossa. Ominaisuus ’rpc_address’ on IP-osoite, johon Cassandran prosessi sidotaan. Paikallinen siirto protokolla (CQL Native Protocol) käyttää tätä osoitetta, jotta Cassandran ajuri voisi kommunikoida palvelimen kanssa (DataStax Documentation www-sivut, 2022, Datastax Java Driver 3.0: Native protocol). Tämän ominaisuuden oletus arvo on ’localhost’, ja vaihdoin sen osoitteeksi ’0.0.0.0’. Jos tämän ominaisuuden vaihtaa edellä mainittuun osoitteeseen, ominaisuus ’broadcast_rpc_address’ täytyy vaihtaa. Tällöin ominaisuuden ’broadcast_rpc_address’ IP-osoite täytyy myös olla joku toinen kuin ’0.0.0.0’. Oletuksena se oli ’1.2.3.4’, joten vaihdoin sen palvelimen IP-osoitteeksi.

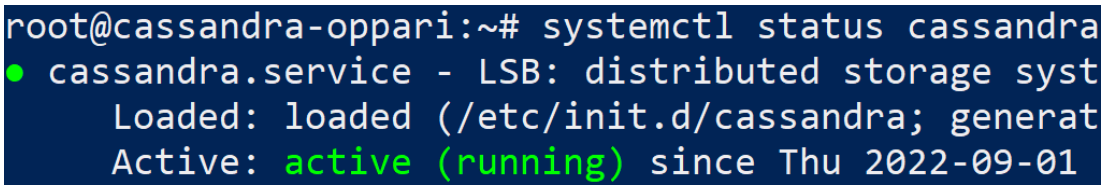
Määrittelyt olivat valmiit ja poistuin konfiguraatio tiedostosta, sen samalla tallentaen. Seuraavaksi solmu piti vielä uudelleen käynnistää. Alla olevalla komennolla tyhjennän solmun, jonka jälkeen uudelleen käynnistys on turvallista.

$ nodetool drain

Tämän jälkeen pysäytän Cassandran ja tarkistan sen statuksen, jotta se on varmasti pysähtynyt. Kun Cassandra on pysähtynyt, käynnistän sen uudelleen. Alla näkyy komennot edellä mainittuihin toimenpiteisiin.

$ systemctl stop cassandra
$ systemctl status cassandra
$ systemctl start cassandra

Solmu on nyt uudelleen käynnistetty ja tarkistan vielä Cassandran statuksen. Status on normaalissa tilassa (Kuva 13). Määrittely on onnistunut ja voin siirtyä seuraavaan vaiheeseen.



Kuva 13. Cassandran status on normaali

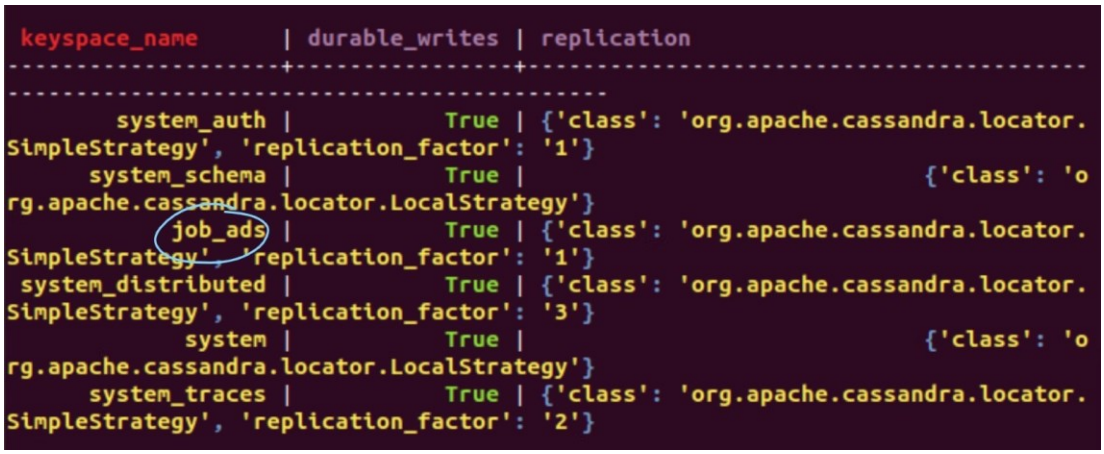
5.4. Skeeman toteutus

Datamallien lopputuloksena sain valmiin skeeman, joka on nähtävissä kuvassa 4. Apache Cassandran asennuksen ja määrittelyn jälkeen, on aika siirtää skeema toimintaan. Olen ensin siirtänyt skeeman virtuaalikoneelle ja sitten vasta palvelimelle. Näin haluan varmistaa, että siirto sujuu onnistuneesti ja ongelmitta.

Ensimmäisenä tarkistan solmun tilan, ja todetessa sen normaaliksi siirryn eteenpäin. Avaan CQL komentokehotteen käyttämällä komentoa ’cqlsh’. Komentokehotteessa ollessani, voin aloittaa skeeman siirron. Ennen taulujen luontia, täytyy olla olemassa tila, johon ne sijoittuvat. Luon tilan alla olevalla ensimmäisellä komennolla. Sen lisäksi tarkistan toisella komennolla, että se on varmasti olemassa.

>CREATE KEYSPACE job_ads WITH replication = {‘class’: ‘SimpleStrategy’, ‘replication_factor’: 1};

>SELECT * from system_schema.keyspaces;

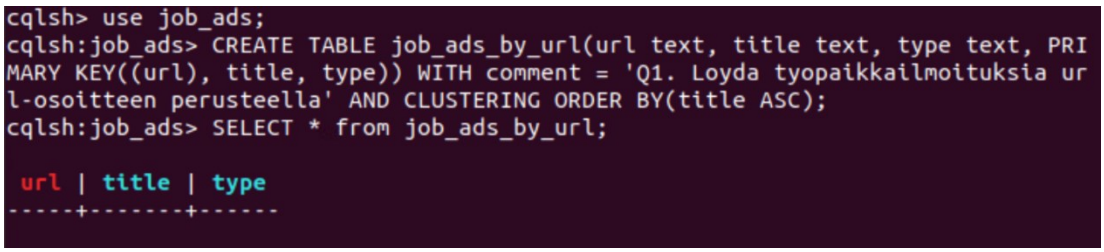


Kuva 14. Tietokannan tilat

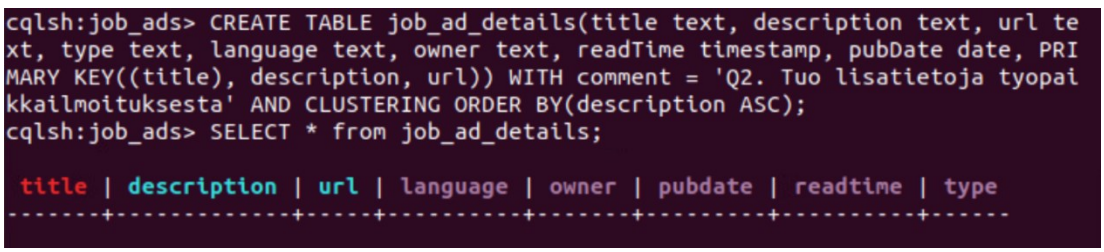
Yllä olevassa kuvassa näkyy koko tietokannan tilat. Sinisellä ympyröity tila on äsken luomani tila. Kun tila on olemassa, voin seuraavaksi luoda kuvassa 4 näkyvät taulut. Ennen taulujen luontia, täytyy tietokannalle kertoa mihin tilaan kyseiset taulut tulevat. Kerron tietokannalle mitä tilaa käytetään alla olevalla komennolla.

>use job_ads;

Oikea tila valittuna ja käytössä, luon seuraavaksi taulut. Kuvissa 15 ja 16 näkyy komennot taulujen luontiin. Sen lisäksi kuvissa näkyy, kun valitsen taulut ja niiden olemassaolo varmistuu.

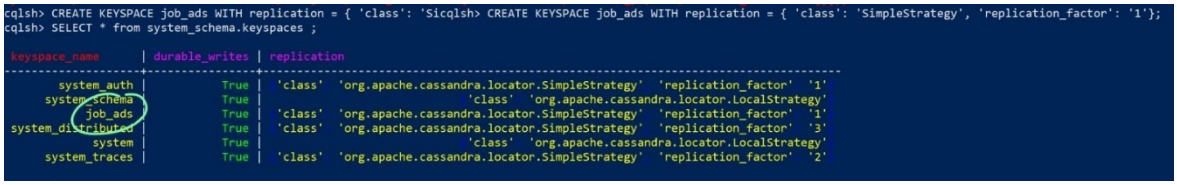


Kuva 15. Ensimmäisen taulun luonti ja valitseminen



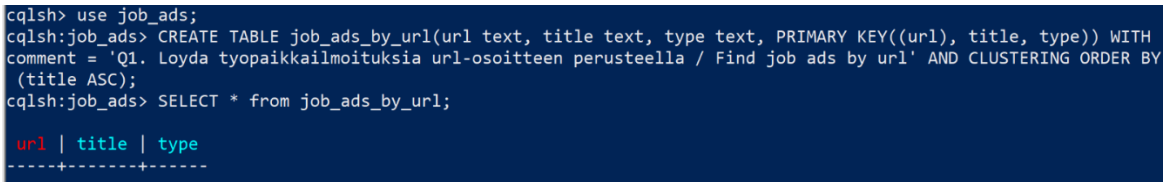
Kuva 16. Toisen taulun luonti ja valitseminen

Skeema on nyt siirretty virtuaalikoneelle. Seuraavaksi siirrän skeeman myös palvelimelle. Etenen palvelimella samankaltaisesti kuin virtuaalikoneellakin. Palvelimella ollessani, tarkistan solmun tilan ja avaan CQL komentokehotteen. Tämän jälkeen luon tilan ja pyydän näyttämään kaikki tilat, jotta tiedän luonnin onnistuneen. Kuvassa 17 näkyy edellä mainitun vaiheen komennot ja tulokset.

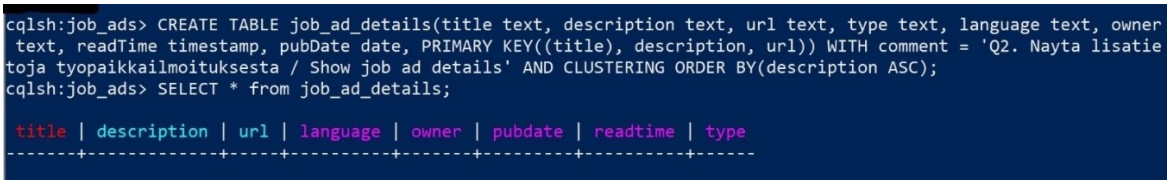


Kuva 17. Tilan luonti ja kaikki tietokannan tilat

Luomani tila on havaittavissa ympyröitynä vihreällä kuvassa 17. Kerron vielä käyttäväni kyseistä tilaa, ja sitten pääsen luomaan taulut sille tilalle. Ensimmäinen taulu vastaa kyselyäni ”Löydä työpaikkailmoituksia url-osoitteen perusteella” (Kuva 18). Luon taulun ja varmistan sen olemassaolon. Lisäsin taulun kommenttiin vielä englanninkielisen käännöksen kyselystä.



Kuva 18. Ensimmäisen taulun luonti ja sen valinta



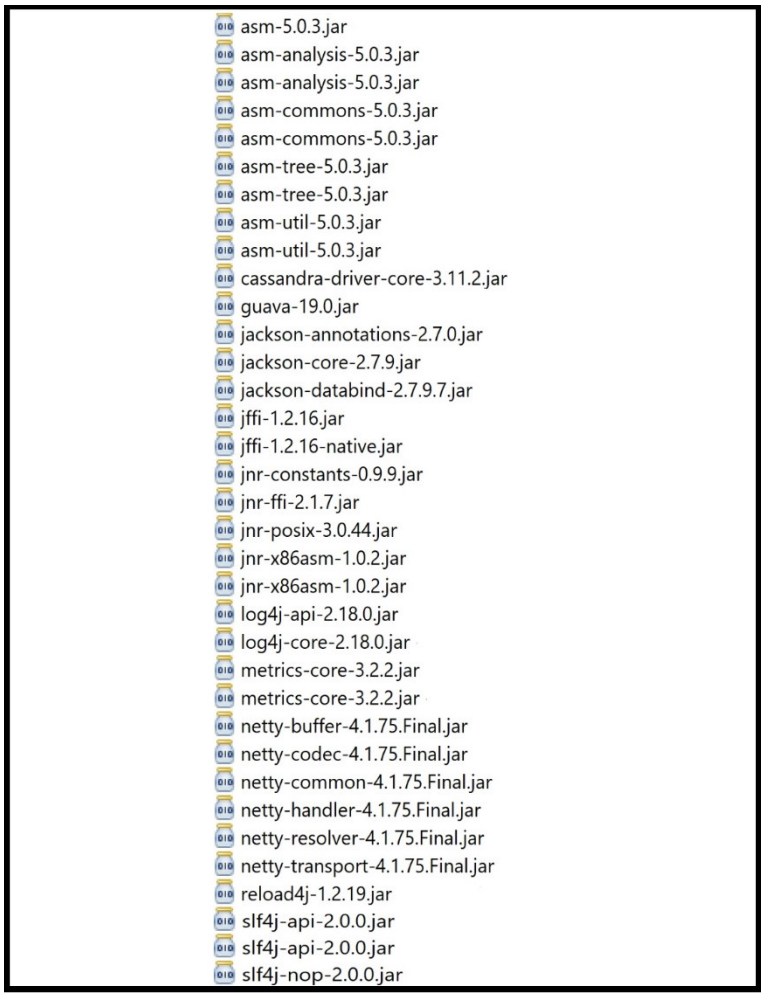
Kuva 19. Toisen taulun luonti ja sen valinta

Toinen taulu vastaa kyselyä ”Näytä lisätietoja työpaikkailmoituksesta”. Tämän taulun luonnin jälkeen, varmistan myös sen olemassaolon. Kuten kuvasta 19 näkyy, lisäsin tähän taulun myös käännöksen kyselystä englanniksi. Skeeman siirto on nyt kokonaisuudessaan tehty, eikä ongelmia ilmennyt siirron aikana.

5.5. Java-API

Muodostaakseni yhteyden tietokantaan, minun täytyy luoda Java-API. Lyhenne API tulee sanoista ’Application Programming Interface’, eli suomeksi ohjelmointirajapinta (Ravikiran, 2022, What is Java API, its Advantages and Need for it). Tulen jatkossa puhumaan rajapinnasta. Rajapinta luo tietynlaisen käyttöliittymän kahden applikaation välille, mahdollistaen niiden kommunikoinnin. Tässä tapauksessa rajapinnan avulla pystytään olla yhteydessä tietokantaan ja lähettämään pyyntöjä noutamaan dataa.

Rajapinnan luonnin olen toteuttanut koodieditorissa nimeltä Eclipse IDE, käyttäen tietenkin Java-ohjelmointikieltä. Aloittaakseni, loin aivan ensimmäiseksi oman Java projektin rajapintaa varten. Ennenkö aloitin koodin kirjoittamisen, tarvitsin projektia varten tarvittavan ajurin. Ajuri sisältää protokollan, jonka ansiosta rajapinta saadaan yhdistettyä tietokannan klusteriin. Käytin projektissani DataStaxin luomaa Java ajuria, joka on suunniteltu juuri Apache Cassandraa varten. Koin tämän ajurin luotettavaksi ja toimivaksi. On olemassa erilaisia tapoja tuoda ajuri projektille, mutta minä päädyin lataamaan koneelleni ajurin JAR-tiedostoja. JAR on lyhennys sanasta ’Java Archive’. JAR on tiedostomuoto, jota käytetään useiden tiedostojen kokoamisessa yhdeksi. Kuvassa 20 näkyy kaikki projektiin lataamani JAR-tiedostot. (DataStax Documentation www-sivut, 2022, DataStax Java Driver for Apache Cassandra; Oracle Documentation www-sivut, 2022, JAR File Overview.)



Kuva 20. Projektin JAR-tiedostot

Ajuri löytyi nyt projektista ja loin projektiin uuden luokan. Luokassa nimeltä ’database_connection’, tullaan rakentamaan sessio tietokantaan yhdistämistä varten. Session rakentaminen vei odotettua enemmän aikaa, tehtävän yksinkertaisuudesta huolimatta. Lopulta sain muodostettua tietokantaan yhteyden. Kuvassa 21 näkyy sessio kokonaisuudessaan.

```
package java_cass;

import com.datastax.driver.core.Cluster;
import com.datastax.driver.core.Session;
import com.datastax.driver.core.Host;
import com.datastax.driver.core.Metadata;


public class database_connection {


    public static void main(String[] args) {

        Cluster cluster;
        Session session;

        cluster = Cluster.builder().addContactPoint("              ").build();
        final Metadata metadata = cluster.getMetadata();

        for (final Host host : metadata.getAllHosts()) {
            System.out.println("driver version " + host.getCassandraVersion());
        }

        session = cluster.connect();
    }

}
```

Kuva 21. Yhteyden muodostaminen tietokantaan

Kuvassa 21 näkyy kokonaisuudessaan luokan ’database_connection’ sisältö. Lisäämällä paketin koodin alkuun, ilmoitan missä kansiossa kyseinen luokka sijaitsee. Seuraavaksi tuon ajurilta koodissani tarvitsemia ominaisuuksia. Näihin ominaisuuksiin kuuluu luokat ’Cluster’, ’Host’, ’Metadata’, sekä rajapinta ’Session’. Rajapinta ’Session’ pitää yllä yhteyksiä klusteriin. Luokka ’Cluster’ sisältää klusterin tilan ja kaiken muun tiedon liittyen klusteriin. Luokka ’Host’ esittää solmua. Viimeisenä luokka ’Metadata’ säilyttää metadataa klusterilta. (Kuva 21.)

Itse session toteutin rakentamalla ensin ominaisuuksista ’Cluster’ ja ’Session’ objektit. Tämän jälkeen loin yhteyden ’builder’ apuluokalla. Apuluokan tueksi lisätään metodi, jossa ilmaistaan tietokannan IP-osoite, jonne yhteys halutaan luoda. Esittelin objektin, joka kerää metadataa klusterilta. Loin vielä toistorakenteen eli for-silmukan, jossa pyydän tulostamaan Cassandran version. Tein tämän siksi, että näkisin jotain konkreettista tietokannasta. Täten pystyin olemaan täysin varma siitä, että yhteys tietokantaan toimii. Lopuksi vielä yhdistin tietokantaan viimeisellä komennolla. (Kuva 21.)

Session toteutettua, yhteys tietokantaan toimi. Todisteena toiminnasta, konsoliin tulostuu pyydetty Cassandran versio. Kuvassa 22 on näkyvillä edellä mainittu tuloste.

```
driver version 4.0.5
```

Kuva 22. Cassandran versio

Yhteyden muodostaminen tietokantaan oli ehdotonta. Yhteyden täytyy toimia, ennekuin mitään muuta pystytään tekemään. Tietokanta on tässä vaiheessa dataa vaille valmis. Luomani rajapinnan avulla pystyn lisäämään dataa tietokantaan. Sijoittamani data on yksinkertainen esimerkki tietokannan potentiaalista. Kuvassa 23 ja 24 näkyy luokassa ’DatabaseConnection’ aikaisemmin luodun session lisäksi datan sijoitus tietokantaan.

```
package java_cass;


import java.util.Date;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import com.datastax.driver.core.BoundStatement;
import com.datastax.driver.core.Cluster;
import com.datastax.driver.core.Session;
import com.datastax.driver.core.Host;
import com.datastax.driver.core.Metadata;
import com.datastax.driver.core.PreparedStatement;


public class DatabaseConnection {

    public static void main(String[] args) {
        // First half of converting string to Date. Java Date type maps cassandra timestamp type.
        DateFormat dateFormat = new SimpleDateFormat("yyyyMMdd");
        // Using LocalDate to get Date type for pubdate.
        com.datastax.driver.core.LocalDate driverLocalDate = com.datastax.driver.core.LocalDate.fromYearMonthDay(2022, 11, 27);

        Cluster cluster;
        Session session;
```

Kuva 23. ’DatabaseConnection’ luokan alkuosa

```
        cluster = Cluster.builder().addContactPoint("94.237.119.79").build();
        final Metadata metadata = cluster.getMetadata();

        for (final Host host : metadata.getAllHosts()) {
            System.out.println("driver version " + host.getCassandraVersion());
        }
        session = cluster.connect();

        //Opening try and catch statement for executing
        try {
            // Second half of converting string to Date format
            Date date = dateFormat.parse("20221127");

            PreparedStatement prepared = session.prepare("INSERT INTO  job_ads.job_ad_details (title, description, url, language, "
                    + "owner, pubdate, readtime, type) VALUES (?,?,?,?,?,?,?,?)");
            BoundStatement bound = prepared.bind("Otsikko3", "kuvaus3", "url-osite3", "suomi", "Sissi", driverLocalDate, date,
                    "tekstityyppi");
            session.execute(bound);
        }
        catch (Exception e) {
            e.printStackTrace();
        }


    }
```

Kuva 24. ’DatabaseConnection’ luokan loppuosa

Toin luokkaan lisää ominaisuuksia, johon lukeutuvat luokat ’BoundStatement’, ’Date’, ’DateFormat’ ja ’SimpleDateFormat’, sekä rajapinta ’PreparedStatement’.

Main -metodin sisällä ensimmäisenä alustin ’SimpleDateFormat’ luokan avulla odotetun muodon, jonka päivämäärä saa. Kommentin jälkeisellä, seuraavalla rivillä hain erikseen määritellyn päivämäärän käyttäen ajurin ’LocalDate’ luokkaa. Alempana näkyy aiemmin tehty sessio yhteyden muodostamiseksi tietokantaan. Session jälkeen käytin try-catch-rakennetta poikkeuksien nappaamiseen. Try-lausekkeen sisällä ensimmäisenä jäsensin ’DateFornat’ luokan avulla annetun päivämäärän merkkijonotyyppisestä muodosta päivämäärätyyppiseen muotoon. Jos yrittäisin syöttää päivämäärän merkkijonotyyppisenä, CQL ei hyväksyisi sitä, koska sarakkeen ’readTime’ datatyypiksi on määritelty aikaleima. CQL kartoittaa Javan päivämäärätyypin aikaleimaksi. Seuraavaksi loin ’PreparedStatement’ objektin nimeltä ’prepared’, jossa valmistetaan tietokannalle välitettävä komento. Komennossa sijoitetaan haluttuun tauluun ja taulussa haluttuihin sarakkeisiin arvoja, jotka määritellään myöhemmin. Olen suorittanut tämän komennon tietokannan molemmille tauluille, vaikka ohjelmassa näkyykin vain yhteen tauluun tehtävä komento. Seuraavan rivin objektissa ’bound’ sidotaan arvot edelle mainitun ’prepared’ objektin muuttujiin. Lopuksi valmistettu sessio suoritetaan ja data siirtyy tietokantaan. Catch-lauseke tulostaisi vielä virheviestin poikkeuksen sattuessa. (Kuva 23; Kuva 24.)

```
cqlsh:job_ads> SELECT * from job_ad_details;

 title    | description | url         | language | owner | pubdate    | readtime                        | type
----------+-------------+-------------+----------+-------+------------+---------------------------------+--------------
 Otsikko3 |     kuvaus3 | url-osite3 |    suomi | Sissi | 2022-11-27 | 2022-11-26 22:00:00.000000+0000 | tekstityyppi

(1 rows)
cqlsh:job_ads> SELECT * from job_ads_by_url;

 url         | title   | type
-------------+---------+-------------
 url-osoite | Otsikko | tekstityyppi

(1 rows)
```

Kuva 25. Dataa tietokannassa

Kuvassa 25 näkyy, että data on onnistuneesti siirtynyt tietokantaan. Tämän esimerkin myötä huomataan, miten data sijoittuu tietokantaan. Tietokannalta kuitenkin löytyy potentiaali mahdolliseen massadataan. Se on suunnitteluvaiheesta lähtien rakennettu sellaiseksi, että se pystyy käsittelemään massadataa. Mahdollista jatkokehitystä varten massadatan lisääminen tietokantaan on hyvä tapa testata sen optimaalisuutta.

6. OPTIMOINTI

6.1. Levytilan keventäminen

Itse suunnitteluvaiheessa jo tapahtuu optimointia. Datamallit antavat yksilöidyn rakenteen Cassandra tietokannalle, ja tarkka määrittely vie pitkälle. Kuitenkin optimoinnin mahdollisen tarpeen tullen, levytilan keventäminen on yksi tapa optimoida tietokannan suoritustoimintaa ja nopeutta. Linux-ytimellä on olemassa järjestelmäkutsu, joka lataa tiedostojen sisältöä välimuistiin. Tällä tavalla tiedostojen käyttöviive on paljon pienempi. Tiedostojen käyttötavat Cassandran kyselyiden kanssa kuitenkin usein johtaa siihen, että ne jäävät käyttämättä, mikä lisää turhaa tilaa välimuistissa. Tietokannan tauluille voidaan tehdä mitoitusanalyysiä, jonka avulla selvitetään, jos osiot ovat turhan suuria. Jos esimerkiksi osioiden arvojen määrä on suuri tai levyltä viety tila liikaa, voidaan näitä pienentää. Taulun osioavaimelle yksinkertaisesti lisätään yksi ylimääräinen sarake, tai siirretään taulussa jo olemassa oleva sarake osioavaimelle. Täten yksi suuri osio saadaan puolitettua. Tämä yksinkertainen teko voi jo huomattavasti vähentää levylle muodostuvaa kuormaa, ja täten parantaa suorituskykyä. (IBM Documentation www-sivut, 2021, Optimizing disk performance for Cassandra; Apache Cassandran www-sivut, 2022, Evaluating and Refining Data Models: Breaking Up Large Partitions.)

6.2. ’Cassandra-stress’ työkalu

Tietokannan klusterin suorituskykyä voidaan mitata, ja kuormitusta testata työkalulla nimeltä ’cassandra-stress’. Tätä työkalua voidaan käyttää usealle eri käyttötarkoitukselle. Työkalu pystyy esimerkiksi tekemään useita samanaikaisia kirjoituksia klusteria vasten, tai suorittaa useita samanaikaisia lukuja. Oikeastaan mitä tahansa näkökulmaa tietokannan toiminnasta voidaan testata, ja täten säästää aikaa tulevaisuudessa, sekä parantaakseen suorituskykyä. (Apache Cassandran www-sivut, 2022, Cassandra Stress.)

7. POHDINTA

Työssäni olen täyttänyt tarpeen tietokannalle, joka pystyy käsittelemään suuria määriä dataa. Loin Apache Cassandra tietokannanhallintajärjestelmän aivan alusta loppuun asti. Suunnitteluvaihe oli eniten aikaa vievä, sillä se vaati todella paljon taustatutkimusta itse Apache Cassandrasta. Apache Cassandran suunnitteluprosessi myös erosi todella paljon relaatiotietokannan suunnittelusta, johon olin tottunut aiemmilta kursseiltani. Halusin myös rauhassa ja huolellisesti edetä suunnittelun kanssa, jotta lopputulos olisi mahdollisimman hyvä. Otin siis oman aikani suunnittelun parissa ja olen erittäin tyytyväinen, että paneuduin kaikkiin tärkeisiin vaiheisiin suunnittelussa. Tietokannan määrittelyssä en aluksi työskennellyt niin tarkasti kuin olisi kannattanut. Tietenkin se oli täysin uutta minulle, ja tässä kohtaa minun olisi kuulunut ottaa hieman enemmän aikaa taustatutkimisen tekemiseen. Tämä johti siihen, että yrittäessä yhdistää rajapintaa tietokantaan, seurasi ongelmia. Yhteyden muodostaminen ei aluksi onnistunut ja en tiennyt mistä voisi olla kyse. Vasta tehtyäni uudelleen ja tarkemmin lisää määrittelyä onnistui yhteyden luonti. Tämän haastavan ongelman kautta opin paljon uutta ja tärkeää tietoa tietokannan määrittelystä. Rajapinnan parissa opin todella paljon uutta Java-ohjelmointikielestä, itse Apache Cassandrasta sekä rajapinnan ja tietokannan välisestä kommunikoinnista. Koen, että itsevarmuuteni etenkin Java-ohjelmointikieltä kohtaan on hurjasti noussut. Optimoinnin kannalta opin, mitkä keinot ovat juuri Apache Cassandran kannalta hyödyllisiä. Kokonaisuudessaan tämä opinnäytetyö mahdollistaa yleisen mitattavuuden, energiankulutuksen ja tehokkuuden testauksen. Tietokanta kantaa myös potentiaalia mahdollista jatkokehitystä varten.

LÄHTEET

Apache Cassandran www-sivut. 2022. Conceptual Data Modeling. Viitattu 3.5.2022. https://cassandra.apache.org/doc/latest/cassandra/data_modeling/data_modeling_conceptual.html

Apache Cassandran www-sivut. 2022. Logical Data Modeling. Viitattu 25.5.2022. https://cassandra.apache.org/doc/latest/cassandra/data_modeling/data_modeling_logical.html

Apache Cassandran www-sivut. 2022. Physical Data Modeling. Viitattu 25.5.2022. https://cassandra.apache.org/doc/latest/cassandra/data_modeling/data_modeling_physical.html

Apache Cassandran www-sivut. 2022. Evaluating and Refining Data Models: Calculating Partition Size. Viitattu 30.5.2022. https://cassandra.apache.org/doc/latest/cassandra/data_modeling/data_modeling_refining.html

Apache Cassandran www-sivut. 2022. Evaluating and Refining Data Models: Calculating Size on Disk. Viitattu 30.5.2022. https://cassandra.apache.org/doc/latest/cassandra/data_modeling/data_modeling_refining.html

Apache Cassandran www-sivut. 2022. Data Definition: Common Definitions. Viitattu 31.5.2022. https://cassandra.apache.org/doc/latest/cassandra/cql/ddl.html

Apache Cassandran www-sivut. 2022. Installing Cassandra: Prerequisites. Viitattu 20.3.2022. https://cassandra.apache.org/doc/latest/cassandra/getting_started/installing.html

Apache Cassandran www-sivut. 2022. Installing Cassandra: Installing the binary tarball. Viitattu 29.3.2022. https://cassandra.apache.org/doc/latest/cassandra/getting_started/installing.html

Apache Cassandran www-sivut. 2022. Installing Cassandra: Installing the Debian packages. Viitattu 30.3.2022. https://cassandra.apache.org/doc/latest/cassandra/getting_started/installing.html

Apache Cassandran www-sivut. 2022. Configuring Cassandra. Viitattu 4.4.2022. https://cassandra.apache.org/doc/latest/cassandra/getting_started/configuring.html

Apache Cassandran www-sivut. 2022. Cassandra Basics. Viitattu 24.9.2022. https://cassandra.apache.org/_/cassandra-basics.html

Apache Cassandran www-sivut. 2022. Evaluating and Refining Data Models: Breaking Up Larger Partitions. Viitattu 14.11.2022. https://cassandra.apache.org/doc/latest/cassandra/data_modeling/data_modeling_refining.html

Apache Cassandran www-sivut. 2022. Cassandra Stress. Viitattu 19.11.2022. https://cassandra.apache.org/doc/latest/cassandra/tools/cassandra_stress.html

DataStax Documentation www-sivut. 2022. CQL data types. Viitattu 30.5.2022. https://docs.datastax.com/en/cql-oss/3.x/cql/cql_reference/cql_data_types_c.html

DataStax Documentation www-sivut. 2022. About Apache Cassandra: Selecting hardware for enterprise implementations. Viitattu 30.5.2022. https://docs.datastax.com/en/cassandra-oss/2.2/cassandra/planning/planPlanningHardware.html

DataStax Documentation www-sivut. 2022. About Apache Cassandra: Internode communications. Viitattu 5.4.2022. https://docs.datastax.com/en/cassandra-oss/2.2/cassandra/architecture/archGossipAbout.html?hl=internode%2Ccommunications

DataStax Documentation www-sivut. 2022. Datastax Java Driver 3.0: Native protocol. Viitattu 5.4.2022. https://docs.datastax.com/en/developer/java-driver/3.0/manual/native_protocol/

DataStax Documentation www-sivut. 2022. DataStax Java Driver for Apache Cassandra. Viitattu 5.10.2022. https://docs.datastax.com/en/developer/java-driver/3.0/

Education-wikin www-sivut. 2021. Onko Cassandra NoSQL? Viitattu 15.3.2022. https://education-wiki.com/

Fisher, T. 2021. What is a TAR File? Viitattu 29.3.2022. https://www.lifewire.com/tar-file-2622386

Gitbook www-sivut n.d. The History of Cassandra. Viitattu 18.3.2022. https://teddyma.gitbooks.io/learncassandra/content/

Geekboots www-sivut. 2019. What is mirror download and how does it works? Viitattu 29.3.2022. https://www.geekboots.com/story/what-is-mirror-download-and-how-does-it-works

GnuPG www-sivut. 2019. Integrity Check. Viitattu 30.3.2022. https://gnupg.org/download/integrity_check.html

IBM Documentation www-sivut. (23.8.2021). Optimizing disk performance for Cassandra. Viitattu 17.11.2022. https://www.ibm.com/docs/en/cloud-paks/cp-management/1.3.0?topic=module-optimizing-disk-performance-cassandra

Matloka, M. (5.9.2019). 7 mistakes when using Apache Cassandra. Viitattu 26.4.2022. https://blog.softwaremill.com/7-mistakes-when-using-apache-cassandra-51d2cf6df519

Miskelly, N. (21.7.2015). Big Data: Challenges, Risk and Solutions. Viitattu 19.3.2022. https://www.bobsguide.com/articles/big-data-challenges-risks-and-solutions/

Oracle Documentation www-sivut. 2022. JAR File overview. Viitattu 5.10.2022. https://docs.oracle.com/javase/8/docs/technotes/guides/jar/jarGuide.html

PyPi www-sivut. 2022. cqlsh 6.0.0. Viitattu 20.3.2022. https://pypi.org/project/cqlsh/

SAS www-sivut n.d. Big Data Insights. Viitattu 19.3.2022. https://www.sas.com/

ScyllaDB www-sivut. 2022. Cassandra Cluster. Viitattu 24.9.2022. https://www.scylladb.com/glossary/cassandra-cluster/

Ravikiran, A. (9.8.2022). What is Java API, its Advantages and Need for it. Viitattu 3.10.2022. https://www.simplilearn.com/tutorials/java-tutorial/java-api