Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

LAURI HAKALA

Dokumenttitietokannan datan analysointi ja hyödyntäminen BI-työkaluilla

SÄHKÖ- JA AUTOMAATIOTEKNIIKKA
2021

Tekijä(t)
Lauri Hakala

Julkaisun laji
Opinnäytetyö, AMK

Päivämäärä
Kuukausi Vuosi

Sivumäärä
25

Julkaisun kieli
Suomi

Julkaisun nimi
Dokumenttitietokannan datan analysointi ja hyödyntäminen BI-työkaluilla

Tutkinto-ohjelma
Sähkö- & automaatiotekniikka

Tiivistelmä

Digitaalisesta aikakaudesta lähtien datan keräämisestä on tullut arkipäivää. Business Intelligence työkalut ovat ratkaisu muuttamaan data helposti tulkittavaan muotoon. Datan tulkitseminen ja johtopäätöksien tekeminen on tärkeä osa nykypäivän liiketoimintaa.

Työ suoritettiin Dyme Solutions Oy:n toimeksiantona, koska yrityksellä ei ollut soveltuvaa ratkaisua valmiina ja asiakkaalla oli tarve BI-ratkaisulle. Tämän opinnäytetyön tavoitteena oli suunnitella ja kehittää dataputki dokumenttitietokannan ja BI-työkalun välille AWS ympäristössä.

Opinnäytetyön tulokseksi saatiin laajamittainen dokumentaatio kahdesta eri tuontimahdollisuudesta ja niiden heikkouksista. Työn tuloksia voidaan käyttää tulevaisuuden datan analysointiprojektien tukena. Valittu prosessi otettiin käyttöön asiakkaan tuotantojärjestelmässä.

Asiasanat
kehittäminen, dokumenttitietokanta, analysointi, liiketoimintatiedon hallinta, AWS,

Author(s)
Lauri Hakala

Type of Publication
Thesis AMK

Date
April 2021

Number of pages
25

Language of publication:
Finnish

Title of publication
Analyzing and utilizing document database with BI tools

Degree program
Electrical and Automation Engineering

Abstract

Since the digital age, data collection has become common. Business Intelligence tools are a solution to convert collected data into more readable format. Interpreting data and drawing conclusions is an important part of today's business development.

The work was commissioned by Dyme Solutions Oy, as the company did not have a suitable solution ready, and the customer needed a BI-tool solution. The aim of this thesis was to design and develop a data pipe between a document database and a BI tool in AWS environment.

The result of the thesis was extensive documentation of two different import opportunities and their weaknesses. The results of the work can be used to support future data analysis projects. The selected process was implemented in customer's production system.

Key words
development, document database, analysis, business intelligence, AWS

SISÄLLYS

SYMBOLI- JA LYHENNELUETTELO (EI PAKOLLINEN)

AWS S3	Amazonin hallinnoima datan varastointi palvelu.
BI-työkalu	On liike-elämän datan analysointiin ja hankintaan kehitetty työkalu. Esimerkiksi Microsoft Power BI ja AWS QuickSight. (Business Intelligence)
Dokumenttitietokanta	On skeematon, horisontaalisesti skaalattava tietokanta, mikä mahdollistaa monimutkaisen datan tallentamisen.
ETL	Menetelmä, jolla kuvataan datan hakeminen, muuttaminen ja tallentaminen uudessa muodossa. (Extract, Transform and Load)
FaaS	Palvelu, jossa kehittäjät vastaavat vain ohjelman koodista ja konfiguroinnista. (Function as a Service)
JSON	Standardoitu tekstipohjainen muoto, joka perustuu JavaScript kielen objektiin. Yleisesti käytetty tiedostomuoto tiedonvälitykseen verkkosovelluksissa. (JavaScript Object Notation)
NoSQL	Termi, joka sisältää kaikki muut tietokannat, mitkä eivät kuulu relaatiotietokantoihin. (Not only SQL)
Palveliton	On menetelmä palveluiden tarjoamiseksi tavalla, jolla käyttäjän ei tarvitse huolehtia infrastruktuurista.
Pilvipalvelu	Verkkoyhteyden saatavilla olevan tietokoneresurssien tarjoaminen eri muodoissa.
Relaatiotietokanta	On tietokantamalli, jossa tiedot tallennetaan moniin tauluihin ja yhdistetään avaimien suhteella. Yleisin tietokanta. Voidaan myös kutsua SQL-tietokannaksi.

SQL	Datan manipulointiin tarkoitettu ohjelmointikieli. Yleisesti käytetty relaatiotietokannan käsittelyyn. (Structured Query Language)

1 JOHDANTO

Datan analysointi on yleistynyt teollisuudessa viime vuosina. Dataa kerätään ja säilötään, jotta sitä voidaan analysoida. Analysointi auttaa tuotannon kehittämisessä ja kustannussäästöissä.

Tämän opinnäytetyön tarkoituksena on suunnitella ja kehittää järjestelmä, jolla dokumenttitietokantaan tallennettu data saadaan helposti valjastettua modernin BI-työkalun käyttöön.

Dokumenttitietokannat, kuten tässä työssä käytetty DynamoDB, mahdollistavat datan tallentamisen eri muodoissa saman taulun sisällä, eikä datan skeema tarvitse olla yhtenäinen. Datan analysointityökalut eivät osaa lukea ja soveltaa eri muodoissa olevaa tietorakennetta, joten useasti data täytyy muuttaa relaatiomalliin ennen sen päätymistä BI-työkalulle.

Tässä opinnäytetyössä hyödynnetään teollisuusprosessin antureista kerättyä dataa, jota säilötään Amazonin DynamoDB tietokantaan. Dataa on jo tallennettu pitkään, mutta ongelmana on ollut sen saaminen analysoitavaan muotoon. Tavoitteena on tutkia dataa BI-työkalulla

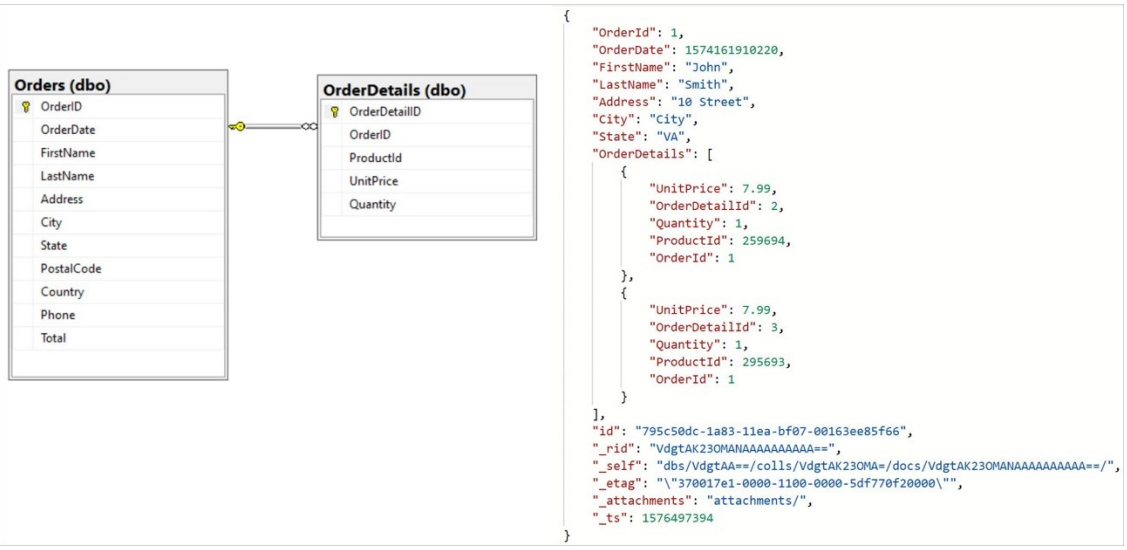
Työssä tutkitaan kahta eri vaihtoehtoa, ensimmäinen vaihtoehto on AWS Glue työkalun käyttäminen ja pakatun analysointi tiedoston muodostaminen. Toinen ratkaisu on avoimen lähdekoodin DynamoDB Connectorin liittäminen tietokannan ja BI-työkalun välille. Käyttäjäystävällisyys asiakkaan näkökulmasta on keskeinen tavoite.

1.1 Työn toimeksiantaja

Toimeksiantaja on vuonna 2014 perustettu IT-alan yritys Dyme Solutions Oy. Yritys keskittyy ohjelmistojen suunnitteluun ja valmistukseen. Dymen toimitusjohtaja ja yrityksen toisena perustajana toimii Kai Vainio. Dymen toimipaikka sijaitsee Porissa. Yritys työllistää tällä hetkellä noin 15 ohjelmistokehittäjää. (Dyme Solutions www-sivut 2021)

1.2 Työn määrittely

Työn tavoitteena oli saada dokumenttitietokannan data AWS QuickSight työkalun käyttöön. Työ rajoitettiin toimimaan vain AWS ympäristössä. Dokumenttitietokannan ja BI-työkalun yhdistämisen suurin ongelma on dokumenttitietokannan mahdollisuus sisältää monimutkaisia tietorakenteita. (Kuva 1 oikealla) QuickSight pystyy tulkitsemaan vain relaatiomallissa olevaa dataa. (Kuva 1 vasemmalla)



Kuva 1. Vasemmalla relaatiomalli ja oikealla dokumenttimalli.

2 JULKISET PILVIPALVELUT YLEISESTI

Pilvipalvelu on verkon kautta käytettävä palvelu, jota suoritetaan tietokone resursseilla. Palvelut voidaan jakaa yleisesti infrastruktuurin, ohjelmistoalustan tai

ohjelmiston käyttämiseen. (NIST www-sivut 2021) Suurimpia palveluntarjoajia ovat Google Cloud, Microsoft Azure ja Amazon Web Services. AWS tarjoaa käyttäjille yli 200 palvelua, josta tässä opinnäytetyössä keskitytään muutamaan. (AWS www-sivut 2021)

2.1 AWS Lambda

Amazon Web Servicen tarjoama on Function as a Service (FaaS) palvelu. Käyttäjä voi halutessaan määrittää, paljonko RAM-muistia funktion käyttöön halutaan varata, jonka mukaan prosessorin tehot skaalautuvat. Lambdalla maksat vain siitä, kuinka monta millisekuntia sitä käytetään. Rajoituksena onkin 15 minuutin suoritusaika. Yleisimpiä lambdan käyttötarkoituksia on tehdä yksinkertainen ja lyhytkestoinen pyyntö käyttäjän ja palvelimen välillä. Esimerkkeinä ovat esimerkiksi: Käyttäjän kuvan tallentaminen tai käyttäjän kommentin haku tietokannasta. (AWS www-sivut 2021)

2.2 AWS DynamoDB

Amazon Web Servicen tarjoama NoSQL tietokanta, eli not only SQL, on täysin Amazonin hallinnoima ja yleinen ratkaisu nykypäivän arkkitehtuurissa. Tietokanta tarjoaa niin avain-arvotietokanta mahdollisuuden, kuin dokumenttirakenteen. Avain-arvotietokanta tallentaa avaimen, jolla on arvo. Dokumenttirakenne voi sisältää laajempia kokonaisuuksia. DynamoDB käyttää tiedon tallennuksessa JSON muotoa. Datan tallennus on ilmaista 25GB asti ja sen jälkeen hinta on 0.283 $/GB/kk. (AWS www-sivut 2021)

Relaatiotietokannoissa nousee esiin ongelmia, kun datan määrä kasvaa, erityisesti vaakasuuntaisen skaalauksen toteuttamisen vaikeus. Dokumenttitietokannat ovat tulleet ratkaisemaan ongelmaa helpottamalla tietokannan vaakasuuntaista skaalausta. Vaakasuuntainen skaalaus tarkoittaa, että lisätään koneita resurssijoukolle. (AWS www-sivut 2021)

2.3 AWS Athena

Amazon Web Servicen tarjoama palvelu, jolla voidaan tehdä kyselyitä niin tietorakenteisiin, kuin tietokantoihin. Kyselykieli on SQL, joten suurin osa valmiista datan hakumahdollisuuksista on relaatiotietokantoihin, johon kysely toimii varmasti. Athena perustuu palvelittomaan arkkitehtuuriin eli maksat vain niistä kyselyistä, mitä käytät. Hinnoittelu on varsin maltillinen 5 $ per skannattu teratavu. Athena käyttää avoimen lähdekoodin SQL moottoria Prestoa, joka on lähtöisin Facebookin aloittamasta projektista. (AWS www-sivut 2021)

2.4 AWS QuickSight

Amazon Web Servicen tarjoama Business Intelligence työkalu, joka on täysin selainpohjainen ja skaalattavissa jopa 10,000 käyttäjään. QuickSight työkalun tarkoituksena on helpottaa datan analysointia. QuickSight tarjoaa kahta versiota Standard ja Enterprise versiot. Ohjelma tarjoaa myös oman sisäisen muistimoottorin nimeltään SPICE, johon pystyy tallentamaan dataa analysointia varten. Jos SPICE moottoria ei käytetä, tapahtuu haku aina datan lähteestä, mikä voi aiheuttaa kuluja. (AWS www-sivut 2021)

QuickSightista voidaan suoraan luoda yhteyksiä useimpiin relaatiotietokantoihin, kuten MySQL, PostreSQL ja AWS Aurora. Athena on yleisesti käytetty liittymistapa QuickSightin ja relaatiotietokantojen välillä. (AWS www-sivut 2021)

QuickSight tukee myös manuaalista datan tuomista. Tuetut dataformaatit: CSV, TSV, ELF, CLF, JSON ja XLSX eli Excel tiedosto. Yleisesti käytössä oleva tapa on säilöä dataa AWS S3 palveluun ja tuoda sieltä analysoitavaksi. (AWS www-sivut 2021)

2.5 AWS Glue

Amazon Web Servicen tarjoama valmis ETL menetelmää toteuttava työkalu, mikä on täysin palveliton ratkaisu. Glue on rakennettu vastaamaan ongelmaan ETL prosessin yksinkertaistamiseksi. ETL palvelun rakentaminen itse alusta lähtien vaatii paljon

resursseja ja osaamista. Palvelua käytetään paljon datan valmistelemisessa analytiikkaan ja koneoppimiseen. (AWS www-sivut 2021)

Glue on monikäyttöinen työkalu, joka perustuu Apache Spark ohjelmistoon. Spark on avoimen lähdekoodin datan käsittelyyn suunniteltu ohjelmisto, jonka rajapinta on kehuttu kehittäjien puolesta. Sparkin tukemat kielet ovat Scala, Python, R, SQL ja Java. Spark on vakiinnuttanut paikkansa Big Datan modernina muokkaustyökaluna. (AWS www-sivut 2021)

Gluen yleinen toimintarakenne muodostuu kahdesta työkalusta: Glue Crawler ja Glue Job. Crawler on työkalu, jolla pystytään helposti hakemaan taulun rakenne ja tallentamaan sen metadata Catalog Tableen. Catalog Table on osa Gluen kokonaisuutta, johon tallennetaan datalähteiden metataulut. (AWS www-sivut 2021)

Glue Job on datan prosessointityökalu. Työkalulle voidaan määritellä lähde eli Catalog Tableen muodostettu metataulu, joka sisältää rakenteen ja linkin tietokantaan. Glue pystyy muokkaamaan datan rakennetta. (AWS www-sivut 2021) Yleistä on muuttaa datan tyyppiä johonkin tehokkaasti pakattuun ja hakuoptimoituun formaattiin.

3 JÄRJESTELMÄN KEHITYS

Seuraavassa kappaleessa käydään läpi, kuinka järjestelmä kehitettiin. Opinnäytetyön tavoitteena oli tuoda dokumenttitietokannan data BI-työkalun käyttöön.

3.1 Toteutus omalla ohjelmalla

Opinnäytetyötä aloittaessa etsin tietoa mahdollisista prosesseista, joita muut olivat jo käyttäneet onnistuneesti. Amazon Glue tuli vastaan usein, mutta yleinen ilmapiiri ja sen monimutkaisuus aiheen ympärillä saivat miettimään muita ratkaisuja.

Valmistauduin tekemään oman työkalun, joka toimisi liitäntänä DynamoDB ja S3 palveluiden välillä. Asioiden yksinkertaistamiseksi valitsin koodin toteuttamiseen Lambdan, josta minulla oli aiempaa kokemusta. Ohjelmistoa tehdessä tuli vastaan asiakkaan tahtotila muodostaa uniikkeja kyselyitä datasta, heidän tuotantokokemuksien perusteella. Haluttu kysely voisi perustua esimerkiksi tuotantolinjan muutaman anturin dataan.

Uniikkien kyselyiden toteuttaminen asiakkaan puolesta tulisi olemaan työlästä. Jokainen uniikki kysely tarkoittaisi uuden lambdafunktion toteuttamista tuotantoon ja uuden S3 tiedoston tallentamista. Ongelma voitaisiin ratkaista tekemällä ohjelmistosta dynaaminen, joka ottaisi vastaan taulun rakennetta vastaavia parametreja, joilla kysely toteutettaisiin.

Dynaaminen ohjelmisto vaatisi käyttöliittymän, jotta käyttäjä voisi tehdä kyselyitä ilman ohjelmistokehittäjän osaamista. Käyttöliittymän toteuttaminen monimutkaistaisi prosessia ja lisäisi vielä yhden ylläpidettävän osan kokonaisuudelle. Tämän ratkaisun vieminen muihin projekteihin olisi hankalaa ja näillä perustein ratkaisua lähdettiin toteuttamaan muilla tavoin.

3.2 Toteutus Glue työkalulla

Toteutus aloitettiin osoittamalla Glue Crawlerille haluttu lähdedata. Tässä tapauksessa osoitin lukemaan DynamoDB tietokantaa. Glue Crawler käy läpi taulun rakenteen ja arvojen datatyypit. Tallensin taulun metadatan Gluen Catalog Tableen.

Crawlerin ajamisen jälkeen valitsin Glue Jobin lähteeksi juuri muodostetun metataulun. Glue Jobilla voidaan muuttaa tulevan taulun rakennetta, poistaa tai lisätä sarakkeita. Tässä opinnäytetyössä data tuli muokata analysoitavaan muotoon. Tutkin mahdollisuuksia muuttaa datan formaattia kyseiseen käyttötarkoitukseen.

Mahdollisia dataformaatteja olivat: CSV, ORC, Parquet ja Apache Avro. Näistä valitsin Parquetin. Suurimpana syynä oli sen nopeus ja sen ominaisuus tukea sisäkkäistä

datarakennetta. Gluen yleisimpiä ohjelmointikieliä ovat Python ja Scala, joista valitsin Pythonin, sillä se on suositumpi.

Parquet, joka on Apachen alla oleva avoimen lähdekoodin datan tallennusmuoto, dataformaatti on sarakekohtainen. Parquetin vahvuudet tulevat sen tavasta käsitellä dataa. Parquet pakkaa datan ja muuttaa sen binääriksi. Yleinen pakkausmuoto on Googlen tekemä snappy. (parquet.apache www-sivut 2021)

Glue Jobin ajamisen jälkeen Parquet tiedosto oli tallennettu S3 palveluun, ja täten haettavissa AWS Athena palvelulla. Alkuperäisessä muodossa data oli noin 2000MB, Parquet muutoksen jälkeen koko oli noin 330MB, joka on huomattavasti pienempi. Kiinnostuin vielä tiedon järjestelemisestä, jolloin hakutapahtuma voitaisiin rajata vain haluttuun osaan. Testiympäristön datamäärällä vaikutus ei olisi niin suuri, mutta määrän kasvaessa tulisi siitä huomattava hyöty.

Datan osiointi eli sen jakaminen osiin mahdollistaa todella nopean hakukyselyn, sillä haun ei tarvitse välittää muista tiedostoista, ainoastaan mihin raja-arvot ylettyvät. Valitsin datan jakamiseen tiedon, jolla hakuja rajattaisiin. Tämä tulisi olemaan aikaperusteinen. Tietorakenteessa oli ISO8601(”yyy-MM-ddTHH:mm:ss.SSSZ”) muodossa oleva merkkijono, mutta ongelmana oli, että se on liian tarkka. Datan hakemisen tarkoituksena oli hakea päiväkohtaista dataa, minuuttikohtainen erittely tehtäisiin itse BI-työkalulla.

Tietorakenteeseen tulisi siis lisätä ylimääräinen sarake, jossa olisi osiin jaettava data. Datan formaatiksi valikoitui ”yyyy-MM-dd”. Ongelmaksi muodostui Gluen käyttämä tietomalli Dynamic Frame, joka on Amazonin itse tehty paranneltu versio Sparkin Data Framesta. Molemmat muistuttavat sarakekohtaisen relaatiotietokannan taulua, mutta sisältävät optimointeja. Sarakkeen lisääminen ja datan ottaminen olemassa olevasta sarakkeesta ei näyttänyt onnistuvan ilman muunnosta Data Frameen ja sitten takaisin Dynamic Frameen.

```
## Create partition for yyyy-MM-dd for faster queries, Dataframe convertion for column manipulation
df_datasource4 = dropnullfields3.toDF()
df_datasource4 = df_datasource4.withColumn('date', date_format(df_datasource4.ts, 'yyyy-MM-dd')).repartition(1)
df_datasource4.show(10, False)
## Convert dataframe back to dynamic frame
datasource4 = DynamicFrame.fromDF(df_datasource4, glueContext, "datasource4")
```

Kuva 2. Glue Script sarakkeen lisääminen.

Datan haun optimointi onnistui ja hakeminen oli tehokasta. Skannattavan datan määrä laski noin 80 %. Tämä ratkaisu olisi vielä tehokkaampi, verrattuna mitä enemmän dataa olisi. (Kuva 3 ja 4) Myös kustannukset tulisivat olemaan pienemmät datan haussa.

Kuva 3. Haku ennen datan jakamista.

Kuva 4. Haku datan jakamisen jälkeen.

Kuva 5. Tietorakenne jakamisen jälkeen S3 palvelussa.

3.2.1 Glue ongelmat

Gluen kokonaisuus on monimutkainen, jolloin käytettävä aika tuloksien aikaansaamiseksi kasvaa. Myös Gluen eroavaisuudet yleisesti käytettyyn Apache Spark ohjelmistoon, vaatii Gluen oppimista.

Kehittäjän näkökulmasta Glue Job ohjelman kirjoittaminen ja testaaminen on suhteellisen haastavaa, sen pitkän käynnistymisajan takia, joten muutoksien tekeminen koodiin ja niiden tulkitseminen on hidasta. Ennen 2.0 versiota käynnistys kesti noin 10 minuuttia ja uuden version myötä Glue lupaa noin 1 minuutin käynnistysajan.

Vaikka datan haku oli nopeaa ja optimoitua, oli sen prosessissa ongelmia. Gluen hinnoittelu alkaa 0.88 $/h ja minimilaskutus on 1 minuutti. Kahden kuukauden datan muuttaminen kesti noin 14 minuuttia. Tämä itsessään ei ole ongelma, vaan se että Glue ei tue mahdollisuutta tehdä osittaista latausta DynamoDB:n kautta. Eli jokainen kerta, kun Gluen ajo ajettaisiin, sen tulisi poistaa nykyinen data, hakea koko taulun datat uudelleen ja kirjoittaa ne uudestaan.

Ratkaisu ei ole pitkäikäinen. Tutkin mahdollisuuksia ratkaista ongelmaa ja vastaan tuli liittää Glue Job DynamoDB streamin päätteeksi. Tämä ratkaisisi, jatkuvasti olemassa olevan datan poistamisen ja hakemisen uudelleen. Ratkaisu olisi hyvä, jos dataa vasta aloitettaisiin säilöä, mutta dataa on säilötty jo vuosien ajan. Stream tukee vain uuden tai poistetun datan siirtämistä putkeen.

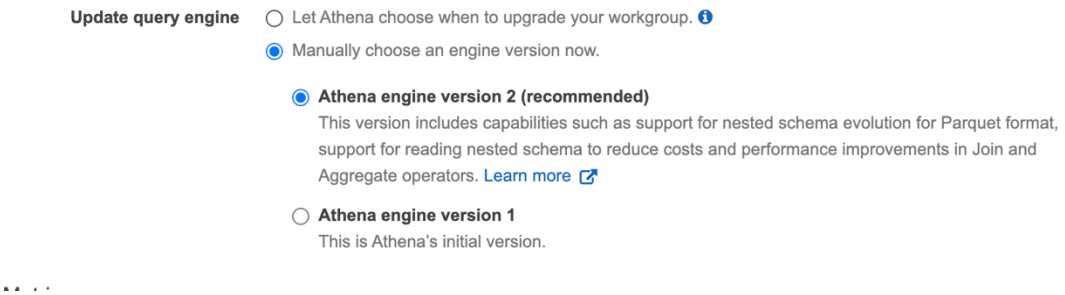
Suunnittelussa on hyvä huomioida ratkaisun uudelleenkäytettävyys muissa projekteissa. Käyttöönotto tulisi aina olemaan erilainen erilaisten datan rakenteiden muodossa. Myös haun optimointi tulisi aina suunnitella ja tarkastella erikseen. Vaikka Glue onkin nopea ja valmis ETL työkalu käyttöönottoon kuluva aika on mahdollinen kompastuskivi.

3.3 Toteutus DynamoDB Connectorilla

Opinnäytetyön alussa oli tiedossa ratkaisuina ongelmaan vain Amazon Glue tai oman ohjelmiston rakentaminen. Vastaan osui kuitenkin blogikirjoitus, joka julkisti Athena

version 2.0 tietyille alueille. Uusi versio puuttuu vielä esimerkiksi Tukholman palvelinkeskukselta. Versio 2.0 on saatavilla vain seuraavilla AWS:n alueilla: ”Asia Pacific (Mumbai), Asia Pacific (Seoul), Asia Pacific (Singapore), Asia Pacific (Sydney), Asia Pacific (Tokyo), Canada (Central), Europe (Frankfurt), Europe (Ireland), Europe (London), Europe (Paris), South America (São Paulo), US East (N. Virginia), US East (Ohio), US West (N. California), and US West (Oregon) Regions.” (AWS www-sivut 2021)

Tämä tieto mahdollisti avoimen lähdekoodin perustuvan ohjelmiston käytön, joka on Athena Federated Query. Työkalusta löytyy monia eri liitäntöjä tietolähteisiin, kuten CloudWatch, Elasticsearch, DynamoDB ja monia muita. Athenan versio tuli päivittää manuaalisesti uudempaan versioon, jotta kyseinen ohjelmisto toimisi. (Kuva 6)



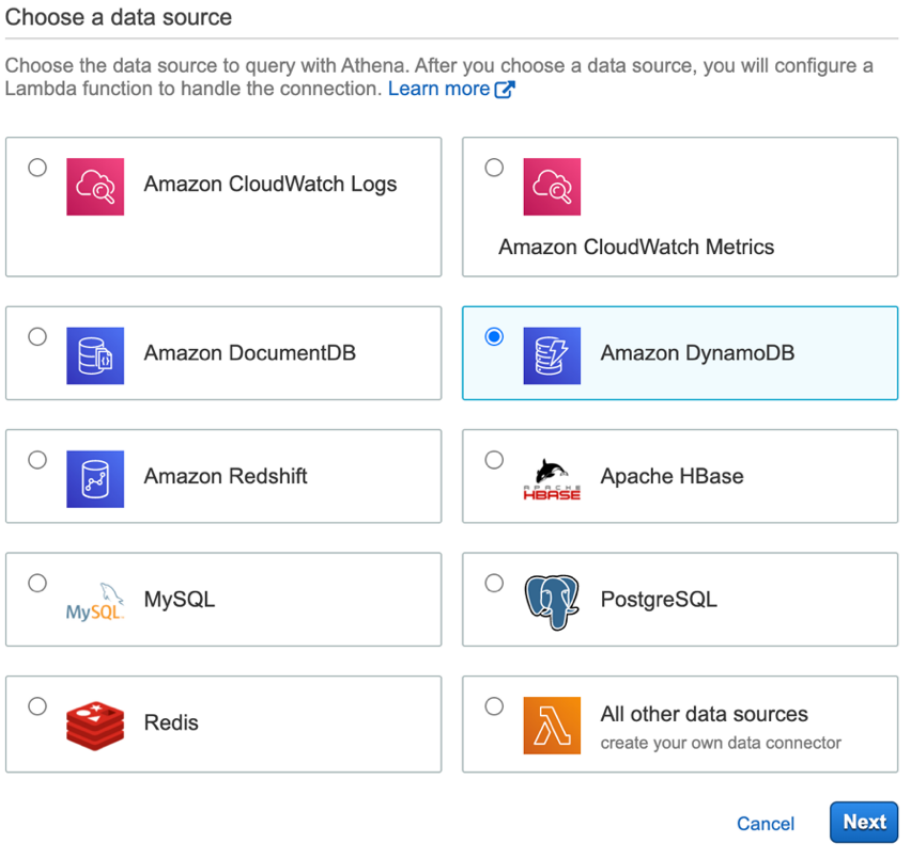
Kuva 6. Athenan versiovalikko.

DynamoDB Connectorin arkkitehtuuri on seuraavanlainen. Käyttäjä pystyttää avoimen lähdekoodin valmiista mallipohjasta lambdan, joka tässä tapauksessa on Java ohjelmointikielellä toteutettu ohjelmisto. Ohjelmisto toimittaa SQL pohjaiset kyselyt DynamoDB:n ymmärtämään syntaksiin, tekee kyselyn ja palauttaa tuloksen Athenalle. (Kuva 7)

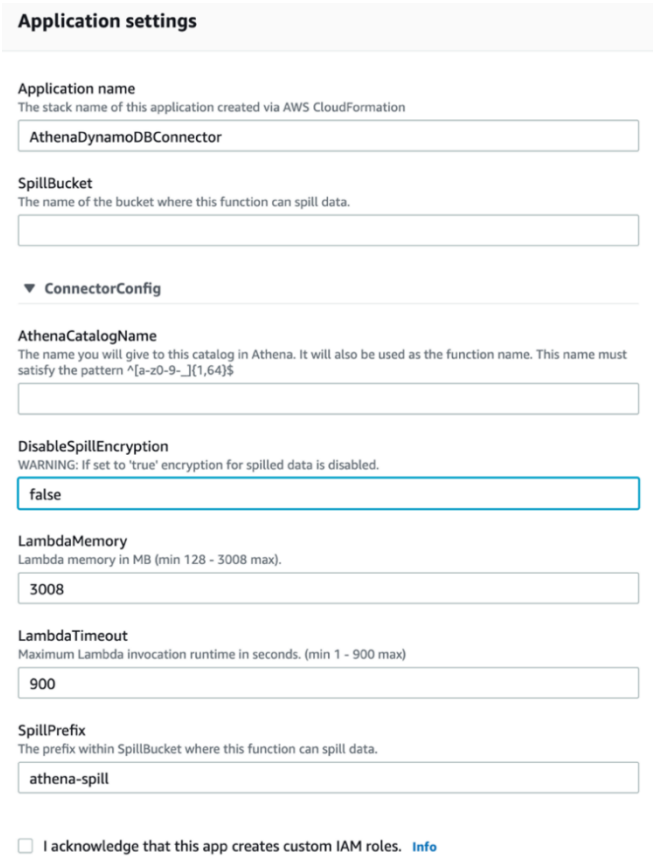
Kuva 7. DynamoDB Connectorin arkkitehtuuri.

Dokumentaatio kyseisestä työkalusta löytyykin ”awslabs/ aws-athena-query-federation” GitHub arkiston alta. Tämän jälkeen käyttöönotto olikin hyvin suoraviivaista. (GitHub awslabs www-sivut 2021)

Athenan valikoista valitaan Data Source, mikä tarkoittaa tietolähdettä. (Kuva 8) Tässä opinnäytetyössä Data Source on DynamoDB.



Kuva 8. Athenan data source valikko.



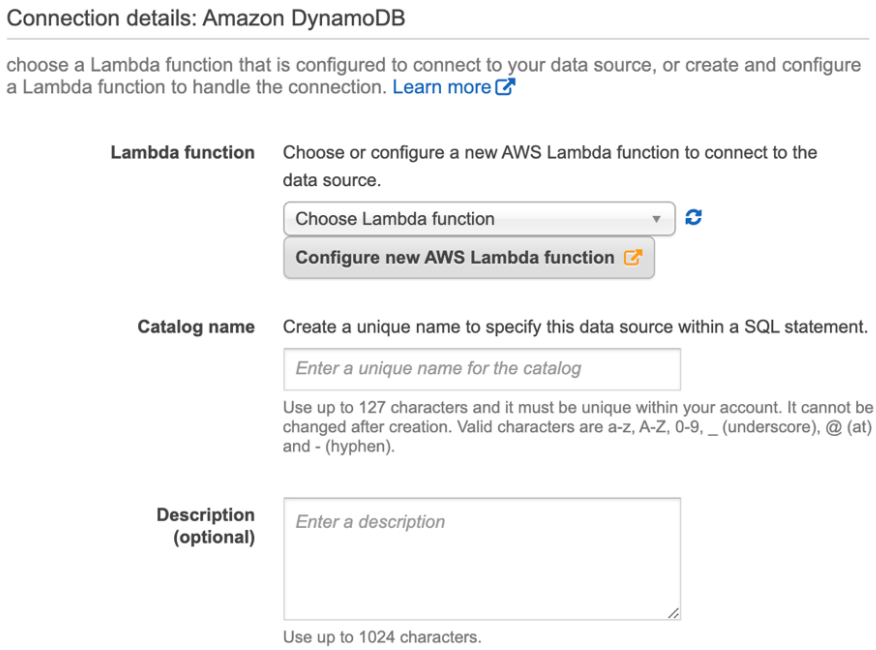
Kuva 9. Lambdan käyttöönotto.

Kuvan 9 selitykset:

- Application name – Sovelluksen nimi, johon lambda liitetään.
- SpillBucket – Väliaikainen datan säilytyspaikka (S3 bucket). Vaaditaan, jotta lambdan sisäinen muisti ei lopu kesken.
- AthenaCatalogName – Athena Catalog nimi, myös funktio nimetään tämän mukaan.
- DisableSpillEncryption – Valinta salataanko datat S3 palvelussa.
- LambdaMemory – Funktiolle varattu muisti, vaikuttaa myös prosessorin tehokkuuteen.
- LamdaTimeOut – Funktiolle varattu aika, jolloin funktion täytyy saada haku toteutettua.
- SpillPreFix – Väliaikaisen datan kansiorakenne S3 ympäristössä. Eli SpillBucket/SpillPrefix/väliaikaiset haut

AWS pystyy suoraan antamaan tarvittavat oikeudet. Lambda tallentaa hakutulokset AWS S3 palveluun. Tulisi siis toteuttaa lifecycle policy AWS S3 palveluun. Lifecycle policy on AWS S3 palveluun kuuluva ratkaisu, jolla voidaan toteuttaa tiedostojen siirtäminen tai poistaminen valitun ajan jälkeen

Kun DynamoDB Connectorille on annettu validit parametrit, funktio julkaistaan. Voidaan valita, että Athena käyttää juuri tätä funktiota haun suorittamiseen. (Kuva 10)



Kuva 10. Athenan Katalogin määrittely valikko, jossa valitaan funktio.

Athenan käyttämä SQL eli Structured Query Language on laajasti käytössä relaatiotietokannoissa, joten ohjelmoijille on tuttua kirjoittaa kyselyitä tietokantoihin kyseisellä kielellä.

Kuva 11. Esimerkki SQL kysely DynamoDB Connectorilla.

3.3.1 DynamoDB Connector

DynamoDB Connector on avoimen lähdekoodin ohjelmisto, eli käyttäjät voivat itse tutustua, kuinka ohjelmisto on rakennettu ja voivat muokata sitä omaan käyttöön sopivaksi.

Tässä opinnäytetyössä oli tarkoitus hakea dataa, mutta Connector ei täysin tukenut DynamoDB:n haun syntaksia, joka johti virheisiin. Tästä johtuen tuli tutustua Connectoriin tarkemmin, mikä johti virheen havaitsemiseen ja korjaamiseen. Tein myös issuen ja pull reviewin avoimen lähdekoodin projektiin.

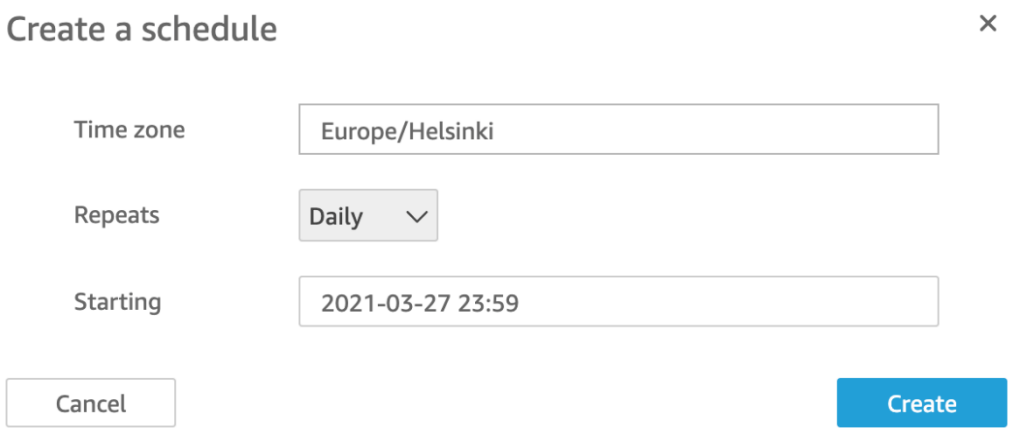
DynamoDB Connectorin kautta tapahtuva kysely kestää kauemmin ja Athenan skannattava datan määrä on suurempi. Tämä tulee ottaa huomioon, jos on odotettavissa, että kyselyitä tapahtuu suuri määrä päivässä.

4 DATAN ANALYSOINTI AWS QUICKSIGHT TYÖKALULLA

Tässä opinnäytetyössä datan analysointiin käytettävä työkalu on AWS QuickSight. Työn rajauksena pidettiinkin datan saattaminen analysointityökalun saataville.

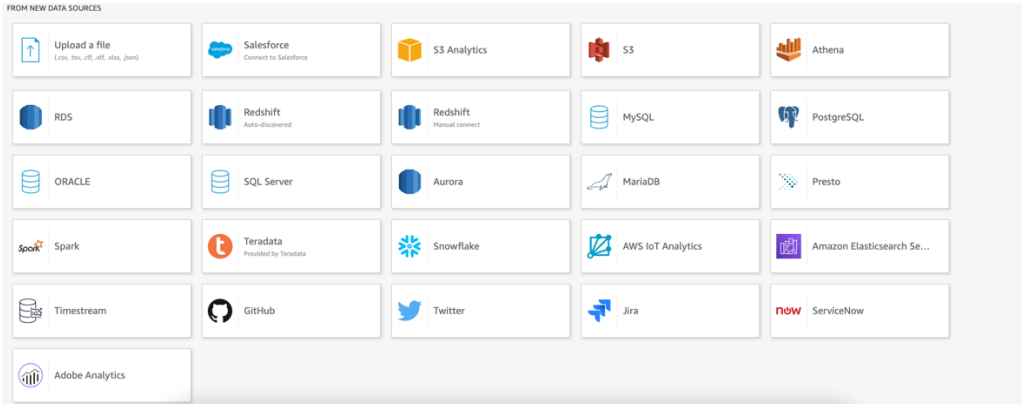
Tässä opinnäytetyössä, toteutettiin valmiiksi sopivia kyselyitä asiakkaiden tarpeiden mukaisiksi. Datan tuominen AWS QuickSight ympäristöön toteutettiin AWS Athenan avulla. Kyselyesimerkkinä on hakea viimeisen kuukauden kaikkein antureiden data aina valmiiksi asiakkaan käytettäväksi. Datan pystyy myös päivittämään manuaalisesti käyttöliittymästä, jos käyttäjä haluaa reaaliaikaisen datan. Automaattisessa hakukyselyssä voidaan käyttää muuttujaa current_date, jolla voidaan hakea nykyisestä ajanhetkestä verrannollinen ajanjakso.

Antureista osa on merkitty asiakkaan toimesta erityisen mielenkiintoisiksi. Näistä antureista toteutettiin erillinen datasetti, joka sisältää päivityshetkestä vuoden datat. Data voidaan päivittää halutun väliajoin. (Kuva 12)



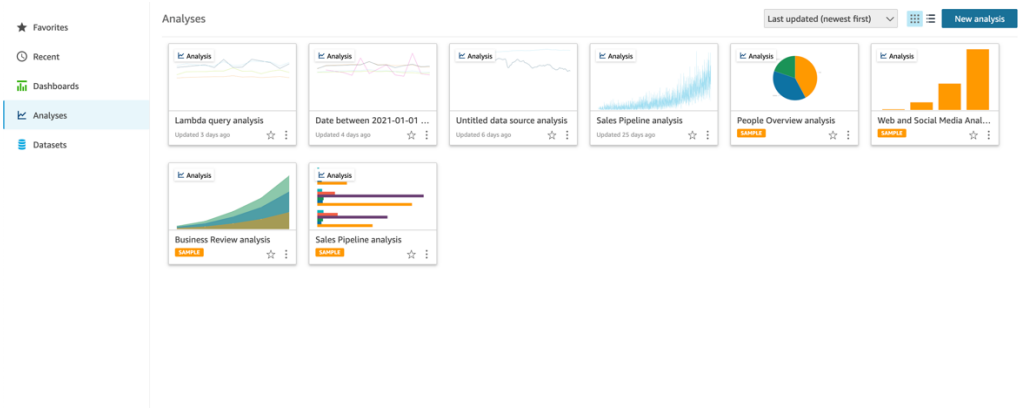
Kuva 12. QuickSightin tarjoama aikataulutus datan päivitykselle.

QuickSight mahdollistaa datan haun monista AWS:n sisällä toimivista datalähteistä. Athenan kautta kysely mahdollistaa suuresta määrästä haettavan datan jakamisen, jolloin analyysien tutkiminen on helpompaa.

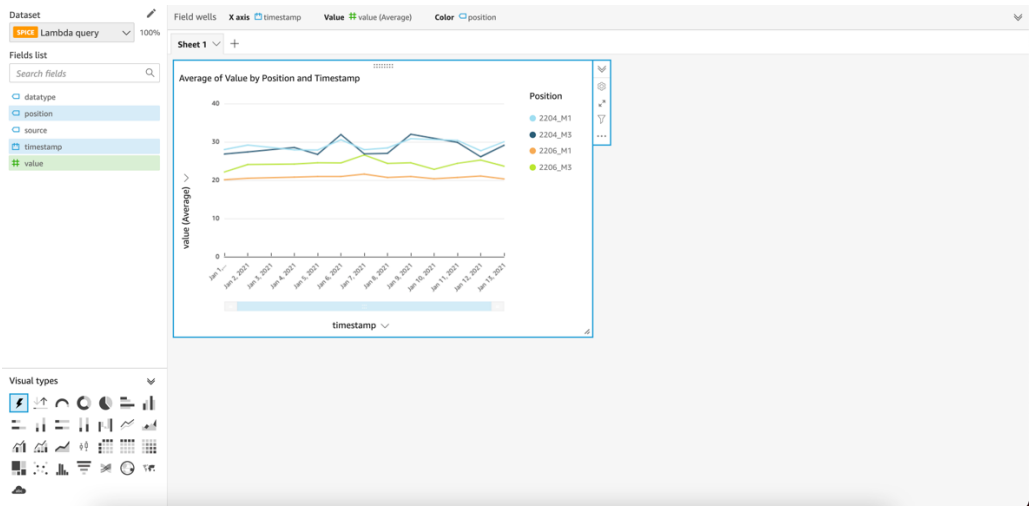


Kuva 13. QuickSightin käyttöliittymä data sourcen lisäämiselle.

Analyyseja on mahdollista tehdä valmiiksi. Analyysit voivat sisältää esimerkiksi ympyrädiagrammeja, pylväs- ja viivakaavioita. (Kuva 14) Analyysit voivat sisältää eri dataa eri lähteistä.

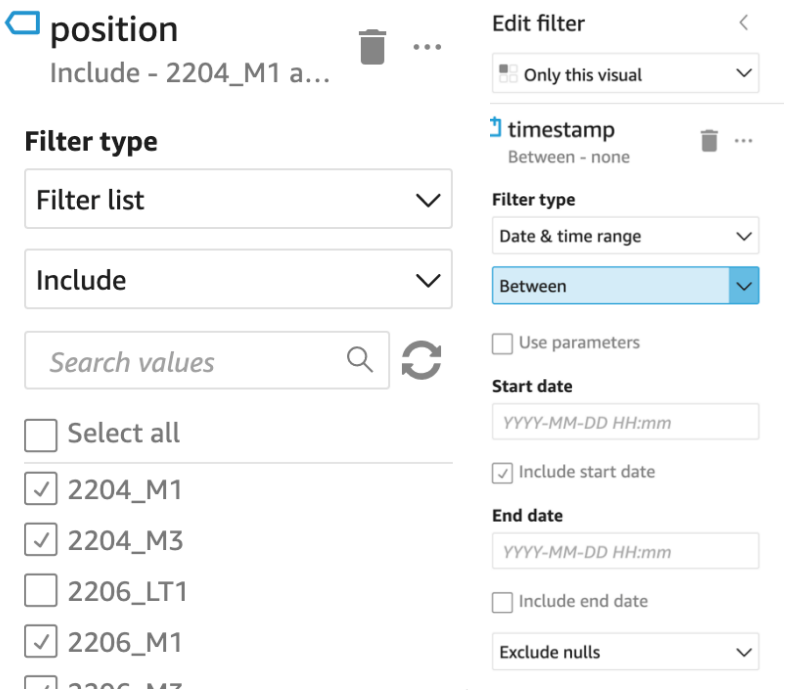


Kuva 14. QuickSight analyyseja.



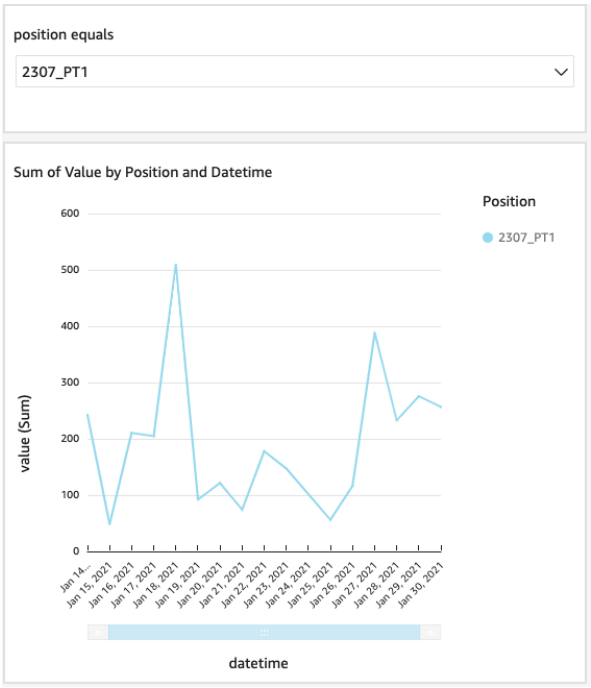
Kuva 15. QuickSight käyttöliittymän analyysi.

Datan rajaaminen on tärkeä osa analyysien tekoa, jotta saadaan ainoastaan haluttu data nähtäville. Itse QuickSightin käyttöliittymä on käyttäjäystävällisempi, kuin SQL-kyselyllä rajaaminen. (Kuva 3, 4 ja 11)



Kuva 16. QuickSight käyttöliittymän esimerkki filttereitä.

Datan rajaaminen voidaan tuoda omana elementtinä analyysille. (Kuva 17) Tällöin voidaan tehdä datan rajaaminen ja analyysien muuttaminen nopeaksi. Myös visuaalisesti nähtävät rajaukset auttavat käyttäjää ymmärtämään muutoksia.



Kuva 17. QuickSight filtteri elementti analyysissä.

5 YHTEENVETO

Tässä opinnäytetyössä tutustuttiin kahteen erilaiseen prosessiin hakea dataa DynamoDB:stä BI-työkalulle. Molemmista prosesseista voidaan tehdä johtopäätöksiä.

ETL ratkaisussa ongelmaksi nousivat datan replikointi, datan pitäminen ajan tasalla alkuperäisen datalähteen kanssa, prosessointiin kuluva aika ja kustannus. Python- ja Spark-ominaisuuksien opettelu ja oikean dataformaatin valitseminen. Dataformaatin valitsemisen jälkeen täytyi data suunnitella käyttökohteen mukaisesti, jotta kyselyt toimivat optimoidusti. Lopulta, kun järjestelmä saatiin toimimaan kyselyt toimivat nopeasti ja olivat kustannustehokkaita.

Valmiin avoimen lähdekoodin DynamoDB Connectorin kanssa ongelmaksi nousivat ohjelmiston keskeneräisyys. Ohjelmiston virheen etsiminen, tunnistaminen lähdekoodista ja muutoksen tekeminen. Myös lambdan RAM-muistin rajoitukset ja 15 minuutin aikaraja voivat nousta ongelmiksi, mikäli käsiteltävän datan määrä on suuri eikä hakua

ei ole mahdollista optimoida. Hakutapahtuma on myös hitaampaa ja kalliimpaa verrattuna AWS Glue toteutukseen. Tämä voidaan välttää tekemällä vakiokyselyitä, joilla on tietty päivitysaika esimerkiksi keskiyöltä. Yksittäisten kyselyjen kustannusten nousu on tässä opinnäytetyössä datan replikointia halvempaa. Prosessin vieminen muihin DynamoDB tietokantaa hyödyntäviin projekteihin on yksinkertaista, sillä data pysyy samana.

Yhteenvetona voisi pitää, että jos käyttäjiä on useampi ja kyselyitä tapahtuu suuri määrä päivässä, olisi kohtuullista muuttaa data kyselyihin optimoituun formaattiin esim. Parquet. Jos kyselyiden suorittaminen on kuitenkin satunnaista ja harvoin suoritetaan uniikkeja kyselyitä, on parempi jättää data replikoimatta ja hakemalla suoraan tietokannasta.

Kyseisessä työssä tuli myös vastaan, että ongelmaan on todella monia ratkaisuja. Jokainen niistä on varteenotettava vaihtoehto omanlaiseen käyttötarpeeseen. Alussa olikin vaikeuksia ymmärtää erilaisten ratkaisujen hyödyt ja ongelmat, kun aikaisempi kokemus datan analysointityökaluista oli vähäistä. Tässä työssä oli välttämättä paneuduttava moniin ratkaisuihin, jotta yleiskuva tulisi selväksi.

Projektissa tuli monia vaiheita, jotka opettivat paljon nykypäivän ohjelmistokehityksen osista. Tärkeimpänä nostaisin esiin ongelman määrittelemisen. Koodin ymmärtäminen on myös isossa osassa, kun käytetään ja muokataan valmiita työkaluja.

Työ oli merkittävä tilaajalle, sillä datan säilöminen DynamoDB:ssä on osana monia projekteja. Nyt selvitettiin kahden eri mahdollisuuden hyötyjä ja haittoja, joita voidaan pitää dokumentaationa muidenkin projektien analysointimallien havainnollistamisessa.

Tämän opinnäytetyön aikana sain hyvää kokemusta tiedon analysointiin perustuvan ohjelmistokehityksen vaiheista ja työkaluista. Olen tyytyväinen työn lopputulokseen ja uskon tästä kokemuksesta olevan hyötyä myös tulevaisuudessa.

LÄHTEET

Apache Software Foundation, Apache Parquet Viitattu 30.03.2021 https://parquet.apache.org/documentation/latest/

AWS 2021, Amazon Athena features Viitattu 30.03.2021 https://aws.amazon.com/athena/features/?nc=sn&loc=2

AWS 2021, Amazon DynamoDB features Viitattu: 30.03.2021 https://aws.amazon.com/dynamodb/features/

AWS 2021, AWS Lambda Features Viitattu: 30.03.2021 https://aws.amazon.com/lambda/features/

AWS 2021, AWS Glue features Viitattu 30.03.2021 https://aws.amazon.com/glue/features/

AWS 2021, Amazon QuickSight features Viitattu 30.03.2021 https://aws.amazon.com/quicksight/features/

AWS 2021, Introduction to Presto Viitattu 30.03.2021 https://aws.amazon.com/big-data/what-is-presto/

AWS 2021, What is AWS Viitattu 24.04.2021 https://aws.amazon.com/what-is-aws/

GitHub 2021, Athena Query Federation – DynamoDB Viitattu 30.03.2021 https://github.com/awslabs/aws-athena-query-federation/tree/master/athena-dynamodb

IBM 2021, Cloud Computing Viitattu 08.04.2021 https://www.ibm.com/cloud/learn/cloud-computing

IBM 2021, ETL explained Viitattu 12.04.2021
https://www.ibm.com/docs/en/spm/7.0.11?topic=explained-etl

NIST 2021, Cloud Computing Viitattu 24.04.2021
https://csrc.nist.gov/projects/cloud-computing