
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

ATTE PELTOLA

# Tietovarastoratkaisun pilotointi

TIETOJENKÄSITTELY TUTKINTO-OHJELMA
2023

TIIVISTELMÄ

Peltola, Atte: Tietovarastoratkaisun pilotointi
Opinnäytetyö, AMK
Tietojenkäsittelyn tutkinto-ohjelma
Joulukuu 2023
Sivumäärä: 37

Tämän opinnäytetyön toimeksiantajana toimii Teollisuuden Voima Oyj. Työn tarkoituksena oli arvioida ja testata NoSQL-tietovarastoratkaisuja, jonka avulla tavoiteltaisiin tietokannan kykyä datan käsittelyyn tietovarastossa. Työn lopputuloksena dokumentoitiin NoSQL-tietokantojen ominaisuuksia, jotka edistävät projektin toteuttamista. Osana työnlopputulosta tuotin tiivistelmätaulukon, joka sisältää yhteenvedon tietokantojen ominaisuuksista.

Työn tavoitteena oli perehtyä NoSQL-tietokantojen tietokantamalleihin ja ohjelmiin. Suunnittelu vaiheessa mallinnettiin NoSQL-tietokantoja, jotka vastasivat toimeksiantajan vaatimuksiin. Tietokanta-ohjelmista valittiin kaksi ohjelmaa testattavaksi. Testattaviksi tietokannoista valitsin dokumentti pohjaisen MongoDB tietokannan ja aikasarjatietokannan InfluxDB:n. Ohjelmistojen valinnan jälkeen alkoi pilotointi vaihe, joka alkoi tietokantojen käyttöönotolla. Työn edetessä pääsin arvioimaan tietokantojen suorituskykyä, API-rajapintaa ja tietoturvallisuutta. Lopussa käytiin läpi pilotoinnin kokonaisuus ja havaitut asiat läpi, jonka jälkeen laadin projektista tiivistelmätaulukon.

Avainsanat: NoSQL-tietokanta, InfluxDB, MongoDB, Dokumentti pohjainen tietokanta, Aikasarjatietokanta

Abstract

Peltola, Atte: Piloting a Data Warehouse solution
Bachelor's thesis
Business Information Systems
December 2023
Number of pages: 37

This thesis was commissioned by Teollisuuden Voima Oyj. The purpose of the project was to assess and test NoSQL data warehouse solutions aimed at enhancing a database's capability for handling data in the data warehouse. The outcome of the work documented the features of NoSQL databases that facilitate the implementation of the project. As part of the result, I produced a summary table providing an overview of the database features.

The goal of the project was to familiarize with the database models and programming of NoSQL databases. During the planning phase, NoSQL databases were modeled to meet the client's requirements. Two database programs were selected for testing. I chose to test a document-based MongoDB database and a time-series database InfluxDB. After selecting the software, the pilot phase begun with the implementation of the database. As the project progressed, I assessed the performance, API interface and security of the databases. At the end I reviewed the databases performance, followed by the creation of a summary table summarizing the project.

Keywords: NoSQL databases, InfluxDB, MongoDB, Document based database, time series database

# ALKUSANAT

Tahtoisin kiittää toimeksiantajaa Teollisuuden Voima Oyj:tä aiheesta ja ohjaajiani Jussi Niemistä ja Jarno Sarinia tuesta ja vinkeistä opinnäytetyötä kohden.

SISÄLLYS

# SYMBOLI- JA LYHENNELUETTELO (EI PAKOLLINEN)

| | |
|---|---|
| TVO | Teollisuuden Voima Oyj |
| SQL | Relaatiotietokanta tiedon tallentamiseen ja käsittelyyn |
| NoSQL | Tietokantajärjestelmä, joka on suunniteltu käsittelemään erilaisia tietomalleja |
| API | Application Programming Interface eli ohjelmointirajapinta, jonka avulla ohjelmat toimivat keskenään |
| JSON | JavaScript Object Notation tiedostomuoto, joka perustuu JavaScript-ohjelmointikielen |
| Python | Ohjelmointikieli |
| GUI | Tulee sanoista Graphical User Interface ja tarkoittaa graafista käyttöliittymää |
| CLI | Tulee sanoista Command Line Interface ja tarkoittaa tekstipohjaista käyttöliittymää |
| CSV | Tulee sanoista Comma-separated Values ja tarkoittaa tiedostotyyppiä, johon tallennetaan tietoa taulukkomaisesti |

# 1 JOHDANTO

Tämän opinnäytetyön tavoitteena on tutkia ja pilotoida Teollisuuden Voima Oyj:lle (TVO) avoimen lähdekoodiin perustuva NoSQL-tietokanta, jonka tehtävänä on vastaanottaa ja hallinnoida Olkiluoto laitoksilta tulevaa laitostietoa. Tässä työssä ei kuitenkaan käsitellä laitoksilta olevaa tiedonsiirtoyhteyttä, koska tarkoituksena on hankkia taustatietoa ja osaamista ennen kuin tiedonsiirtoyhteydet aikanaan aukeavat. Työssä tehdään tietokantaratkaisu, jonka tarkoituksena on näyttää kuinka tietovarasto toimisi loppukäyttäjille. Tietokantaratkaisuista on tarkoitus tehdä tiivistelmätaulukko, jonka tavoitteena on luoda tietokantojen ominaisuuksista yksi kokonaisuus, jotta tietokannan valinta olisi helpompaa.

Tässä opinnäytetyössä käydään ensimmäisenä läpi TVO yrityksenä yleisesti, jonka jälkeen käydään läpi NoSQL-tietokanta yleisesti ja niiden tietokantamalleja. Teoriaosion jälkeen käydään läpi toimeksiantajan nykytilannetta, jonka jälkeen lähdetään tutkimaan ja pilotoimaan valitsemieni tietokantojen nopeutta, suorituskykyä ja rajapinnallisuutta. NoSQL-tietokantaohjelmia on nykypäivänä paljon, joten tässä työssä aihe on rajattu kahteen tietokantaan, sillä aihe olisi liian laaja useammalle tietokantavertailulle. Työn edetessä käydään läpi yhteenveto molemmista tietokannoista ja havaintoja tietokantojen ominaisuuksista. Viimeiseksi tietokantojen ominaisuuksia vertaillaan tiivistelmätaulukossa.

# 2 TOIMEKSIANTAJA (TVO)

Opinnäytetyön toimeksiantaja, Teollisuuden Voima Oyj (TVO) on ydinvoimalla sähkö tuottava yhtiö. TVO on vuonna 1969 perustettu ja on listaamaton julkinen osakeyhtiö. TVO:lla on Eurajoen Olkiluodossa ydinvoimalaitoksia, jossa sijaitsee laitosyksiköt Olkiluoto 1, 2 ja 3. Olkiluoto 1 ja 2-laitosyksiköt on rakennettu aikanaan tyydyttämään Suomen sähköntarvetta. Laitosyksiköt Olkiluoto 1 ja 2 ovat identtisiä kiehutusvesireaktoreita. Olkiluoto 3 -laitosyksikön rakennushanke alkoi vuonna 2005 ja tuotti ensimmäistä kertaa sähköä valtakunnalle koekäyttöohjelman aikana, joka päättyi huhtikuussa 2023. Olkiluoto 3 -laitosyksikkö valmistui vuonna 2023 ja on Suomen suurin ydinvoimala ja sen kautta TVO tuottaa Suomen sähköntuotannosta lähes kolmanneksen (TVO, n.d.-a)

Teollisuuden Voima -konserniin kuuluvat myös TVO Nuclear Services Oy (TVONS) sekä ydinjätehuoltoyhtiöön keskittyvä Posiva Oy että Posiva Solutions Oy. TVONS on kokonaan TVO:n omistama tytäryhtiö ja Posiva Oy on 60 % TVO:n omistama yhteisyhtiö Fortum Oy:n kanssa. Posiva Solutions Oy on kokonaan Posiva Oy:n omistama yhtiö (TVO, n.d.-b)

# 3 NOSQL

## 3.1 Yleistä

NoSQL, tunnetaan myös nimillä ”Not only SQL” tai ” non-SQL”, on relaatiotietokantaan perustuva tietovarasto, joka mahdollistaa relaatiotietokannan perinteisen tallennuksen ja kyselyn rakenteiden eriyttämisen. Sen avulla dataa voidaan tallentaa, muotoilla ja siitä voidaan tehdä kyselyjä. NoSQL-tietokantojen tarkoituksena on sallia uusia erilaisia tapoja tallentaa dataa tietovarastoon käyttämättä yleisintä relaatiotietokantaa ja sen kaavoja. NoSQL-tietokannat luotiin vastaamaan rajoituksiin ja tietyntyyppisten tietojenkäsittelyyn, jotka aiheuttavat skaalautuvuushaasteita perinteisessä relaatiotietokannassa.

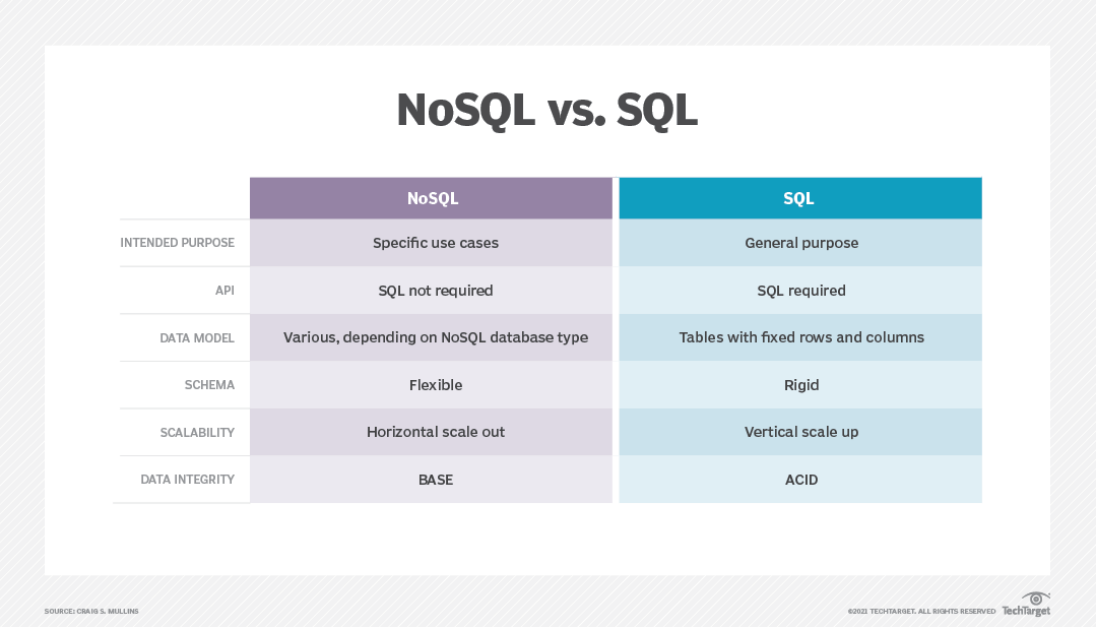
Erilaisia NoSQL-tietokantoja on tästä syystä paljon ja jokaisella NoSQL-tietokannalla on ainutlaatuinen lähestymistapa tietojen tallentamiseen. Uusien lähestymistapojen avulla NoSQL-tietokannat luovat monipuolisen tavan lisätä tietoa. NoSQL-tietokannan käyttö on yleistynyt teknologian kehittyessä, sillä nykypäivänä datan määrä ja sen tärkeys on suuressa kasvussa. NoSQL-tietokannat antavat yrityksille ja käyttäjille mahdollisuuden tallentaa dataa, mitä ei ole ollut mahdollista perinteisessä relaatiotietokannassa. Näiden lisäksi NoSQL-tietokantoihin saatetaan myös siirtyä, koska ne tarjoavat nopeuden ja skaalautuvuuden massadatan käsittelyyn. (IBM. n.d.)

## 3.2 NoSQL- ja SQL-tietokantojen eroavaisuudet

Nykypäivän tietovarastoratkaisun valinnassa on yritysten ja käyttäjien tiedettävä tietokantojen eroavaisuuksista. Käyttötarve tietovarastoille määrittelee eroavaisuudet. Ensimmäinen ja suurin päätös nykypäivän tietokantaratkaisun valinnassa on tehtävä NoSQL- tai perinteisen SQL-tietokannan välillä. Molemmat järjestelmät tarjoavat käyttäjille ainutlaatuisia etuja ja vastaavat erilaisiin tarpeisiin. Järjestelmän käyttötarpeet tulevat kokonaisuudessaan yritysten datan muodosta ja laajuudesta. Mikäli data on jäsentämätöntä kuten dokumenteissa, NoSQL-tietokanta on kannattavampi valinta, koska monet NoSQL-tietokannat tarjoavat joustavan tietokantamallin. Yritysten on myös hyvä tarkastella entisiä ratkaisujaan ennen tietovaraston valintaa, sillä vanhemmat järjestelmät eivät tue NoSQL-tietokantoja, jolloin perinteinen relaatiotietokanta on hyvä valinta tietovarastolle. (Smallcombe, 2023)

Eroavaisuuksia NoSQL- ja SQL-tietokantojen välillä on siis monia. Nämä eroavaisuudet määrittävät järjestelmän käyttötarpeen. Suurimmat eroavaisuudet tulevat joko niiden käyttötarkoituksesta, tietomallista, skeemasta, skaalautuvuudesta tai tietojen eheydestä (Kuva 1.). Ensimmäisenä isona eroavaisuutena on se, että relaatiotietokanta käyttää toimiakseen kiinteää tietomallia (rigid schema), johon tietomalli tallennetaan kiinteisiin tauluihin. Taulun sarakkeet ja rivit ovat rakenteeltaan aina samanlaiset ja eivät erotu toisistaan. NoSQL-tietokannoissa tietoa ei tallenneta taulukoihin. Sen sijaan se tallentaa

tietomallin esimerkiksi dokumentteihin tai avainarvopareihin. NoSQL-tietokannat ovat siis dynaamisia ja sallivat eroavaisuuksia tietojen välillä, kun taas relaatiotietokanta käyttää kiinteää mallia. Toisena isona eroavaisuutena on tietokantojen skaalautuvuus. Skaalaus tarkoittaa järjestelmän kykyä kasvaa tai pienentyä, jotta pystyy vastaaman vaatimuksiin. NoSQL-tietokannoissa käytetään vaakasuuntaista skaalautuvuutta. Tämä tarkoittaa sitä, että palvelimeen lisätään lisää samanlaisia resursseja. Relaatiotietokannoissa käytetään pystysuuntaista skaalautuvuutta eli päivitetään itse palvelinta, kuten lisäämällä suorituskykyä. (Mullins, 2021-a)

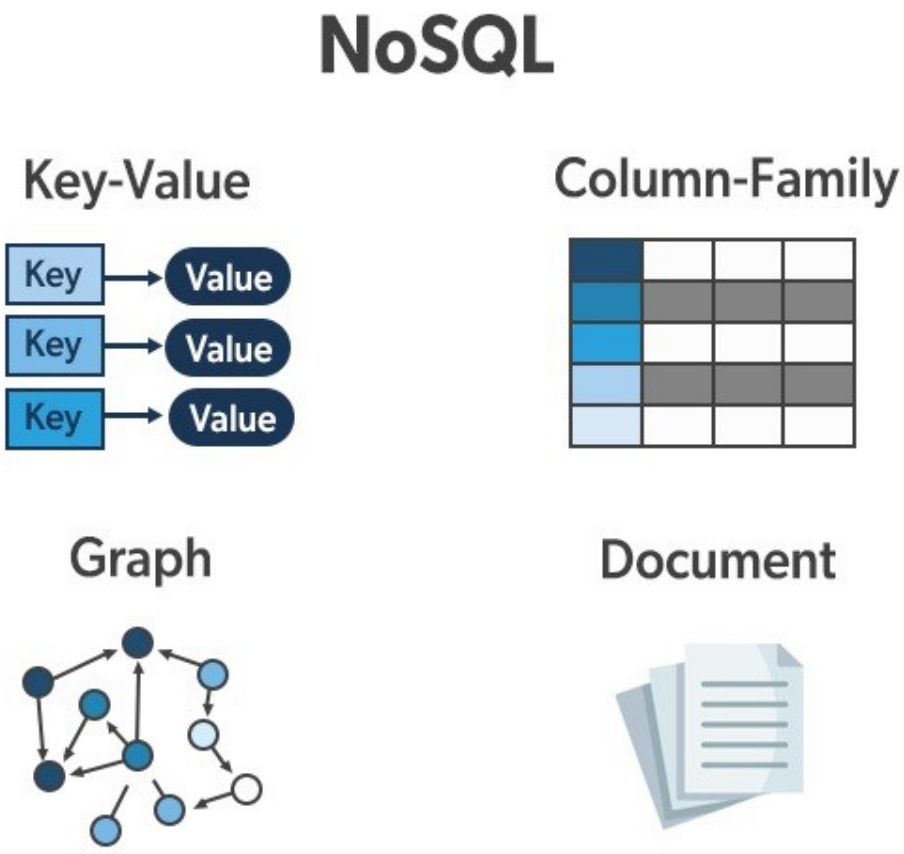



Kuva 1. NoSQL- ja SQL-tietokantojen eroavaisuudet tiivistettynä. (Mullins, 2021-a)

## 3.3 NoSQL:n tietokantamallit

NoSQL-järjestelmät tarjoavat monipuolisia tietokantamalleja, jotka mahdollistavat datan soveltamisen tehokkaasti ja joustavasti omiin käyttötarpeisiinsa. Monipuolisten tietorakenteiden kautta NoSQL-tietokantoja pystytään soveltamaan yhä enemmän data-analytiikkaan, massadatan käsittelyyn tai sovellusten kehittämiseen. NoSQL-järjestelmät toimivat usein yhden tietokantamallin perustana, jonka pohjalle on rakennettu tietojärjestelmä. Nämä tietomallit jaetaan päätyyppeihin, jotka ovat asiakirja-, avain-arvo-, sarakeperhe-, graafinen-

tai aikasarjatietomallit (Kuva 2). Jokaisella tietomallilla on omat ainutlaatuinen tapa tallentaa ja tuottaa kyselyitä järjestelmän käyttäjille. monipuolistaen datan käyttöä suuresti. (Oracle. n.d.)




Kuva 2. NoSQL-tietokannan yleisimmät tietokantamallit (Geeks for geeks, 2023.)

### 3.3.1 Avain-arvo tietokannat

Avain-arvo tietokanta (Key-Value database) käyttö perustuu avain-arvomenetelmään. Avain-arvomenetelmää käytetään esimerkiksi silloin, kun organisaatiossa halutaan tallentaa sovelluksen istuntoja, jotka ovat vaatineet käyttäjältä kirjautumisen. Tässä esimerkissä käyttäjän sisäänkirjautuminen kirjattaisiin järjestelmään ja käyttäjän uloskirjautuminen tallennettaisiin tietovarastoon. Avain-arvomenetelmä, joka tallentaa tiedon järjestelmään on nopea tallentamaan ja saamaan dataa samanaikaisesti. Avain-arvo-tietokanta ei kuitenkaan

sovellu muihin kuin avainperustaisiin hakuihin. Tietokannan nopeus on yksi parhaista ominaisuuksista, jota avaimeen perustuvalla tietokannalla on. Tämän lisäksi avain-arvon yksinkertaisuus, skaalautuvuus ja luotettavuus ovat pääominaisuuksista, mitä avaimeen perustuva tietokanta tarjoaa organisaatioille. Avain-arvo tietomalliin perustuvia tietokantoja on monia, mutta yksi suosituimmista avoimen lähdekoodiin perustuva avainarvo tietokanta on Redis. (Williams, 2021)

### 3.3.2 Asiakirjatietokannat

Asiakirjatietokanta (Document database) toiselta nimeltään dokumenttivarasto (Document store) luokitellaan monipuoliseksi tietokannaksi, joka tallentaa saamansa tiedot asiakirjoihin. Asiakirjat eli dokumentit voidaan tallentaa asiakirjatietokantaan monella erilaisella dokumenttityypeillä, jotka ovat esimerkiksi JSON- (JavaScript Object Notation) tai XML- (Extensible Markup Language) tiedostotyyppiä. Asiakirjoissa data on yleensä tallennettu siten, että se sisältää tietoja yhdestä objektista ja objektiin liittyvistä ominaisuuksista. Objektin tiedot saattavat sisältää myös tietoa suhteista toisiin objekteihin. Nämä objektit tallennetaan tietojärjestelmän eri kokoelmiin. Kokoelmat sisältävät ryhmän dokumentteja, jossa voidaan tehdä kyselyitä. Asiakirjaan perustuva tietokannan yksi suurimmista hyödyistä on mallin joustavuus. Joustettavuuden takia kokoelma ei vaadi käyttäjältä samoja kenttiä eli objektin ominaisuuksia tai suhteita tallennettaessa dokumenttia tietokantaan. Dokumenttipohjaisen tietokannan hyötyihin kuuluu sen helppokäyttöisyys, sillä se mahdollistaa tietojen mallinnuksen ilman ennalta määriteltyä skeemaa. Vaikka asiakirjatietokannan hyötyihin kuuluvat sen joustavuus ja helppokäyttöisyys, sen soveltuvuus riippuu useista tekijöistä. Asiakirjatietokanta ei välttämättä sovi parhaiten, kun käsittelyssä on pienet määrät tietoa. Dokumenttipohjaiseen tietokantaan löytyy monta erilaista ohjelmaa esimerkiksi avoimeen lähdekoodiin perustuva MongoDB. (MongoDB, n.d.-a)

### 3.3.3 Sarakeperhetietokannat

Sarakeperhetietokanta (Column-family database) on NoSQL-tietomalli, jossa tietokannan sarakkeiden nimet ja muodot voivat muuttua eri riveiltä ja jopa samasta taulukosta. Sarakeperhetietokanta erottautuu normaalista relaatiotietokannasta siten että sarakeperhetietokanta lukee datan sarakkeet itsenäisesti ja jokainen sarake pitää sisällään avaimen, jonka kautta sarakkeita luetaan nopeasti ja kätevästi. Sarakeperhetietokannat ovat loistavia laajaan ja hajautettuun tallennustarpeeseen. Vaikka sarakeperhetietokanta ja avaimiin perustuva tietokanta kuulostavat samanlaisilta, löytyvät tietokannoista eroavaisuuksia. Avaimiin perustuva tietokannassa jokainen avain sisältää vain yhden arvon, kun taas sarakeperhetietokannassa jokainen avain voivat liittyä mahdollisesti useampaan sarakkeisiin, jotka sisältävät enemmän kuin yhden arvon. Huomioitavaa on se, että sarakeperhetietokannat eivät sovellu ratkaisuksi, kun kyseessä on pieni tai keskikokoinen tietokanta. (ScyllaDB, n.d.)

### 3.3.4 Graafiset tietokannat

Graafinen tietokanta (Graph Database) on tietokantamalli. Graafisen tietokannan (Graph database), jonka käyttö pohjautuu hyvin monimutkaisten suhteiden käsittelyyn tai verkostoituneiden tietojen läpikäyntiin. Ne on suunniteltu tallentamaan ja näyttämään tietojen väliset suhteet kätevästi. Tämän avulla nämä tietokannat tarjoavat organisaatioille tehokkaita ja skaalautuvia ratkaisuja monimutkaisten datan suhteiden käsittelyyn. Yleisiä käyttökohteita ovat monimutkaiset tietorakenteet, jotka on liitetty toisiinsa. Suosittuja graafisia tietokantoja ovat esimerkiksi Neo4j tai Amazon Neptune. (Ghazaryan, 2023)

Graafiset tietokannat ovat hyvin monipuolisia ja käteviä työkaluja. Graafisuus sallii käyttäjien keskittyä tietojen syvempiin suhteisiin. Tietokannan etuna on myös visualisoitavuus, sillä sen avulla on helpompaa tulkita tietoja. Vaikka visualisoitavuus nähdään etuna, on se myös kalliimpaa käyttää ja ylläpitää. Graafiset tietokannat eivät myöskään sovellu tietovarastoksi, jos tietojen väliset suhteet ovat vähäisiä. (Oracle, n.d.)

### 3.3.5 Aikasarjatietokannat

Aikasarjatietokanta (Time-Series database) on yleistynyt tietokantamalli, jonka käyttötarve on erityisesti suunniteltu aikaleimattujen tai aikasarjatietojen käsittelyyn. Aikasarjalla tarkoitetaan yleisesti jatkuvan muuttujan arvon muuttumista aikajanalla. Nämä voivat olla esimerkiksi sovellusten suorituskyvyn mittaukset, verkkotiedot tai anturitiedot. Aikasarjatietokantojen hyödyt tulevat perinteisen relaatiotietokannan toiminnasta, sillä relaatiotietokanta ei ole suunniteltu käsittelemään aikasarjallista tietoa. Tehokkuuksien ansiosta aikasarjatietokannat pystyvät tarjoavat massiivisen suorituskyvyn parannuksia. Nämä suorituskykyparannukset näkyvät tiedonottonopeuksista ja kyselyiden kyvykkyydestä. Aikasarjatietokannat suosio on nousussa, sillä yritysten investoinnit data-analytiikkaan ovat kasvussa. Aikasarjatietokannat antavat organisaatioille sekä näkemyksiä historiallisista että reaaliaikaisista tiedoista, joita pystytään käyttämään liiketoiminnan hyödyntämiseen. Aikasarjatietokannan valinnassa on kuitenkin hyvä huomioida, että ne ovat erityisesti suunniteltu käsittelemään aikasarjatietoja. (Timescale, 2023)

## 3.4 NoSQL-tietokantoja

### 3.4.1 MongoDB

MongoDB on avoimeen lähdekoodiin perustuva dokumenttipohjainen tietokanta. MongoDB on perustettu vuonna 2007 ja on saatavilla kolmena eri tuoteversiona: Enterprise, Community ja Atlas. Kaikki tuoteversiot tarjoavat sekä yrityksille että käyttäjille joustavan ja laadukkaan tietokantakokemuksen. Joustava dokumenttimalli tallentaa tiedot JSON-tyyppisiin dokumentteihin, mikä tarkoittaa sitä, että tiedot vaihtelevat toisistaan eikä niiden tietorakenne tarvitse olla samanlainen. Näitä dokumentteja kutsutaan myös nimellä BSON. BSON tulee sanoista Binary JSONista. JSON-tyyppinen tallennusmuoto on yksi MongoDB hyödyistä, sillä BSON sallii tietojen siirtämisen nopeasti ohjelmistokehittämiseen. Tämä lisää tietokannan rajapintojen hyödyntämisen mahdollisuuksia. Tietorakennetta pystytään myös tarvittaessa muuttamaan ajan myötä

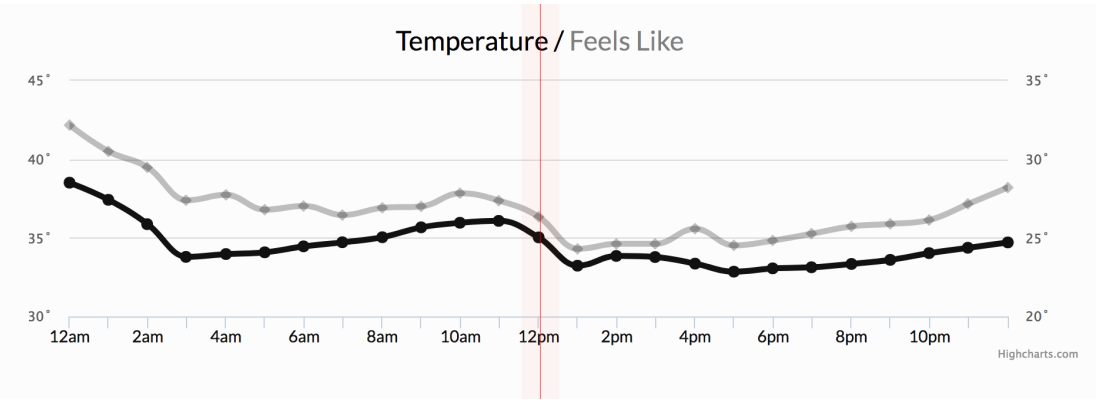
sopivaksi omaan käyttöä varten. MongoDB on luokiteltu yksi suosituimmista NoSQL-tietokannaksi, koska tietokanta on hyvin suunniteltu käyttäjä ystävälliseksi ja tarjoaa käyttäjilleen parhaimman mahdollisuuden tallentamaan tiedon korkealla käytettävyydellä. MongoDB käyttää tietokantakyselyissään MQL-kyselykieltä, joka pohjautuu JavaScript-ohjelmointikieleen. JavaScriptiin pohjautuva kyselykieli, helpottaa kyselyiden tekemisissä. MongoDB on saatavilla omana pilvipalveluna nimeltään MongoDB Atlas, jossa tietokanta toiminta perustuu pilvimahdollisuuksiin. MongoDB tarjoaa myös Enterprise ja Community versioista paikallisen palvelimen. (Gillis, 2023)

### 3.4.2 InfluxDB

InfluxDB on InfluxDatan vuonna 2013 julkaistu avoimeen lähdekoodiin perustuva aikasarjatietokanta. InfluxDB tarjoaa myös käyttäjille pilvipohjaisia tietokantaratkaisuja mukaan lukien datan integroimista ja visualisointia. InfluxDB:n tämänhetkinen versio 2.0, on kirjoitettu uudella Flux-ohjelmointikielellä. Flux on InfluxDatan avoimen lähdekoodin perustuva oma projekti ja on pääsääntöisesti tarkoitettu aikasarjatietokantoihin. Ohjelmointikieli perustuu suosittuun JavaScript-ohjelmointikieleen, joka on helppo oppia ja laajentaa. Yksi Flux-ohjelmointikielen merkittävä ominaisuus on se, että sen avulla tietokanta voidaan integroida erilaisten kolmannen osapuolen API-sovellusten kanssa. InfluxDB tarjoaa myös organisaatioille oman Enterprise-version, joka sisältää yrityksille tietokannan ylläpitoa ja kulunvalvontalaitteita. InfluxDB Enterprise on saatavilla omana pilvipalveluna tai paikallisena palveluna. (Technical matters, 2023)

InfluxDB tarkoituksena on tallentaa aikasarjatietoa (Kuva 3). Aikasarjatietokantoja käytetään yleensä esimerkiksi IoT-laitteiden kanssa, koska IoT-laitteet voivat lähettää ajallista tietoa, jonka avulla voidaan tehdä toimenpiteitä riippuen tilanteesta. Aikasarjatietokantoja käytetään myös yrityksissä reaaliaikaiseen data-analysointiin ja laitteiden seurantaan. Verrattuna tavallisiin relaatiotietokantoihin InfluxDB tarjoaa käyttäjille selkeitä nopeuteen liittyviä etuja mittaustietojen tallentamiseen ja sen käsittelyyn. InfluxDB etuihin kuuluvat

samalla suuren kirjoitusnopeuden ylläpito, koska järjestelmä käyttää toiminnassaan yksinkertaisia indeksejä. (Technical matters, 2023)




Kuva 3. Aikasarjatieto kertoo lämpötilan muutokset päivän aikana. (Influxdata, n.d.-a)

## 3.5 API-rajapinta

API on lyhenne sanoista “Application Programming Interface”, eli ohjelmointirajapinta tai rajapinta. Rajapinnan tarkoituksena on mahdollistaa ohjelmistojen integraation. Rajapinnan avulla pystytään tekemään pyyntöjä järjestelmille, josta halutaan noutaa tai tuoda tietoja. API:n integraation avulla pystytään saavuttamaan suuria kustannussäästöä tietojen liikkuessa järjestelmien välillä automaattisesti. API voi käyttää toiminnassaan CRUD menetelmää. CRUD tulee sanoista “Create, Read, Update, Delete” ja ovat perustoimintoja, jotka ovat tietojen lukemisen ja muokkaamisen perustoiminta. (Valjas, 2019)

Tietokannat käyttävät rajapintaa toimiakseen toisten järjestelmien kanssa. Yleisesti rajapintojen avulla kehittäjät pystyvät keskustella tietokannan kanssa ja poimia tärkeitä tietoja tämän avulla. Rajapintoja käytetään tietokannassa, koska rajapinnat toimivat minkä tahansa järjestelmän kanssa. Tämä edistää työntekijöiden tuottavuutta, koska muuten työntekijät siirtyisivät monen ohjelmien välillä. API:t lisäävät myös tietokannan tietoturvaa. Ne mahdollistavat sen, että vain kelvollisilla käyttäjillä on pääsy tietokantaan ja pääsy muokkaamaan tietokannan tietoja. Tämän lisäksi rajapinnat edistävät pilvipalveluita, sillä API:n käyttö on tehokkain tapa yhdistää pilvitietokantaan. (Nguyen, n.d.)

### 3.6 REST-API

REST-API on API, jolla ohjelmistot pystyvät kommunikoimaan keskenään verkon yli. Termi REST tulee sanoista “Representational State Transfer” ja otettiin käyttöön ensimmäistä kertaa vuonna 2000. REST-API:t käyttävät verkkopalvelun kautta HTTP-pyyntömenetelmiä resurssien käsittelyyn. REST-ympäristössä CRUD-menetelmä vastaa usein HTTP:n menetelmiä POST, GET, PUT ja DELETE. Nämä neljä komentoa ovat siis REST-API:n jatkuvan tallennusjärjestelmän peruselementtejä (Kuva 4). REST-rajapintojen hyötyihin kuuluu niiden nopea tapa skaalautua ja integroitua tarpeensa mukaan. (Altexsoft, 2022)


Kuva 4. CRUD-menetelmät REST-rajapinnassa. (Wallarm, n.d.)

## 4 MASSADATA

### 4.1 Yleistä

Massadata eli Big data, referoi erittäin laajaan ei-lajiteltuun ja lajiteltuun dataan, joka jatkaa kasvamista ajan myötä. Tällainen tietojoukko on todella laaja ja monimutkainen, koska se voi pitää sisällään kaikennäköistä dataa, jota ei tavanomainen ohjelma pysty käsittelemään monipuolisen datan takia. Massadatan käsittelyn tarve kasvaa koko ajan, koska datan tärkeys ja sen hyödyt kasvavat teknologian kanssa. Suurten tietoaineistojen purkaminen edellyttäisi

yrityksille suurta kilpailuetua ja liiketoiminnan kehitystä. Yritykset käyttävät massadatan purkamiseen datavarastoja ja NoSQL-tietokantoja. NoSQL-tietokantoja käytetään massadatan analysointiin, sillä NoSQL- tietokannat sallivat massadatan monipuolisen datan käytön. Massadatan käsittelyyn sisältyy haasteita. Massadatan isoin haaste yrityksissä on tietolähteiden määrä. Usein yrityksillä työskennellään monella erilaisella tietojärjestelmillä eripuolilla organisaatioita. Erilaisten tietolähteiden integroiminen ja saatavuus on monimutkaista, mutta tärkeää, jos yritys haluaa saada datasta arvoa irti. (Google Cloud, n.d.)

### 4.2 Tärkeys

Massadatan tärkeys syntyy isojen organisaatioiden hallinnasta ja liiketoimintapäätöksistä. Massadatan hallinta saavuttaa yrityksille kilpailuetua, sillä ne saavat uusia tapoja optimoida toimintaansa ja vastata paremmin esimerkiksi asiakastarpeisiin. Massadatan hallinnalla pystytään myös tekemään tietoon perustuvia päätöksiä. Tietoon perustuvat päätökset voivat perustua esimerkiksi trendejä ja kuvioita, joista yritykset mahdollistavat strategisemmat ja tehokkaammat päätökset. Yhteenvetona voidaan todeta, että massadata ja sen hallinta on tärkeää, koska hallinta voi muuttaa toimialoja, parantaa päätöksentekoa ja kykyä ratkaista jopa yhteiskunnallisia haasteita. Maailmassa tuotetaan yhä enemmän dataa ja kyky hyödyntää dataa kasvaa entistä kriittisemmin yrityksille. (SAS, n.d.)

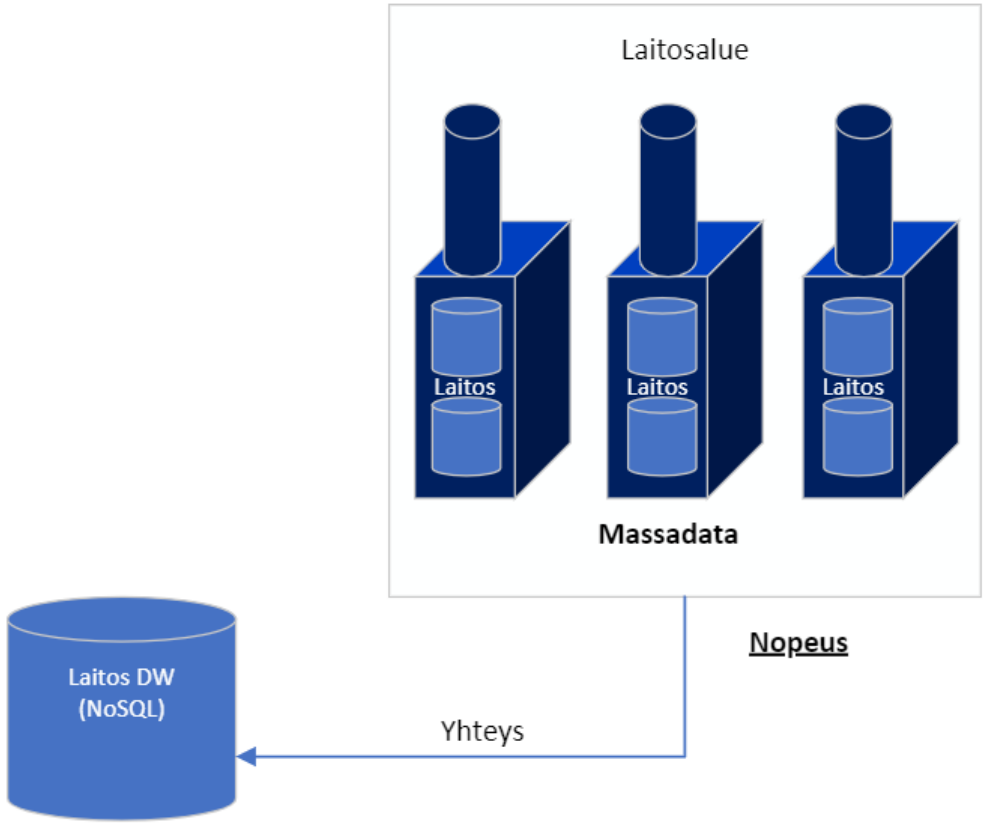
## 5 SUUNNITTELU

### 5.1 Nykytilanne

Toimeksiantajan nykyratkaisu tietovarastointiin sisältää monta erilaista SQL-relaatiotietokantaa. Nämä tietokannat ovat yhdistetty yhteen suureen tietovarastoon, jossa dataa käsitellään. Tietokannat ovat ns. On-premise-tietokantoja

eli ne sijaitsevat toimeksiantajan omissa konesaleissa. Nämä tietokannat ovat olleet käytössä jo pitkään. Tietokannat sisältävät paljon erilaista tietoa eri puolilta Olkiluotoa, mutta laitosalueelta ei pystytä tällä hetkellä tallentamaan tietoa tietoturvasyistä, joten ainoa tapa tallentaa tietoa laitosalueelta on ulkoisen kovalevyn avulla. Ratkaisuksi toimeksiantajalla on suunnittelussa avata tulevaisuudessa dataverkosto, jonka avulla pystytään tallentamaan ja käsittelemään laitoksilta tulevaa tietoa. Dataverkosto haluttaisiin avata, koska toimeksiantaja näkee datan käsittelyn merkityksen ja kuinka datan käsittely hyödyntäisi liiketoimintaa. Laitosdatan hyödyntäminen auttaisi sekä liiketoiminnassa että sen kehityksessä, sillä laitosdatan avulla pystyttäisiin esimerkiksi reagoimaan ajoissa laitosten osien vaihto- tai huoltotarpeisiin. Myös juuri valmistunut Olkiluoto 3 –laitosyksikkö on kasvattanut laitosdatan käsittelyn tarpeita konsernissa, sillä on todettu, että tiedon määrä on tuplaantunut ja sen käsittely on todella raskasta. Toimeksiantajan välisissä keskusteluissa käytiin läpi, että tällä hetkellä oletuksena on, että laitoksilta tuleva tieto on suurta massadataa, jota tämänhetkinen relaatiotietovarasto ei pystyisi käsittelemään jäsentämättömän datan määrän syystä.

Tämän pohjalta toimeksiantajan ajatuksena olisi luoda uusi tietovarastoratkaisu NoSQL-tietokannan pohjalta, jonka tarkoituksena olisi käsitellä kolmelta laitokselta tulevaa massadataa (Kuva 5). Uusi tietovarasto vastaanottaisi kaikista kolmesta laitoksesta tulevaa laitosdataa. Laitoksilta oleva tieto on luultavasti ei-jäsenneltyä dataa. Tämän tiedon suhteen tietovaraston järjestelmä pitäisi olla sellainen, että järjestelmä pystyisi vastaanottamaan massadataa ilman suuria vaikeuksia. Tietovarastoratkaisun vaatimuksina määriteltiin seuraavat asiat: tietokannan API-mahdollisuuksia, tietoturvallisuus, tietokannan pilvimahdollisuuksia, kyky käsitellä erilaisia rakenteellisessa tai ei-rakenteellisessa muodossa olevia tietoja, suorituskykyä ja nopeutta. Myös tietokannan kustannukset otetaan käsittelyyn, kun tietokanta lopulta otetaan käyttöön. Uuden tietovarastoratkaisun hankinta on hyvin merkittävä kokonaisuudessaan ja hankinta avaisi uusia ovia datastrategiasta ja liiketoiminnan kehityksessä. Tämän pohjalta lähdin toteuttamaan toimeksiantajalle tietovarastoratkaisuja, jotka täyttävät konsernin vaatimuksia.




Kuva 5. Tietovarastoratkaisun tarkoitus

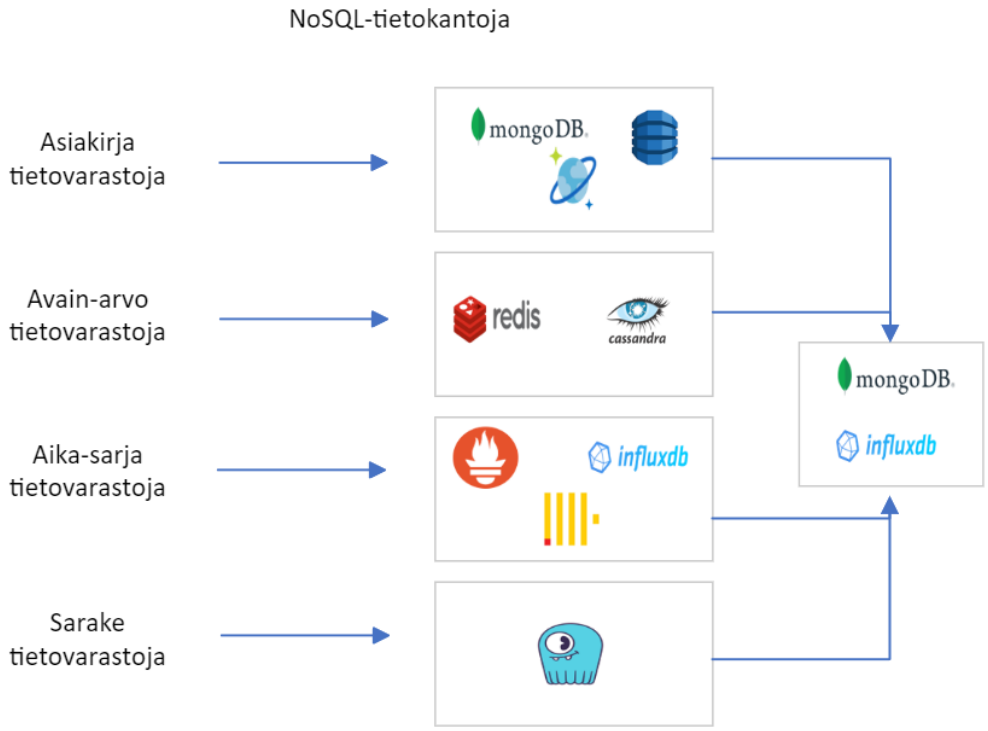
## 5.2 Miksi tutkia NoSQL-tyyppisiä tietokanta-alustoja?

NoSQL-tietovarastopohjaisen hankinta on ollut mietinnässä toimeksiantajalla jo pitkään. Osa mietinnän syystä ovat olleet sekä digitalisaation että tiedolla johtamisen tarpeiden kasvu maailmalla. Digitalisaation ja tiedolla johtamisen syistä halutaan avata NoSQL-tietokannan mahdollisuuksia varastoida monia erityyppisiä tietolähteitä ja tietokokonaisuuksia sekä useassa eri rakenteellisessa muodossa olevaa tietoa. Samalla halutaan pitää relaatiotietokannat ajan tasalla siten, että uusi tietovarastoratkaisu sisältäisi REST-API:n kaltaisia ominaisuuksia, joilla voidaan olla vuorovaikutuksessa tietoja tuottavien ja hyödyntävien järjestelmien kanssa. Näistä tiedoista lähdin pohtimaan tietokantaratkaisuja ja esittämään valintoja toimeksiantajalle.

## 5.3 Tietokantojen valinnat

Tietokantojen valitseminen pilotointia varten tuotti aluksi haasteita, sillä NoSQL-tietokantavaihtoehtoja on nykypäivänä moneen eri tarpeeseen. Tärkeimpänä asiana tietokannan valintaan oli sen kyky käsitellä jäsentelemätöntä dataa. Seuraavaksi lähdin pohtimaan muita ominaisuuksia kuten suorituskykyä ja rajapintoja. Kävin myös toimeksiantajan kanssa keskusteluja tietovarastoratkaisuista ja tarpeista (Kuva 6). Keskusteluissa käytiin läpi NoSQL-tietokantoja ja niiden käyttötarkoituksia, jotka vaikuttavat valintoihin. Jotta työn sisältö ei kasvaisi liian isoksi, vertailuun valitsin kaksi tietokantavaihtoehtoa. Sain käyttööni testitietokoneen, johon asensin tulevat tietokannat. Testitietokoneessa on 16 gigatavua keskusmuistia (RAM), 512 gigatavua SSD-muistia ja suorittimena Intelin Core i5-sarjan prosessori, joka sisältää 10 ydintä.

Ensimmäisenä tietokantana valitsin dokumenttipohjaisen tietokannan MongoDB:n. MongoDB on minulle entuudestaan tuttu ja haluan testata, kuinka tämän tietokantavaihtoehdon joustavuus ja skaalautuvuus mahdollistaisi massadatan käytön. Pilotoinnissa haluan saavuttaa tietokannan äärirajoja tehokkuuden testaamisessa. Toiseksi pilotoitavaksi tietokannaksi valitsin InfluxDB:n. Valinta perustuu aikaisempiin keskusteluihin mahdollisista tietovarastoratkaisuista. Laitoksilla sijaitsee paljon aikasarjatietoa, joten luokittelimme keskusteluissa aikasarjatietokannan pilotoinnin tarpeelliseksi. InfluxDB on aikasarjatietokanta, joka on erityisesti suunnattu isojen voimalaitosten tietojenkäsittelyyn. InfluxDB:stä minulla ei ollut ennen työn aloittamista mitään aikaisempaa kokemusta, mutta keskusteluiden jälkeen aloitin tiedonhaun. Asennuksia varten sain käyttööni uuden Windows-käyttöjärjestelmän, johon asennan tietokannat alusta alkaen.




Kuva 6. Tietokanta-ohjelmien valinnat.

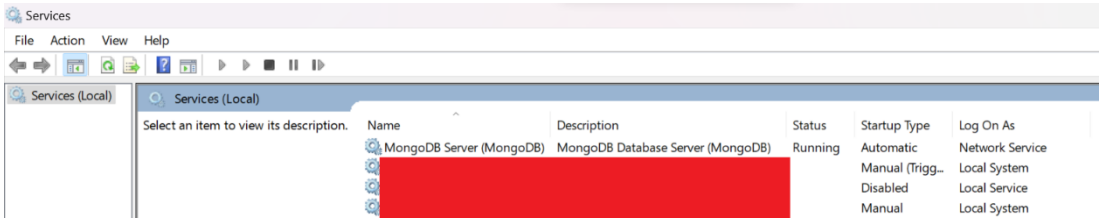
# 6 TIETOKANTOJEN PILOTOINTI

## 6.1 MongoDB:n käyttöönotto

Valitsin MongoDB asennusta varten MongoDB Community Serverin uusimman versio, joka on tätä opinnäytetyötä tehdessäni versio 7.0.2. Community Server on On-Premise -tietokanta, joka tarjoaa käyttäjilleen ilmaisen ja monipuolisen MongoDB tietokannan menettämättä tallentamisen ja hakemisen pääominaisuuksia. Community Editionista on myös tarjolla MongoDB Atlas maksullinen pilvipalvelu, mutta toimeksiantajan tämänhetkisten suositusten mukaan valinta kohdistuu On-Premise palveluun. Asennus ei tuottanut minulle ongelmia, sillä aikaisempi kokemukseni auttoi asennuksessa.

Community Editionin asentamisessa varmistan, että tietojärjestelmäni täyttää tarvittavat esivaatimukset ja että ohjelmisto on saatavilla Windows-käyttöjärjestelmälle. Asennusta varten suoritin paketista olevan mongodb_windows-x86_64-7.0.2.msi (Microsoft Software Installer) tiedoston. Suorittamalla asennustiedoston pääsee valitsemaan asennustyypin. Valitsin vaihtoehdon “Complete”, sillä tämä asentaa kaikki mahdolliset ominaisuudet, jotka ovat saatavilla MongoDB:ssä. MongoDB halutaan asentaa On-Premise ympäristöön, joten valitsin palvelimen asennuksen. Viimeisenä pääsin valitsemaan asennuksen lisäyksenä lataamaan MongoDB Compass-ohjelman, joka on palvelimen GUI käyttöjärjestelmä (Graphical User Interface). GUI on graafinen käyttöliittymä, jonka kautta käyttäjä pääsee vuorovaikuttamaan elektronisten laitteiden, tässä tapauksessa tietokanta palvelimen kanssa.  Asensin samalla komentorivipohjaisen käyttöliittymän Mongo Shellin.

MSI:n asennuksen loputtua käynnistin Windows-järjestelmä uudelleen, jotta kaikki ominaisuudet tulevat käyttöön. Uudelleen käynnistyksen jälkeen suoritin Windows PowerShellissä komennon “Net start Mongod”, joka käynnistää palvelimen. Käynnistyksen jälkeen kävin katsomassa Windows Services -sovelluksessa MongoDB palvelimen tilan. MongoDB status on merkitty ”Running”, joten palvelin on nyt toiminnassa ja asettuu oletuksena aina päälle (Kuva 7).



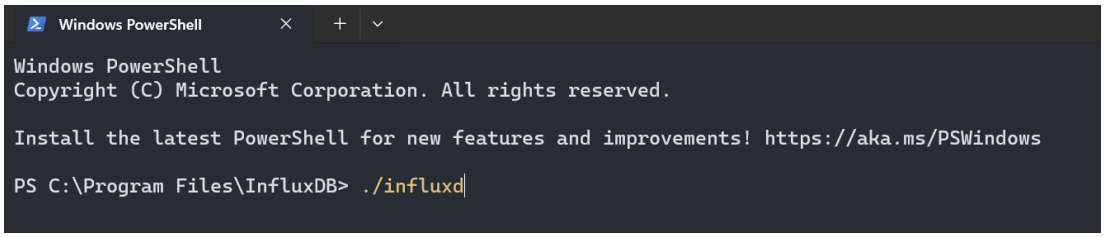

Kuva 7. MongoDB:n palvelimen varmistaminen

## 6.2 InfluxDB:n käyttöönotto

InfluxDB-tietokannasta latasin version 2.7.1, joka on tämän opinnäytetyön hetkenä uusin versio ohjelmistosta. Tietokannan lataaminen Windows-käyttöjärjestelmälle ei tuottanut haasteita vaan oli kovin työläs. Asennus alkoi ZIP-tiedoston lataamisella ja purkamisella, jonka jälkeen InfluxDB palvelimen

käynnistys toimii PowerShellin tai komentokehotteen avulla. Järjestelmän käynnistämiseen tarvitsee mennä komentokehotteen avulla sijaitsevaan kansioon. Tiedoston sijainnin syöttämisen jälkeen tietokanta käynnistetään syöttämällä komento komentokehotteeseen ”./influxd” (Kuva 8). Tämä aloittaa palvelimen käynnistyksen, jonka jälkeen on käytettävissä. Ohjelma avaa komentokehotteen avulla paikallisen palvelimen, joka käyttää oletuksena porttia 8086.  InfluxDB käyttää HTTP-rajapintaa, joten palvelimen pääsyä varten käytetään ”localhost:8086” osoitetta selaimessa.

Ensimmäinen vierailu palvelimeen vaatii käyttäjältä käyttäjätunnuksen ja salasanan laatimisen. Käyttäjätunnuksen tekemällä InfluxDB luo minulle ensimmäisen API-rajapinnan. Tällä API-rajapinnalla käyttäjäni on merkitty tietokannan omistajaksi ja voin myöhemmin muokata ohjelmia omistajan käyttäjätunnuksilla.



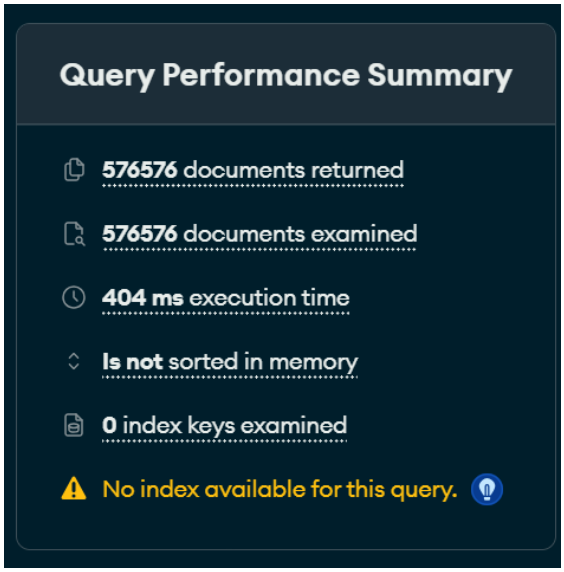

Kuva 8. Palvelimen käynnistys

## 6.3 Tietokantojen suorituskyvyn testaus

### 6.3.1 MongoDB:n suorituskyky

Ohjelmat ovat nyt käyttövalmiit ja toimivat oletettuun tapaan. Käyttöönoton jälkeen lähden testaamaan tietokantojen suorituskykyä ja nopeutta. MongoDB suorituskykyä haluan testata MongoDB Compassissa ja Mongo Shellissä. Ajatuksena on testata tietokannan suorituskykytoimintoja lisäämällä tietokantaan monipuolista dataa ja suorittamalla kuormittavia kyselyjä samalla. Compassin ja Shellin suorituskyvyn vertaamisen kautta saan samalla katsottua ohjelmien suorituskykyjen eroja. MongoDB Compass GUI sisältää kyselyjä tehdessä ”Explain” painikkeen, jonka avulla näen kyselyn suorittamisen aikana

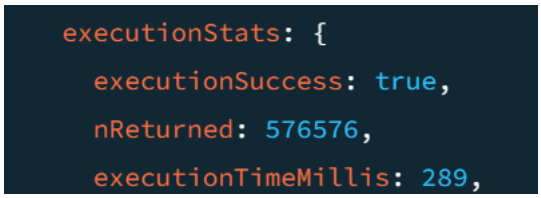
tapahtuneet toimenpiteet ja kuinka kauan kyselyn vaiheiden tekemisessä on kulunut aikaa. Lisäsin tietokantaan avoimesti olevaa dataa siten, että kokoelma sisälsi yli 575 tuhatta dokumenttia. Dokumenttien määrä hidasti palvelinta ja se näkyi kyselyitä tehdessäni. Dokumenttien lisäämisen jälkeen suoritin tyhjän kyselyn, joka haki kaikki tietokannassa olevat dokumentit ja tietokannan nopeudeksi tuli 404 millisekuntia (Kuva 9). Ennen suuren tiedon lisäämistä palvelin suoritti ensimmäisen kyselyn alle 10 millisekunnissa. Suoritin vielä erilaisia peruskyselyitä tietokannassa ja keräsin niistä keskimääräisen suoritusajan, joka oli noin 293 ms jokaista kyselyä kohden. Tässä on hyvä huomioida se, että MongoDB tallentaa käytettyjä kokoelmia sisäiseen muistiinsa, joka nopeuttaa seuraavia kyselyjä. Tämän takia tietojärjestelmä käyttää runsaasti muistia, joka johtaa isoissa tietokannoissa siihen, että järjestelmä vaatii suuren määrän muistia, joka nostaa järjestelmän hinnoittelua.




Kuva 9. Mongo Compassin ensimmäisen kyselyn tulokset.

Seuraavaksi lähdin testaamaan samaa kyselyä CLI-liittymällä, jotta saan samalla selville, löytyykö Compass ja Mongo Shellin välillä suorituskyky eroavaisuuksia. Mongo Shell kyselykomennoista löytyy komento ”.explain(executionStats)”. Tämä komento antaa kyselyn suorituksen jälkeen tilastoja tietokannan suorituksesta, joka sisältää myös kyselyn suorittamisen nopeuden. Suorittamalla samanlaisia kyselyitä Mongo Shellissä kyselyn nopeudeksi sain keskimääräisesti 289 millisekuntia, joka on samaa tasoa kuin GUI-

käyttöliittymässä (Kuva 10). Testaukseni perusteella näiden kahden käyttöliittymän välillä ei ole suorituskyvyn kanssa eroavaisuuksia.

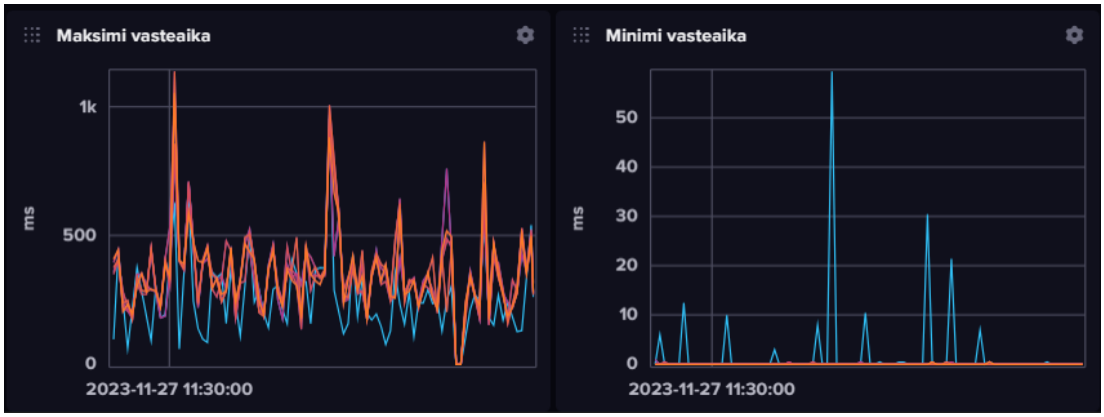



Kuva 10. Mongo Shellin kyselyn tulokset

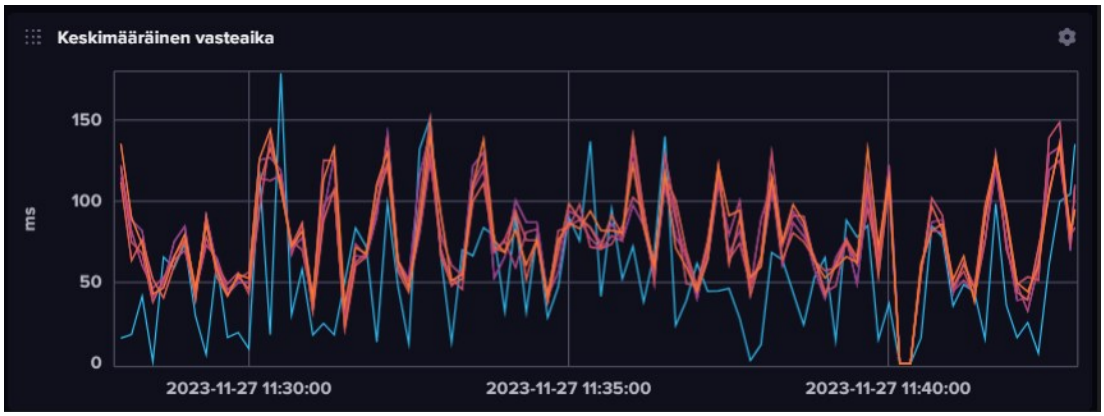
### 6.3.2 InfluxDB:n suorituskyky

InfluxDB tuotti aluksi haastetta suorituskyvyn testaamisessa, koska mitattavaa aikasarjatietoa oli vaikeaa saada palvelimelle. Pitkän tutkinnan jälkeen löysin tavan testata suorituskykyä. Koska InfluxDB käyttää HTTP-rajapintaa, lähdin testaamaan sen selaimen kuormitusta. Suorituskyvyn testaamiseen käytin Apache JMeter -ohjelmaa, joka tarkoituksena on luoda testikäyttäjiä palvelimeen. Nämä testikäyttäjät tekevät toiminnassaan CRUD-toimenpiteitä palvelimeen, jotka aiheuttavat kuormitusta. Tämän avulla loin JMeteriin testisuunnitelman, joka lisää palvelimeen samanaikaisesti 250 testikäyttäjää. Asetan nämä testikäyttäjät suorittamaan palvelimeen GET-pyyntöjä, jotka tekevät tietokantakyselyitä. Samalla testikäyttäjät suorittavat myös POST-pyyntöjä, jotka taas lisäävät tietokantaan tietoja. JMeter-ohjelma kerää näiltä käyttäjiltä vasteaikoja, jotka tallennetaan tietokantaan. Jotta Windows-järjestelmäni ei kuormittuisi liikaa, päätin valita 250 samanaikaista käyttäjää palvelimelle. Keräsin näistä käyttäjistä 15 minuutin ajan maksimi- ja minimivasteajan omaan visuaaliseen taulukkoon (Kuva 11). Keräsin myös keskimääräisen vasteajan omaan taulukkoon (Kuva 12). Tarkastellessani keräämiäni vasteaikataulukoita huomasin, että keskimääräinen vasteaika ja maksimi vasteaika heittelevät paljon. Palvelimen keskimääräinen vasteaika oli kuitenkin 110 ms eli noin sekunti. Maksimi vasteaika oli suurimmillaan yli 1139 ms eli noin 11 sekuntia. Tämä johtui testikäyttäjien samanaikaisesta tapahtumasta. Minimi vasteaika pysyi todella matalalla. Kokonaisuudessaan InfluxDB:n suorituskyky on nopea, mutta heittelevää samanaikaisten tapahtumien syystä. Käyttäjämäärä ei kuitenkaan vaikuttanut huomattavasti testituloksiin ja pysyi samana riippuen

käyttäjämäärästä. Testissäni käytetty Windows-järjestelmä ei käyttänyt testien aikana paljon resurssejaan, joten vasteajat eivät riippuneet resurssien määrästä.




Kuva 11. Maksimi- ja minimivasteajat




Kuva 12. Keskimääräinen vasteaika

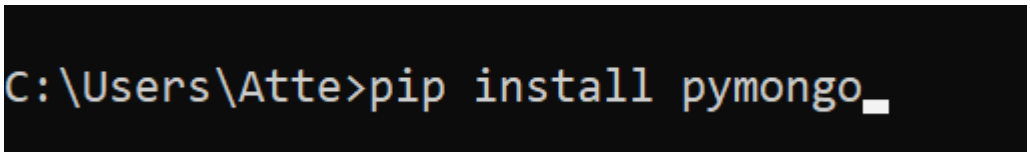
## 6.4 Tietokantojen API-mahdollisuudet

### 6.4.1 MongoDB:n API-integraatio

Lähdin pilotoimaan MongoDB:n rajapintaa koodieditori Visual Studio Code:ssa. Integraation testaamisen tarkoituksena olisi saada yhdistettyä MongoDB tietokanta koodieditoriin ja yhdistämisen jälkeen luoda tietokantaan uusi kokoelma Python-ohjelmointikielen avulla. MongoDB käyttää rajapinnallisuuteen asennettavia ajureita. Näiden ajureiden kautta pystytään yhdistämään projekteista tullutta dataa. Ensimmäisenä halusin testata MongoDB:n REST-API-rajapintaa. Tietoa hakiessani ilmeni, että REST-API:a ei ollut mahdollista

ilmaisversiossani. Tämä oli kuitenkin mahdollista maksullisessa Atlas pilvipalvelun kautta. Atlas pilvipalvelu tarjoaa käyttäjälle Atlas Data API-ominaisuuden, jonka avulla pystytään työskentelemään turvallisesti tietokannan kanssa (MongoDB n.d.-c). Tämän vuoksi lähden testaamaan yleistä integraatiota Python-kielen avulla. Visual Studio Code sisältää MongoDB laajennuksen. Tämä laajennus sallii palvelimen yhdistämisen, jonka kautta pääsen käyttämään tietokantaa koodieditorissani. Laajennuksen avulla näen nopeasti lisäämäni kokoelmat Python-koodillani ja pystyn muokkaamaan tietokantaa.

API-integraation toteutan lataamalla pymongo-asennuspaketin, jonka avulla pääsen käyttämään Python-kieltä rajapinnan testaamisessa. Komennolla ”pip install pymongo” asennan tämän paketin (Kuva 13). Käytän tätä pakettia Pythonin kanssa työskentelyyn. Integraation aloitan luomalla uuteen Python-projektiin muuttujan ”palvelin” (Kuva 14). Tämä muuttuja pitää sisällään yhteyden Windows-järjestelmässäni olevaan MongoDB palvelimeen ”Mongdb://localhost:27017/”. Varmistaakseni yhteyden toiminnan luon toisen muuttujan y, joka etsii palvelimen version ja tulostaa sen. Yhteyden varmistuksen jälkeen loin uudet muuttujat, jotka sisältävät tietokannan ja sen kokoelman nimen. Tämän jälkeen luon vielä asiakirjamuuttujan, johon syötän syötettävät tiedot. Tietojen lisäämisen jälkeen syötän antamani tiedon komennolla ”insert_many()”. Tämän jälkeen tulostan kaikki tietokannassa olevat kokoelmat komennolla "list_collection_names()". Lopuksi tulostan kokoelmassani olevat tiedot, jonka jälkeen suljen palvelimen yhteyden.




Kuva 13. Pymongo-paketin asentaminen

```python
#Pymongo paketin hakeminen
import pymongo

#Tietokanta palvelimeen yhdistäminen
palvelin = pymongo.MongoClient("mongodb://localhost:27017")

#MongoDB version tulostaminen
y = palvelin.server_info()["version"]
print("MongoDB versio: " + y)

#Tietokannan valitseminen (MongoDB perustaa uuden tietokannan automaattisesti vastaanottaessaan tietoa)
tietokanta = palvelin["Pilotointi"]

#Kokoelman muodostaminen
kokoelma = tietokanta["Yritykset"]

#Syötettävät tiedot
#Yrityksien perustietoja
mydict = [
    {"Yritys" : "TVO", "Osoite": "Olkiluodontie", "Y-tunnus" : "0196656-0",
      "Toimiala": "Sähköntuotanto ydinvoimalla"},
    {"Yritys" : "Posiva", "Osoite" : "Olkiluodontie", "Y-tunnus" : "1029258-8",
      "Toimiala": "Ongelmajätteen käsittely ja hävittäminen"},
    {"Yritys" : "TVONS", "Osoite" : "Olkiluodontie", "Y-tunnus" : "1468037-7",
      "Toimiala" : "Kone- ja prosessisuunnittelu"},
    {"Yritys" : "Posiva solutions", "Osoite" : "Olkiluoto", "Y-tunnus": "2765436-2",
      "Toimiala": "Maa- ja vesirakentamisen tekninen palvelu"}
        ]

#Tietojen syöttäminen kokoelmaan
kokoelma.insert_many(mydict)

#Tietokannan kokoelmien tulostaminen
kokoelmat = tietokanta.list_collection_names()
print(kokoelmat)

#Kysely, jossa tulostetaan kaikkien tietojen Yritysten nimi.
for x in kokoelma.find({}, {"_id": 0, "Yritys": 1}):
    print(x)

#Yhteyden katkaiseminen
palvelin.close()
```

Kuva 14. Python-koodi MongoDB palvelimen yhdistämiseen ja tietokannan luomiseen.

### 6.4.2 InfluxDB:n API-integraatio

InfluxDB:n rajapinnan toimintaa lähden kokeilemaan lisäämällä valmiina olevaan kokoelmaan satunnaista tietoa Python-kielen avulla. Jotta rajapinta toimisi Pythonin kanssa, tarvitsen influxdb-client-paketin, jonka asennan myös pip-komennon avulla. InfluxDB:n rajapinnan toiminta perustuu API-tunnusten käyttöön. Nämä tunnukset varmistavat turvallisen käytön kolmannen osapuolen kanssa identifioimalla InfluxDB:n käyttäjän käyttöoikeudet organisaatiossa (InfluxData, n.d.-b). API-tunnuksen asettamisen jälkeen luon yhteyden tietokantaan käyttämällä Python-kieltä. Luon uuden Python-projektin, johon luon muuttujat “token”, “org” ja “url”, jotka pitävät sisällään palvelimen tiedot yhdistämistä varten. InfluxDB-client-paketti pitää sisällään InfluxDBClient-luokan,

johon syötän palvelinmuuttujat. Varmistan aina lopuksi muuttujien ja yhteyden toiminnan pingaamalla palvelinta, sillä yhteyden luominen on ehdotonta datan lisäämistä varten (Kuva 15). Tämä toiminto toimii käyttämällä HTTP-pyyntöä GET-toimintaa, joka hakee rajapinnasta palvelimen tiedot.

```python
import influxdb_client, os, time
from influxdb_client import InfluxDBClient, Point, WritePrecision
from influxdb_client.client.write_api import SYNCHRONOUS

#Tietokannan tiedot
token = "[redacted]"
org = "Kokoelma"
url = "http://localhost:8086"

#Yhdistäminen tietokantaan muuttujien avulla
write_client = influxdb_client.InfluxDBClient(url=url, token=token, org=org)

#Yhteyden varmistaminen
Ping = write_client.api_client.call_api('/ping', 'GET')
print(Ping)
```

Kuva 15. Python-koodi tietokannan yhdistämiseen

Yhteys on nyt dataa vaille valmis. Datan lisääminen tietokantaan lisäsi haastetta, sillä InfluxDB ei suostunut lisäämään tietoa, joka ei sisällä mitattavaa tietoa. Tämä on siksi, että InfluxDB vaatii tiettyjä tiedon syöttöjä, kuten aikaleiman ja mittarin, jotta se pystyisi käsittelemään tiedon oikein. Datan kuitenkin pystyy lisäämään siten, että luon Mittari1 nimisen muuttujan, joka vastaanottaa satunnaista numerotietoa (Kuva 16). Merkitsen vielä tietoon merkin, jotta löydän tiedon helpommin merkin avulla. Tietokanta vaatii vielä kentän, johon tuleva numero asetetaan. Teen mittarin sisään vielä “for” toiston, joka nostaa numeron arvoa toiston toistuessa, jotta numero arvot eroavaisivat toisistaan. Lopuksi laitan ohjelman tallentamaan numeron tietokantaan. Sujuvuuden kannalta lisään vielä komennon ”time.sleep(1)”, joka pitää sekunnin tauon ennen uuden tiedon lisäämistä.

```python
write_api = write_client.write_api(write_options=SYNCHRONOUS) #Synkronoi HTTP
#Loop, joka lisää numeron arvon tietokantaan
for x in range(12):
  numero = (
    Point("Mittari1") #Mittarin luominen
    .tag("numeronimi", "numeroarvo")
    .field("numero", x)
  )
  write_api.write(bucket=bucket, org="Kokoelma", record=numero) #Tallentaa numeron kokoelmaan
  time.sleep(1) #Päivittämistä varten pitää yhden sekunnin tauon ennen lisäämästä uuden luvun
```

Kuva 16. Python-koodi tietokannan tiedon lisäämiseen.

## 6.5 Tietoturva

Tietokannan tietoturva on nykypäivänä tärkeä ominaisuus ja vaatii käyttäjiltä huolellisuutta ja säännöllisiä tarkastuksia tietokannan ylläpitämiseen. Molemmat tietokannat tarjoavat laajan valikoiman tietoturva-asetuksia ja työkaluja, jotka parantavat tietoturvallisuuden kokonaisuutta. Tarkastelin tietokantojen eri tietoturvamahdollisuuksia. Molemmat ohjelmat sisältävät runsaan määrän tietoturva-asetuksia, mutta käyn tässä kappaleessa läpi tietokannan käyttöönotossa esiin tulleita havaintojani.

MongoDB sisältää TLS/SSL-salauksen, joka salaa käyttäjän ja tietokannan välisen verkkoliikenteen (MongoDB, n.d-c). Tämä varmistaa sen, että MongoDB:n tiedot on vain käyttäjän luettavissa. Tietoturva tarjoaa myös joustavan käyttäjätodennuksen, jonka avulla tietyt käyttäjät näkevät vain heille merkittyjä tietoja. Tarkastellussani huomasin, että MongoDB:en ei sisälly automaattista käyttäjän todennusta vaan se joudutaan asettamaan päälle manuaalisesti.

InfluxDB:n tietoturva näkyi heti käyttöönotossa, sillä ennen käyttöönottoa InfluxDB pyytää käyttäjää asettamaan käyttäjätunnukset. Myös tietokantojen luomisessa käyttäjälle annetaan kerran nähtäväksi rajapinnallisuusmerkintä, jonka jälkeen sitä ei voi saada enää näkyviin. Palvelimessa on myös automaattinen uloskirjautuminen, kun palvelin huomaa käyttäjän olevan epäaktiivinen. Ohjelma tarjoaa samat tietoturva-asetukset kuin MongoDB:ssä, mutta lisäksi tiedon salaukseen InfluxDB käyttää https-salausta, joka estää tiedonkalastelun. Pilotoinnin käytössäni oleva ilmaisversio ei sisällä edistynyttä tietoturvaa, joka on saatavilla ainoastaan maksullisessa versiossa.

## 6.6 Tietokannat tiivistettynä

Olen nyt saanut perusnäkemyksen valittujen tietokantojen toimivuudesta. Tietokannat toimivat oletettuun tapaan. Suorituskykyä testatessani huomasin

MongoDB:ssä sen, että suorituskyvyn toiminta hidastuu tiedon lisääntyessä. Tietokanta kompensoi hidastuessaan tallentamalla dataa sisäiseen muistiin, jonka vuoksi tietokanta on muistipainotteinen. InfluxDB:n vasteaika oli vaihteleva, mutta aikasarjatietokantana InfluxDB oli nopea. Myöskään käyttäjämäärää ei vaikuttanut tietokantaan suuresti. Molemmat tietokannat tarjoavat myös hyvin erilaisia rajapintavaihtoehtoja. MongoDB tarjoaa monipuolisesti erilaisia laajennuspaketteja eri ohjelmointikielille, kun taas InfluxDB tarjoaa HTTP-rajapinnan, jota se käyttää myös käyttöliittymänä. Tämä rajapinta tukee CRUD-toimenpiteitä ja noudattaa REST-API:a. Tietoturva on huomioitu asianmukaisesti molemmissa tietokannoissa. Molemmat tarjoavat laajan valikoiman tietoturva-asetuksia. Näistä asioista koostin tiivistelmätaulukon, jonka tarkoituksena on tukea toimeksiantajaa tietokannan valinnassa (taulukko 1).

Taulukko 1. Tietokantojen ominaisuudet tiivistettynä.

| **Ominaisuus:** | **MongoDB:** | **InfluxDB 2.0:** |
|---|---|---|
| **Tietokannan tyyppi** | Dokumenttipohjainen NoSQL-tietokanta | Aikasarjaan perustuva NoSQL-tietokanta |
| **Tietojen hakeminen** | Sisältää GUI-ohjelman monimutkaisten kyselyiden kokoamiseen. | Tietojen hakeminen sisältää valmiina olevia graafisia analysointi työkaluja tietojen kokoamiseen |
| **Suorituskyky** | Hakukone, jonka nopeus riippuu käytössä olevasta muistin määrästä. | Hakukone, jonka nopeus riippuu ohjelmistojen kuormitusten määrästä. |
| **Kyselykieli** | MQL-kyselykieli monimutkaisiin kyselyihin, joka pohjautuu JavaScript-kieleen. | Flux-skriptikieli, tehty pääsääntöisesti käsittelemään aikasarja dataan |
| **Tietoturva** | Sisältää tiedon salausjärjestelmän ja muita | Sama kuin MongoDB, mutta lisäksi tietoturva koostuu API- |

| | | |
|---|---|---|
| | monivaiheisia todennuksia. | merkinnöistä, jotka ovat saatavilla, vain tietokannan järjestelmävalvojalla. |
| **Pilvimahdollisuudet** | Oma Atlas pilvipalvelu, joka sisältää tietokannan seurantaa ja optimointia tukevia ominaisuuksia | Tarjoaa täysin hallitun pilviinfrastruktuurin, joka on tarkoitettu työnkuormitukselle. |
| **Käyttötarkoitus** | Yleiseen käyttöön tarkoitettu tietojen säilytykseen | IoT (Internet of Things) ja reaaliaikaisen tiedon analysointiin |
| **Hyödyt** | Helposti käyttöönotettava, käyttäjäystävällinen, monipuolinen tiedon syöttö | Suorituskyky kohdallaan, Tietoturvan oletus asetukset, Visualisointi mahdollisuudet |
| **Haitat** | Tietoturvan oletus asetukset, rajallinen tiedon koko, Muistin käyttö suuremmissa tietokannoissa | Monimutkainen ja aikaa vievä kokonaisuus, vaikea oppia, erityiseen käyttöön tarkoitettu. |

# 7 YHTEENVETO

Tämän opinnäytetyön tarkoituksena oli pilotoida NoSQL-tietovarastoratkaisuja. Testien tarkoituksena oli kokeilla tietokantojen toimivuutta ja ominaisuuksia. Testauksien jälkeen oli tarkoituksena luoda tiivistelmätaulukko testattujen tietokantojen ominaisuuksista. Taulukon avulla pystyttäisiin tekemään päätöksiä tietokannan hankinnan suhteen.

Opinnäytetyössäni haastetta tuotti eniten InfluxDB. Tämä tietokanta oli minulle hyvin monimutkainen, koska sen käyttö vaati syvällistä ymmärrystä. Uudet käytännöt hidastivat alkuvaiheen työtäni ja vaativat ylimääräistä panostusta dokumentaatioiden tutkimiseen. MongoDB:ssä helpotti se, että se oli minulle valmiiksi tuttu. Käyttöönoton edetessä InfluxDB tuli minulle tutummaksi ja opin paljon uutta tietokantojen mahdollisuuksista. Mielestäni tietovarastojen pilotoinnit onnistuivat vastaamaan toimeksiantajan vaatimuksiin. Olen siis tyytyväinen opinnäytetyön haastavuuteen.

Opinnäytetyön aikana tunnistettiin tietokantojen useita hyötyjä ja haittoja, joka sisälsi myös asioita, joita ei työn rajaamisen takia en voinut käydä läpi. Esille nousseita huomioita olivat tietokannan optimointi, visualisointi ja tietokantojen käyttökustannukset. Nämä huomiot mahdollistavat projektin tulevaisuuden jatkokehityksen tarpeita.

Tiivistelmätaulukko tekee tehtävänsä, sillä se antaa mielestäni lyhyen katsauksen kokonaisuudesta. Päädyin tekemään taulukon, koska se tarjoaisi lyhyen lopputuloksen työstäni. Sen tarkoitus oli tehdä yleisvaikutelman työstä ja koota työn asiat yhteen pakettiin. Tiivistelmätaulukon myötä mielestäni molemmat tietokannat ovat käteviä ratkaisuja omiin tarpeisiin, mutta jatkon kannalta suosittelisin MongoDB tietokantaa. Valintani perustuu helppokäyttöisyyteen ja monipuolisuuteen sekä se täyttää suurimman osan toimeksiantajan antamista vaatimuksista.

Yhteenvetona voin todeta, että vaikka opinnäytetyö oli haastava kokemus, se tarjosi hyvin opettavaisen ja monipuolisen kokonaisuuden. Työ hyödynsi toimeksiantajaa, sillä työn avulla saatiin taustatietoa tulevaan hankkeeseen. Opinnäytetyön merkityksellisyys motivoi minua paljon ja uskon, että työssäni saaneet tiedot ja taidot tulevat auttamaan minua tulevaisuudessa.

# LÄHTEET


Altexsoft. (21.10.2022). What is an API: Definition, Types, Specifications, Documentation. https://www.altexsoft.com/blog/what-is-api-definition-types-specifications-documentation/

GeeksforGeeks. (14.3.2023). Types of NoSQL Databases. https://www.geeksforgeeks.org/types-of-nosql-databases/

Ghazaryan, Ani. (14.7.2023). What is a Graph Database? https://memgraph.com/blog/what-is-a-graph-database

Gillis, A. (13.3.2023). Definition MongoDB. https://www.techtarget.com/searchdatamanagement/definition/MongoDB

Google Cloud. (n.d.). What is Big Data? Haettu 16.10.2023 osoitteesta https://cloud.google.com/learn/what-is-big-data

IBM. (n.d.). What are NoSQL databases? Haettu 28.9.2023 osoitteesta https://www.ibm.com/topics/nosql-databases

InfluxData. (n.d.-a). Time series data. Haettu 13.12.2023 osoitteesta https://www.influxdata.com/what-is-time-series-data/

InfluxData. (n.d.-b). Using the InfluxDB HTTP API. Haettu 15.11.2023 osoitteesta https://docs.influxdata.com/influxdb/v2/api/#operation/PatchAuthorizationsID

Nguyen, S. (n.d.). Understanding the Connection Between Databases and APIs. Haettu 17.11.2023 osoitteesta https://blog.dreamfactory.com/understanding-the-connection-between-databases-and-apis/

Mullins, C. Vaughan, J. Beal, B. (1.4.2021). NoSQL (Not Only SQL database). https://www.techtarget.com/searchdatamanagement/definition/NoSQL-Not-Only-SQL

MonboDB. (n.d.-a). What is a Document Database? Haettu 11.10.2023 osoitteesta https://www.mongodb.com/document-databases

MongoDB. (n.d.-b). Atlas Data API. Haettu 13.12.2023 osoitteesta https://www.mongodb.com/docs/atlas/app-services/data-api/

MongoDB. (n.d.-c). TLS/SSL (Transport Encryption). Haettu 17.11.2023 osoitteesta https://www.mongodb.com/docs/manual/core/security-transport-encryption/

Oracle. (n.d.). What is NoSQL? Haettu 17.10.2023 osoitteesta https://www.oracle.com/database/nosql/what-is-nosql/

Oracle. (n.d.-b). Graph Database Defined. Haettu 10.11.2023 osoitteesta https://www.oracle.com/autonomous-database/what-is-graph-database/

SAS. (n.d.). Big Data, what it is and why it matters. Haettu 17.10.2023 osoitteesta https://www.sas.com/en_us/insights/big-data/what-is-big-data.html

Scylladb. (n.d.). Wide-column Database. Haettu 13.10.2023 osoitteesta https://www.scylladb.com/glossary/wide-column-database/

Smallcombe, M. (9.10.2023). SQL vs NoSQL: 5 Critical Differences. https://www.integrate.io/blog/the-sql-vs-nosql-difference/#:~:text=SQL%20databases%20are%20relational%2C%20and,NoSQL%20databases%20are%20horizontally%20scalable.)

Technical Matters. (12.5.2023). What is InfluxDB. https://www.ionos.com/digitalguide/hosting/technical-matters/what-is-influxdb/

Timescale. (10.2.2023). Time-Series Database: An Explainer. https://www.timescale.com/blog/what-is-a-time-series-database/

TVO. (n.d.-a). TVO yhtiö. Haettu 25.10.2023 osoitteesta https://www.tvo.fi/yhtio.html

TVO. (n.d.-b). TVO-konserni. Haettu 25.10.2023 osoitteesta https://www.tvo.fi/yhtio/hallintojajohtaminen/tvo-konserni.html

Valjas. (2.5.2019). Mitä integraatio, rajapinta ja api tarkoittavat? https://valjas.fi/opi/blogi/mita-integraatio-rajapinta-ja-api-tarkoittavat/

Wallarm. (n.d.). CRUD – Create, Read, Update, And Delete. Haettu 17.11.2023 osoitteesta https://www.wallarm.com/what/crud-meaning

Williams, A. (17.8.2021). NoSQL database types explained: Key-value store. https://www.techtarget.com/searchdatamanagement/tip/NoSQL-database-types-explained-Key-value-store