AVOIMEN LÄHDEKOODIN TIETOKANTOJEN SOVELTUVUUS IQ-DATAN TALLENTAMISEEN

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tekniikan ammattikorkeakoulututkinnon opinnäytetyö

Tieto- ja viestintätekniikka, Riihimäki

Syksy 2021

Outi Kettunen

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tieto- ja viestintätekniikka Tiivistelmä

Riihimäki

Tekijä	Outi Kettunen	Vuosi 2021
Työn nimi	Avoimen lähdekoodin tietokantojen soveltuvuus IQ-datan tallentamiseen	
Ohjaajat	Petri Kuittinen	

TIIVISTELMÄ

Opinnäytetyön tavoitteena on selvittää avoimen lähdekoodin tietokantojen soveltuvuutta IQ-datatallenteiden ja niihin liittyvän metadatan tallentamiseen. IQ-data on digitaalisessa signaalinkäsittelyssä ja esimerkiksi ohjelmistoradioissa käytettävä signaalin kompleksista IQ-esitystapaa hyödyntävä tallenne. Työn tilaaja on Puolustusvoimien Tutkimuslaitoksen Informaatiotekniikkaosasto.

Opinnäytetyössä arvioidaan tapaustutkimuksena kymmenen avoimen lähdekoodin tietokannan hallintajärjestelmän soveltuvuutta IQ-datan tallentamiseen. Vertailtavista tietokantojen hallintajärjestelmistä kolme on relaatiotietokantoja ja seitsemän NoSQL-tietokantoja. Tapaustutkimuksen lisäksi työssä testataan käytännössä soveltuvimmaksi arvioitua MongoDB-dokumenttitietokantaa ja sen suurten tiedostojen tallentamiseen tarkoitettua GridFS-ominaisuutta.

Työn perusteella avoimen lähdekoodin tietokantoja voidaan hyödyntää rajoituksin IQ-datan tallentamiseen. IQ-datan tallentamista suurena binäärimuotoisena tietotyyppinä (BLOB) ei nähty potentiaalisena vaihtoehtona. Binäärimuotoiset IQ-datatallenteet suositellaan tallennettavan sellaisenaan omalle palvelimelle tiedostojärjestelmään ja niihin liittyvä metadata tietokantaan. Vaihtoehtoisesti tallenteet sekä niiden metadata voidaan tallentaa tietokantaan erityisesti suurten tiedostojen tallentamiseen tarkoitettujen toiminnallisuuksien, kuten MongoDB GridFS:n avulla.

Avainsanat avoimen lähdekoodin tietokanta, IQ-signaali, IQ-data, tietokanta

Sivut 43 sivua ja liitteitä 3 sivua

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Information and Communication Technology
Riihimäki

Abstract

Author Outi Kettunen
Year 2021

Subject Suitability of the open-source databases for the IQ-data

Supervisors Petri Kuittinen

ABSTRACT

The aim of this thesis is to examine how open-source databases are suited for storing binary IQ-datafiles and their metadata. IQ-data is a complex form of representing the sinusoidal signal and it is used in digital signal processing, for example in software defined radios. The commissioner of this thesis is Information Technology Division of The Finnish Defence Research Agency.

This thesis compares ten open-source database management systems as a case study and analyses their suitability to store IQ-datafiles. Three of the compared database management systems are relational databases and seven NoSQL-databases. In addition, this thesis performs the test in practice for MongoDB- document database management system and its GridFS-specification, which was considered the most suitable for IQ-data in the case study.

Based on the observations in this study, open-source databases can be used to store IQ-datafiles with certain limits. IQ-datafiles in binary form are recommended to be stored in the server's file system and the metadata of the files in the database. Other option is to store both binary files and the metadata in the database using the database features for unstructured large files, such as MongoDB GridFS.

Keywords database, IQ-signal, IQ -data, open-source database

Pages 43 pages and appendices 3 pages

Sisällys

Kuvat, taulukot ja kaavat

Liitteet

Liite 1 Tietokannan suunnitteluprosessi

Liite 2 Tietokantojen vertailutulokset

Liite 3 Ohjelmakoodi IQ-datatiedostojen ja metadatan tuontiin

Sanasto ja lyhenteet

A/D-muunnos	Analogia-digitaalimuunnos. Analogisen signaalin muuttaminen digitaaliseen muotoon.
Alipäästösuodatin	Rakenne, joka päästää läpi matalataajuiset signaalit ja suodattaa tai vaimentaa korkeataajuiset signaalit.
API	Application Programming Interface, ohjelmointirajapinta. Mahdollistaa tiedonsiirron ohjelmien ja komponenttien välillä.
BLOB	Binary Large Object. Tietokantaan tallennettava tietotyyppi, joka sisältää dataa binäärimuodossa, esimerkiksi kuva- tai audiotiedosto.
BSON	Binary JSON. Laajennettu JSON-tiedostomuoto, jossa on tuki 32- ja 64-bittisille kokonaisluvuille, binääridatalle ja binääridatataulukoille. BSON tukee kaikkia JSON-datatyyppejä.
Clustering	Klusterointi. Malli, jossa tyypillisesti tietokannan yksi solmu jakaa tehtäviä muiden solmujen kesken transaktioiden nopeuttamiseksi.
D/A-muunnos	Digitaali-analogiamuunnos. Digitaalisessa muodossa olevan signaalin muuttaminen analogiseksi.
DBMS	Database Management System, tietokannan hallintajärjestelmä. Ohjelmisto, jolla käsitellään tietokantaan tallennettua tietoa.
DDC	Digital Down Converter, digitaalinen alassekoitin. Käytetään esimerkiksi muuttamaan A/D-muunnettu signaali välitaajuudelta kantataajuiseksi kompleksiseksi IQ-signaaliksi.

Docker	Sovelluskontteihin perustuva ohjelmistokehityksen teknologia. Sovellukset kehitetään ja paketoidaan standardoiduissa konteissa, joita voidaan siirtää sellaisenaan alustalta toiselle.
Demodulointi	Informaatiota sisältävän signaalin erottaminen kantoaallosta. Vrt. modulointi.
Diskreettiaikainen	Epäjatkuva.
Entiteetti	Yleisnimitys oloille, oliojoukoille ja ainemäärille.
FFT	Fast Fourier Transform, nopea Fourier-muunnos. Tehokas algoritmi Fourier-muunnoksen ja sen käänteismuunnoksen laskemiseen.
Fourier-muunnos	Integraalimuunnosta hyödyntävä algoritmi. Käytetään signaalinkäsittelyssä esimerkiksi taajuusanalyysiä vaativissa sovelluksissa.
FSK	Frequency shift-keying, taajuusavainnus. Digitaalinen modulointitekniikka.
GNU Radio	Avoimen lähdekoodin ohjelmistokehitystyökalu ohjelmistoradioiden toteuttamiseen.
HDFS	Hadoob Distribute File System. Apache Hadoop-ohjelmistokehykseen perustuva hajautettu tiedostojärjestelmä.
Hilbertin muunnos	Signaalinkäsittelyssä käytettävä integraalimuunnos, joka lisää +/-90-asteen vaihe-eron signaalin taajuuskomponentteihin.

Indeksointi	Relaatiotietokannan tietojen järjestämistä ja hakuja nopeuttava tekniikka. Indeksi on järjestetty kopio taulun sarakkeesta. Sen avulla voidaan tehdä uusia hakuja taulusta, joka on järjestetty toisen sarakkeen tietojen perusteella.
IQ tai I/Q	In phase -quadrature. Signaalin esittäminen kompleksisessa muodossa kahtena keskenään 90-asteen vaihe-erossa olevana osana.
JSON	JavaScript Object Notation. Tekstimuotoinen avoin tiedostostandardi.
Kvadratuuri	90-asteen kulmaero.
Massadata	Big Data. Suuri määrä muodoltaan vaihtelevaa dataa, joka lisääntyy jatkuvasti.
Modulointi	Signaalin muokkaaminen toisella signaalilla. Tiedonsiirrossa modulaation avulla siirretään informaatiosignaali siirtotiellä toisen signaalin yhteydessä. Vrt. demodulointi.
Msps	Megasamples per second, millions of samples per second. Näytteenottonopeuden kerrannaisyksikkö, voidaan ilmaista myös näytteenottotaajuutena (Mhz).
NCO	Numerically Controlled Oscillator. Ohjelmoitava, digitaalinen signaaligeneraattori, jota käytetään esimerkiksi ohjelmistoradioissa ja tutkajärjestelmissä.
NoSQL	Not only SQL. Vakiintunut nimitys tietokannoista, jotka perustuvat johonkin muuhun kuin relaatiomalliin.

Oskillaattori	Värähtelevä piirielementti, jota käytetään taajuussekoittimessa signaalin siirtämiseksi halutulle väli- tai radiotaajuudelle.
PMR	Personal Mobile Radio, radioluvasta ja taajuusmaksuista vapautettu radiopuhelinstandardi.
Skeema	Tietokannan tietomallin määrämuotoinen esitystapa esimerkiksi kaaviona.
Spatiaalinen	Avaruudellinen. Tilaa, sijaintia tai välimatkaa koskeva.
Spektri	Mitatun suureen jakaantuminen osiin taajuuden suhteen.
SQL	Structured Query Language. Relaatiotietokannoissa käytettävä kyselykieli.
Strukturoitu	Rakenteellinen, jäsennelty.
Taajuus	Toistuvan ilmiön tapahtumien määrä aikayksikköä kohti. Taajuuden yksikkö on 1/s eli hertsi (Hz).
Tavujärjestys	Määrittelee, missä järjestyksessä tietokone käsittelee suurempia kuin yhden tavun kokonaisuuksia. Little endian tarkoittaa, että vähiten merkitsevät bitit tallentuvat ensin. Big endian tarkoittaa, että eniten merkitsevät bitit tallentuvat ensin.
Transaktio	Tietokannassa tehtävä hauista ja tallennuksista muodostuva operaatio.

1 Johdanto

Avoimen lähdekoodin sovellukset ovat yleistyneet viime vuosina ja niiden käyttökohteita on lukuisia. Sovelluksia hyödynnetään paitsi yksityisissä projekteissa myös kaupallisissa järjestelmissä. Avoimen lähdekoodin ohjelmien kehitystyö on yleensä nopeaa ja joustavaa, joten niillä pystytään vastaamaan monipuolisiin tarpeisiin. Massadatan yleistyminen ja muun muassa tekoälyn hyödyntäminen sovelluksissa on luonut uusia tarpeita tietokantaratkaisuille. Tietokantaan ei tallenneta enää pelkästään yksittäisiä tarkkaan strukturoituja dataelementtejä, vaan joissain käyttötapauksissa tietokantaan halutaan tallentaa myös laajempia kokonaisuuksia erilaisissa tiedostomuodoissa. Tähän tarpeeseen löytyy vaihtoehtoja erityisesti NoSQL-tietokannoista.

Digitaalisessa signaalinkäsittelyssä ja esimerkiksi ohjelmistoradioissa hyödynnettävä kompleksinen IQ-data on mielenkiintoinen kohde tiedon varastoinnin näkökulmasta. IQ-datalle ei ole määritelty standardia, vaan datan tallennusformaatti vaihtelee laitevalmistajittain. Tämän opinnäytetyön tavoitteena on selvittää avoimen lähdekoodin tietokantojen soveltuvuutta IQ-muotoisen signaalidatan tallentamiseen. Työn tilaaja on Puolustusvoimien Tutkimuslaitoksen Informaatiotekniikkaosasto.

Tietokanta on hyvin yleinen opinnäytetyön aihe Tieto- ja viestintätekniikan koulutusalalla. Ammattikorkeakoulujen avoimesta opinnäytetyöportaalista Theseuksesta löytyy 2318 työtä hakusanalla ”tietokanta”. Digitaalista signaalinkäsittelyä käsitteleviä töitä löytyy samalta koulutusalalta muutamia kymmeniä, mutta IQ-dataa käsitteleviä töitä vain yksi. Molempia aiheita sivuavia töitä portaalista löytyy myös yksi. Työssä käsitellään lääketieteen neurofysiologisten mittalaitteiden datatiedostojen arkistointia palvelimelle.

Opinnäytetyön teoria jakautuu kahteen osaan. Työn toisessa luvussa käydään läpi tietokantojen teoriapohja. Luvussa käsitellään yleisimpiä tietomalleja ja tietokantatyyppejä. Lisäksi luvussa kuvataan lyhyesti tietokannan suunnitteluprosessia. Kolmannessa luvussa käsitellään perusteoriaa digitaalisesta signaalinkäsittelystä ja IQ-datasta. Luku antaa pohjatiedon IQ-datasta ja sen tyypillisistä käyttökohteista. Luvussa käsitellään myös, miten ja missä muodossa IQ-dataa tallennetaan erilaisissa markkinoilla olevissa sovelluksissa.

Työn neljännessä luvussa kuvataan tapaustutkimuksena toteutettu selvitystyö, jossa vertaillaan kymmentä avoimen lähdekoodin tietokantaa ja arvioidaan niiden soveltuvuutta IQ-datan tallentamiseen. Luvussa kuvataan myös opinnäytetyön aiheena olevan tietokannan käyttötapaus ja tehtäväselostus. Tapaustutkimuksen tavoitteena on vastata seuraaviin tutkimuskysymyksiin:

- Miten tapaustutkimukseen valitut avoimen lähdekoodin tietokannat soveltuvat yleisesti IQ-datan tallentamiseen?
- Mitkä tutkimuksen kohteena olevista tietokannoista soveltuvat IQ-datan tallentamiseen?
- Mikä on tutkimuksen pohjalta käyttöön soveltuvin tietokanta?
- Millaisia käyttöliittymiä soveltuvimmaksi arvioidun tietokannan kanssa voidaan käyttää?

Työn viidennessä luvussa esitellään tutkimuksen pohjalta toteutettu käytännön testaus. Luvussa kuvataan testijärjestelyt ja testauksessa käytetty tietokantaratkaisu. Käytännön testauksen tavoitteena on syventää arviota soveltuvimmaksi analysoidun tietokannan sopivuudesta IQ-datan tallentamiseen. Tapaustutkimuksen ja käytännön testin tavoitteena on raportoida työn tilaajalle, miten avoimen lähdekoodin tietokantoja voisi hyödyntää IQ-datatallenteiden ja niihin liittyvän metadatan varastointiin.

2 Tietokanta

Tietokanta on sähköinen, ennalta määritellyllä tavalla järjestetty kokoelma dataa (Coronel & Morris, 2015, s. 7). Datalla tarkoitetaan koneluettavaa ja digitaalisesti tallennettua käsittelemätöntä raakatietoa, jonka avulla voidaan muodostaa informaatiota. Informaatio on käsiteltyä, tulkittua ja jalostettua tietoa. (Kansalliskirjasto, 2016) Tietokantaan tallennettu data voi olla loppukäyttäjän tarvitsemaa dataa tai siihen liittyvää metadataa. Metadata on tietoa tietokannassa olevan tiedon ominaisuuksista. Metadataa voi olla esimerkiksi yksittäisen tietokannassa olevan dataelementin nimi tai tyyppi. (Coronel & Morris, 2015, s. 7) Data voi olla muodoltaan strukturoitua, semistrukturoitua tai strukturoimatonta (Castrounis, n.d.).

Tietokannat eroavat tekstinkäsittelyohjelmilla tehdyistä luetteloista ja laskentataulukoista siten, että tietokanta on suunniteltu usean käyttäjän suorittamaan suuren tietomäärän hallintaan ja käsittelyyn. Taulukot, kuten Microsoft Excel toimivat ensisijaisesti yhden tai maksimissaan muutaman käyttäjän työkaluina kohtalaisen tietomäärän käsittelyyn. (Oracle, n.d.a) Taulukossa olevan tiedon määrän kasvu saattaa johtaa päällekkäisyyksiin ja vaikealukuisuuteen, jolloin tiedon siirtäminen tietokantaan on suositeltavaa (Microsoft, n.d.).

Tietokanta voi olla käyttötietokanta tai analyyttinen tietokanta. Käyttötietokantoja käytetään tiedon keräämiseen, muokkaamiseen ja varastointiin. Käyttötietokantaan tallennettava data on yleensä dynaamista; sitä päivitetään ja muokataan säännöllisesti. Analyyttiseen tietokantaan tallennetaan ajasta riippuvaa staattista historiatietoa, joka ei pääsääntöisesti muutu tallentamisen jälkeen. (Hernandez, 2000, s. 3–4) Tietokantaa hallitaan tietokannan hallintajärjestelmällä (DBMS), joka toimii rajapintana käyttäjän ja tietokannan välillä (Coronel & Morris, 2015, s. 7). Tietokannan hallintajärjestelmällä ylläpidetään, muokataan ja käsitellään tietokantaa sekä luodaan uusia tietokantoja (Hernandez, 2000, s. 17).

2.1 Tietomallit

Tietokannan rakenne perustuu tyypillisesti johonkin seuraavista tietomalleista:

- relaatiomalli
- oliomalli
- hierarkkinen malli
- verkkomalli
- ER-malli (Coronel & Morris, 2015, s. 40-41).

Relaatiomallissa tieto tallennetaan relaatioihin, eli tauluihin rivi- ja sarakemuotoon. Taulun rivit ovat uniikkeja tietueita ja sarakkeet kenttiä eli tietueiden attribuutteja. Jokaisella sarakkeella tulee olla määritelty ja uniikki nimi. Data tallennetaan relaatiomallissa taulun yksittäisiin soluihin. Rivien ja sarakkeiden järjestyksellä tai tiedon fyysisellä sijainnilla ei ole tiedonhaun näkökulmasta merkitystä. (Kroenke & Auer, 2011, s. 69) Oliomalli perustuu olio-

ohjelmoinnin tavoin olioihin. Olio sisältää dataa, kuten tekstiä, ääntä, videota tai kuvaa sekä metodeita. Metodit ovat ohjelmakoodia, joka sisältää olion sisältämän datan mahdolliset toiminnot. Oliotietokantoja käytetään olio-ohjelmointiin perustuvilla ohjelmointikielillä, kuten C++ tai Python. (Panwar, 2021) ER-malli perustuu olioiden sijasta entiteetteihin, jotka voivat olla olioita tai olioryhmiä (Coronel & Morris, 2015, s. 44; Tieteen termipankki, 2016).

Hierarkkiseen tietomalliin perustuvassa tietokannassa data järjestetään puumaiseksi tasorakenteeksi. Tietokannassa olevan datan keskinäinen suhde määritellään vanhempi-lapsi-näkökulmasta. Vanhemmalle voidaan määrittää useampi lapsi, mutta lapsella voi olla vain yksi vanhempi. (Panwar, 2021) Hierarkkinen tietokanta mahdollista yleensä nopean tiedonhaun yksiselitteisten vanhempi-lapsi-linkitysten ansiosta (Hernandez, 2000, s. 6). Toisaalta hierarkkinen malli ei mahdollista lapsi-tietorakenteiden keskinäisten suhteiden määrittämistä, joten usein uuden tiedon tai tietotyypin lisääminen johtaa koko tietokannan uudelleenjärjestämiseen. Esimerkiksi IBM:n tietokanta Information Management System hyödyntää hierarkkista tietomallia. Yksi hierarkkisen tietomallin tyyppi on verkkomalli, joka perustuu entiteettien välisiin suhteisiin. Toisin kuin hierarkkisessa mallissa, verkkomallissa lapsella voi olla useampi vanhempi. (Panwar, 2021)

2.2 Relaatiotietokannat

Relaatiomalliin perustuvat relaatiotietokannat ovat suosituimpia ja käytetyimpiä tietokantoja. Relaatiotietokantoja käytetään SQL-kyselykielellä. (Tobin, 2020) Relaatiomallin mukaisesti data tallennetaan tauluihin riveihin ja sarakkeisiin (Oracle, n.d.b). Taulujen väliset suhteet muodostetaan viiteavainten avulla (Hernandez, 2000, s. 44). Relaatiotietokantaan tallennetun datan virheettömyys ja saavutettavuus pyritään takaamaan eheyssäännöillä, joiden avulla voidaan esimerkiksi estää tuplarivien muodostaminen (Oracle, n.d.b). Tiedon eheyden saavuttamiseksi relaatiotietokannoissa noudatetaan tyypillisesti ACID-mallia, joka koostuu seuraavista periaatteista:

- Jakamattomuus (Atomicity): tietokannassa tehtävä transaktio suoritetaan kokonaan.
- Eheys (Consistency): tietokannassa oleva tieto säilyy muuttumattomana ja eheänä.
- Eristyneisyys (Isolation): transaktiot ovat riippumattomia toisistaan.

- Pysyvyys (Durability): transaktion myötä tehtävät muutokset dataan ovat pysyviä. (Oracle, n.d.b)

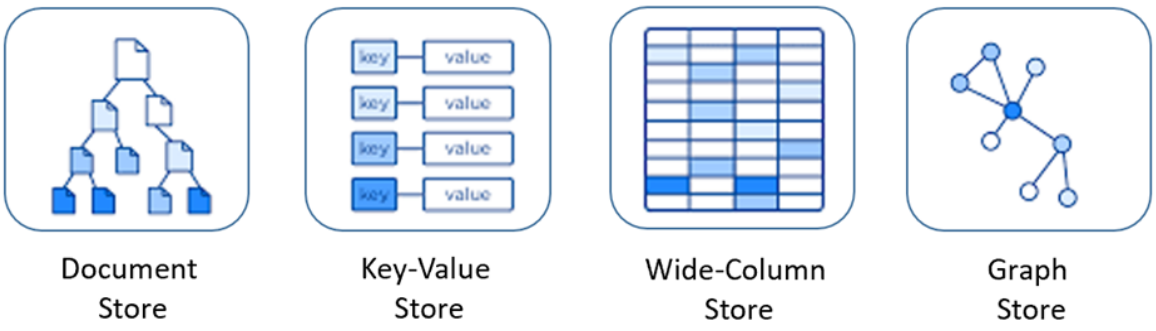
Relaatiotietokannassa taulujen ja niiden tietueiden välille määritellään aina tietynlainen yhteys. Yhteystyyppejä ovat yhdestä yhteen, yhdestä moneen ja monesta moneen. Yhdestä yhteen -yhteys tarkoittaa, että taulun tietue liittyy vain yhteen toisen taulun tietueeseen ja sama päinvastoin. Yhdestä moneen -yhteydessä tietue voi liittyä toisen taulun yhteen tai useampaan tietueeseen, mutta toisen taulun tietue voi edelleen liittyä vain yhteen ensimmäisen taulun tietueeseen. Monesta moneen -yhteydessä taulujen tietueet voivat liittyä puolin ja toisin useampaan vastakkaisen taulun tietueeseen. (Hernandez, 2000, s. 46–49)

Relaatiotietokanta soveltuu erityisesti tilanteisiin, joissa tallennettava data koostuu vain muutamista erilaisista tietotyypeistä, ja datan määrä sekä laatu pysyvät koko tietokannan elinkaaren ajan vakioituina. Haasteena relaatiotietokannassa on skaalautuvuus tilanteessa, jossa tietokannan kokoa on tarpeen kasvattaa. Lisäksi relaatiotietokannan rakennetta on haastavaa muuttaa jälkikäteen. (Tobin, 2020) Suosittuja avoimen lähdekoodin relaatiotietokannan hallintajärjestelmiä ovat MySQL, PostgreSQL ja MariaDB (DB_Engines, 2021a).

2.3 NoSQL-tietokannat

Tietokannoista, jotka eivät käytä relaatiomallia ja SQL-kyselykieltä, käytetään yleensä nimitystä NoSQL-tietokanta. NoSQL-tietokannat eivät hyödynnä vain yhtä standardoitua tietomallia. (Coronel & Morris, 2015, s. 51) NoSQL-tietokantatyyppejä ovat muun muassa avain-arvotietokannat, saraketietokannat, dokumenttitietokannat, verkkotietokannat, spatiaaliset tietokannat sekä aikasarjatietokannat (Drake, 2014; DB-Engines, 2021c). Sivulla 6 olevassa kuvassa 1 on esitetty perusrakenne tyypillisimmistä NoSQL-tietokantatyypeistä dokumenttitietokannasta, avain-arvotietokannasta, laajasta saraketietokannasta sekä verkkotietokannasta.

Kuva 1. Yleisiä NoSQL-tietokantarakenteita. (Microsoft, 2021)



Suosittuja avoimen lähdekoodin NoSQL-tietokannan hallintajärjestelmiä ovat:

- Cassandra (laaja saraketietokanta),
- HBase (laaja saraketietokanta),
- InfluxDB (aikasarjatietokanta),
- MongoDB (dokumenttitietokanta),
- Neo4j (verkkotietokanta) sekä
- Redis (avain-arvotietokanta) (DB-Engines, 2021c).

Avain-arvo-tietokantaan tallennetaan dataa avain-arvopareina, joissa avain yksilöi arvon. Tietokantaan tallennettuja arvoja voi hakea avaimen avulla. Avain-arvo-tietokannassa ei käytetä tauluja, rivejä eikä sarakkeita, vaan data tallennetaan tietokantaan hakemistoihin ilman ennakkoon määriteltyä rakennetta. Arvojen tietotyypit voivat olla joko hyvin yksinkertaisia tai monimutkaisempia kokonaisuuksia, kuten kokonaislukuja, merkkijonoja tai olioita esimerkiksi JSON-muodossa. Avain-arvo-tietokantoja käytetään esimerkiksi välimuistitallennuksissa, käyttöjärjestelmien viestijonoprosesseissa sekä sessionhallinnassa. (Drake, 2014)

DB-Engines-sivuston (2021a) mukaan suosituin avoimen lähdekoodin avain-arvotietokannan hallintajärjestelmä on Redis. Redis-tietokannassa arvoja käsitellään yksinkertaisilla SET- ja GET-operaatioilla. SET-operaatio asettaa avaimen ja sille arvon. GET-operaatio hakee tietokannasta pyydetylle avaimelle arvon. (Redis Ltd, n.d.) Redis-tietokantaa voidaan käyttää tietokantana, mutta myös välimuistina tai viestinvälittäjänä. Sitä voidaan hyödyntää myös tietokantakyselyiden nopeuttamiseksi toisen tietokannan rinnalla. (Vanhatapio, 2020)

Saraketietokannassa data tallennetaan sarakkeisiin, jotka sijaitsevat omissa tiedostoissaan tai palvelimen järjestelmämuistin osissa. Sarakkeiden tiedot linkittyvät toisiinsa merkintäjärjestyksessä. Tämän ansiosta haut ja kyselyt voidaan suorittaa relaatiotietokantaa nopeammin, kun tiedot haetaan vain haun kannalta oleellisista sarakkeista. (Drake, 2014) The Apache Foundationin Cassandra ja HBase ovat käytetyimpiä avoimen lähdekoodin saraketietokannan hallintajärjestelmiä (DB-Engines, 2021a). Cassandra-tietokanta voidaan hajauttaa useaksi solmuksi eri palvelimille. Tietokannan hajautetut solmut liikennöivät keskenään hyödyntäen gossip-protokollaa. Solmut ovat keskenään samanarvoisia, jolloin arkkitehtuuri ei vaadi master-solmun nimeämistä. Data tallennetaan tauluihin, sarakkeisiin ja riveihin. Cassandra-tietokannassa käytetään CQL-kyselykieltä, joka on syntaksiltaan hyvin samanlainen SQL-kyselykielen kanssa. (The Apache Softaware Foundation, 2021a)

HBase-tietokanta on suunniteltu erityisesti massadatan käsittelyyn. HBasea voidaan käyttää sekä hajautettuna että paikallisesti. Hajautettu käyttö perustuu HDFS-tiedostojärjestelmään ja paikallisessa käytössä tietokanta hyödyntää palvelimen paikallista tiedostojärjestelmää. Cassandran tavoin data tallennetaan tauluihin, sarakkeisiin ja riveihin. HBase-tietokannan taulussa on rivejä, joilla on jokaisella oma riviavaimensa. Lisäksi riviin voi liittyä yksi tai useampia sarakkeita arvoineen. Sarakkeita voidaan yhdistellä perheiksi. Vaikka terminologia on samanlainen relaatiomallin kanssa, ei saraketietokannan tauluja voi rinnastaa relaatiotietokannan tauluihin. Saraketietokannan taulut voidaan enemmänkin kuvata eräänlaisia moniulotteisina karttoina. (The Apache Software Foundation, 2021b)

Dokumenttitietokannan rakenne on hyvin joustava ja sen vuoksi se soveltuu hyvin semi-strukturoidun datan tallentamiseen (Tobin, 2020). Dokumenttitietokanta voidaan mieltää myös avain-arvotietokannaksi, jossa dokumentit ovat arvoja ja jokaisella dokumentilla on uniikki avain. Tyypillisesti dokumenttitietokannassa käytetään XML-, JSON-, BSON- tai YAML-tiedostomuotoja, mutta dokumentit voivat olla muodoltaan ja rakenteeltaan myös muunlaisia. Dokumenttitietokantoja käytetään esimerkiksi verkkokaupoissa, blogisivustoilla ja sisällönhallintajärjestelmissä. Yksi käytetyimmistä dokumenttitietokannan hallintajärjestelmistä on MongoDB. (Drake, 2014)

MongoDB -tietokanta tallentaa datan JSON-muotoiseen tiedostoon binäärisessä BSON-muodossa. Tietokannan rakenne eli skeema ei vaadi määrittelyä etukäteen, jolloin tietokannan rakennetta voi muuttaa ja päivittää joustavasti tarpeiden mukaan. (MongoDB, 2020, s. 3) MongoDB mahdollistaa monimutkaiset kyselyt esimerkiksi säännöllisten lausekkeiden avulla ja sen API-toiminnallisuudet ovat monipuoliset. MongoDB-tietokantaa käyttävät muun muassa GitHub, SourceForge ja bit.ly. (Kuittinen, 2011)

Verkkotietokannassa tieto tallennetaan dokumenttitietokannan tavoin tietynlaisiin dokumentteihin. Verkkotietokannassa korostuvat dokumenttien väliset suhteet. Verkkotietokannan peruselementtejä ovat solmu (node), kaari (edge) ja ominaisuus (property). Solmu on yksittäinen dokumentti, jolla voi olla erilaisia ominaisuuksia. Kaarilla kuvataan dokumenttien välisiä suhteita. Verkkotietokantoja käytetään erityisesti tapauksissa, jossa entiteettien välisillä suhteilla on merkitystä, kuten sosiaalisessa mediassa, maksuvälinepetosten ennaltaehkäisyssä ja suosittelusovelluksissa. Yksi suosituimmista avoimen lähdekoodin verkkotietokannoista on Neo4j. (Drake, 2014) Neo4j on helppokäyttöinen, ACID-mallin mukainen ja hyvin skaalautuva tietokanta (Neo4j, n.d.).

Spatiaalisia tietokantoja käytetään muun muassa välimatkaa tai sijaintia koskevan datan tallentamiseen. Spatiaalisia tietokantoja käytetään erityisesti paikkatietojärjestelmissä. Aikasarjatietokantaan vastaavasti tallennetaan entiteettejä, joilla on aina aikaleima. Dataa aikasarjatietokantaan voivat tuottaa esimerkiksi IoT-järjestelmien sensorit, sähkömittarit tai RFID-tunnisteet. (DB-Engines, 2021c)

NoSQL-tietokantoihin voidaan tallentaa monipuolisesti erilaista dataa ja niitä on tyypillisesti helppo skaalata ja laajentaa (Tobin, 2020). Tunnettuja NoSQL-tietokantojen käyttäjiä ovat muun muassa Amazon, YouTube, Google Maps ja Facebook (Coronel & Morris, 2015, s. 51). Relaatiotietokantaan verrattuna NoSQL-tietokannat ovat yleensä melko yksinkertaisia suunnitella. Koska NoSQL-tietokannat ovat vielä uudehkoja sovelluksia, niihin ei ole tarjolla yhtä laajaa käyttäjätukea kuin relaatiotietokantoihin. Samasta syystä tietokannoista saattaa puuttua esimerkiksi tarpeellisia analyysityökaluja. NoSQL-tietokantoja ei myöskään ole välttämättä mahdollista integroida osaksi muita järjestelmiä tai sovelluksia. (Tobin, 2020)

2.4 Tietotyypit

Tietokantaan tallennettavan datan luonne kuvataan tietotyypillä. Yleisiä tietotyyppejä ovat aakkosnumeerinen data, numeerinen data, päiväys ja kellonaika. Aakkosnumeerisiin tietotyyppeihin kuuluvat kirjaimet, numerot, laajennetut merkit ja erikoismerkit sekä niiden yhdistelmät. Numeerisia tietotyyppejä ovat kokonaisluvut ja reaaliluvut. (Hernandez, 2000, s. 250) Käytettävät tietotyypit ja niiden jaottelu vaihtelevat tietokannan hallintajärjestelmittäin. Esimerkiksi Apache Cassandra -tietokannan hallintajärjestelmässä tietotyypit jaetaan natiiveihin (native), kokoelmiin (collection), käyttäjän määrittelemiin (user-defined), monikkoihin (tuple) ja mukautettuihin (custom) tietotyyppeihin (The Apache Software Foundation, 2021a). Avain-arvotietomallin perustuvassa Redis- tietokannan hallintajärjestelmässä vastaavasti jaotellaan tietotyypit merkkijonoihin(strings), listoihin (lists), joukkoihin (sets), tiivisteisiin (hashes), lajiteltuihin joukkoihin (sorted sets), bittikarttoihin ja HyperLogLog -algoritmeihin (Redis Ltd., n.d.).

Digitaalisen median yleistyttyä sovellusten käyttämien olioiden koko on kasvanut. Suuria olioita voidaan tallentaa tiedostoina tiedostojärjestelmään tai suurina binääri-tietotyyppeinä (BLOB) tietokantaan. Tiedostojärjestelmää käytettäessä tietokantaa voidaan hyödyntää tiedostojen metadatan tallentamiseen. (Sears, van Ingen & Gray, 2006, s. 2, 5) Searsin, van Ingenin ja Grayn (2006, s. 2) Microsoft Researchille ja Kalifornian yliopistolle tekemässä tutkimuksessa vertailtiin tiedostojärjestelmän ja tietokannan käyttöä sekä suorituskykyä suuria tiedostoja tallennettaessa. Tutkimuksessa todettiin, että BLOB-tietotyyppi ja tietokanta soveltuvat hyvin alle 256 kilotavun kokoisten olioiden tallentamiseen. Tiedostojärjestelmää suositeltiin yli 1 megatavun kokoisten ja suurempien olioiden tallentamiseen. 256 kt-1 Mt kokoisille olioille sopiva tallennustapa riippuu käyttötapauksesta ja erityisesti siitä, kuinka intensiivisesti tietokantaan tallennettavia tietoja aiotaan muuttaa ja replikoida. (Sears ym., 2006, s. 11)

2.5 Tietokannan valinta ja suunnittelu

Sopivaa tietokannan hallintajärjestelmää valitessa tulee selvittää, millaista dataa tietokantaan tallennetaan ja mitä tallennetulla datalla halutaan tehdä (Chima, 2018). Tietokannan vaatimusmäärittely tehdään käyttäjän tarpeiden näkökulmasta (Watt & Eng,

2014, s. 85). Tietokantaa suunnitellessa laaditaan tehtäväselostus, joka määrittelee tietokannan tarkoituksen. Lisäksi määritellään tehtävän tavoitteet, jotka kuvaavat niitä tehtäviä, joita käyttäjien on tarkoitus toteuttaa tietokantaan tallennettavien tietojen avulla. (Hernandez, 2000, s. 68)

Kun tiedetään, minkä tyyppistä dataa tietokantaan tullaan tallentamaan, määritellään sen pohjalta käytettävä tietomalli sekä tietokannan tietorakenne. (Hernandez, 2000, s. 69; Watt & Eng, 2014, s. 85) Tietokannan suunnittelussa tulee lisäksi määritellä tietokantaan tallennettavan datan keskinäinen suhde sekä dataelementtien tietotyypit (Watt & Eng, 2014, s. 17). Tietokannan suunnittelu voidaan jakaa loogiseen ja fyysiseen suunnitteluun. Loogisen suunnittelun tuotteena syntyy käytettävän tietokannan hallintajärjestelmän tietomalli. Fyysisen suunnittelun tavoitteena on määrittää tietokannan rakenne, datan järjestäminen ja indeksointi. (Watt & Eng, 2014, s. 17)

Tietokannan suunnitteluprosessi viimeistellään normalisoimalla tai de-normalisoimalla. Normalisointia käytetään erityisesti relaatiomallissa. Normalisoinnin tavoitteena on poistaa tietokannasta päällekkäisyyksiä ja taata tiedon eheys siten, että data tallennetaan tietokantaan vain kerran ja yhteen paikkaan. (Krug, 2019) De- normalisoinnilla pyritään lisäämään tietokannan suorituskykyä ja nopeuttamaan tietokantakyselyitä. Tämä voidaan tehdä esimerkiksi lisäämällä harkiten tiedon päällekkäisyyttä tai ryhmittelemällä tietoa uudelleen. De-normalisointia käytetään tyypillisesti NoSQL -tietokannoissa. (Krug, 2019) Liitteeseen 1 on koottu kuvaus tietokannan suunnitteluprosessista.

2.6 Avoimen lähdekoodin tietokanta

Avoimen lähdekoodin ohjelmiston lähdekoodi on vapaasti kaikkien saatavilla ja hyödynnettävissä. Avoimen lähdekoodin käyttöä edistävä Open Source Initiative -järjestö on määritellyt avoimen lähdekoodin ohjelmille seuraavat kriteerit:

- Avoimen lähdekoodin ohjelmistoa saa levittää ja muokata vapaasti. Myös muokattua versiota tai niistä tehtyjä ohjelmistoja saa levittää vapaasti. Lisenssi voi kuitenkin edellyttää, että muokatuilla versioilla on esimerkiksi eri nimi tai versionumero alkuperäiseen verrattuna.

- Avoimen lähdekoodin lisenssistä ei veloiteta lisenssimaksuja.
- Avoimen lähdekoodin ohjelma ei saa syrjiä ihmisiä eikä ihmisryhmiä.
- Avoimen lähdekoodin ohjelman käyttökohteita ei saa rajoittaa.
- Avoimen lähdekoodin lisenssi ei saa olla riippuvainen tietystä tuotteesta tai ohjelmistokokonaisuudesta, vaan sitä tulee voida hyödyntää myös erillään laajemmasta kokonaisuudesta.
- Avoimen lähdekoodin lisenssi ei saa rajoittaa muiden ohjelmien toimintaa tai toteutustapaa. Lisenssillä ei esimerkiksi saa asettaa ehtoa, joka kieltää muiden kuin avoimen lähdekoodin ohjelmien käytön yhdessä lisenssin kanssa.
- Avoimen lähdekoodin lisenssi ei saa olla riippuvainen tietystä teknisestä toteutuksesta. (Open Source Initiative, 2007)

Muun muassa sosiaalinen media ja IoT-teknologia käyttävät valtavia datamääriä, jolloin datan tallentamiseen ja analysointiin tarvitaan joustavia ratkaisuja. Avoimen lähdekoodin tietokantojen ominaisuuksia hyödynnetään myös monimutkaisten keskinäisten yhteyksien etsimiseen suurista datamääristä, esimerkiksi suosittelupalveluissa ja maksuvälinepetosten ennaltaehkäisyssä. Avoimen lähdekoodin tietokannat mahdollistavat ohjelman muokkaamisen kustannustehokkaasti juuri tiettyyn tarpeeseen sopivaksi. (OmniSci, n.d.)

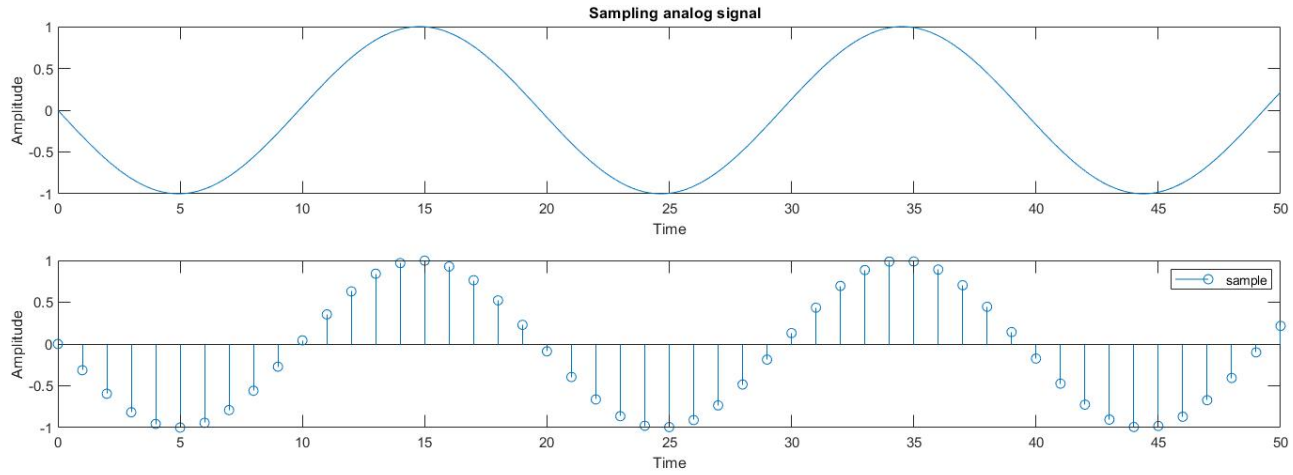
3 Digitaalinen signaalinkäsittely

Digitaalinen signaalinkäsittely on signaalin muokkaamista digitaalitekniikan ja matemaattisten algoritmien avulla. Digitaalisen signaalinkäsittelyn tavoitteena voi olla esimerkiksi informaation erottaminen signaalista. Yleensä digitaalinen signaalinkäsittely toteutetaan signaaliprosessoreilla tai korkean tason ohjelmointikielillä. (Jokinen, 2003, s. 17) Tyypillinen digitaalisen signaalinkäsittelyn sovelluskohde on tietoliikennetekniikka, jossa digitaalista signaalinkäsittelyä käytetään esimerkiksi moduloinnissa, demoduloinnissa sekä virheenkorjauksessa. Muita sovelluskohteita ovat muun muassa tutkajärjestelmät, kaikuluotaimet, navigointijärjestelmät, audiosignaalinkäsittely, kuvankäsittely sekä lääketieteen kuvantamismenetelmät. (Stranneby, 2001, s. 2–3)

3.1 Digitaalisen tiedonsiirron perusteita

Signaali voi olla jatkuva-aikainen eli analoginen tai diskreettiaikainen (Stranneby, 2001, s. 4). Diskreettiaikainen signaali ei ole jatkuva, vaan se on määritelty vain tiettyinä ajanhetkinä (Jokinen, 2003, s. 18). Useimmat sähköiset ja reaalimaailmassa ilmenevät signaalit ovat analogisia, esimerkiksi ääni audiotaajuisena signaalina. Analoginen signaali saadaan muunnettua digitaaliseksi ottamalla siitä signaalinäytteitä säännöllisin näytteenottovälein. (Räisänen & Lehto, 2011, s. 219) Näytteiden määrä sekunnissa määrittää signaalin näytteenottotaajuuden ja se saadaan laskemalla näytteenottovälin käänteisarvo (Jokinen, 2003, s. 45, 193). Alla olevassa kuvassa 2 on Matlab-ohjelmistolla luotu esimerkki analogisen signaalin näytteenotosta. Ylemmässä kuvaajassa on jatkuva-aikainen analoginen sinisignaali. Alemmassa kuvaajassa signaalista on otettu näytteitä näytteenottovälillä 1 ja näytteenottotaajuudella 1.

Kuva 2. Analoginen sinisignaali näytteistettynä



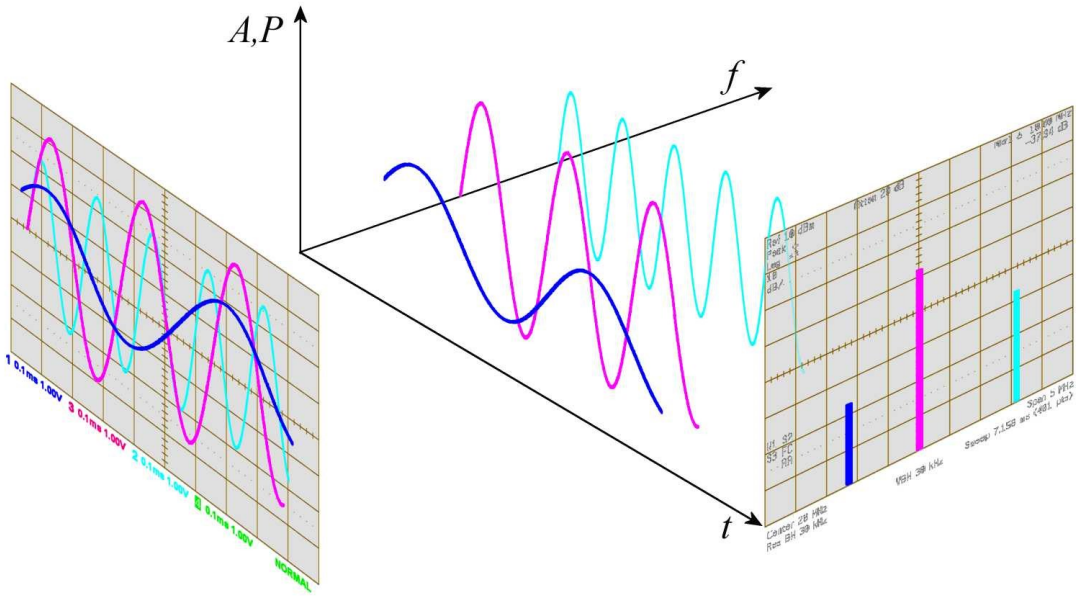
Signaalille sopiva näytteenottoväli ja -taajuus määritetään näytteenottoteoreeman eli Nyqvistin teoreeman avulla. Näytteenottoteoreeman mukaan alkuperäistä vastaava signaali voidaan muodostaa uudelleen signaalinäytteiden avulla, mikäli näytteenottotaajuus on vähintään kaksinkertainen verrattuna alkuperäisen signaalin suurimpaan taajuuskomponenttiin. (Huttunen, 2014, s. 4; Rouphael, 2014, s. 264; Stranneby, 2001 s. 6)

Analoginen signaali muutetaan digitaalisiksi signaalinäytteiksi analogia-digitaali- eli A/D-muuntimella (Stranneby, 2001, s. 36). Muunnettava signaali voi olla mitä tahansa

mitattavissa olevaa aikasarjaa, esimerkiksi ääntä, puhetta tai pulssia. Digitaalisen signaalinkäsittelyn menetelmillä signaalia suodatetaan käytettävän järjestelmän tai sovelluksen kannalta soveltuvampaan muotoon. Suodattamisen tavoitteena on esimerkiksi kohinan poistaminen signaalista tai signaalin kiinnostavien piirteiden erottaminen muun signaalin joukosta. Digitaalinen signaali voidaan muuttaa tarvittaessa takaisin analogiseksi digitaali-analogia- eli D/A-muuntimella. (Huttunen, 2014, s. 1–2)

Digitaalisessa signaalinkäsittelyssä signaalit voidaan esittää aikatasossa tai taajuustasossa eli spektrinä (Lichtman, n.d.). Alla olevassa kuvassa 3 on tarkasteltu kolmea sinimuotoista signaalia kolmiulotteisesti. Kolmiulotteisesta esityksestä ilmenee sekä oskilloskoopilla tarkasteltava signaalin aikataso että spektrianalysaattorilla tarkasteltava taajuustaso. (Radartutorial, n.d.) Signaalin taajuusjakauma selvitetään Fourier-integraalimuunnoksella (Lichtman, n.d.). Huttunen (2014, s. 33) kuvailee Fourier-muunnoksen tavoitteeksi selvittää ”kuinka paljon signaalissa on kutakin taajuutta”.

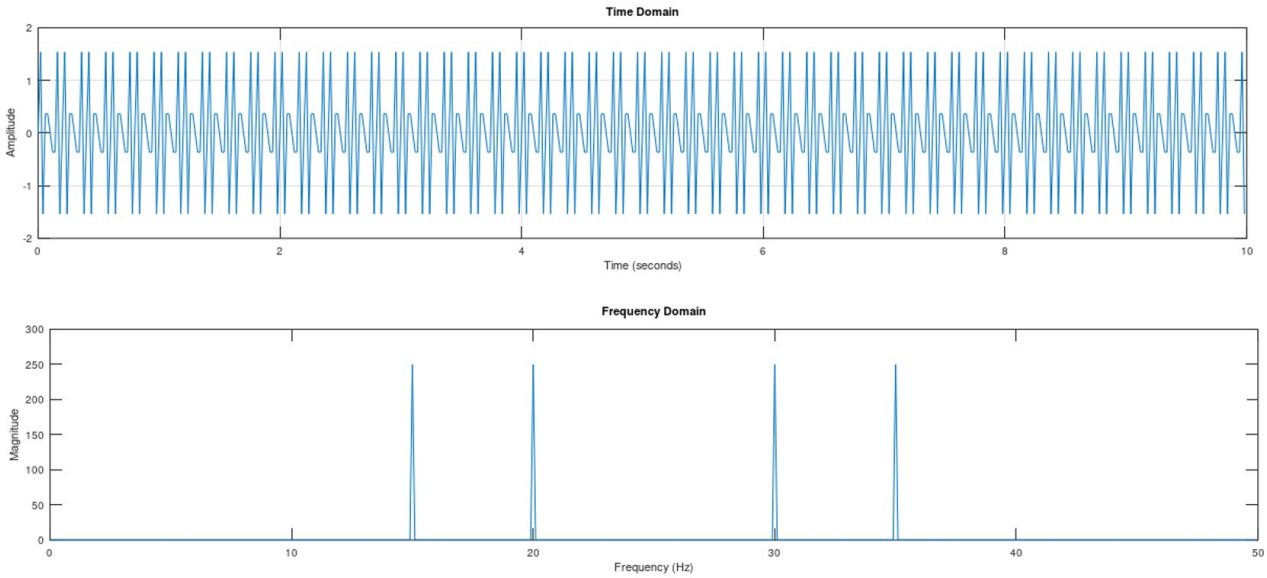
Kuva 3. Signaalin aikataso (t) ja taajuustaso (f) (Radartutorial, n.d.)



Usein signaalinkäsittelyssä käytetään algoritmia Fast Fourier Transform, eli nopea Fourier-muunnos (Lichtman, n.d.). Sivulla 14 olevassa kuvassa 4 on Matlab-ohjelmistolla esitetty signaali sekä aika- että taajuustasoissa. Taajuustason selvittämiseksi signaalille on tehty nopea Fourier-muunnos. Fourier-muunnoksen toteuttaminen sellaisenaan edellyttää, että

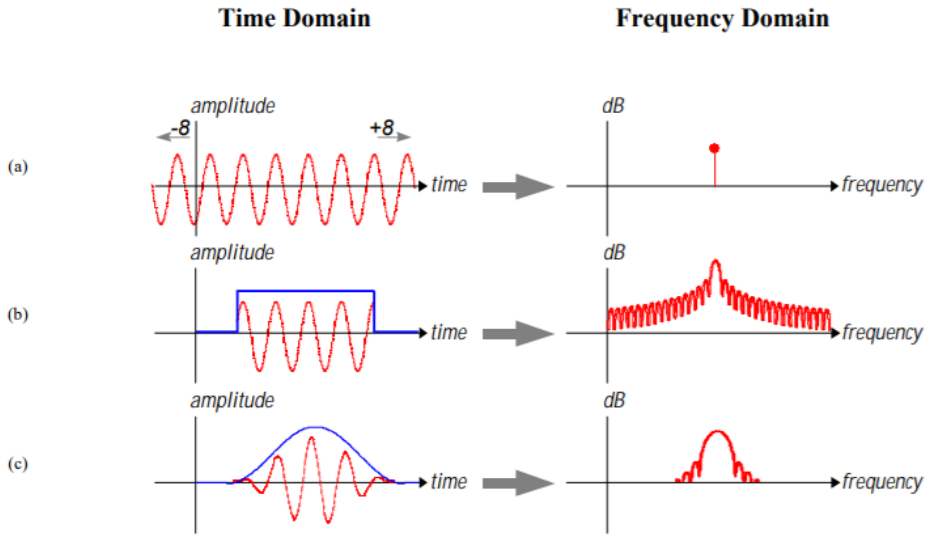
signaali on käytettävissä äärettömänä. Koska signaalia voidaan käytännössä tarkastella ja taltioida vain äärellisesti, tulee taajuusjakauman selvittämiseksi käyttää ikkunointia. (Henderson, 2004, s. 2)

Kuva 4. Signaalin muutos aikatasosta taajuustasoon nopealla Fourier-muunnoksella



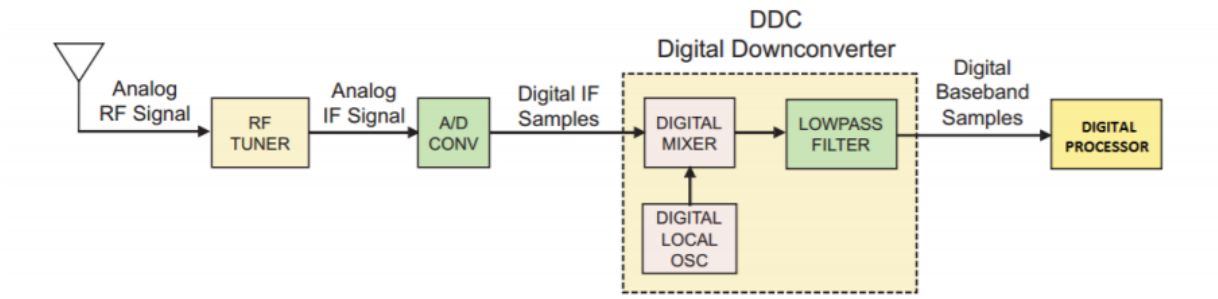
Nopea Fourier -muunnos olettaa, että tarkasteltava näyte on tasan jaksollisen signaalin yksi tai useampi jakso, jolloin näytteen alku- ja loppupää ovat keskenään jatkuvia. Yleensä kuitenkin tarkasteltava signaalin alku ja loppu ovat keskenään epäjatkuvia, jolloin nopea Fourier-muunnos aiheuttaa signaalin taajuusjakauman vuotamisen sellaisille taajuuksille, joita alkuperäisessä signaalissa ei ole. (National Instruments, n.d., s. 8–9) Tämän estämiseksi signaali kerrotaan ikkunafunktiolla. Ikkunafunktio rajaa tarkasteltavaa signaalia ajallisesti esimerkiksi kaventaen signaalin alku- ja loppupään amplitudin arvoa kohti nollaa. Tämä vähentää alun ja lopun keskinäistä epäjatkuvuutta sekä signaalivuotoa. (Henderson, 2004, s. 9) Ikkunafunktioita on erilaisia ja tilanteeseen sopiva ikkunafunktio riippuu kulloinkin tarkasteltavasta signaalista (National Instruments, n.d., s. 10). Sivulla 15 olevassa kuvassa 5 on esitetty äärettömän pituisen samanlaisena jatkuvan sinisignaalin ideaali taajuusjakauma (a) sekä saman signaalin äärellisesti tarkasteltava taajuusjakauma suorakaideikkunalla (b) ja Hanning-ikkunalla (c) kerrottuna (Henderson, 2004, s. 9).

Kuva 5. Signaalin ikkunointimenetelmiä (Henderson, 2004, s. 9)



Alla olevassa kuvassa 6 on esimerkki digitaalista signaalinkäsittelyä hyödyntävän ohjelmistoradiovastaanottimen lohkokaaviosta. Analoginen radiotaajuinen signaali vastaanotetaan ilmateitse antennilla ja muutetaan sopivalle välitaajuudelle edelleen analogisena. A/D-muunnin muuttaa välitaajuudella olevan analogisen signaalin digitaalisiksi signaalinäytteiksi. (Núñez & Mascareñas, 2016) Välitaajuiset signaalinäytteet muutetaan digitaalisella alassekoittimella kantataajuisiksi näytteiksi. Vastaanottimen alassekoitin sisältää digitaalisen taajuussekoittimen ja paikallisoskillaattorin sekä alipäästösuodattimen. Lopuksi näytteet käsitellään digitaalisella prosessorilla esimerkiksi demoduloiden tai dekoodaten. (Núñez & Mascareñas, 2016)

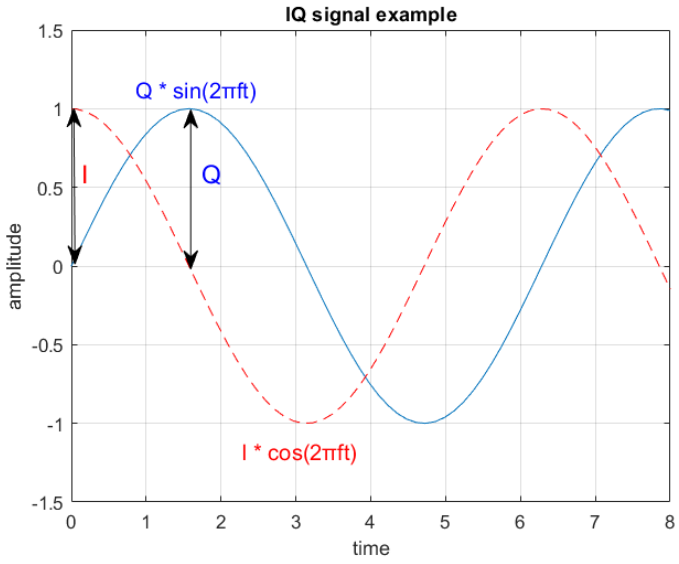
Kuva 6. Esimerkki ohjelmistoradion lohkokaaviosta (Núñez & Mascareñas, 2016)



3.2 IQ-signaali

IQ-signaali eli kvadratuurisignaali on digitaalisessa signaalinkäsittelyssä hyödynnettävä signaalin kompleksinen ja kaksiulotteinen esitysmuoto (Wolke, 2015). IQ-signaaliformaatissa reaalinen, sinimuotoinen signaali esitetään kahtena toisistaan 90-asteen vaihe-erossa olevana signaalina (All About Circuits, n.d.). Signaalit ovat tällöin keskenään kvadratuurissa. Toinen signaali on IQ-signaalin kosinimuotoinen komponentti I (in phase) ja toinen sinimuotoinen komponentti Q (quadrature). (Wolke, 2015) Kuva 7 esittää Matlab-ohjelmistolla simuloituna kahden keskenään kvadratuurissa olevan signaalin amplitudin eli värähdyslaajuuden kuvaajan ajan funktiona. Kuvan signaalit voidaan esittää alla olevien kaavojen 1 ja 2 mukaisesti, joissa I ja Q = amplitudi, f = taajuus ja t = aika (Lichtman, n.d.).

Kuva 7. Kvadratuurissa olevat signaalit



Kaava 1. I-signaalin kaava

$$I * \cos(2\pi f t) \tag{1}$$

Kaava 2. Q-signaalin kaava

$$Q * \sin(2\pi f t) \tag{2}$$

IQ-signaali voidaan esittää kompleksilukuna, jossa signaalin I-komponentti on luvun reaaliosa ja Q-komponentti luvun imaginaariosa (Lyons, 2000). Kompleksiluvun yleinen esitysmuoto on alla olevan kaavan 3 mukainen. Kaavassa a = reaaliosa, b = imaginaariosa ja i = imaginaariyksikkö. (Mäkelä, Soininen, Tuomola & Öistämö, 2005, s. 10) Imaginaariyksiköstä voidaan käyttää i-kirjaimen sijasta myös merkintää j (Valkama, 2001, s. 7).

Kaava 3. Kompleksiluvun yleinen esitysmuoto

$$z = a + b\mathrm{i} \tag{3}$$

Signaalin IQ-esitysmuodossa kompleksiluvun reaali- ja imaginaariosa ovat signaalin I- ja Q-komponenttien amplitudien arvot (Lichtman, n.d.). Sivulla 16 esitetyn kuvan 7 signaalien muodostama kompleksiluku z voidaan tällöin esittää alla olevan kaavan 4 yhtälön mukaisesti.

Kaava 4. Kvadratuurissa olevien signaalien esitys kompleksilukuna

$$z = 1 + 1 * \mathrm{i} \tag{4}$$

Kun kuvan 7 kvadratuurissa olevista signaaleista muodostetaan yksi reaalinen signaali, sen funktio on tällöin alla olevan kaavan 5 mukainen (Lichtman, n.d.).

Kaava 5. Kvadratuurissa olevien signaalien yhdistäminen

$$\mathrm{x(t)} = I * \cos(2\pi f t) + Q * \sin(2\pi f t) = 1 * \cos(2\pi f t) + 1 * \sin(2\pi f t) \tag{5}$$

Kaavan 5 ratkaisuun käytetään Tekniikan kaavaston (Mäkelä ym., 2005, s. 14) mukaisia trigonometrian perusyhtälöitä, jotka on kuvattu seuraavissa kaavoissa 6,7 ja 8.

Kaava 6. Trigonometrian perusyhtälö 1

$$a\sin(\alpha) + b\cos(\alpha) = r\sin(\alpha + \varphi) \tag{6}$$

Kaava 7. Trigonometrian perusyhtälö 2

$$r = \sqrt{a^2 + b^2} \tag{7}$$

Kaava 8. Trigonometrian perusyhtälö 3

$$\varphi = \arctan(\frac{b}{a}) \tag{8}$$

Trigonometrian perusyhtälöiden mukaan kuvan 7 I- ja Q-komponenteista saadaan tällöin muodostettua reaalinen signaali alla olevan kaavan 9 laskutoimituksen mukaisesti.

Kaava 9. Reaalisen signaalin muodostaminen I- ja Q-komponenteista

$$\mathrm{x}(t) = 1.414213562 * \sin(2\pi f t + 0.7853981634) \approx 1.41 * \sin(2\pi f t + 0.79) \tag{9}$$

Kaavan 5 lisäksi IQ-signaali voidaan esittää Eulerin kaavan avulla. Eulerin kaava on esitetty kaavassa 10 ja IQ-signaalin esitys Eulerin kaavan avulla kaavassa 11. (Mäkelä ym., 2005, s. 11; Lyons, 2000).

Kaava 10. Eulerin kaava

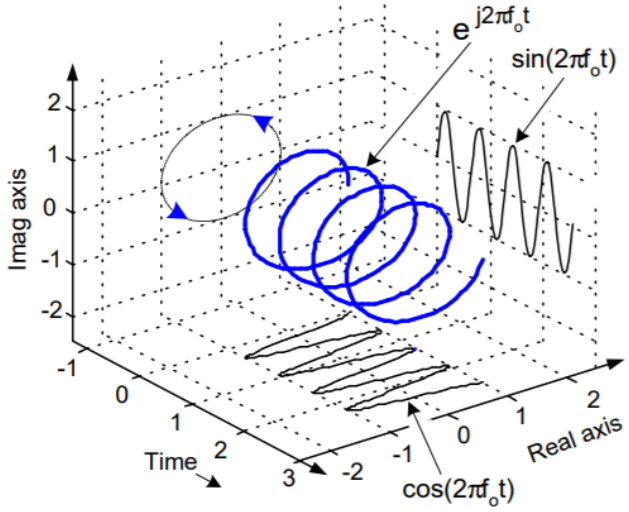
$$\mathrm{e}^{\mathrm{i}\varphi} = \cos\varphi + \mathrm{i} * \sin\varphi \tag{10}$$

Kaava 11. IQ-signaalin esitys Eulerin kaavan mukaisesti

$$\mathrm{x(t)} = \mathrm{e}^{\mathrm{i}2\pi f t} \tag{11}$$

Sivulla 19 olevassa kuvassa 8 on kolmiulotteinen esitys IQ-muotoisesta signaalista. Kuvaajassa näkyy signaalin imaginaariosa ja reaaliosa sekä niiden yhdistelmä Eulerin kaavan avulla esitettynä.

Kuva 8. Signaali esitettynä kompleksisessa IQ-muodossa (Lyons, 2000)



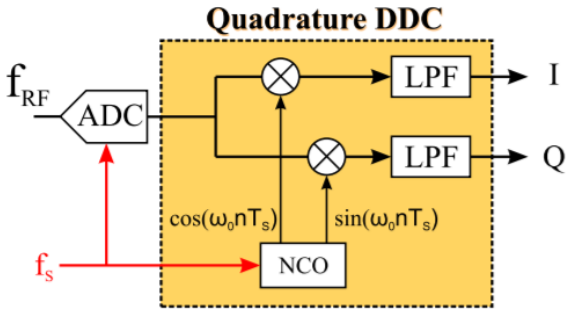
3.3 Reaalisignaalin muuttaminen IQ-muotoon

Kaikki reaalimaailmassa lähetettävät ja vastaanotettavat signaalit ovat aina reaalisia ja IQ-signaali on ainoastaan reaalisen signaalin teoreettinen esitysmuoto. Useat nykyaikaiset ohjelmistomääritellyt ja digitaaliset radiot hyödyntävät IQ-esitysmuotoa. (Lichtman, n.d.) IQ-esitysmuotoa hyödynnetään yksityiskohtaisessa signaalianalyysissä, koska se mahdollistaa tarkan amplitudi-, taajuus- ja vaihetiedon taltioinnin signaalista (ITU-R, 2017, s. 3–4). Formaatin käyttökohteita ovat tietoliikennetekniikka, tutkajärjestelmät, antennin keilanmuodostussovellukset sekä aikaeromittaukseen perustuvat suuntimojärjestelmät (Lyons, 2000). Lisäksi IQ-esitysmuotoa voidaan hyödyntää esimerkiksi uusien yleislähetysstandardien luomisessa (Schipper, 2013, s. 4).

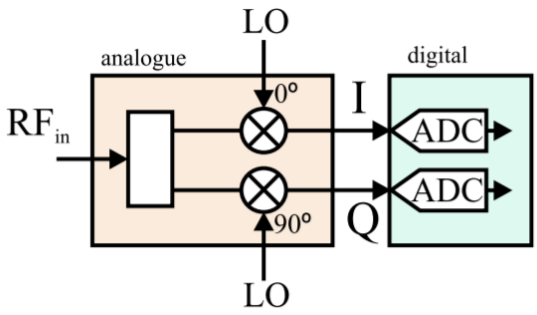
Reaalisignaalin muuttaminen IQ-muotoon tehdään Hilbertin integraalimuunnoksella. Hilbertin muunnos lisää positiiviseen signaaliin -90-asteen vaihe-erolla olevan komponentin ja negatiiviseen signaaliin +90-asteen vaihe-erolla olevan komponentin. (Valkama, 2001, s. 10) Sivulla 20 olevassa kuvassa 9 on esimerkki ohjelmistoradiovastaanottimen IQ-näytteistykseen soveltuvan digitaalisen alassekoittimen (DDC) rakenteesta. Aiemmin esitettyyn vastaanottimeen (kuva 6) verrattuna signaalia ei muuteta sellaisenaan välitaajuudelle. Digitaaliseksi muunnettu analoginen radiotaajuinen signaali sekoitetaan paikallisoskillaattorilla (NCO) ja taajuussekoittimilla kahteen osaan I- ja Q-haaraksi. Tämän jälkeen molemmat komponentit käsitellään alipäästösuodattimella (LPF). Sivulla 20 olevassa

kuvassa 10 on esimerkki vastaanottimesta, jossa analoginen radiotaajuinen signaali jaetaan I- ja Q-komponentteihin jo ennen A/D-muunnosta ja digitaalista signaalinkäsittelyä. (Schilcher, 2008, s. 254, 259)

Kuva 9. IQ-näytteenotto DDC:llä (Schilcher, 2008)



Kuva 10. Signaalin jakaminen I- ja Q-komponentteihin ennen A/D-muunnosta (Schilcher, 2008)



3.4 IQ-datan tallentaminen

Signaalidataa tallennetaan muun muassa sen manuaalista analysointia varten (Lichtman, n.d.). IQ-tiedosto tai IQ-aaltomuototiedosto sisältää radiotaajuisesta signaalista otettujen digitaalisten näytteiden arvoja. IQ-datan lähteenä voi olla esimerkiksi tallenne vastaanotetusta reaalimaailman signaalista. (Schipper, 2013, s. 4) IQ-signaalinäytteet tallennetaan usein binääritiedostoina. Binäärimuodossa tallentaminen säästä tallennustilaa, erityisesti suuria näytteenottotaajuuksia käytettäessä. Signaali tallennetaan tällöin liukulukuja sisältävänä rivinä, josta ne voidaan uudelleen järjestää kompleksiluvuiksi. Esimerkiksi binääritiedostoon tallennettu lukusarja IQIQ on kompleksilukuina esitettynä [I + Qi, I + Qi]. (Lichtman, n.d.)

Kansainvälisen televiestintäliiton radioviestintäsektori ITU-R:n (ITU-R, 2018, s. 1) mukaan IQ-datan tallennusmuodot vaihtelevat laitteiden ja valmistajien välillä. IQ-datalle ei ole toistaiseksi määritelty standardia (Schipper, 2013). ITU-R (2018, s. 1) suosittelee, että laitteet ja sovellukset, joilla käsitellään IQ-dataa, ovat yhteensopivia ITU-R:n määrittelemän ja Hierarchical Data Format (HDF5) -muotoon perustuvan tiedostomuodon kanssa. Standardien puutteesta johtuen tähän työhön on koottu muutamien laitteiden ja ohjelmistojen toimintoja IQ-datan tallentamiseen liittyen.

Yhdysvaltalaisen Nuandin BladeRF-ohjelmistoradiolla voidaan tallentaa vastaanotettua signaalia binäärimuodossa tai CSV-muodossa. Molemmissa tapauksissa I- ja Q-näytteet tallennetaan 16-bittisinä näytteinä. Vastaanottimelle määritetään näytteenottotaajuus sekä näytteiden kokonaismäärään vaikuttava kerroin. (Ghilduta, 2019) Pythonin NumPy -kirjastoa käytettäessä digitaalisessa signaalinkäsittelyssä hyödynnetään yleensä 32-bittisiä näytteitä np.complex64-muodossa. IQ-muotoisissa signaalinäytteissä binääritiedostojen tallennusmuoto on iq-päätteinen. (Licthman, n.d.)

Avoimen lähdekoodin ohjelmistoradioiden toteutukseen tarkoitettua kehitysalusta GNU Radiota käytettäessä IQ-datan tallentamiseen voidaan luoda oma File Sink -lohko, jonne data tallennetaan kompleksisessa binäärimuodossa. Käytettävä näytteenottotaajuus tallennetaan omaan lohkoonsa. GNU Radion kautta tallennettuja IQ-muotoisia tiedostoja voidaan käsitellä muun muassa Matlab- tai Octave-ohjelmistoilla. (Northern Arizona University, 2017, s. 5–6) Saksalainen RF-tekniikkaa valmistava Rohde & Schwarz käyttää laitteissaan IQ-datan tallentamiseen pääsääntöisesti binääritiedostoja, jonne IQ-data tallennetaan 8–64-bittisinä näytteinä. Metatiedon tallentamiseen käytetään xml-tiedostoa. Yksi IQ-datatallenne sisältää molemmat tiedostot, jotka pakataan iq.tar-muotoisiksi tiedostoksi. (Rohde & Schwarz, 2015, s. 3)

Yhdysvaltalaisen testaus- ja mittalaitteiden valmistaja Tektronixin RSA3408A-spektrianalysaattorilla voidaan tallentaa digitaalista IQ-raakadataa dat-tiedostomuotoon. I- ja Q-näytteet tallennetaan 16-bittisinä näytteinä noudattaen little endian -tavujärjestystä. Esimerkiksi tallennettaessa 36 Mhz taajuuskaistaa 51,2 Msps näytteenottonopeudella tulee 10 sekunnin IQ-datatallenteesta 1,9 Gt kokoinen. Yhden minuutin IQ-datatallenne vastaavilla

parametreilla on 11,44 Gt kokoinen. (Tektronix, 2006, s. 2–3) Combitech Oy:n tutkimus- ja tuotekehityspäällikkö Timo Mustonen (2021) esittelee artikkelissaan Ettus Research USRP X300 -ohjelmistoradion ja TwinRX-vastaanotinkortin käyttöä IQ-datan tallentamiseen. IQ-muodossa voidaan tallentaa maksimissaan 80 Mhz taajuuskaistaa 100 Msps näytteenottonopeudella 32-bittisinä näytteinä. I- ja Q-näyte ovat kumpikin 16-bittisiä. (Mustonen, 2021)

4 Tapaustutkimus: Tietokantojen vertailu

Tapaustutkimuksen tavoitteena on saada mahdollisimman monipuolinen kuva yhdestä tai useammasta ennalta määritellystä tapauksesta. Tapaustutkimus ei pyri yleistämään, vaan kuvaamaan ja havainnollistamaan rajattua tutkimuskohdetta. Tapaustutkimuksen kohteena voi olla yksi tai useampi tapaus. Mikäli tapaustutkimuksen kohteena on useampi tapaus, tehdään niistä yleensä vertailevaa analyysia. Tällöin tutkimuksen kohteeksi kannattaa valita toisistaan eroavia kohteita. (Vuori, n.d.) Opinnäytetyön tapaustutkimuksen kohteena oli kymmenen avoimen lähdekoodin tietokantaa. Tapaustutkimuksena toteutetun tietokantojen vertailun avulla pyrittiin vastamaan asetettuihin tutkimuskysymyksiin:

- Miten yleisimmät avoimen lähdekoodin tietokannat soveltuvat IQ-datan tallentamiseen?
- Mitkä tutkimuksen kohteena olevista tietokannoista soveltuvat IQ-datan tallentamiseen?
- Mikä on tutkimuksen pohjalta käyttöön soveltuvin tietokanta?

4.1 Tietokannan käyttötapaus

Ennen tietokantojen vertailua kuvattiin, mitä tietokantaan halutaan tallentaa ja miten tietokantaa halutaan käyttää. Tätä varten määritettiin tehtäväselostus tietokannan tarkoituksesta sekä tehtävän tavoitteet. Tehtävän tavoitteiden avulla kuvattiin käyttäjän toimenpiteet, joita hänen on tarkoitus suorittaa tietokannan ja siellä olevien tietojen avulla. Lisäksi tietokannalle kuvattiin käyttötapaus, joka on esitetty taulukossa 1 sivulla 24.

Tilaajalla oli tarve pysyvälle tietovarastolle IQ-muotoista signaalidataa ja siihen liittyvää metadataa varten. Datatallenteiden tiedostomuoto ja rakenne ei ole toistaiseksi vakioitu. Yksittäinen tallenne on kooltaan keskimäärin 200–500 Mt. Tietokannassa olevaa tietoa käytetään, vertaillaan ja analysoidaan muissa sovelluksissa. Tietokantaan tuodaan tallenteita pääosin massana, noin 100–1000 tallenteen kokoelmina. Tietokannassa olevan datan määrä ei ole vakio.

Tehtäväselostuksen pohjalta määriteltiin tehtävän tavoitteet:

- Käyttäjä haluaa tallentaa tietokantaan IQ-datatiedostoja binäärimuodossa tai muussa tiedostomuodossa.
- Käyttäjä haluaa siirtää ja tallentaa IQ-datatiedostoja massana, 100–1000 tallennetta kerrallaan.
- Käyttäjä haluaa tallentaa tietokantaan IQ-datatiedostojen metadataa esimerkiksi tekstinä, numeroina tai csv-tiedostona.
- Käyttäjä haluaa hyödyntää tietokannan kanssa ulkoista ohjelmaa tai skriptiä, joka lukee IQ-datatiedoston ja tallentaa siitä metadataa tietokantaan.
- Käyttäjä haluaa hyödyntää tietokantaan tallennettua dataa muissa sovelluksissa.

Tehtävän tavoitteissa mainittu ulkoinen ohjelma tai skripti metadatan lukemiseksi tallenteesta rajattiin tämän opinnäytetyön ulkopuolelle ja sitä ei toteutettu. Ratkaisun toteutus ja siihen liittyvä taustaselvitystyö olisivat jo sellaisenaan olleet aiheena niin laajat, että työstä olisi pitänyt jättää varsinaisiin tutkimuskysymyksiin liittyvät osa-alueet pois. Tämän vuoksi kaikkia osa-alueita ei nähty mahdolliseksi toteuttaa yhdessä opinnäytetyössä.

Taulukko 1. Tietokannan käyttötapaus.

Tavoite	Tietokanta, jonne voi tallentaa IQ-datatallenteita sekä niiden metadataa.
Esiehdot	Käyttäjällä on hallussaan valmiita IQ-datatallenteita.
Onnistunut lopputulos	Käyttäjä pystyy siirtämään ja tallentamaan tietokantaan massana IQ-tallenteita, joista ulkoinen järjestelmä tai skripti populoi tarvittavan metadatan tietokantaan. Käyttäjä voi hakea tietokannasta erilaisilla hakuparametreilla haluamiaan tallenteita ja ladata niitä käytettäväksi muissa sovelluksissa.
Kuvaus käyttötapauksesta	1. Ulkoinen skripti tai järjestelmä lukee IQ-tallenteesta halutun metadatan. 2. Käyttäjä vie massana IQ-datatallenteet ja niiden metadatan tietokantaan. 3. Käyttäjä hakee tietokannasta halutuilla parametreilla IQ-tallenteita. 4. Käyttäjä kopioi tai siirtää tietokannasta hakemansa tallenteet muissa sovelluksissa tai järjestelmissä käsiteltäviksi ja käytettäviksi.

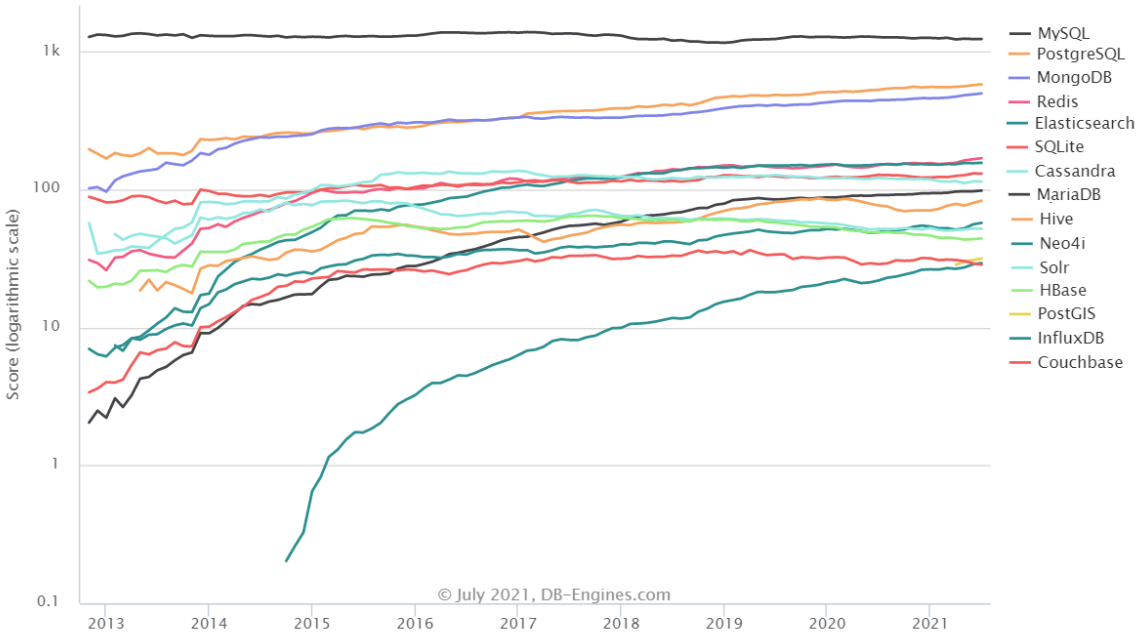
4.2 Tietokantojen valinta tapaustutkimukseen

Opinnäytetyön tapaustutkimukseen valittiin kymmenen olemassa olevaa avoimen lähdekoodin lisenssillä jaettavaa tietokantaa. Tietokantojen valinta perustui DB-Engines sivuston kuukausittain päivittyvään tietokantojen vertailuluetteloon, jossa tietokantoja pisteytetään muun muassa Google Trends-sivustolta saatavien hakumäärien, Twitter-mainintojen sekä erilaisten tekniikkaan liittyvien verkkosivujen keskusteluaiheiden perusteella (DB-Engines, 2021d). Lisäksi käytettiin Shanhong Liun (2021) Statista-sivustolla julkaistua raporttia suosituimmista tietokannoista vuonna 2021. Sivulla 25 olevassa taulukossa 2 on DB-Engines-sivuston (2021a) mukaan 15 suosituinta avoimen lähdekoodin tietokantaa heinäkuussa 2021. Kuva 11 esittää tilaston kyseisten tietokantojen suosion kehityksestä vuosina 2013–2021 (DB-Engines, 2021b).

Taulukko 2. Suosituimmat avoimen lähdekoodin tietokannat heinäkuussa 2021 (DB-Engines, 2021a)

Tietokanta	Sijoitus	Tietokantamalli
MySQL	1.	relaatio, dokumentti, spatiaalinen
PostgreSQL	2.	relaatiotieto, dokumentti, spatiaalinen
MongoDB	3.	dokumentti, spatiaalinen, hakukone
Redis	4.	avain-arvo, dokumentti, verkko, spatiaalinen, aikasarja
Elasticsearch	5.	hakukone, dokumentti, spatiaalinen
SQLite	6.	relaatio
Cassandra	7.	laaja sarake
MariaDB	8.	relaatiotieto, dokumentti, spatiaalinen
Hive	9.	relaatio
Neo4j	10.	verkko
Solr	11.	hakukone, spatiaalinen
HBase	12.	laaja sarake
PostGIS	13.	spatiaalinen, relaatio
InfluxDB	14.	aikasarja, spatiaalinen
Couchbase	15.	dokumentti, spatiaalinen, avain-arvo

Kuva 11. Avoimen lähdekoodin tietokannat 2013–2021 (DB-Engines, 2021b)



Tapaustarkasteluun valittavien tietokantojen määrä rajattiin kymmeneen. Vertailuun valittiin erityyppisiä tietomalleja hyödyntäviä avoimen lähdekoodin tietokantoja. Koska mahdollisen käyttöönotettavan tietokannan haluttiin olevan mahdollisimman pitkäikäinen ratkaisu, valittiin vertailuun suosittuja ja useita vuosia käytössä olleita ratkaisuja. Relaatiomallia käyttäviä tietokantoja 15:sta suosituimmasta oli viisi: MySQL, PostgreSQL, MariaDB, SQLite ja Hive. Vertailusta päädyttiin jättämään pois SQLite ja Hive, koska muilla mainituista viidestä tietokannasta pystytään hyödyntämään relaatiomallin lisäksi myös muita tietokantamalleja. 15 suosituimman tietokannan ulkopuolelta vertailuun valittiin digitaalisen sisällön, kuten videon ja kuvan tallentamiseen suunnattu Jackrabbit-tietokanta. Lopulliseen vertailuun valittiin seuraavat tietokannat:

- Cassandra
- HBase
- InfluxDB
- Jackrabbit
- MariaDB
- MongoDB
- MySQL
- Neo4j
- PostgreSQL
- Redis

4.3 Aineiston keräys ja analyysi

Tutkimukseen valittuihin tietokannan hallintajärjestelmiin perehdyttiin pääasiassa virallisen dokumentaation avulla. Lisäksi hyödynnettiin muuta julkista materiaalia, kuten tietokannan hallintajärjestelmistä tehtyjä artikkeleita, tutoriaaleja ja käyttäjäarvioita. Tietokannoista pyrittiin selvittämään niiden oleellisia ominaisuuksia ja toiminnallisuuksia sekä tyypillisiä käyttötapoja ja -kohteita. Erityisenä tarkastelun kohteena oli, miten tietokantaan pystyy tallentamaan kokonaisia tiedostoja, kuten IQ-muotoisia signaalitallenteita.

Kerätyn aineiston avulla tutkimuksen kohteena olevia tietokannan hallintajärjestelmiä vertailtiin sekä toisiinsa että opinnäytetyössä määriteltyyn käyttötapaukseen. Aineiston

analyysissä vertailun avulla tarkastellaan määriteltyjä ominaisuuksia kahdesta tai useammasta tapauksesta (Routio, n.d.). Tarkasteltaviksi ominaisuuksiksi on hyvä määrittää sellaisia ominaisuuksia, jotka eivät ole samanlaisia kaikissa tapauksissa. Ominaisuuksia voidaan lisätä tai poistaa analyysin edetessä. (Routio, n.d.)

Koska aineiston määrä oli suuri, ei jokaisen tietokannan hallintajärjestelmän yksityiskohtainen tarkastelu eikä testaaminen ollut mahdollista. Tämän vuoksi aineistosta tarkasteltavat ominaisuudet rajattiin seuraaviin kokonaisuuksiin:

- Tietokannan tietomalli, kyselykieli ja ohjelmointikieli
- Tietokannan julkaisuvuosi ja käytettävä lisenssi
- Esimerkkejä käyttötapauksista ja käyttäjistä
- Datan tuonti tietokantaan massana
- BLOB-tietotyypin maksimikoko
- Yhteensopivuus Matlab-ohjelmiston sekä muiden ulkoisten järjestelmien ja sovellusten kanssa (API-rajapinnat)
- Soveltuvuus IQ-datan tallentamiseen
- Tietokannasta haku ja tiedon suodattaminen
- Skaalautuvuus: valmiin tietokannan laajennettavuus ja muokkautuvuus
- Yhteensopivat alustat
- Yhteensopivat CLI- tai GUI-käyttöliittymät

4.4 Tulokset

Vertailussa perehdyttiin kolmeen relaatiotietokannan hallintajärjestelmään ja seitsemään NoSQL-tietokannan hallintajärjestelmään. Kaikille vertailluille tietokannoille yhteistä on se, että niitä pystytään käyttämään vähintäänkin Linux- ja Windows-alustoilla. Suurinta osaa pystytään käyttämään myös Docker-konttiratkaisussa. Tyypillisiä käyttökohteita tietokannoille ovat IoT-järjestelmät, sosiaalisen median sovellukset, soittolistat, pilvipalvelut, suosittelusovellukset ja maksuvälinepetosten ehkäisyyn liittyvät ratkaisut. Vertailutulokset on koottu taulukkoon liitteessä 2.

Lähes kaikki vertaillut tietokannan hallintajärjestelmät tukevat vähintään yhtä API-rajapintaratkaisua ja mahdollistavat datan tuonnin massana tietokantaan ainakin jollain tavalla. Relaatiotietokannoissa on hiukan monipuolisemmin vaihtoehtoja datan massatuontiin muihin tietokantoihin verrattuna, joskin lähes kaikkiin vertailtuihin tietokantoihin on mahdollistaa yhdistää erilaisilla lisätyökaluilla ja -kirjastoilla laadittuja omia sovelluksia datan tuontiin ja transaktioihin.

IQ-datan tallentamiseen käytettävän tietokantaratkaisun on tarpeen olla helposti laajennettavissa ja skaalattavissa. On mahdollista, että tietokantaan tallennetaan eri tiedostomuotoisia tallenteita. Lisäksi tallenteista saatavan metadatan rakenne ja sisältö voi vaihdella tallenteen alkuperän mukaan. Tämän vuoksi tietokantaratkaisu, joka vaatii tarkkaan määritellyn ja pysyvän skeeman, ei välttämättä sovellu tähän tarpeeseen. Koska relaatiotietokannan hallintajärjestelmät pohjautuvat tyypillisesti ennalta määriteltyyn skeemaan, ei relaatiomalliin perustuvia ratkaisuja pidetty tästä syystä potentiaalisena ratkaisuna käyttötarpeeseen.

IQ-datan tallentamisen näkökulmasta suurimmat rajoitteet relaatiotietokannan käytössä liittyvät strukturoimattoman datan tallentamiseen. Relaatiotietokantoja ei ole suunniteltu tilanteisiin, joissa käsiteltävänä on strukturoimatonta dataa. Lisäksi relaatiotietokannat vaativat tarkan skeeman määrittelyn etukäteen ja sen muuttaminen valmiiseen tietokantaan on haastavaa. Parhaiten IQ-datan tallentamiseen soveltuvat NoSQL-tietokannan hallintajärjestelmät, joissa on löyhästi määritelty skeema tai jotka ovat osittain jopa täysin skeemattomia. Tällöin valmiissa tietokannassa on mahdollista päivittää skeemaa joustavasti ja tallentaa esimerkiksi alkuperäisestä skeemasta poikkeavaa dataa. Tällaiseen käyttöön soveltuvia tietokannan hallintajärjestelmiä ovat esimerkiksi MongoDB sekä Jackrabbit.

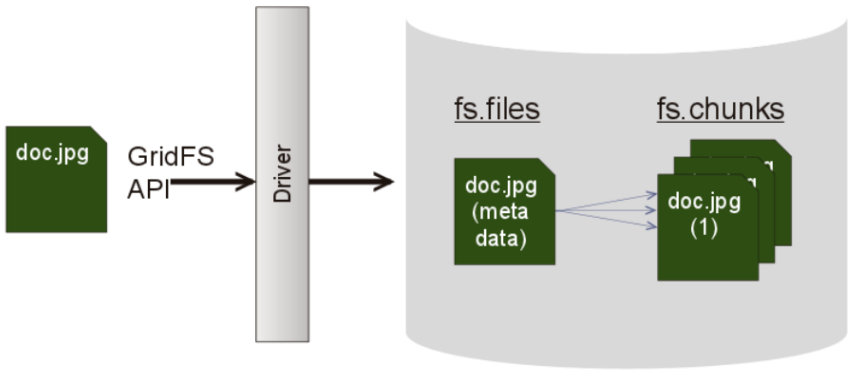
Tulosten perusteella voidaan kuitenkin arvioida, että hiukan erilaisella käyttöperiaatteella myös relaatiotietokannan hyödyntäminen voisi olla toimiva ratkaisu. Käyttötapauksena voisi tällöin olla tietokannan hyödyntäminen ainoastaan metadatan ja signaaliin liittyvien parametrien tallentamiseen. Varsinaiset IQ-datatallenteet tallennettaisi tiedostojärjestelmään palvelimelle ja tietokannassa olevaan metadataan lisättäisi viittaus tiedoston sijaintiin. Tällöinkin tietokantaratkaisu vaatisi rakenteen määrittelyn etukäteen. Vaihtoehtoisesti tällaisessa käyttötapauksessa voisi käyttää relaatiotietokannan sijasta myös

sellaisia NoSQL-tietokantoja, jotka eivät suoraan sovellu strukturoimattoman datan tai suurten tiedostojen varastointiin. Tällaisia vaihtoehtoja ovat esimerkiksi Cassandra ja Redis.

Eroja tietokantojen välillä havaittiin BLOB-tietotyypin käytössä ja sen maksimikoossa. Osassa tietokannoista, kuten Neo4j, ei BLOB:n käyttöä suositeltu lainkaan. Relaatiotietokantojen BLOB-tietotyypin maksimikoko vaihteli 1 Gt ja 4 Gt välillä. NoSQL-tietokantojen BLOB-tietotyypin maksimikoko vaihteli 512 Mt ja 2 Gt välillä. InfluxDB- ja HBase-tietokantojen dokumentaatio ei ottanut suoraan kantaa BLOB-tietotyypin käyttöön eikä maksimikokoon. MongoDB-dokumenttitietokannassa on erillinen GridFS-toiminto suurten tiedostojen hallintaan ja sen arvioitiin olevan käyttökelpoisempi vaihtoehto IQ-datatiedostojen varastointiin BLOB-tietotyyppiin verrattuna. Arviossa hyödynnettiin Searsin, van Ingenin ja Grayn (2006, s. 2) Microsoft Researchille ja Kalifornian yliopistolle tekemää tutkimusta, jonka perusteella BLOB-tietotyyppi soveltuu pääsääntöisesti pienille, alle 256 kilotavun kokoisille tiedostoille.

MongoDB GridFS yhdistää tietokannan ja tiedostojärjestelmän toiminnallisuuksia. GridFS jakaa tietokantaan tallennettavat tiedostot pienempiin dokumenttimuotoisiin osiin. Alkuperäinen binääritiedosto jaetaan 255 kt kokoisiksi chunks-dokumenteiksi ja yhdeksi files-dokumentiksi. Files-dokumenttiin tallennetaan alkuperäisen tiedoston metadata. (Runkel, 2014) Alla olevassa kuvassa 12 on esitetty GridFS-toiminnallisuuden rakenne. Dokumentaation ja muiden käytettyjen lähteiden perusteella MongoDB:n katsottiin olevan potentiaalisin vertailluista kymmenestä vaihtoehdoista IQ-datan tallentamiseen.

Kuva 12. MongoDB GridFS -rakenne (Runkel, 2014)



4.5 Tulosten luotettavuuden arviointi

Tietokannan hallintajärjestelmien vertailu tapaustutkimuksena soveltuu hyvin aiheeseen perehtymiseksi ja vaihtoehtojen rajaamiseksi. Ilman käytännön testaamista ei kuitenkaan voida todentaa aukottomasti, soveltuuko jokin ratkaisu haluttuun käyttötarkoitukseen. Vertailevalla tutkimuksella on kuitenkin mahdollista rajata suuresta joukosta mahdollisia vaihtoehtoja tarkempaa tutkimusta tai testaamista varten.

Aiheeseen liittyvää aineistoa on paljon ja oleellisen tiedon löytäminen on haasteellista. On myös riski, että pelkkään kirjalliseen dokumentaatioon perehtymällä jää jotain oleellista havaitsematta. Tietokannan hallintajärjestelmien virallisen dokumentaation laajuus vaihtelee, jolloin pelkkien virallisten dokumentaatioiden avulla on haastava analysoida tasapuolisesti kunkin tietokannan soveltuvuutta. Käyttäjäkokemuksia vertaillessa on muistettava, että kokemukset ovat aina subjektiivisia ja eivät sellaisenaan välttämättä suoraan rinnastettavissa omaan käyttötarpeeseen. Käyttäjäkokemukset ja tutoriaalit saattavat olla myös vanhoihin ohjelmistoversioihin perustuvia.

Tutkimus tehtiin tietokannan hallintajärjestelmien uusimmista versioista ja niiden dokumentaatiosta elokuussa 2021. Avoimen lähdekoodin ohjelmistoja kehitetään ja päivitetään jatkuvasti, joten on mahdollista, että uudempiin ohjelmistoversioihin on lisätty uusia ominaisuuksia tai niistä on poistettu jotain. Vertailluista tietokannan hallintajärjestelmistä kuuteen oli julkaistu uusi ohjelmistoversio lokakuussa 2021, noin kaksi kuukautta vertailun toteuttamisen jälkeen. Vertailun tulokset eivät tämän vuoksi ole täysin rinnastettavissa uudempiin ohjelmistoversioihin.

5 Tietokannan testaus

Soveltuvimmaksi arvioidulla MongoDB -tietokannan hallintajärjestelmällä toteutettiin käytännön testi pienellä datamäärällä. Tavoitteena oli syventää arviota tietokannan soveltuvuudesta ja testata tietokannan hallintajärjestelmän toiminnallisuuksia.

5.1 Testijärjestelyt

Testitietokantaan tallennettavan datan tuottamisessa sovellettiin vuonna 2016 Hämeen Ammattikorkeakoulussa julkaistua Lehtosen (2016) opinnäytetyötä ”Ohjelmistovastaanottimen soveltaminen radiotekniikan opetukseen”. Radiolupavapaalla PMR-radiolla lähetettiin taajuusmoduloitua audiota 12,5 kHz kaistanleveydellä kahdella PMR-käyttöön tarkoitetulla kanavalla. Radioteitse lähetettävä audio tuotettiin tietokoneen äänikortin kautta lähettimen mikrofonille avoimen lähdekoodin Minimodem-ohjelmalla. Ohjelmaa voidaan käyttää FSK-modeemina (Mostafa, n.d.).

Lähetettävää PMR-signaalia tallennettiin BladeRF -ohjelmistovastaanottimella. Lähetin ja vastaanotin olivat samassa tilassa, alle 10 metrin päässä toisistaan. Vastaanottimen keskitaajuudeksi asetettiin kulloinkin lähettimessä käytettävä taajuus ja kaistanleveydeksi 2 Mhz. Näytteenottotaajuus oli 1 Mhz. IQ-raakadata tallennettiin sekä binääri- että csv-muodossa. Tallenteista rajattiin kapeampikaistaisia binäärimuotoisia tallenteita avoimen lähdekoodin Universal Radio Hacker (URH) -ohjelmalla. Ohjelma on suunniteltu saksalaisessa Stralsundin ammattikorkeakoulussa langattomien tiedonsiirtoprotokollien analysointiin ja simulointiin esimerkiksi arvioitaessa IoT-laitteiden tietoturvaa. (Pohl & Noack, 2018) Alkuperäiset ja niistä URH-ohjelmalla muokatut tallenteet olivat tiedostokooltaan yhteensä noin 3,5 Gt. Yksittäiset tallenteet olivat kooltaan 40–163 Mt ja niitä oli yhteensä 30 kappaletta.

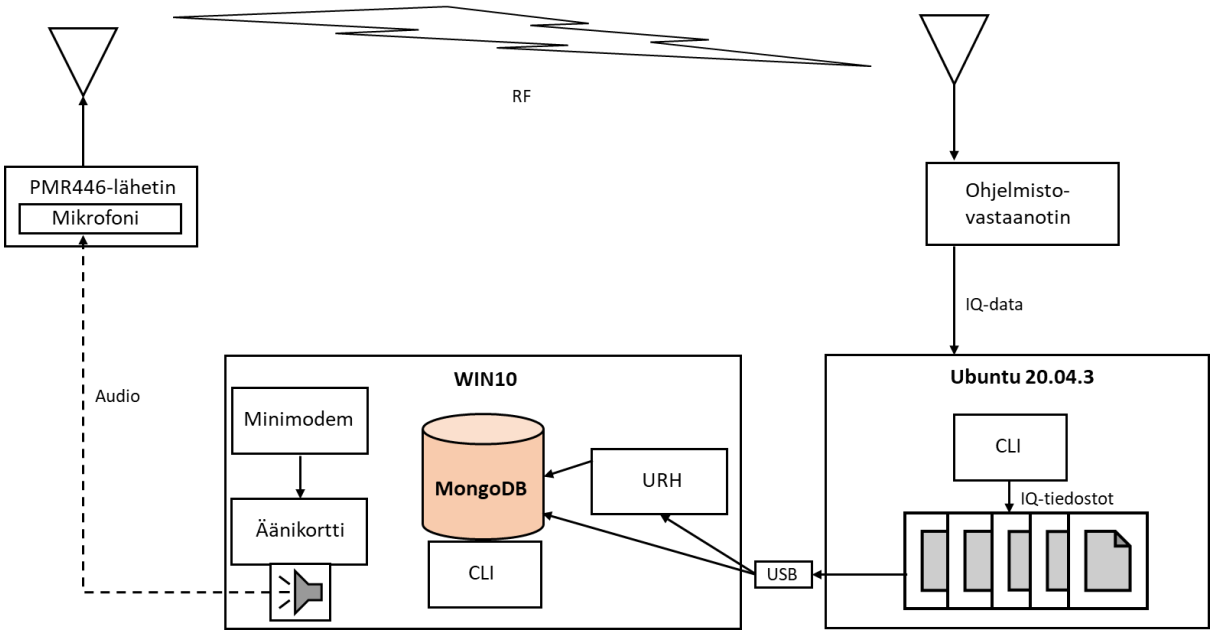
IQ-tallenteiden metadata tallennettiin manuaalisesti JSON-muotoisiksi tiedostoiksi. Metadatana käytettiin seuraavia parametreja:

- keskitaajuus (middlefrequencyHz)
- kaistanleveys (bwHz)
- näytteenottotaajuus (samplerateHz)
- tallenteen aloitusaika (recording_starttime)
- tallenteen kesto (recording_length_seconds)

Testauksessa käytettiin Lenovo IdeaPad 330 -kannettavaa työasemaa, jossa on 64-bittinen Windows 10 -käyttöjärjestelmä, Intel® Core™ i5-8250U-suoritin, 256 Gt SSD-muisti ja 6 Gt

RAM-muisti. Lisäksi käytettiin Asuksen kannettavaa työasemaa, johon on asennettu Linuxin 64-bittinen Ubuntu 20.04.3 -käyttöjärjestelmä. Testissä käytetty MongoDB- tietokannan hallintajärjestelmä ja sen käyttöliittymä asennettiin Lenovo-työasemalle WIN10-käyttöjärjestelmään. Testijärjestelyt on esitetty alla olevassa kuvassa 13.

Kuva 13. Testijärjestelyt



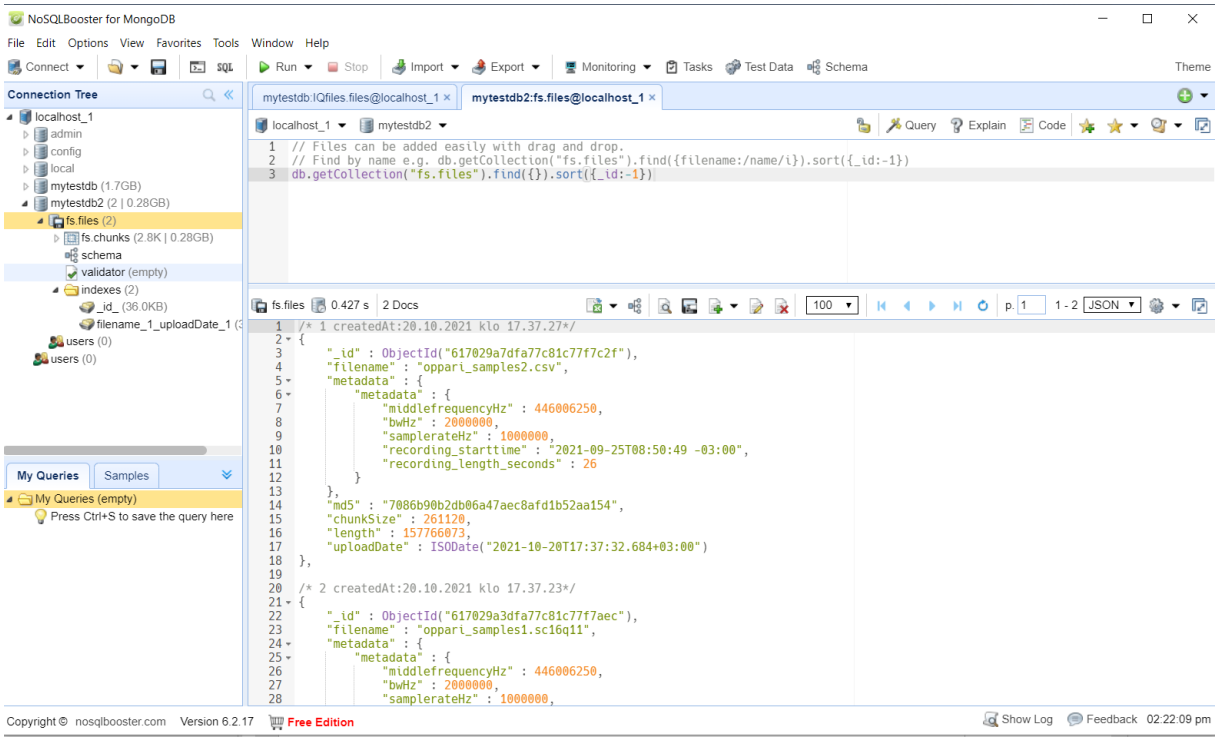
5.2 Tietokannan hallinta- ja käyttöliittymä

Tietokannan käyttöliittymänä testattiin NoSQL Booster for MongoDB -sovelluksen versiota 7.0.2. Sovellus on helppokäyttöinen ja se mahdollistaa erityisesti GridFS-toiminnon hyödyntämisen helposti. Sovelluksessa on valikot GridFS files- ja chunks-kokoelmien luomiseksi sekä tiedostojen siirtämiseksi tietokantaan. Ohjelmasta on saatavilla ilmainen lisenssi sekä kaksi maksullista lisenssiä: personal ja commercial. Testissä ohjelmaa käytettiin ilmaislisenssillä. Käyttöliittymä on helppokäyttöinen ja siinä on valmiita toimintoja esimerkiksi hakuihin ja tallennettujen tiedostojen lataamiseen tietokannasta. Sovelluksessa on myös työkalu MongoDB:n omalla MQL-kyselykielellä luotujen kyselyiden muuttamiseksi kahdeksalle eri ohjelmointikielelle, kuten Python, Java, C# ja Ruby.

Suurimmat rajoitteet ilmaisessa lisenssissä ovat mongoimport-massatoiminnon ja virheenjäljitystoimintojen puuttuminen sekä autentikointiin ja koodien sekä skriptien käsittelyyn liittyvien toimintojen rajoitteet. Tämän vuoksi on mahdollista, että ilmainen

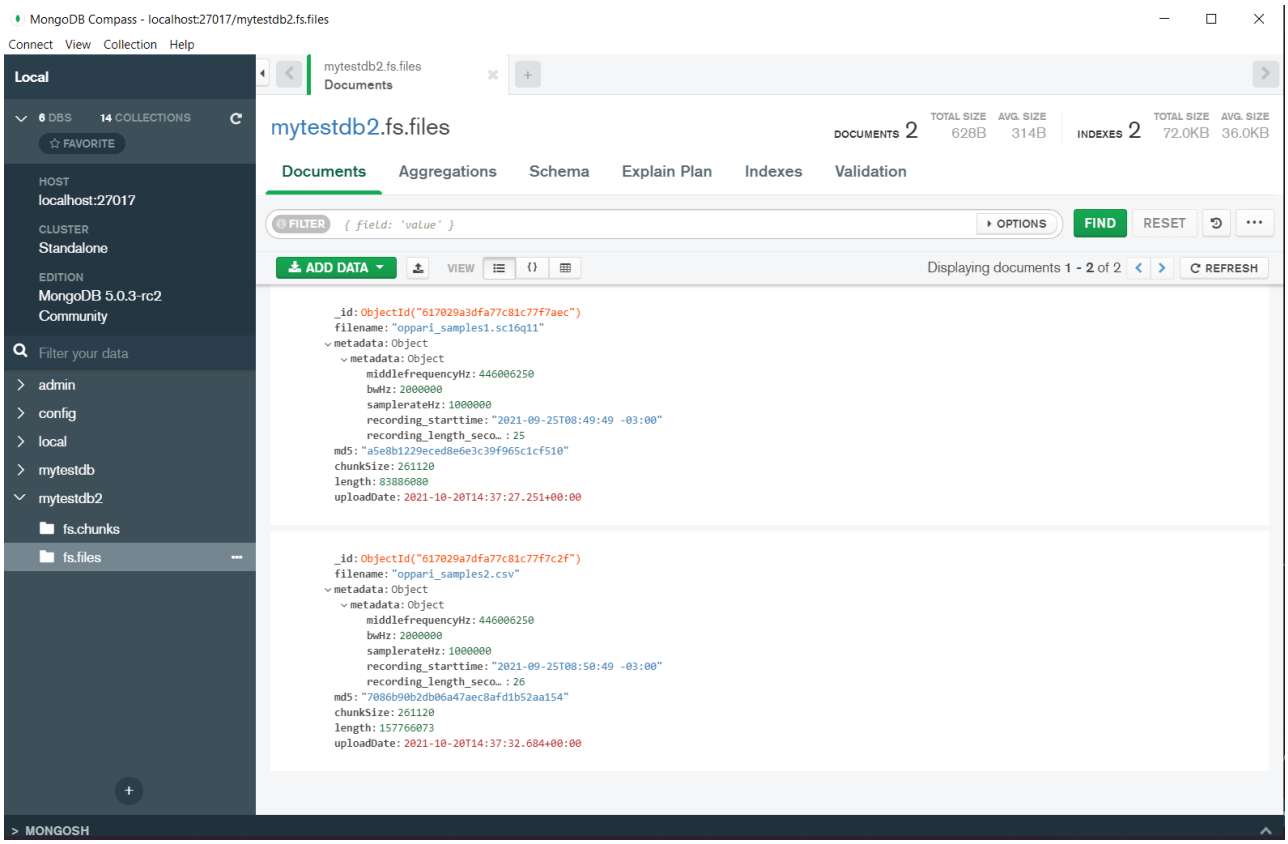
lisenssi on toiminnoiltaan liian rajoitettu opinnäytetyössä määritettyyn käyttötarpeeseen. Alla olevassa kuvassa 14 on NoSQL Booster -käyttöliittymä.

Kuva 14. NoSQL Booster -käyttöliittymä



Lisäksi testattiin MongoDB Compass-käyttöliittymän versiota 1.28.4. Myös Compass on helppokäyttöinen ja mahdollistaa perustoiminnot, kuten uuden tietokannan luomisen ja tiedon lisäämisen. Compass-käyttöliittymässä ei ole suoraan vaihtoehtoa GridFs-kokoelmien luomiseen, eikä tiedostojen lataamiseen, mutta käyttöliittymässä on mahdollista tehdä olemassa olevaan tietokantaan hakuja. Compass on ulkoasultaan NoSQL Boosteria hiukan viimeistellymmän näköinen ja siinä on myös yksinkertaisia analyysityökaluja esimerkiksi hakujen optimoimiseksi. Sivulla 34 olevassa kuvassa 15 on Compass-käyttöliittymä.

Kuva 15. MongoDB Compass -käyttöliittymä

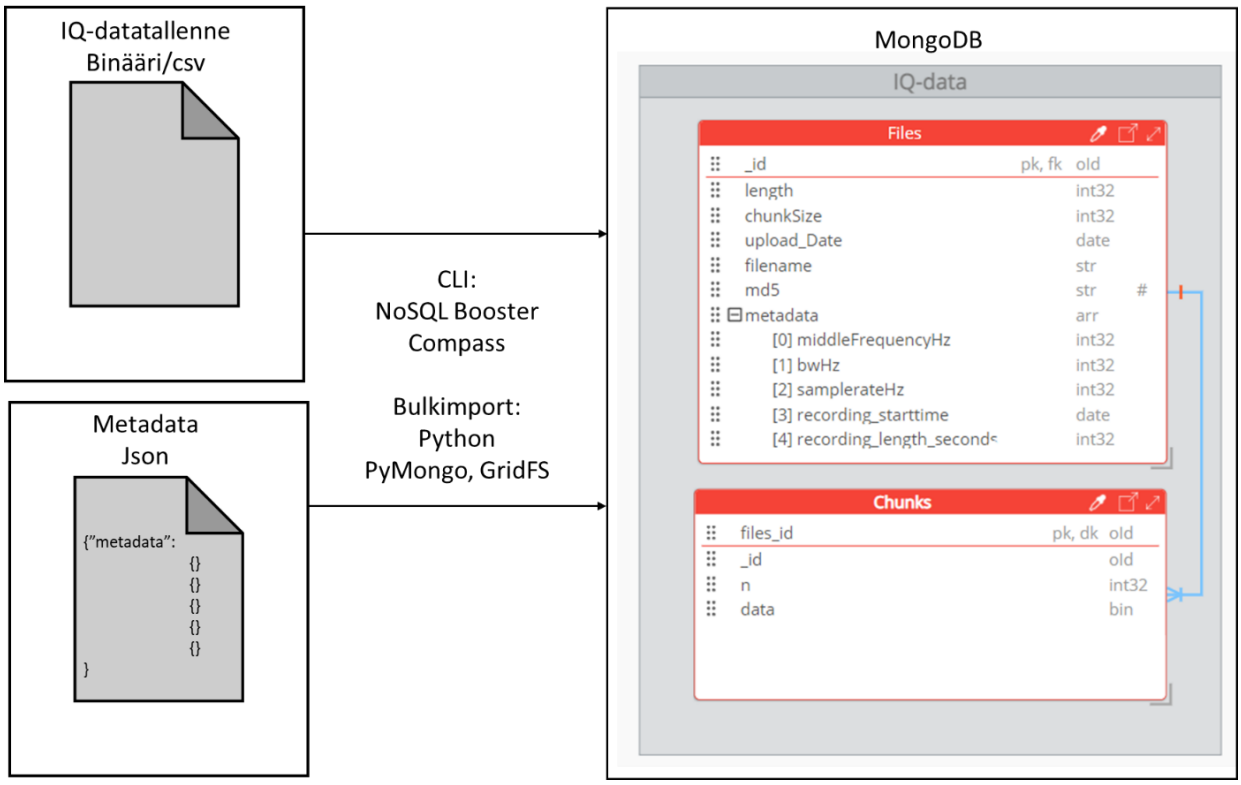


NoSQL Booster mahdollistaa tiedostojen lataamisen massana esimerkiksi suoraan valitusta kansiosta, mutta metadatan lataamiseksi tarvitaan erillinen työkalu. Testiä varten luotiin yksinkertainen PyMongo-kirjastoa käyttävä Python-ohjelma. Ohjelman muodostaa yhteyden tietokantapalvelimeen, hakee ohjelmaan määritellyistä sijainneista IQ-datatiedostot sekä niiden metadata-tiedostot. Ohjelma siirtää IQ-datatiedostot sekä metadatan PyMongo-kirjaston GridFS-työkalulla tietokantaan GridFS files- ja chunks-dokumenteiksi. Ohjelman lähdekoodi on liitteessä 3.

5.3 Tietokannan rakenne

Testauksessa käytettiin MongoDB GridFS-toiminallisuutta. Yhdestä IQ-datatiedostosta tehtiin tietokantaan useita 255 kt kokoisia chunks-dokumentteja ja yksi files-dokumentti. Jokaisella files-dokumentilla on yksilöllinen ID-tunnus (_id), joka vastaa saman IQ-datatallenteen chunks-dokumenttien files_id-tunnusta. Luvussa 5.1 kuvattu manuaalisesti luotu IQ-datatallenteiden metadata tallennettiin kunkin IQ-datatiedoston files-dokumenttiin. Sivulla 35 olevassa kuvassa 16 on testitietokannan rakenne.

Kuva 16. Testitietokannan rakenne



Lisäksi testattiin erikseen myös CSV-tiedostoon kootun IQ-datatiedon purkamista tietokantaan omaan dokumenttiin. Tämä osoittautui hitaaksi ja raskaaksi. Esimerkiksi 82 Mt kokoisesta CSV-tiedostosta saatiin kahdessa minuutissa tuotua tietokantaan vain 25 prosenttia datasta. Testitilanteessa tiedonsiirron hitauteen vaikutti todennäköisesti testissä käytetyn työaseman riittämätön RAM-muistin suorituskyky. Testin perusteella CSV-muotoisen IQ-datatiedoston kompleksisten arvojen purkamisesta MongoDB-dokumenttiin ei nähty saatavan lisäarvoa IQ-datan varastoinnin näkökulmasta.

6 Johtopäätökset

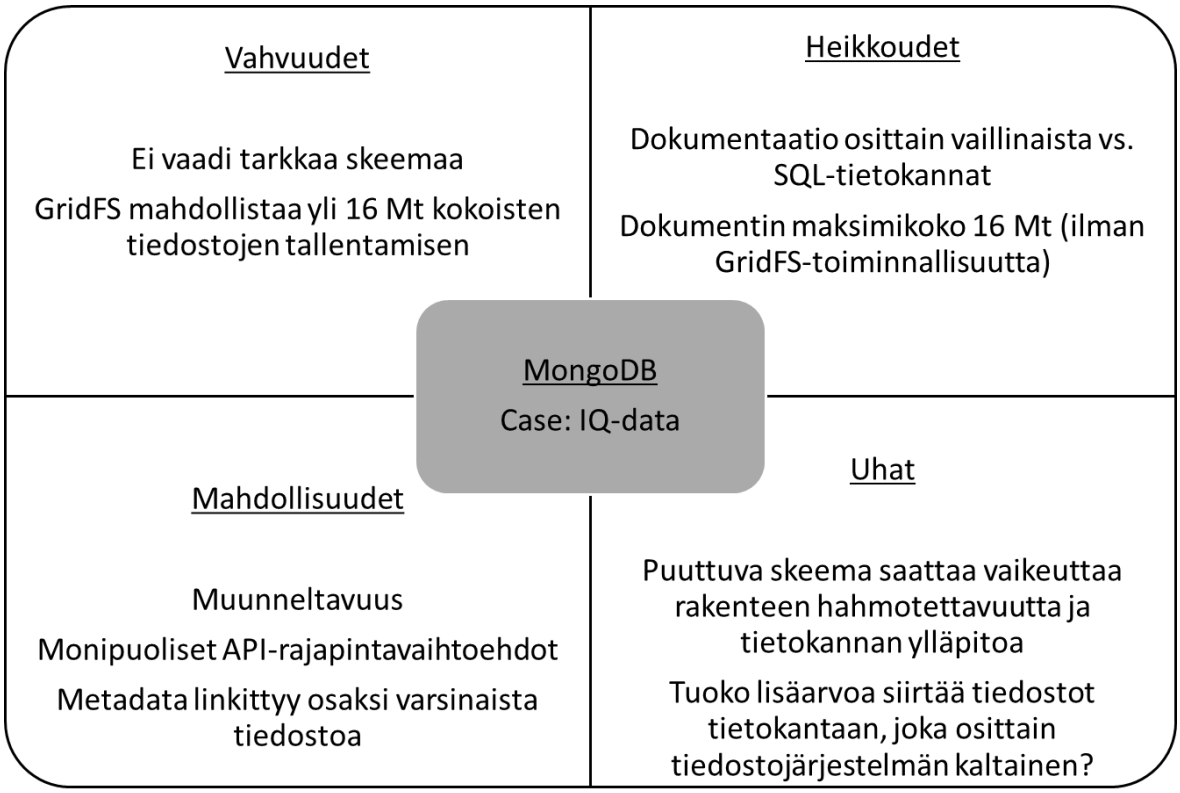
Työssä toteutetun tapaustutkimuksen ja käytännön testauksen perusteella avoimen lähdekoodin tietokantoja voidaan hyödyntää rajoituksin IQ-datan tallentamiseen. Binäärimuotoiset tiedostot suositellaan tallennettavan sellaisenaan omalle palvelimelle tiedostojärjestelmään tai tietokantaan erityisesti tiedostojen tallentamiseen tarkoitettujen toiminnallisuuksien, kuten MongoDB GridFS:n avulla. Tiedostojen tallentamisesta BLOB-tietotyyppinä tietokantaan ei työn perusteella nähty saatavan suurta etua

tiedostojärjestelmään verrattuna. Sen sijaan IQ-datatallenteista saatavan metatiedon tallentamiseen tietokanta vaikuttaa soveltuvalta ratkaisulta tiedonhallinnan näkökulmasta.

Opinnäytetyössä käytettyjen IQ-tallenteiden metadata luotiin manuaalisesti ja metadatana käytettiin vastaanotettuun RF-signaaliin liittyvää parametritietoa, kuten keskitaajuus ja näytteenottotaajuus. Jossain tapauksissa signaalia voi olla tarpeen tarkastella bittitasolla ja tiedot signaalin sisältämästä datasta bitteinä halutaan varastoida. Tässä opinnäytetyössä testattua metadatan varastointiratkaisua on teknisesti mahdollista hyödyntää myös signaalitallenteesta mitatun bittivirran tekstimuotoiseen tallentamiseen.

Dokumenttitietokanta MongoDB on potentiaalinen vaihtoehto IQ-datan tallentamiseen. Sen joustava rakenne mahdollistaa tietokannan käytön erilaisten tiedostomuotojen sekä vaihtelevan metadatan tallentamiseen. Käytännön testauksen perusteella tietokannasta laadittiin SWOT-analyysi, jossa arvioitiin tietokannan vahvuuksia, heikkouksia, mahdollisuuksia ja uhkia IQ-datan näkökulmasta. Analyysin pääkohdat on esitetty alla olevassa kuvassa 17.

Kuva 17. MongoDB-tietokannan SWOT-analyysi



Opinnäytetyön aihe osoittautui laajaksi, joten työtä jouduttiin rajaamaan jonkin verran vielä työn edetessä. Erilaisten tietokannan hallintajärjestelmien dokumentaatioon perustuva vertailu oli aikaa vievää ja analyysissä pystyttiin keskittymään vain muutamien kokonaisuuksien arviointiin. Tästä huolimatta työssä pystyttiin kuitenkin vastaamaan asetettuihin tutkimuskysymyksiin ja käytännön testauksessa todentamaan alustavasti tietokannan käytettävyys.

Ennen lopullisen tietokantaratkaisun käyttöönottoa on suositeltavaa testata MongoDB-tietokantaa ja sen suorituskykyä autenttisella datalla sekä tässä työssä tietokantapalvelimena käytettyä työasemaa tehokkaammalla palvelinratkaisulla. Muita työn edetessä esiin nousseita jatkotutkimusideoita ovat muun muassa IQ-datatallenteiden metadatan populointiin liittyvän skriptin laatiminen, testaaminen ja linkittäminen osaksi tietokantaa, kokonaisuuden tietoturvan parantaminen sekä tietokannan hajautetun käytön testaaminen.

Lähteet

All About Circuits. (n.d.). *Understanding I/Q Signals and Quadrature Modulation*. Haettu 10.5.2021 osoitteesta Practical Guide to Radio-Frequency Analysis and Design: https://www.allaboutcircuits.com/textbook/radio-frequency-analysis-design/radio-frequency-demodulation/understanding-i-q-signals-and-quadrature-modulation/

Castrounis, A. (n.d.). *How to Choose the Right Database System: RDBMS vs. NoSql vs. NewSQL.* Haettu 15.8.2021 osoitteesta Innoarchitech: https://www.innoarchitech.com/blog/how-choose-right-database-system-relational-rdbms-vs-nosql-vs-newsql

Chima, R. (2018). *How To Choose A Database For Your Business.* Haettu 20.7.2021 osoitteesta Blueberry Consultants: https://www.bbconsult.co.uk/blog/choose-database-business

Coronel, C. & Morris, S. (2015). *Database Systems: Design, Implementation and Management* (11. p.). Stamford: Cengage Learning.

DataStax. (2021). *CQL for Apache Cassandra 3.0.* Haettu 7.8.2021 osoitteesta https://docs.datastax.com/en/cql-oss/3.3/index.html

DBeaver Community. (n.d.). *DBeaver Documentation*. Haettu 7.8.2021 osoitteesta https://github.com/dbeaver/dbeaver

DB-Engines. (2021a). *DB-Engines Ranking*. Haettu 29.4.2021 osoitteesta https://db-engines.com/en/ranking

DB-Engines. (2021b). *DB-Engines Ranking - Trend Popularity*. Haettu 29.4.2021 osoitteesta https://db-engines.com/en/ranking_trend

DB-Engines. (2021c). *Encyclopedia*. Haettu 29.4.2021 osoitteesta https://db-engines.com/en/articles

DB-Engines. (2021d). *Method of calculating the scores of the DB-Engines Ranking*. Haettu 29.4.2021 osoitteesta https://db-engines.com/en/ranking_definition

Drake, M. (2014). *A Comparison of NoSQL Database Management Systems and Models.* Haettu 20.7.2021 osoitteesta https://www.digitalocean.com/community/tutorials/a-comparison-of-nosql-database-management-systems-and-models

Ghilduta, R. (2019). *bladeRF CLI Tips and Tricks*. Haettu 18.9.2021 osoitteesta https://github.com/Nuand/bladeRF/wiki/bladeRF-CLI-Tips-and-Tricks#Receiving_samples

Henderson, P. D. (2004). *The Fundamentals of FFT-Based Audio Measurements in SmaartLive®.* Haettu 18.9.2021 osoitteesta https://www.rationalacoustics.com/files/FFT_Fundamentals.pdf

Hernandez, M. J. (2000). *Tietokannat - suunnittelu ja toteutus käytännössä* (2. p.). Edita, IT Press.

Huttunen, H. (2014). *Signaalinkäsittelyn perusteet.* Haettu 10.4.2021 osoitteesta Trepo: http://urn.fi/URN:ISBN:978-952-15-3222-1

InfluxData. (2021). *InfluxDB*. Haettu 8.8.2021 osoitteesta https://www.influxdata.com/products/influxdb/

ITU-R. (2017). *Technical identification of digital signals.* Haettu 27.6.2021 osoitteesta https://www.itu.int/rec/R-REC-SM.1600-3-201709-I/en

ITU-R. (2018). *Data format definition for exchanging stored I/Q data for the purpose of spectrum monitoring.* Haettu 27.6.2021 osoitteesta https://www.itu.int/rec/R-REC-SM.2117-0-201809-I/en

Jokinen, R. (2003). *Digitaalinen signaalinkäsittely.* Salo: Taiga Tehnology.

Kansalliskirjasto. (2016). *Finto - suomalainen sanasto- ja ontologiapalvelu*. Haettu 8.6.2021 osoitteesta https://finto.fi/fi/

Kroenke, D. M. & Auer, D. J. (2011). *Database Concepts* (5. p.). Boston: Pearson.

Krug, P. (2019). *The Normalization Spectrum.* Haettu 15.8.2021 osoitteesta https://dzone.com/articles/the-normalization-spectrum

Kuittinen, P. (2011). *SQL or NoSQL? Solutions for the Web.* Häme University of Applied Sciences.

Lehtonen, J. (2016). *Ohjelmistovastaanottimen soveltaminen radiotekniikan opetukseen.* Opinnäytetyö. Tietotekniikan koulutusohjelma. Hämeen ammattikorkeakoulu.

Lichtman, M. (n.d.). *A Guide to SDR and DSP using Python*. Haettu 10.5.2021 osoitteesta PySDR: https://pysdr.org/

Liu, S. (2021). *Most popular database management systems worldwide 2021.* Haettu 29.4.2021 osoitteesta https://www.statista.com/statistics/809750/worldwide-popularity-ranking-database-management-systems/

Lyons, R. (2000). *Quadrature signals: Complex, But Not Complicated*. Haettu 2.7.2021 osoitteesta https://paginas.fe.up.pt/~amoura/Tele2/QuadSignals.pdf

MariaDB. (n.d.). *MariaDB*. Haettu 7.8.2021 osoitteesta https://mariadb.com

MathWorks. (n.d.). *Database Toolbox*. Haettu 21.7.2021 osoitteesta https://se.mathworks.com/help/database/index.html?s_tid=CRUX_lftnav

Microsoft. (2021). *Relational vs. NoSQL data*. Haettu 8.6.2021 osoitteesta https://docs.microsoft.com/en-us/dotnet/architecture/cloud-native/relational-vs-nosql-data

Microsoft. (n.d.). *Perustiedot tietokannasta*. Haettu 8.6.2021 osoitteesta https://support.microsoft.com/fi-fi/office/perustiedot-tietokannasta-a849ac16-07c7-4a31-9948-3c8c94a7c204

MongoDB. (2020). *MongoDB Architecture Guide: Overview.* Haettu 12.8.2021 osoitteesta https://info-mongodb-com.s3.us-east-1.amazonaws.com/MongoDB_Architecture_Guide.pdf

Mostafa, K. (n.d.). *Minimodem - general-purpose software audio FSK modem*. Haettu 24.9.2021 osoitteesta https://github.com/kamalmostafa/minimodem

Mustonen, T. (2021). Joustavaa ELSO-suorituskykyä ohjelmistovastaanottimilla. *Viestimies*(3/2021), s. 18-21.

Mäkelä, M.;Soininen, L.;Tuomola, S. & Öistämö, J. (2005). *Tekniikan kaavasto.* Tampere: Tammertekniikka/Amk-Kustannus Oy.

National Instruments. (n.d.). *Understanding FFTs and Windowing.* Haettu 12.5.2021 osoitteesta https://download.ni.com/evaluation/pxi/Understanding%20FFTs%20and%20Windowing.pdf

Neo4j. (n.d.). *Neo4j Graph Database*. Haettu 8.8.2021 osoitteesta https://neo4j.com/product/neo4j-graph-database/

Northern Arizona University. (2017). *I/Q Data Quide v.1 - Guide for acquiring I/Q data with the Gnu Radio Framework.* (M. Finley, Toim.) Haettu 15.7.2021 osoitteesta Northern Arizona University - Dynamic and Active Systems Laboratory: https://uavrt.nau.edu/wp-content/uploads/2017/06/I_Q_Data_Documentation_v1.pdf

Núñez, J. & Mascareñas, C. (2016). *Software Defined Radio (SDR) on radiocommunications teaching.* Haettu 15.7.2021 osoitteesta https://doi.org/10.21125/INTED.2016.1244

OmniSci. (n.d.). *Open Source Database Definition*. Haettu 8.6.2021 osoitteesta https://www.omnisci.com/technical-glossary/open-source-database

Open Source Initiative. (2007). *The Open Source Definition*. Haettu 8.6.2021 osoitteesta https://opensource.org/osd

Oracle Corporation. (n.d.). *MySQL - The world's most popular open source database*. Haettu 20.7.2021 osoitteesta https://www.mysql.com/

Oracle. (n.d.a). *What Is a Database?* Haettu 7.6.2021 osoitteesta https://www.oracle.com/database/what-is-database/

Oracle. (n.d.b). *What is a Relational Database (RDBMS)?* Haettu 7.6.2021 osoitteesta https://www.oracle.com/database/what-is-a-relational-database/

Panwar, A. (2021). *Types of Database Management Systems*. Haettu 19.7.2021 osoitteesta C#Corner: https://www.c-sharpcorner.com/UploadFile/65fc13/types-of-database-management-systems/

Pohl, J. & Noack, A. (2018). *Universal Radio Hacker: A Suite for Analyzing and Attacking Stateful Wireless Protocols.* Haettu 25.9.2021 osoitteesta https://www.usenix.org/system/files/conference/woot18/woot18-paper-pohl.pdf

Radartutorial. (n.d.). *Time-Domain versus Frequency-Domain*. Haettu 24.9.2021 osoitteesta https://www.radartutorial.eu/10.processing/sp53.en.html

Redis Ltd. (n.d.). *Redis*. Haettu 7.8.2021 osoitteesta https://redis.io/

Rohde & Schwarz. (2015). *R&S iq-tar - File Format Specification.* Haettu 11.7.2021 osoitteesta https://cdn.rohde-schwarz.com/pws/dl_downloads/dl_common_library/dl_manuals/gb_1/f/fsw_1/RS_iq-tarFileFormatSpecification_02.pdf

Rouphael, T. J. (2014). *Wireless Receiver Architectures and Design - Antennas, RF, Synthesizers, Mixed Signal and Digital Signal Processing.* Oxford: Elsevier.

Routio, P. (n.d.). *Tutkimusmenetelmät, Toteava tutkimus*. Haettu 13.7.2021 osoitteesta Taideteollinen korkeakoulu, Virtuaaliyliopisto: http://www.uiah.fi/virtu/materiaalit/tuotetiede/html_files/14112_totea.html

Runkel, J. (2014). *Building MongoDB Applications with Binary Files Using GridFS: Part1-2*. Haettu 18.10.2021 osoitteesta https://www.mongodb.com/blog/post/building-mongodb-applications-binary-files-using-gridfs-part-1

Räisänen, A. & Lehto, A. (2011). *Radiotekniikan perusteet.* Helsinki: Gaudeamus Helsinki University Press.

Schilcher, T. (2008). RF applications in digital signal processing. Teoksessa *Digital Signal Processing.* Geneva: CERN Accelerator School. Haettu 25.6.2021 osoitteesta http://dx.doi.org/10.5170/CERN-2008-003.249

Schipper, M. (2013). *I/Q Waveform File Conversion for Use with Precise Broadcast Signal Generators - Application Note.* Haettu 21.8.2021 osoitteesta https://cdn.rohde-schwarz.com/pws/dl_downloads/dl_application/application_notes/7bm79/7BM79_2E.pdf

Sears, R.;van Ingen, C. & Gray, J. (2006). *To BLOB or Not To BLOB: Large Object Storage in a Database or a Filesystem?* Haettu 25.5.2021 osoitteesta https://www.microsoft.com/en-us/research/publication/to-blob-or-not-to-blob-large-object-storage-in-a-database-or-a-filesystem/

Stranneby, D. (2001). *Digital Signal Processing: DSP and Applications.* Oxford: Newnes.

Tektronix. (2006). *Digital I-Q Data Capture and Correction.* Haettu 3.9.2021 osoitteesta https://download.tek.com/document/37W-19297-1.pdf

The Apache Software Foundation. (2021a). *Apache Cassandra - Open Source NoSQL Database*. Haettu 8.8.2021 osoitteesta https://cassandra.apache.org/_/index.html

The Apache Software Foundation. (2021b). *Apache HBase*. Haettu 8.8.2021 osoitteesta https://hbase.apache.org/

The Apache Software Foundation. (2021c). *Apache Jackrabbit*. Haettu 8.8.2021 osoitteesta https://jackrabbit.apache.org/jcr/index.html

The PostgreSQL Global Development Group. (n.d.). *PostgreSQL: The World's Most Advanced Open Source Relational Database*. Haettu 7.8.2021 osoitteesta https://www.postgresql.org/

Tieteen termipankki. (2016). *Entiteetti*. Haettu 8.6.2021 osoitteesta https://tieteentermipankki.fi/wiki/Kielitiede:entiteetti

Tobin, D. (2020). *Which Modern Database Is Right for Your Use Case?* Haettu 8.6.2021 osoitteesta Xplenty: https://www.xplenty.com/blog/which-database/

Valkama, M. (2001). *Advanced I/Q Signal Processing for Wideband Receivers: Models and Algorithms.* Tampereen Teknillinen korkeakoulu-Tietotekniikan osasto. Haettu 14.3.2021 osoitteesta http://urn.fi/URN:NBN:fi:tty-200810021094

Vanhatapio, J. (2020). *Redis*. Haettu 14.8.2021 osoitteesta https://www.zoner.fi/redis/

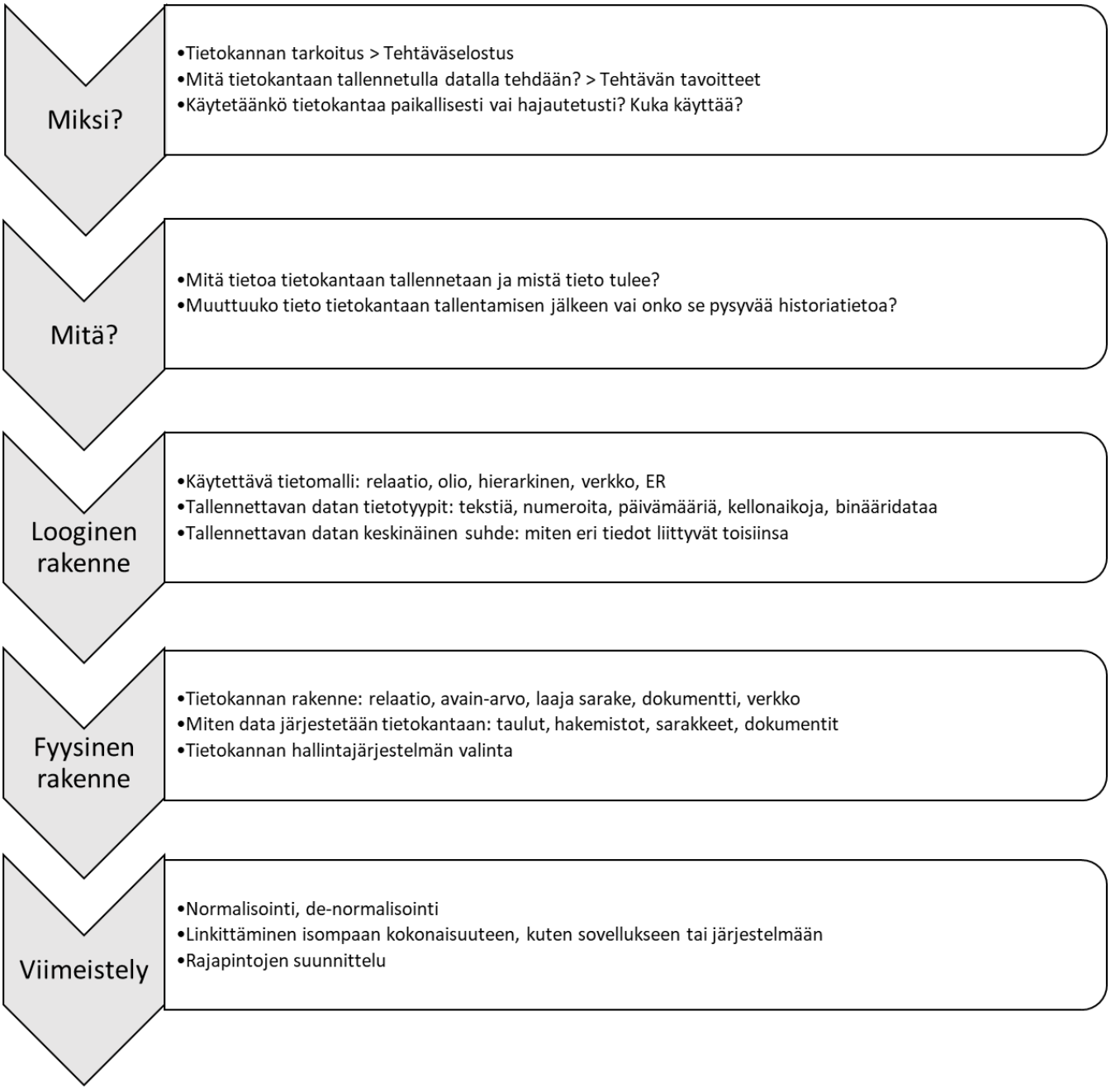
Vuori, J. (n.d.). Tapaustutkimus. Teoksessa J. Vuori (Toim.), *Laadullisen tutkimuksen verkkokäsikirja.* Tampere: Yhteiskuntatieteellinen tietoarkisto. Haettu 14.8.2021

osoitteesta https://www.fsd.tuni.fi/fi/palvelut/menetelmaopetus/kvali/tutkimusasetelma/tapaustutkimus/

Watt, A. & Eng, N. (2014). *Database Desgin - 2nd Edition.* Victoria, B.C.: BCcampus. Haettu 25.7.2021 osoitteesta https://opentextbc.ca/dbdesign01/

Wolke, A. (2015). *What's Your IQ - About Quadrature Signals: Quadrature IQ Signals Explained*. Haettu 29.4.2021 osoitteesta Tektronix: https://www.tek.com/blog/quadrature-iq-signals-explained

Liite 1: Tietokannan suunnitteluprosessi



Liite 2: Tietokantojen vertailutulokset

	Cassandra	Hbase	InfluxDB	Jackrabbit	MariaDB	MongoDB	MySQL	Neo4j	PostgreSQL	Redis
Tietomalli	Laaja sarake	Laaja sarake	Aikasarja, spatiaalinen	Tietovarasto	Relaatio, dokumentti, verkko, spatiaalinen	Dokumentti, spatiaalinen, aikasarja	Relaatio, dokumentti, spatiaalinen	Verkko	Relaatio, dokumentti, spatiaalinen	Avain-arvo, dokumentti, verkko, spatiaalinen, aikasarja
Kyselykieli	Cassandra QL	Apache Drill (SQL hakukone)	Flux tai InfluxQL	Xpath/SQL Parser	SQL	MongoDB QL	SQL	Cypher QL	SQL	Redis
Ohjelmointikieli	Java	Java	Go	Java	C ja C++	C++	C ja C++	Java ja Scala	C	C
Julkaisuvuosi ja lisenssi	2008, Apache	2008, Apache	2013, MIT-license	2006, Apache	2009, GPL	2009, MongoDB Inc Server Side public license	1995, GPL	2007, GPL	1989, BSD	2009, BSD
Versio 08/2021	3.11.11 (28.7.2021)	2.4.5 (31.7.2021)	2.0.8 (13.8.2021)	2.20.3 (11.6.2021)	10.6.4 (6.8.2021)	5.0.2 (4.8.2021)	8.0.26	4.3.3 (9.8.2021)	13.4 (12.8.2021)	6.0
Yhteensopivat alustat	Docker, Linux, MacOS, Windows	Docker, Linux, MacOS, Windows	Docker, Linux, MacOS, Windows	*Linux, OS X*	Docker, Linux, Windows	Docker, Linux, MacOS, Windows	Docker, Linux, MacOS, Windows	Docker, Linux, MacOS, Windows	Dokcer, Linux, MacOS, Windows	Docker, **Linux**, MacOS
Käyttökohteita	IoT, maksuvälinepetosten ehkäisy, suosittelusovellukset, viestisovellukset, soittolistat	IoT, maksuvälinepetosten ehkäisy, suosittelusovellukset, viestisovellukset, soittolistat	IoT, DevOps	Versionhallinta, pääsynhallinta	Internetsivustot, pilvipalvelut, SaaS	IoT, pelit, maksusovellukset, analytiikka ja tekoäly	Verkkosivustot ja -sovellukset	Maksuvälinepetosten ehkäisy, verkonhallinta ja -valvonta, pääsynvalvonta, sosiaalinen media, lohkoketjujen hallinta	Verkkosivustot ja -sovellukset	Maksuvälinepetosten ehkäisy, sosiaalinen media, välimuistitallennuksen hallinta
Datan tuonti massana	SSTables, ei tue muita formaatteja	Spark	Batch write	-	cpimport, json	mongoimport	LOAD DATA INFILE	CSV	CSV, PGLoader, pg_bulkload	Redis Mass Insertion
Matlab-yhteensopivuus	kyllä	ei	ei	ei	ei	kyllä	kyllä	kyllä	kyllä	rajoitettu
API	Java API	Rest, Thrift, JDO, Scala, Jython	InfluxDB API	JCR API	Rest API	MongoDB Query API, Atlas API	Rest API	Rest API	PostgreREST (RESTful API)	Rest API
Skaalautuvuus	Solmujen määrää nostamalla	Horisontaalisesti	InfluxDB Clustering (vain kaupallisessa tuotteessa)	Jackrabbit Oak	MaxScale	Horisontaalisesti	MySQL Cluster	Sirpaloimalla ja yhdistämällä dataa	-	Solmujen määrää nostamalla
BLOB (vast.) maksimikoko	2 GB (1 MB)	-	-	2 GB	4 GB	GridsFS (ei kokorajoitusta)	4 GB	Ei suositeltu	1 GB	512 MB
CLI/GUI	Cassandra GUI Client, DBeaver	Hbase Shell CLI	Influx CLI, Flux UI, DBeaver	-	Dbeaver, dbForge Studio, HeidiSQL, phpMyAdmin	MongoDB CLI, Dbeaver, NoSQL Booster	Dbeaver, dbForge Studio, Workbench, HeidiSQL, phpMyAdmin	Neo4j Desktop	Dbeaver, HeidiSQL, Workbench	Redis CLI, DBeaver
Arvio haku- ja suodatusmahdollisuuksista	monipuoliset	monipuoliset	monipuoliset	monipuoliset	monipuoliset	monipuoliset	monipuoliset	monipuoliset	monipuoliset	yksinkertaiset
Arvio soveltuvuudesta IQ-dataan	kohtalainen	huono	huono	hyvä	kohtalainen	hyvä	kohtalainen	huono	kohtalainen	huono
Lähteet	The Apache Software Foundation, 2021a; DataStax, 2021; DBeaver Community, n.d.; Mathworks, n.d.	The Apache Software Foundation, 2021b	InfluxData, 2021; DBeaver Community, n.d.	The Apache Software Foundation, 2021c	MariaDB, n.d.	MongoDB, 2020; DBeaver Community, n.d.; Mathworks, n.d.	Oracle Corporation, n.d.; DBeaver Community, n.d.; Mathworks, n.d.	Neo4j, n.d.; Mathworks, n.d.	The PostgreSQL Global Development Group, n.d.; DBeaver Community, n.d.; Mathworks, n.d.	Redis Ltd., n.d.; DBeaver Community, n.d.

Liite 3: Ohjelmakoodi IQ-datatiedostojen ja metadatan tuontiin

```
# Import IQ-data -files and their metadata from defined local directories to MongoDB GridFs
# Prerequirements:
#   * IQ-data -file and its metadata-file must be named exactly same way
#       e.q my_iq_data.csv and my_iq_data.json
#   * Metadata must be in json-format

from pymongo import MongoClient
import gridfs, os, json

# Variable for database-connection
def db_connection():
    try:
        connection = MongoClient(host='localhost', port=27017)
        print("Connection to database ok: ", connection)
        return connection.mytestdb2
    except Exception as e:
        print("Error while connecting to database: ", e)

# Create database-connection
db = db_connection()

# Find files from defined directory
with os.scandir('C:\\...\\IQfile\\') as files:
    for file in files:
        # Open and read the IQ-file
        get_data = open(file, "rb")
        read_data = get_data.read()

        # Open and read metadata-file
        meta_data = file.name
        meta_location = 'C:\\...\\Metadata\\' + meta_data.split('.')[0] + '.json'
        with open(meta_location, "rb") as m:
            meta_data_ = json.load(m)

        # Store data to database
        fs = gridfs.GridFS(db)
        fs.put(read_data, filename = file.name, metadata = meta_data_)

print("Files imported")
```