Tampereen ammattikorkeakoulu

Spatiotemporaalinen karttasovellus

Teknologioiden vertailu

Daniel Schmidt

OPINNÄYTETYÖ
Toukokuu 2024

Tietotekniikan tutkinto-ohjelma
Ohjelmistotekniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma
Ohjelmistotekniikka

SCHMIDT, DANIEL:
Spatiotemporaalinen karttasovellus
Teknologioiden vertailu

Opinnäytetyö 43 sivua, joista liitteitä 5 sivua
Toukokuu 2024

Opinnäytetyössä vertaillaan eri tapoja toteuttaa verkkoselaimessa käytettävä sovellus ja sen vaatima ympäristö olioiden sijaintihistorian kartalla esittämistä varten. Toimeksiantajayritys perustelee tarvetta opinnäytetyölle sillä, että saatavilla on niukasti valmiita ohjelmistoja menneen tilannekuvatiedon tuottamiseen. Saatavilla olevat vaihtoehdot ovat nekin toiminnoiltaan hyvin rajalliset. Datan näyttäminen staattisina kerroksina kartalla on melko suoraviivaista, mutta tehokkaan ja dynaamisen esityksen toteuttamista varten on syytä perehtyä tarjolla oleviin menetelmiin huolellisesti.

Sijaintitietoa lähettävien olioiden esittämistä varten on tarjolla runsaasti erilaisia ohjelmallisia ratkaisuja. Työkaluja voi luovasti yhdistellä, ja sovellusten toteutustavat vaihtelevatkin päiväkohtaisten tietueryppäiden näkymistä yksittäisen olion tarkkoihin paikkamääreisiin ajassa. Yhteistä eri sovelluksille on, että datan paljouden ongelmaa on tavalla tai toisella jouduttu väistämään. Suorituskykyä koettelevat tietokantakyselyt, tiedonsiirto ja tietueiden graafiset esitykset.

Saatavilla olevista teknologioista vertailuun valittiin tehokkuutta korostavat vaihtoehdot. Työssä vertailtavat tietokantajärjestelmät ovat Apache Arrow, DuckDB, MongoDB ja PostgreSQL. Karttasovellusten kehittämiseen käytettiin ohjelmistokehyksiä OpenLayers sekä Deck.gl.

Työ aloitettiin tietokantajärjestelmien suorituskykymittauksilla. Tulosten perusteella Apache Arrow soveltuu hyvin yksinkertaisen tilannekuvan välittämiseen ja DuckDB data-analyttisiin tietokantakyselyihin. Näiden järjestelmien varaan kehitettiin kahta käyttötapausta vastaavat karttasovellukset: Ensimmäinen sovellus oli paikka- ja aikaulottuvuuksisen tilannekuvatiedon esittämistä varten, ja toisessa sovelluksessa painotetaan monimuotoista data-analytiikkaa.

Asiasanat: tietokantaohjelmat, karttapalvelut, maantieteelliset koordinaatit, temporaalisuus, massadata

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in ICT Engineering
Software Engineering

SCHMIDT, DANIEL:
Spatiotemporal Map Application
Technology Benchmarking

Bachelor's thesis 43 pages, appendices 5 pages
May 2024

The aim of this thesis was to explore and compare various technological approaches to developing a web-based application for presenting the location history of objects on a map. The thesis sought to identify the most effective methods and technologies, considering the limitations of the existing software in producing past situational data. The goal was to enhance understanding of how different database systems and mapping frameworks could be optimised for such tasks, ultimately providing insights into creating more dynamic and efficient visual representations of location data.

The study focused on comparing the database systems known for their performance capabilities, like Apache Arrow, DuckDB, MongoDB, and PostgreSQL. Frameworks like OpenLayers and Deck.gl were used for map application development. The initial database performance tests showed that Apache Arrow was suited for simple situational data displays while DuckDB was better for data-analytic queries, leading to the development of two case-specific map applications.

Key words: database programs, map services, geographical coordinates, temporality, big data

SISÄLLYS

1 JOHDANTO 5
2 KÄYTETYT TEKNOLOGIAT 6
2.1 Laitteistot ja ohjelmistot 6
2.2 OLAP- ja OLTP-järjestelmät 6
2.3 Vertailuun valitut tietokantajärjestelmät 7
2.4 Karttasovellusten tiedonsiirtoprotokollat 9
2.5 Karttasovellusten JavaScript-kirjastot 10
3 TIETOKANTOJEN KÄYTTÖÖNOTTO 11
3.1 Indeksoinnin teoriaa 11
3.2 AIS-data 12
3.3 Parquet-migraatio 13
3.4 MongoDB- ja PostgreSQL-migraatiot 14
4 TIETOKANTOJEN VERTAILU 15
4.1 Ennakkovalmistelut 15
4.2 Spatiotemporaaliset kyselyt 16
4.3 Analyyttiset kyselyt 19
5 SOVELLUSKOKEILUT 23
5.1 Spatiotemporaalinen sovellus 23
5.2 Lämpökarttasovellus 26
6 POHDINTA 28
6.1 Relaatiotietokantojen vertailu 28
6.2 Parquet-tiedostot tietokantana 29
6.3 Tietokanta osana asiakasohjelmistoa 30
6.4 Johtopäätökset sovelluskokeiluista 31
6.5 Käyttäjälle olennainen data 32
6.6 Riskialtis WebSocket 35
LÄHTEET 36
LIITTEET 39
Liite 1. Näyte AIS-datasta 39
Liite 2. Attribuuttien datatyypit eri tietokantajärjestelmissä 41
Liite 3. MongoDB:n indeksit 42
Liite 4. PostgreSQL:n indeksit 43

1 JOHDANTO

Olioiden sijaintihistorian esittäminen web-pohjaisella karttasovelluksella ei työn tekemisen aikaan ollut kovinkaan yleinen konsepti. Tällainen paikkatietojen dynaaminen esittäminen tuo mukanaan datan paljouden ongelman, jonka olemassa olevat sovellukset ovat tavalla tai toisella joutuneet ratkaisemaan: dataa on esimerkiksi valmiiksi koostettuna tietokantaan taikka näkymä rajoitetaan yhteen karttaolioon kerrallaan. Toimeksiantaja halusi selvitettävän, millaisella yhdistelmällä moderneja teknologioita verkkoinfrastruktuurin eri tasoilla spatiotemporaalinen karttasovellus kannattaisi toteuttaa. Mitkä menetelmät ovat viisas sijoitus tulevaisuuteen?

Opinnäytetyön prosessi käynnistyi karttasovellukseen tarvittavien ohjelmallisten komponenttien kartoittamisella. Sovelluksessa yhdistyvät suurilla datamäärillä tehtävät tietokantakyselyt, tiedonsiirto sekä verkkoselaimessa visualisointi. Kartoituksen pohjalta valittiin tehtävään soveltuvat tietokantajärjestelmät ja sovellusohjelmistot. Työssä vertailtiin eri tavoin erikoistuneita järjestelmiä: karttasovelluksille tyypillisiä sekä toisaalta datanpaljoudenkin keskellä nopeita tietokantoja. Tiedonsiirrossa huomioitiin mahdollisuus siirtää dataa pakattuna tavuvirtana. Karttanäkymän voi esittää verkkosivulla grafiikkakiihdytettynä. Tietokantojen suorituskykyjä vertailtiin, ja tulosten perusteella kaksi näistä valittiin osaksi karttasovelluksia.

Työn alussa on lueteltu sovelluskehityksessä ja suorituskykymittauksissa käytetyt laitteistot ja ohjelmistot, vertailuun valitut tietokantajärjestelmät ja sovelluskokeiluissa käytetyt tiedonsiirtoprotokollat sekä JavaScript-kirjastot. Tämän jälkeen on kerrottu kehityksessä käytetyn AIS-datan luonteesta ja käsittelystä sekä datan lisäämisestä tietokantoihin. Seuraavaksi on esitetty tulokset tietokantojen suorituskykymittauksista, ensin datan suodattamisen ja sitten koostamisen ja analysoinnin näkökulmasta. Karttasovelluksen kehittämisen vaiheista ja lopputuloksista on raportoitu työn loppupuolella. Viimeinen luku on yhteenveto mittaustuloksista ja sovellusten toiminnasta, ja sisältää myös ehdotelmia jatkokehityksestä ja lisätutkimuksista.

2 KÄYTETYT TEKNOLOGIAT

Tässä luvussa esitellään työssä käytetyt keskeiset teknologiat, kuten kehitysympäristön laitteistot ja ohjelmistot, tietokantajärjestelmät, JavaScript-ohjelmistokehykset sekä tiedonsiirtoprotokollat. Vertailuun valituista teknologioista kullakin oli jokin karttasovellusta ajatellen kiinnostava piirre. Yleisesti ottaen vertailuun valittiin saatavilla olevista vaihtoehdoista tiedon hakemisen, siirtämisen tai esittämisen tehokkuutta korostava vaihtoehto, riittävä helppokäyttöisyys ja yhteisön tuki huomioiden. Tietokantojen tai karttaohjelmistojen käytöstä ei aiheutunut lisenssikustannuksia.

2.1 Laitteistot ja ohjelmistot

Tietokone

- Suoritin: AMD Ryzen 5 1600 3 GHz, kuusi fyysistä ja 12 virtuaalista ydintä
- Keskusmuisti: kaksi kappaletta Corsair 4 GB DDR4 2666 MHz -muistikampaa
- Kiintolevy: Samsung SSD 850 EVO 250 GB
- Näytönohjain: NVIDIA GeForce GTX 1050 Ti 4 GB

Ohjelmistot

- Käyttöjärjestelmä: Windows 10 22H2
- Kehitystyökalut: Visual Studio Code v. 1.87, Google Chrome v. 122.0 & DevTools, OpenAI GPT-4 -kielimalli, Draw.io-kaaviotyökalu

2.2 OLAP- ja OLTP-järjestelmät

Tietokantajärjestelmät voidaan karkeasti jakaa käyttötarkoituksensa mukaan kahteen pääluokkaan: Verkon yli tehtävä tiedon analyyttinen prosessointi, OLAP ja verkon yli tehtävä tiedonvaihto, OLTP. Vaikka monet järjestelmät selkeästi edustavat jompaa kumpaa käyttötarkoitusta, on joihinkin OLTP-järjestelmiin

teknologisen toimintaympäristön muuttumisen myötä lisätty OLAP-ominaisuuksia vastaamaan nykypäivän tarpeita.

OLAP-järjestelmät (online analytical processing) ovat erikoistuneet suurten datamäärien tehokkaaseen koostamiseen ja analysointiin. Järjestelmiä käytetään yleisesti yrityksissä päätöksenteon tukena tuottamaan tietoa liiketoiminnasta. Yritys voi haluta esimerkiksi tutkia asiakaskäyttäytymistä analysoimalla asiakasdataa: Sijaintitietoja, ostotapahtumia ja asiakastyytyväisyyttä. Dataa käsitellään usein OLAP-kuutioina, jolla tarkoitetaan useamman attribuutin koottua, moniulotteista esitystä. (Amazon Web Services. n.d.; Sinha, T. 2021.)

OLTP-järjestelmissä (online transactional processing) palvelu keskitytään tarjoamaan useille käyttäjille samanaikaisesti. Tietoa tulee voida hakea, lisätä, muokata ja poistaa reaaliaikaisesti, luotettavasti ja nopeasti. Tyypillisiä käyttökohteita ovat esimerkiksi asiakastietojen tai maksutapahtumien käsittelyt. (Amazon Web Services. n.d.; Sinha, T. 2021.)

2.3 Vertailuun valitut tietokantajärjestelmät

Apache Arrow on ohjelmistokehitysalusta, joka mahdollistaa tiedon tehokkaan käsittelemisen RAM-muistissa. Dataa voidaan tehokkaasti analysoida suorittimia ja näytönohjaimia hyödyntäen, ja alusta tukee datan lukemista muistista ilman kopioimista. Arrow:n ohjelmistokirjastot ovat saatavilla lukuisille eri ohjelmointikielille, mm. C++, C#, Java, MATLAB, Python, R ja Rust. Alusta määrittää standardoidun muodon datalle, joka on saatavilla eri järjestelmien välillä rajapintojen kautta muunnoksitta. (The Apache Software Foundation n.d.a., n.d.b.; Ahmad, Al-Ars & Hofstee 2022, Introduction.)

Arrow käsittelee tietueita Record Batch -taulukoissa, ja data sijoitetaan muistiin sarakkeittain. Kutakin attribuuttia voi vastata metadatavektori, joka voi tietotyypistä riipppuen sisältää esimerkiksi tietueen validiteettibitin, taikka seuraavan tietueen arvon alkuindeksin, jos kyseessä on muuttuvapituuksinen datatyyppi. (Ahmad ym. 2022, Introduction.) Arrow-projektin Flight-

ohjelmistokehys mahdollistaa tiedonsiirron eri ohjelmointikielten ja -kehysten sekä verkkolaitteiden välillä ilman serialisoinnin kuormaa. Flight toimii gRPC-protokollalla, joka HTTPS:n tapaan käyttää yhteyden salaamiseen TLS/OpenSSL-protokollaa. (Ahmad ym. 2022, Background.)

DuckDB SQL-syntaksin tarjoava OLAP-tietokantajärjestelmä, joka tarjoaa rajapinnan usealle ohjelmointikielelle, esimerkiksi C, C++, Java, Go ja Node.js. Järjestelmää voidaan ulkopuolisen ohjelman käyttämänä ajaa isäntäprosessin sisällä, jolloin vältytään prosessien välisen kommunikaation aiheuttamalta kuormitukselta. Vaikka DuckDB onkin relaatiotietokanta, määrittää se Arrow:n tapaan sarakkeisen datan säilytysmuodon. DuckDB-Wasm mahdollistaa järjestelmän ajamisen myös verkkoselaimissa ja älylaitteilla. DuckDB on omiaan monimutkaisten yhdisteiden tuottamiseen suurista datamassoista. (DuckDB Foundation, Amsterdam NL n.d..)

MongoDB on asiakirjatietokantajärjestelmä, jossa tietueet säilötään erillisinä dokumentteina BSON-muodossa (Binary JSON) (MongoDB, Inc. n.d.a, n.d.b). MongoDB-tietokanta ei edellytä attribuuttien lukumäärien tai tietotyyppien määrittelyä, toisin kuin relaatiotietokannat (Chodorow 2013, kappale 1). MongoDB-tietokanta on laajennettavissa horisontaalisesti (MongoDB, Inc. n.d.a; Chodorow 2013, kappale 1). Tämä tarkoittaa, että tietokantainfrastruktuuria voidaan laajentaa palvelimia lisäämällä yksittäisten palvelinten suorituskyvyn kasvattamisen sijaan. MongoDB tukee perinteisten kyselyiden lisäksi mm. indeksointia ja datan koostamista (Chodorow 2013, kappale 1).

PostgreSQL on vapaan lähdekoodin olio-relaatiotietokantajärjestelmä, jolla on kattava tuki SQL-standardille (The PostgreSQL Global Development Group n.d.a.). Relaatiotietokannassa data säilötään kaksiulotteisiin taulukkoihin, ja datatyyppeinä voidaan käyttää esimerkiksi kokonaislukuja, merkkijonoja ja aikaleimoja. Olio-relaatiotietokanta tukee edellisten lisäksi taulukkoa datatyyppinä, taulujen periytymistä sekä SQL-kyselyssä toteutettavia funktiokutsuja. (Drake & Worsley 2002, Introduction to Relational Databases.)

Järjestelmään on saatavilla geospatiaalisen datan talletusta, indeksointia ja hakemista varten kehitetty laajennus, PostGIS. Tämä tarjoaa mm. tuen

avaruudellisia tietotyyppejä varten: piste, viiva ja monikulmio, kaksi- tai kolmiulotteisessa avaruudessa. (PostGIS PSC & OSGeo n.d..) PostgreSQL on perinteisesti nähty OLTP-tietokantana, mutta versiosta 9.6 alkaen järjestelmällä voi tehdä rinnakkaisajoa ja OLAP-piirteet ovat järjestelmässä lisääntyneet (Richter 2021). PostgreSQL tarjoaa erityyppisiä indeksointitapoja, joista tämän työn yhteydessä käytettiin kahta: B-Tree ja GiST. PostgreSQL:ssa B-Tree-algoritmilla voidaan käsitellä yhtäsuuruus- ja arvojoukkokyselyitä järjestettävälle datalle. GiST määrittelee infrastruktuurin eri indeksointitavoille ja sitä voidaan käyttää esimerkiksi lähin naapuri -hakujen optimoinnissa. (The PostgreSQL Global Development Group n.d.b..)

2.4 Karttasovellusten tiedonsiirtoprotokollat

HTTP (hypertext trasfrer protocol) on sovelluskerroksessa toimiva, verkkopohjaisten tietojärjestelmien tiedonvaihtoa varten tehty protokolla. HTTP:n toiminta perustuu asiakasohjelmiston pyynnöille ja palvelimen vastauksille, ja protokollan viestintä tapahtuu yleensä TCP/IP-yhteyksien päällä. (Fielding, Gettys, Mogul, Frystyk, Masinter, Leach & Berners-Lee 1999, Introduction.)

Versiosta 1.1 alkaen HTTP-yhteyden oletetaan pysyvän auki, kunnes se otsikossa määrätään suljettavaksi (Fielding ym. 1999, Connections). Päivitys julkaistiin vuonna 1997 ja se oli loikka nykyaikaan: yhden vastauksen tuottama datamäärä harvoin mahtuu yksittäiseen protokollaviestiin, ja ilman pysyvää yhteyttä alkutiedot täytyy liittää osaksi jokaista viestiä. Reaaliaikainen viestintä HTTP:llä tapahtuu kuitenkin edelleen kiertokyselyillä: Asiakasohjelmisto lähettää palvelimelle pyynnön määrävälein, ja jokaista pyyntöä kohden avataan ja suljetaan erillinen yhteys (Medium 2023).

WebSocket (WS) on tiedonsiirtoprotokolla, joka mahdollistaa viestimisen palvelimen ja asiakasohjelman välillä molempiin suuntiin yhden TCP-yhteyden yli. Protokolla luotiin reaaliaikaista tiedonvaihtoa varten, ja se hyödyntää olemassa olevaa HTTP-infrastruktuuria. Asiakasohjelman avaa yhteyden HTTP-yhteensopivalla kättelyllä, jossa se pyytää palvelinta vaihtamaan käytettävän protokollan WebSocket:iin. WS:n merkittävä etu HTTP:iin verrattuna on, että

kerran muodostettua yhteyttä käytettäessä HTTP-otsikko välitetään vain yhteyden avaamisen ja sulkemisen yhteydessä. (Fette & Melnikov 2011, Introduction.)

2.5 Karttasovellusten JavaScript-kirjastot

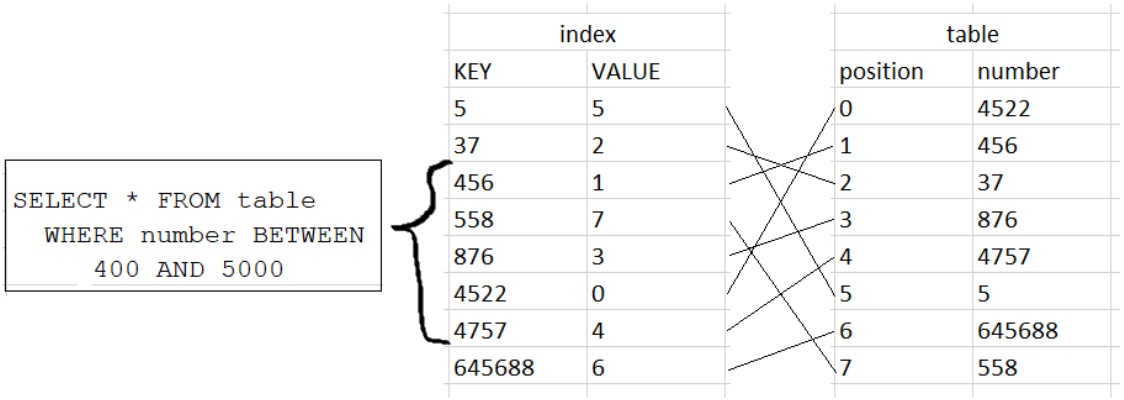
OpenLayers on web-pohjaisten karttasovellusten luomiseen tarkoitettu JavaScript-kirjasto. Kirjaston avulla voidaan esittää karttakuvia ja vektoridataa verkkosivulla. (OpenLayers n.d..) Opinnäytetyön sovelluskokeiluissa käytetään OpenLayers-kirjastoa **OpenStreetMap**-kartasta sekä Deck.gl-oliosta luotujen kerrosten esittämiseen verkkosivulla. OpenStreetMap:lla viitataan tässä web-sovellukseen (tile server), jonka samanniminen palvelu tarjoaa myös itsenäisen palvelun maailmankartan web-sivulla näyttämiseen.

Deck.gl on grafiikkakiihdytettyyn datan web-selaimessa visualisointiin erikoistunut ohjelmistokehys. Sovelluksissa datasta (yleensä JSON-olioita) voidaan luoda kerroksia, joita verkkosivuesityksessä lisätään pohjakartan päälle. (OpenJS Foundation n.d.a., n.d.b..) Dataa voi esittää hyvin erilaisissa muodoissa, tämän työn sovelluskokeiluissa on esimerkki lämpökarttakerroksen käytöstä.

3 TIETOKANTOJEN KÄYTTÖÖNOTTO

3.1 Indeksoinnin teoriaa

Tietokantahakuja voi tuntuvasti nopeuttaa indeksoinneilla perinteisissä tietokantajärjestelmissä, joissa tietueet säilötään riveittäin. Yksinkertaisimmillaan indeksoinnin tuotos, indeksi, voidaan kuvitella vektorina avain–arvo-pareja. Parin avain edustaa attribuutin arvoa ja parin arvo edustaa attribuutin sijaintia taulussa. Indeksin avain–arvo-parit järjestetään ja lisätään osaksi tietokantaa. Attribuutin arvojen sijaintien kartoittaminen taulussa tehdään yhtäsuuruus- tai arvojoukko-operaatioiden tapauksessa indeksin perusteella, ja kyselyehdon ulkopuoliset avaimet jätetään huomioimatta. Menettely on näin tavallista tietokantahakua todennäköisesti nopeampi, kun jokaisen taulun rivin läpi käymisen sijaan järjestelmän tarvitsee vain soveltaa jotakin hakualgoritmiä indeksiin. Indeksointia on havainnollistettu kuviossa 1. Tässä indeksi on jonorakenteessa, mutta tyypillisesti järjestelmissä käytetään tehokkaampia tietorakenteita, esimerkiksi puita.



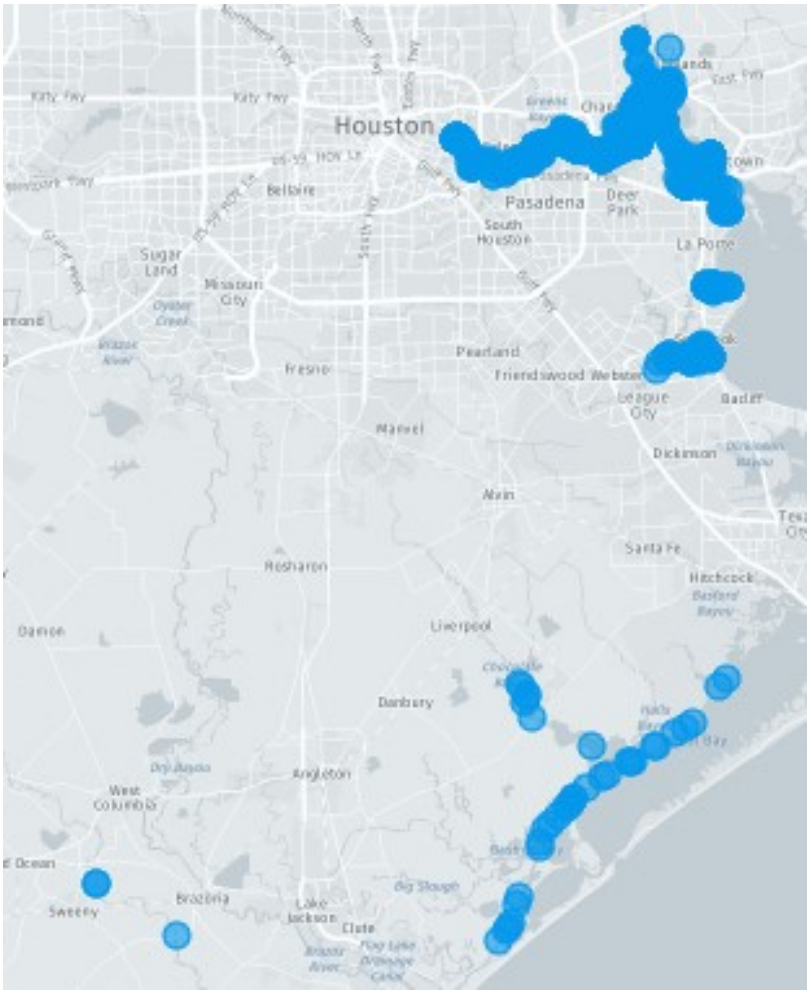
Kuvio 1: Havainnollistus jonoperustaisen indeksin toimintaperiaatteesta

Hakualgoritmit ovat yksinkertaisimmillaan peräkkäis- tai binäärihakuja, mutta joissakin tilanteissa puurakenteiden hyödyntäminen arvoalueiden haarukoinnissa indekseistä voi olla tehokasta. Puurakenteissa eteneminen muistuttaa puolitushakua: puun solmusta haaraudutaan seuraavaan solmuun haettavan arvon mukaisesti, kunnes tämä saavutetaan. B-puu (B-tree) on tietokantojen indeksoinnissa laajasti käytetty tietorakenne (Comer 1979,

Introduction). B-puun rakenne sen kasvaessa säilyy itsestään tasapainoisena, toisin kuin vaikkapa binääripuun rakenne (Comer 1979, The basic B-tree, Balancing).

3.2 AIS-data

Työtä varten hyödynnettiin vapaasti saatavilla olevaa AIS-dataa, liitteessä 1 on tästä näyte. AIS (Automatic Identification System) on laivojen seurantajärjestelmä, jota käytetään alusten tunnistamiseen ja sijaintitiedon välittämiseen (NATO Shipping Centre n.d.). Datakokoelma sisälsi erilaisten merialusten tuottamia AIS-lähetyksiä Meksikonlahden tietämiltä (kuva 1) koko vuoden 2020 ajalta. Data oli jäsenneltynä CSV-tiedostoihin niin, että yhdessä tiedostossa oli kokonaisen vuorokauden aikana vastaanotetut AIS-lähetykset. Yksittäisen tiedoston pakkaamaton koko oli n. 700–800 MB.



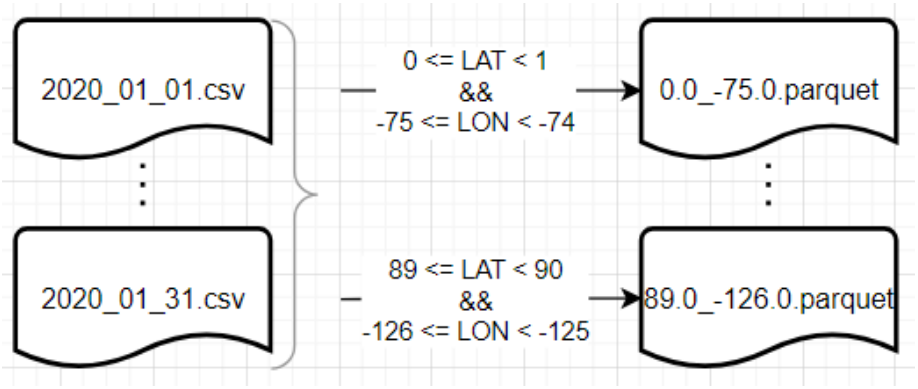
Kuva 1: AIS-lähetysten sijainnit kartalla

Teknologioita vertailtaessa datan sisällöllä ei tietokantakyselyiden suorituskykyjen mittaamisen näkökulmasta ole paljoa merkitystä. Aidosta sijaintidatasta oli kuitenkin hyötyä sovelluskokeiluissa, sillä AIS-datan käsitteleminen herätti kehitysideoita ja kysymyksiä datatyyppeihin ja merialusten esittämiseen liittyen. Dataa tietokantoihin siirrettäessä pyrittiin attribuuteille käytännöllisyyden rajoissa valitsemaan mahdollisimman taloudelliset tietotyypit. Nämä löytyvät järjestelmittäin lueteltuina liitteestä 2.

3.3 Parquet-migraatio

Päiväkohtaisesti jäsennellyistä CSV-tiedostoista luotiin aluksi vastaavia Parquet-tiedostoja. Parquet on sarakkeinen tiedostomuoto, jonka luomisessa hyödynnetään tehokkaita koodaamis- ja pakkaamisalgoritmeja (Apache Parquet n.d.a). AIS-datan tapauksessa Parquqet-tiedoston koko saatiin jopa alle viidennekseen vastaavan CSV-tiedoston koosta. Kokeiluissa Apache Arrow sekä DuckDB käyttivät tietokantana Parquet-tiedostoa.

Myöhemmin todettiin, että karttasovelluksen tapauksessa on hyödyllisempää kategorisoida data **koordinaattialueittain**. Tällä tavalla kiintolevyn lukemiskertoja voidaan karttasovelluksen käytön yhteydessä tuntuvasti vähentää. Data jaettiin yhden pituus- ja leveysasteen suuruisiin kokonaisuuksiin. Työssä päädyttiin käyttämään vain tammikuun 2020 AIS-lähetyksiä. Samassa yhteydessä päiväysmuotoiset aikaleimat muutettiin UNIX Epoch -sekunneiksi. UNIX Epoch -aikaleima ilmaisee, montako sekuntia arvon evaluointihetkellä on kulunut ajankohdasta 1.1.1970 klo 0:00 koordinoitua yleisaikaa (UnixTime.org n.d.). Koska dataa ei ollut jokaiselta maailman kolkalta, jätettiin valtaosa näistä yksittäisiä pituus–leveysaste-neliöitä vastaavista Parquet-tiedostoista luomatta. Prosessia on havainnollistettu kuviossa 2.



Kuvio 2. Tietueiden kategorisointi pituus- ja leveysasteiden mukaan CSV–Parquet-migraatiossa

3.4 MongoDB- ja PostgreSQL-migraatiot

Tammikuun tietueet lisättiin CSV-tiedostoista myös MongoDB- ja PostgreSQL-tietokantoihin. MongoDB:n kohdalla pituus- ja leveysasteattribuuteista luotiin yhdistetty 2d-indeksi, jonka toiminta perustuu alueittaisiin hajautusarvoihin: järjestelmä jakaa kaksiulotteisen kartan neljään osaan, joilla kullakin on oma hajautusarvonsa. Prosessi toistetaan haluttavan tarkkuuden mukaan rekursiivisesti niin, että alineljännesten lopulliset arvot saadaan aina lisäämällä näiden arvot vastaavan yläneljänneksen sisällä tämän arvon perään. Lopuksi hajautusarvot indeksoidaan B-puu–algoritmiä käyttäen. (MongoDB, Inc. n.d.c..)

PostgreSQL:n yhteyteen asennettiin PostGIS-lisäosa. Leveys- ja pituusasteesta muodostettiin geometria-attribuutti, jolle luotiin GiST-indeksi. Kahden järjestelmän kaikki indeksit ovat lueteltuina liitteissä 3 ja 4. Sekä MongoDB että PostgreSQL käyttävät B-puuta tavallisen indeksoinnin yhteydessä.

4 TIETOKANTOJEN VERTAILU

Tässä luvussa keskitytään tietokantajärjestelmien vertailemiseen keskenään. Järjestelmät ovat toimintaperiaatteiltaan erilaiset, pääasiassa muistinhallintaan liittyen. Koeasetelmassa kaikki AIS-data oli Apache Arrow - ja DuckDB-järjestelmille saatavilla yhdessä noin 222 MB Parquet-tiedostossa. Tämä voidaan katsoa kyseisten järjestelmien eduksi suorituskykyä lievästi vääristävänä seikkana, kun käytännön sovelluksessa muistiin saatettaisiin joutua lukemaan useampia tiedostoja. MongoDB- ja PostgreSQL-järjestelmien tapauksessa kyselyt kohdistettiin kummassakin tapauksessa yhteen tietokantaan.

4.1 Ennakkovalmistelut

Vertailut toteutettiin Python-kielellä, versiolla 3.11.1. Vaikka eri järjestelmillä on eri tavat hyödyntää välimuistia ja toisaalta lukea tietoa kiintolevyltä, oli ne vertailua varten pyritty asettamaan samaan asemaan. Tämä huomioitiin seuraavilla menettelyillä:

- Yhteys Parquet-tiedostoon (Apache Arrow ja DuckDB) taikka tietokantaan (MongoDB ja PostgreSQL) avattiin kerran ennen mittauksia ja yhteys suljettiin vasta mittausten jälkeen.
- Mitattavia tietokantakyselyitä edelsi kaksi lämmittelykierrosta, näin huomioitiin tietokantajärjestelmän mahdollisuus hyödyntää välimuistia. Lämmittelykierrosten tuloksia ei huomioitu mittauksissa
- Kyselyt tehtiin yhdelle tietokannalle kerrallan, jolloin suuremmat datamassat eivät ylikirjoittaneet toisen tietokannan välimuistiin tallettamaa dataa.
- Tietokannat eivät hyödyntäneet kyselyissä aiemmin muodostettua tietorakennetta, vaan tietueet haettiin hakukriteerien perusteella aina uudelleen.

Käyttöjärjestelmän käynnissä olevien prosessien aiheuttama kuorma pyrittiin pitämään samana sulkemalla käynnissä olevat etualan prosessit skriptien ajon

ajaksi. Skriptit ajettiin Python-tulkissa Windows 10:n komentokehotteen kautta. Eri järjestelmille kohdistettujen kyselyiden samankaltaisuutta arvioitiin spatiotemporaalisten hakujen tapauksessa palautettujen rivien lukumäärää mittaamalla. Analyyttisten kyselyiden tapauksessa valvottiin, että tulokset ovat samoja järjestelmästä riippumatta. Tätä varten tuloksista luotiin tarvittaessa erikseen yhdenmukaiset esitykset Pandas-kirjaston DataFrame-tietorakenteisiin. Yksittäisen tiedonhaun kestoksi laskettiin tietokantakyselyyn sekä tuloksen tietorakenteeseen saattamiseen kuluneiden aikojen summa. Kukin kysely toistettiin 10 kertaa, ja lopputuloksena kirjattiin kyselyihin keskimäärin kulunut aika.

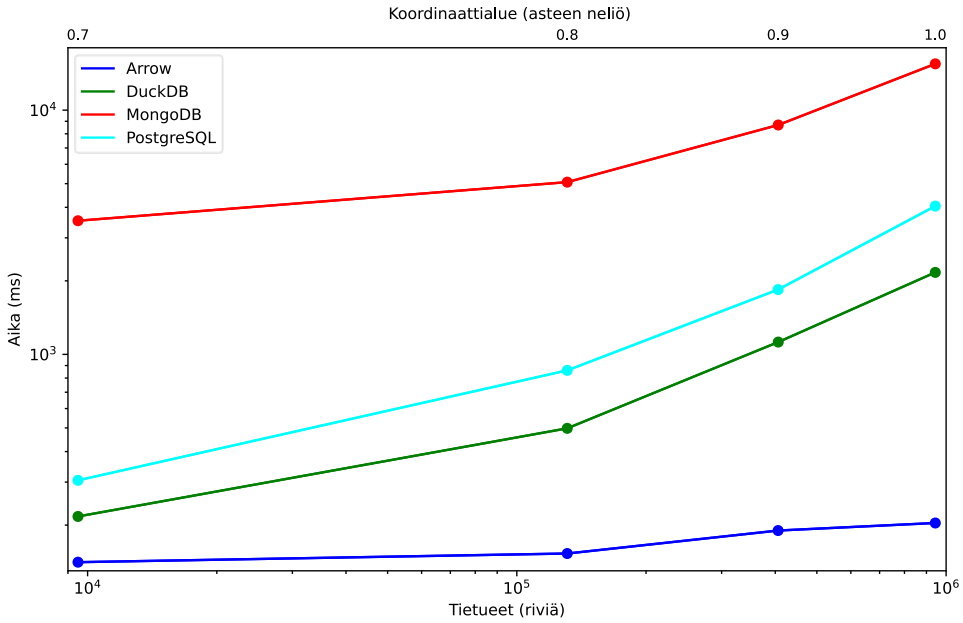
4.2 Spatiotemporaaliset kyselyt

Spatiotemporaalisen haun kokeiluissa nopeuksia mitattiin eri aikajänteiden ja koordinaattialueiden parametrien permutaatioilla. Pienimmät ja nopeimmat taulukoiduista kyselyistä palauttivat muutaman tuhatta tietuetta alle sekunnissa. Suurimmat kyselyt palauttivat suunnitellusti kaikki tietokannan tietueet, näissä kyselyissä ei kuitenkaan eksplisiittisesti pyydetty kaikkia tietueita. Prosessia havainnollistava pseudokoodi:

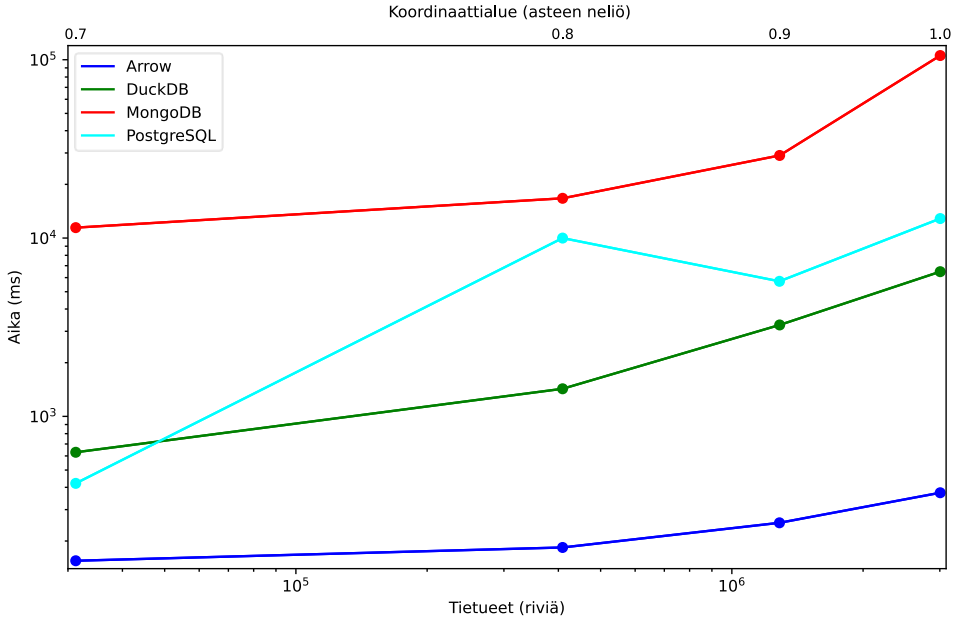
```
def benchmark(T, A):
  for _ in range(2): # warm-up
    data = db.query(T, A)
  time_sum = 0
  for _ in range(10): # benchmarking
    t1 = time()
    data = db.query(T, A)
    t2 = time()
    time_sum += t2 - t1
    verify(data.to_pandas())
  return time_sum / 10

db.connect() # Load Parquet OR init DB conn
for T in time_range:
  for A in coordinate_range:
    time_avg = benchmark(T, A)
    save_results(T, A, time_avg)
db.close()
```

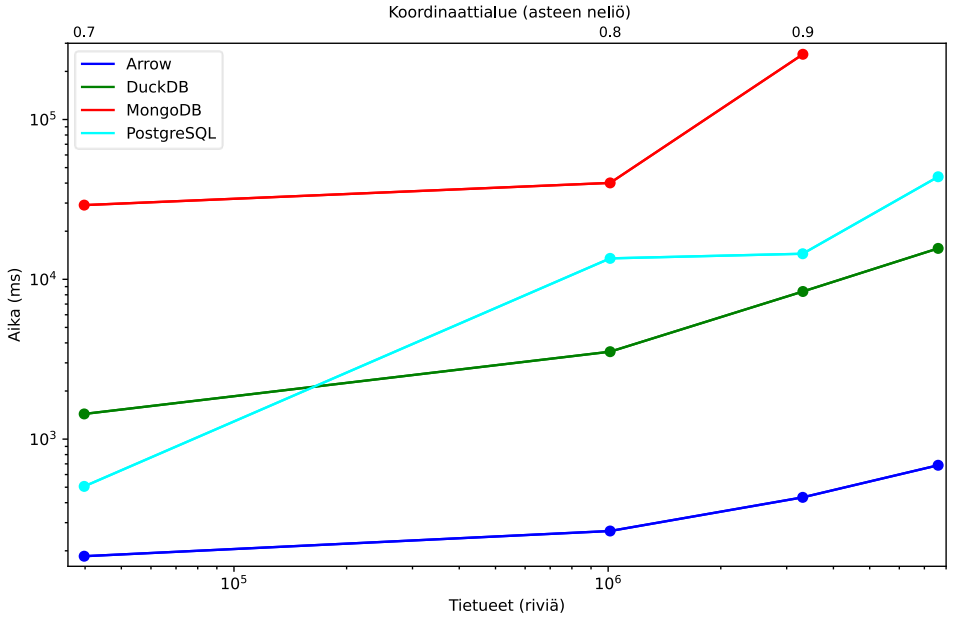
Kuvioissa 3, 4 ja 5 ovat kuvattuina kunkin kymmenen kyselyn keskimääräiset kestot ja niiden riippuvuudet aika- ja alueparametrien määrittämistä laajuuksista. *Koordinaattialue* kertoo, kuinka suuri alue kyselyissä välitettiin parametrinä kussakin kymmenen mittauksen tapahtumassa. *Tietueet*-akseli ilmaisee kyselyn palauttamien tietueiden lukumäärän. *Aika* tarkoittaa kyselyyn kulunutta aikaa. Kuviossa 3 ovat mittaustulokset neljän vuorokauden aikajänteellä, kuviossa 4 vastaavasti 12 vuorokaudelta ja kuviossa 5 koko kuukauden ajalta. Kuvioon 5 ei ole sisällytetty MongoDB:n suurimman tietuemäärän käsittävää kyselyä, sillä kysely aikakatkaistiin useamman tunnin odottelun jälkeen.



Kuvio 3: Kyselyiden kestot (alhaisempi on parempi) koordinaattialueittain neljän vuorokauden aikajänteellä



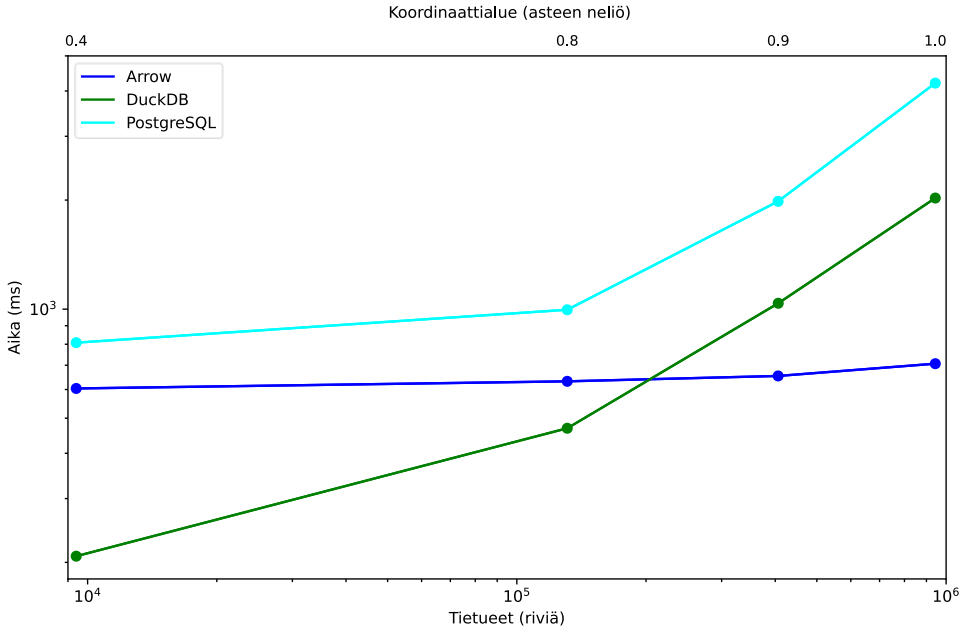
Kuvio 4: Kyselyiden kestot koordinaattialueittain 12:n vuorokauden aikajänteellä



Kuvio 5: Kyselyiden kestot koordinaattialueittain 32:n vuorokauden aikajänteellä (MongoDB:n kaikkien tietueiden hakuaikaa ei tiedetä)

Kuviossa 6 on kylmäkäynnistykset sisältävien mittausten tulokset kuvion 3 mittauksia vastaavilla parametreilla. Kylmäkäynnistyksellä tarkoitetaan, että yhteys tietokantaan avataan ja suljetaan kyselykohtaisesti. Kuvioita 3 ja 6 vertailemalla huomataan, mikä vaikutus tietokantayhteyden avaamisella on kyselyn tehokkuuteen. Tämä koejärjestely toimii välimuistin hyödyntämisen

osalta vastaesimerkkinä tämän luvun alustuksessa esitettyjen periaatteiden mukaisille järjestelyille. Mittausten eroavaisuudet korostuvat pienemmillä datamäärillä: Yhteyden avaamiseen ja sulkemiseen kuluva aika pysyy suunnilleen vakiona, joten tämä aika on suhteessa kyselyn kokonaisaikaan suurempi aina pienemmillä datamäärillä.



Kuvio 6: Mittaustulokset neljän vuorokauden aikajänteellä (ei koske MongoDB:a): Yhteys tietokantatiedostoihin avattiin ja suljettiin kyselykohtaisesti (kylmäkäynnistys)

Kyselyiden tuottamien tietueiden lukumäärissä oli enimmillään alle promillen poikkeamia. Hakutuloksia keskenään vertailtaessa tämän huomattiin johtuvan eroavaisuuksista arvojen pyöristystarkkuuksissa. Apache Arrow jätti esimerkiksi sisällyttämättä hakutulokset pituusasteväliltä $[-96{,}0, -95{,}1]$, mikäli tietueen vastaava arvo oli -95.099998. Kyseinen tietue löytyi kuitenkin DuckDB:n ja MongoDB:n hakutuloksista.

4.3 Analyyttiset kyselyt

Tietokantajärjestelmien suoriutumista analyyttisistä kyselyistä arvioitiin seuraavalla tavalla: Ensin muodostettiin karttasovelluksen käyttötarkoitusta ajatellen mielekkäitä kyselyehtoja SQL-kielellä. Tämän jälkeen kysely sovitettiin

vastaamaan eri järjestelmien syntakseja. Kyselyitä laadittiin kolme erilaista, ja neljän järjestelmän suoriutumista arvioitiin jälleen kyselyyn vastaamiseen kuluneen ajan perusteella. Tietokantoja pyydettiin mm. esittämään tietyn aikavälin AIS-lähetysten laivan tyypin mukaan muodostettuja joukkoja, joiden sisältämien tietueiden arvoilla edelleen tehtiin laskentaa. Tällaista oli esimerkiksi laivajoukon keskinopeus, suurin syväys ja nopeuden keskihajonta. Kuviossa 7 on 7.1.2020 klo 0:00 alkaen tasan viikon aikavälin tietueiden käsittelyn tulokset. Alla on lueteltu tietokantakyselyt 1–3 SQL-kielelle muotoiltuna. Vihreällä (kysely 2) ja sinisellä (kysely 3) tekstillä osoitetaan lisäykset edeltävään kyselyyn.

Kysely 1:

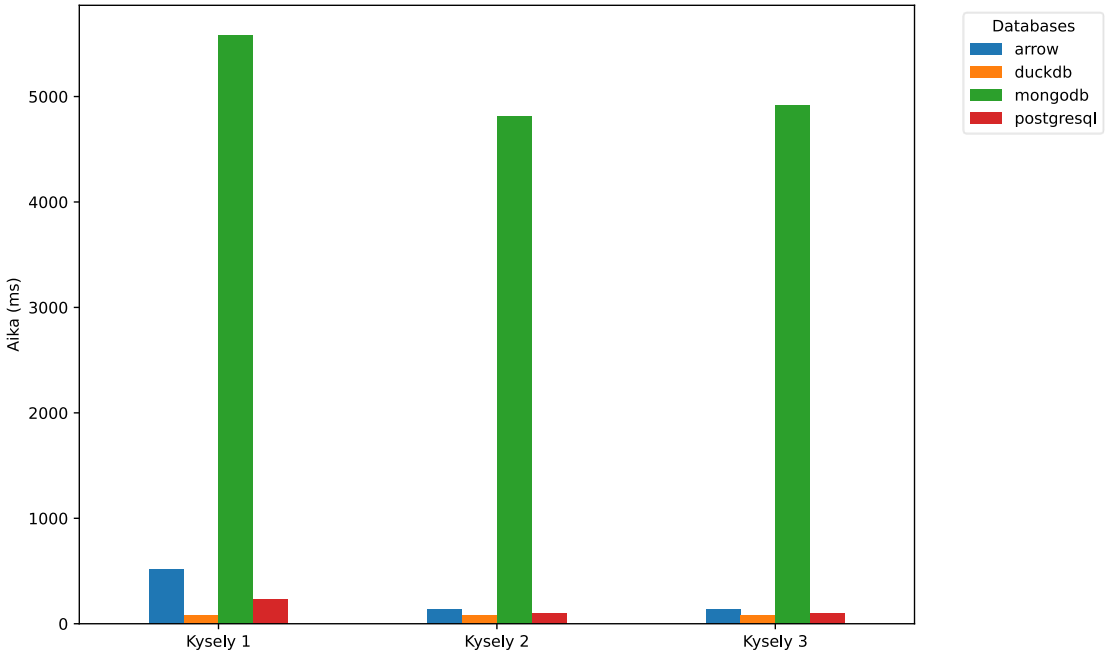
```
SELECT VesselType,
       AVG(SOG) AS AverageSpeed,
       MAX(Draft) AS MaximumDraft
FROM records
WHERE Timestamp BETWEEN 1578355200 AND 1578960000
GROUP BY VesselType
ORDER BY VesselType;
```

Kysely 2:

```
SELECT VesselType,
       AVG(SOG) AS AverageSpeed,
       MAX(Draft) AS MaximumDraft,
       MIN(COG) AS MinimumCOG
FROM records
WHERE Timestamp BETWEEN 1578355200 AND 1578960000
      AND Draft > 0
GROUP BY VesselType
ORDER BY VesselType;
```

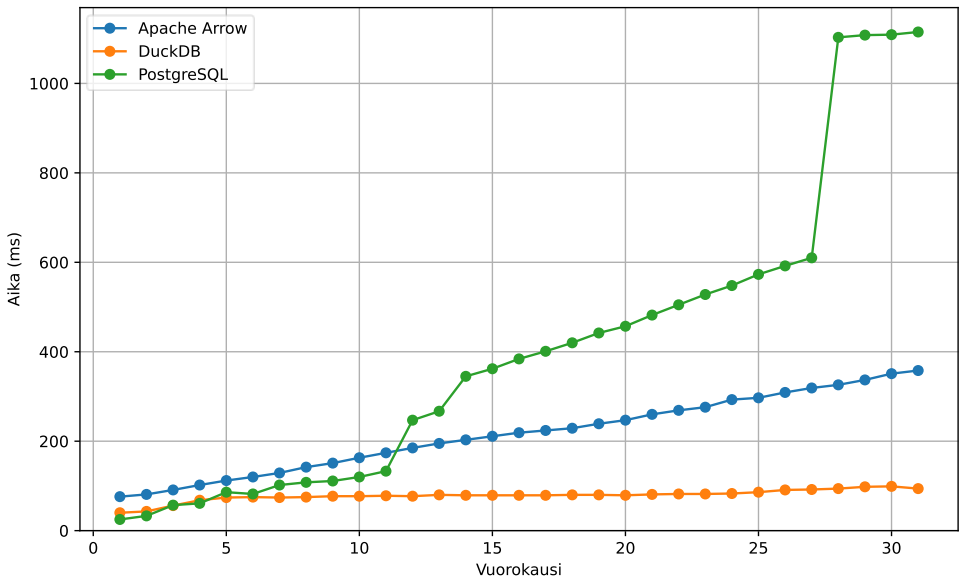
Kysely 3:

```
SELECT VesselType,
       AVG(SOG) AS AverageSpeed,
       MAX(Draft) AS MaximumDraft,
       MIN(COG) AS MinimumCOG,
       STDDEV_SAMP(SOG) AS SpeedStdDev
FROM records
WHERE Timestamp BETWEEN 1578355200 AND 1578960000
      AND Draft > 0
GROUP BY VesselType
HAVING AVG(SOG) > 2
ORDER BY VesselType;
```



Kuvio 7: Analyyttisten kyselyiden tulokset viikon aikaväliltä alkaen 7.1.

MongoDB karsittiin pois tässä vaiheessa, ja muiden järjestelmien suorituskykyä mitattiin edelleen. Kyselyehdon aikajännettä kasvatettiin vuorokausi kerrallaan, alkaen 1.1.2020 klo 0:00 yhdestä vuorokaudesta ja lopulta käsittämään koko kuukauden AIS-lähetykset. Muutoin kyselyehto vastasi kyselyä 3. PostgreSQL:n suorituskyky heikkeni selvästi päivien 27 ja 28 välillä, joten tuloksia tarkasteltiin lähemmin `EXPLAIN ANALYZE` -käskyllä. Tulostuksista ilmeni, että järjestelmä lakkasi käyttämästä indeksiä päivästä 28 eteenpäin. Tätä ennen järjestelmä hyödynsi attribuuttien aikaleima, syväys ja aluksen tyyppi yhdistettyä indeksiä. Mittaustulokset on esitetty kuviossa 8.



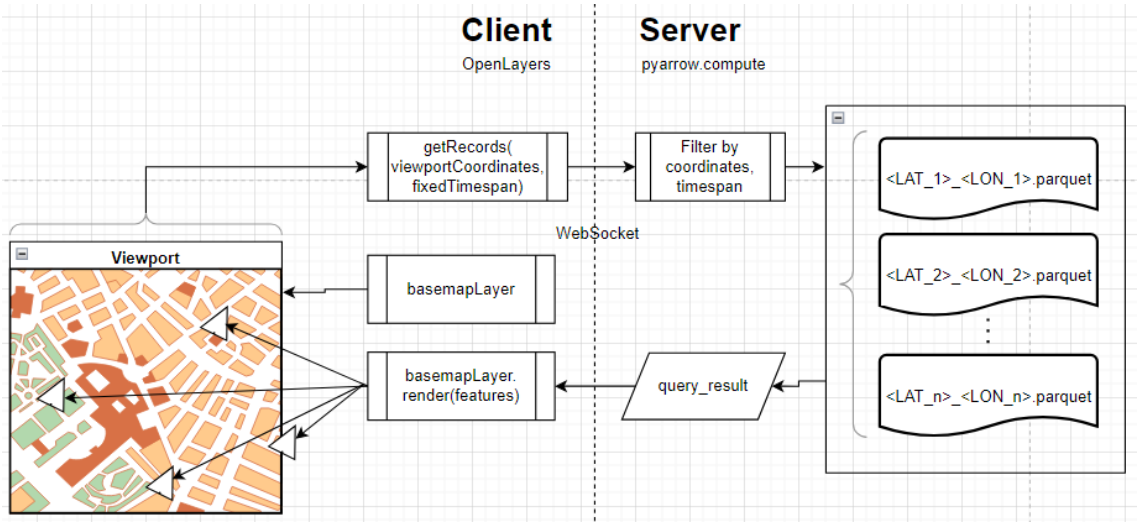
Kuvio 8: Analyyttisen kyselyn tulokset (MongoDB:a ei sisällytetty). Kyselyehdon aikajännettä kasvatettiin vuorokausi kerrallaan

5 SOVELLUSKOKEILUT

Työssä luotiin kaksi kuvitteellista, toimeksiantajan tarpeiden näkökulmasta hyödyllistä käyttötapausta: spatiotemporaalinen sovellus ja lämpökarttasovellus. Käyttötapaukset ovat ehdotelmia eri tavoista lähestyä aikasarjaisen sijaintidatan esittämisen ongelmaa. Molemmat sovellukset käyttävät OpenStreetMap-pohjakarttaa liitettynä OpenLayers-kirjaston kerrosolioihin. Sovelluksissa käytetään AIS-lähetyksiä yhden viikon ajalta 1.1.2020 alkaen. Palvelinohjelmistot toteutettiin Python-kielellä. Selainohjelmistot luotiin Node-ympäristöön ja paketoitiin WebPack-työkalulla.

5.1 Spatiotemporaalinen sovellus

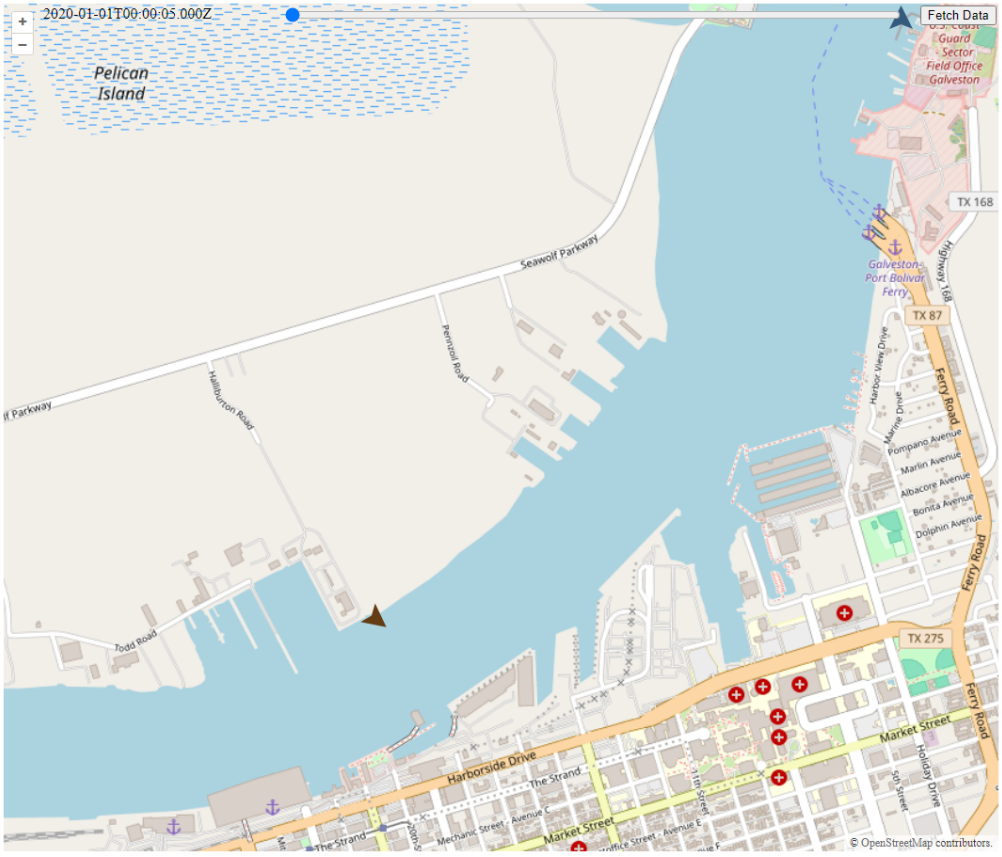
Spatiotemporaalisella web-sovelluksella voidaan tarkastella alusten mennyttä liikehdintää kartalla. Käyttöliittymän *Fetch Data* -painike lähettää palvelimelle pyynnön välittää karttanäkymää vastaavan koordinaattialueen AIS-lähetykset. Prosessi on esitetty kuviossa 9. Palvelimella PyArrow Table -tauluun luetaan kaikki koordinaattialueen piiriin lukeutuvat tietokantatiedostot yksitellen. Taulusta suodatetaan hakukriteeriä vastaavat tietueet PyArrow Compute -kirjaston avulla tuotetuilla binäärimaskeilla. Suodatetut tulokset lisätään samaan listaan. Lopuksi listan alkiot yhdistetään samaan tauluun ja lähetetään tavuvirtana kutsujalle. Data välitetään Arrow IPC -muotoisena (inter-process communication) ja muunnetaan selaimen JavaScript-ympäristössä jälleen Arrow Table -tauluksi. Tiedonsiirto toteutettiin WebSocket:n välityksellä.



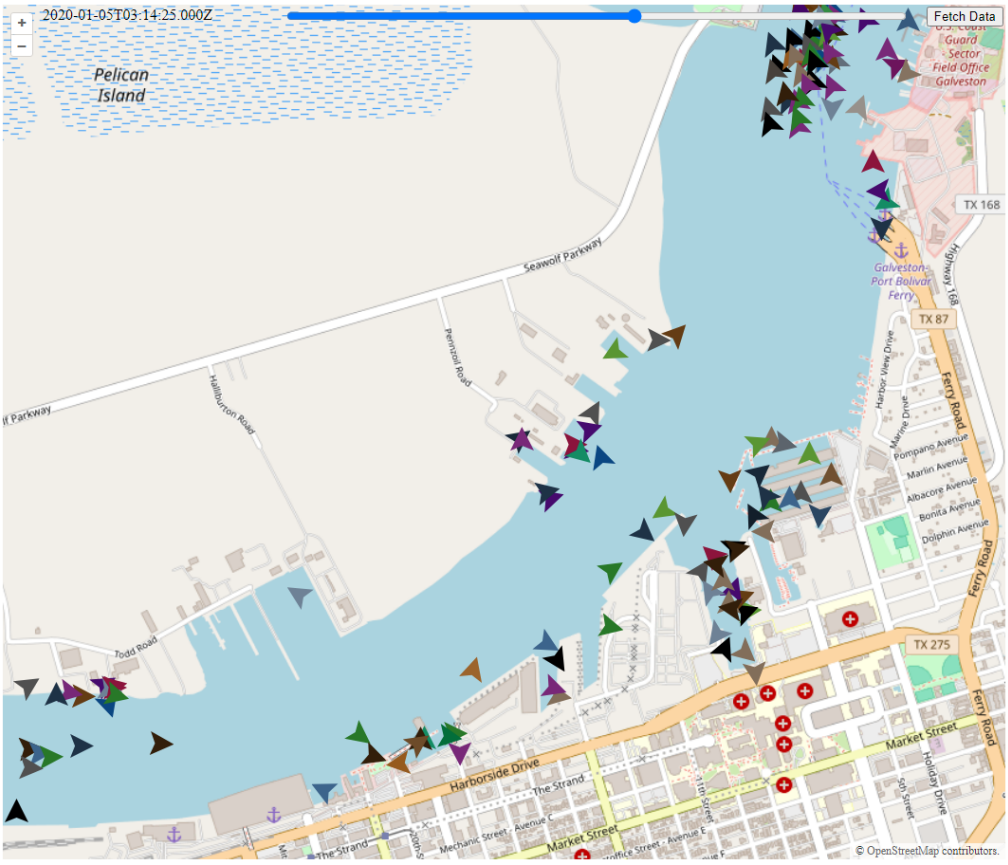
Kuvio 9: Spatiotemporaalisen karttasovelluksen tiedonsiirtoprosessi

Selaimessa AIS-lähetyksistä luodaan karttaolioita, jotka yksilöidään usean aluksen tunnistetiedon yhdistelmällä sekä värikoodataan lähetysten MMSI-tunnusten (Maritime Mobile Service Identity) mukaan. Lähetykset ryhmitellään aikaleimoittain, ja saman ryhmän tietueet tulevat näkymään kartalla samaan aikaan. Alukset tuottavat AIS-lähetyksiä eri aikoihin ja eri aikavälein. Näin ollen aikaryhmien sisältämiä lähetyksiä esitettäessä kartalla oliot katoilisivat sekä ilmestyisivät näkymässä. Tästä syystä kerran aikaryhmään lisätyn olion sama lähetys kopioidaan tuleviin aikaryhmiin, kunnes samalta oliolta on saatavilla uusi lähetyspäivitys.

Selainkäyttöliittymässä näytetään OpenLayers:n karttakerros, jonka päälle ryhmistä koostuvan aikasarjan visuaalinen esitys lisätään. Aikaulottuvuutta voi selata karkeasti vierityspalkilla sekä hienosäätää Ctrl + hiiren vierityspainikkeella. Kuvassa 2 on näytetty aikasarjan ensimmäinen ryhmä. Kuvassa 3 on näytetty ryhmä sarjan keskivaiheilta, kun alusten lähetyksiä on kumuloitunut tuleviin aikaryhmiin. Karttanäkymät ovat Galveston Bay:n eräästä satama-alueesta. Sovellus tarjoaa kuvanauhoitusta muistuttavan toiminnon alusten liikehdintöjen tarkkailuun.



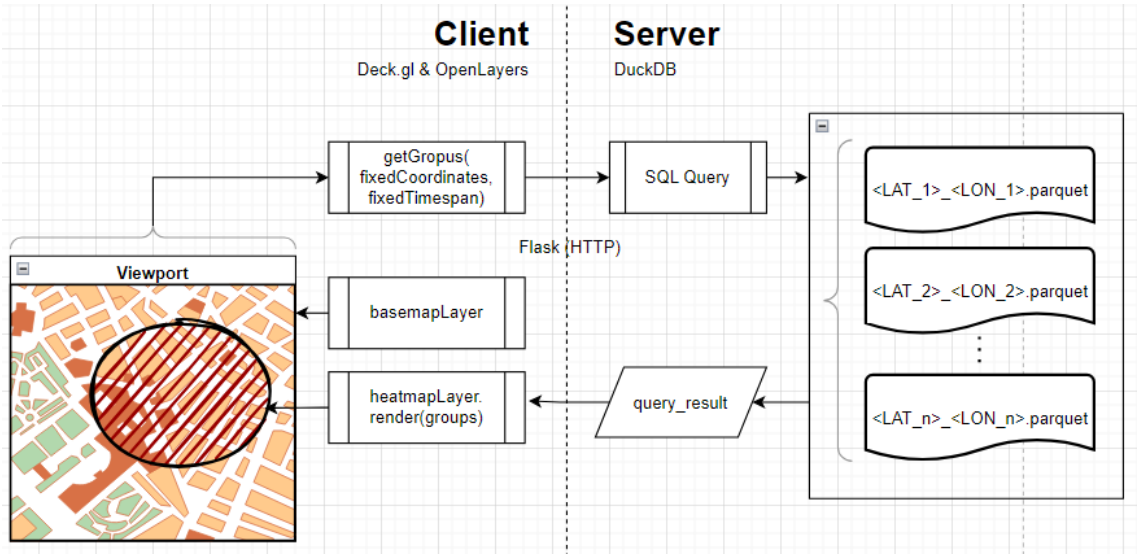
Kuva 2: Karttasovelluksen näkymä aikasarjan ensimmäisestä ryhmästä



Kuva 3: Karttasovelluksen näkymä ryhmästä aikasarjan keskivaiheilla

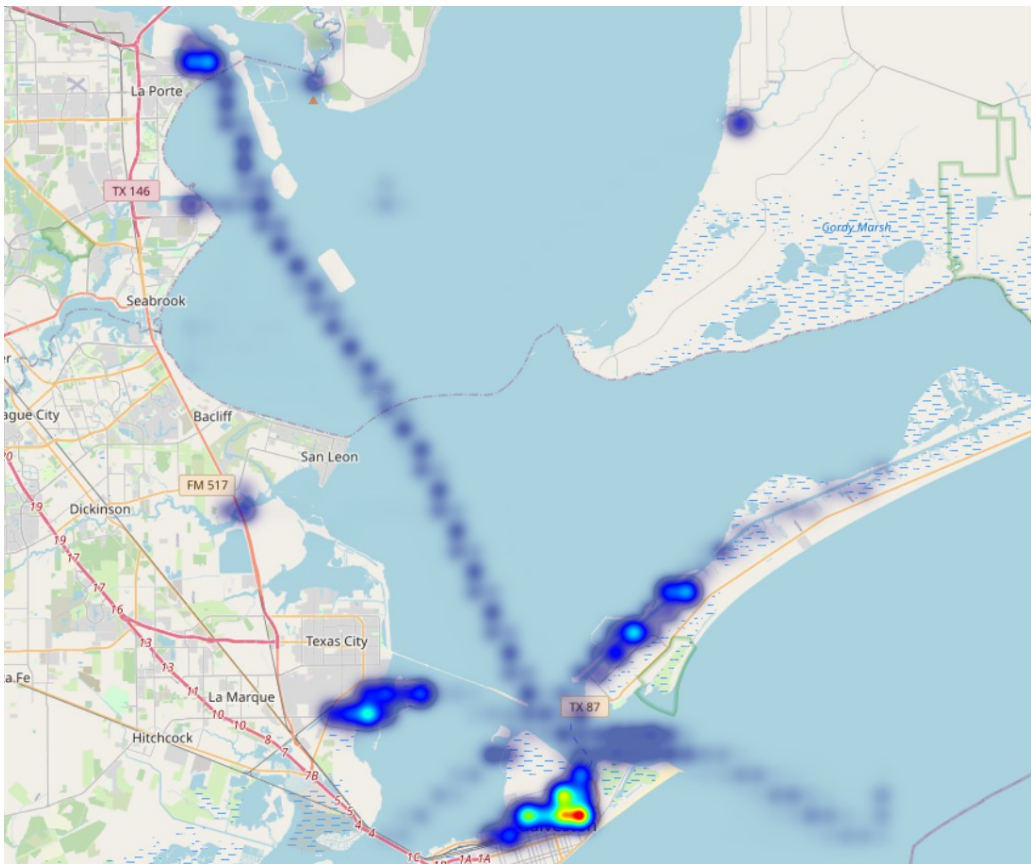
5.2 Lämpökarttasovellus

Lämpökarttasovellus näyttää erään $0{,}4 \times 0{,}4$ leveys- ja pituusasteen suuruisen alueen AIS-lähetysten sijainnit lämpökarttakerroksena pohjakartan päällä. Sovelluksen käynnistämisen yhteydessä palvelimelta haetaan data kiinteillä hakukriteereillä. Palvelimella kerätään listaan hakukriteerien koordinaattialuetta vastaavien tietokantatiedostojen nimet. Hakukriteerien paikka- ja aikatiedot liitetään SQL-kyselylauseen parametreiksi. Yksittäinen palautettava tietue SQL-kyselyissä määräytyy siten, että kyselyssä luodaan koordinaattialueiden mukaisia joukkoja ja näiden yhteyteen liitetään kyseisen alueen AIS-lähetysten summa. Kysely toistetaan jokaiselle koordinaattialuetta vastaavalle tietokantatiedostolle. Tulokset muunnetaan GeoJSON-muotoon ja lähetetään kutsujalle Flask-palvelimen (HTTP) välityksellä. Prosessi on esitetty kuviossa 10.

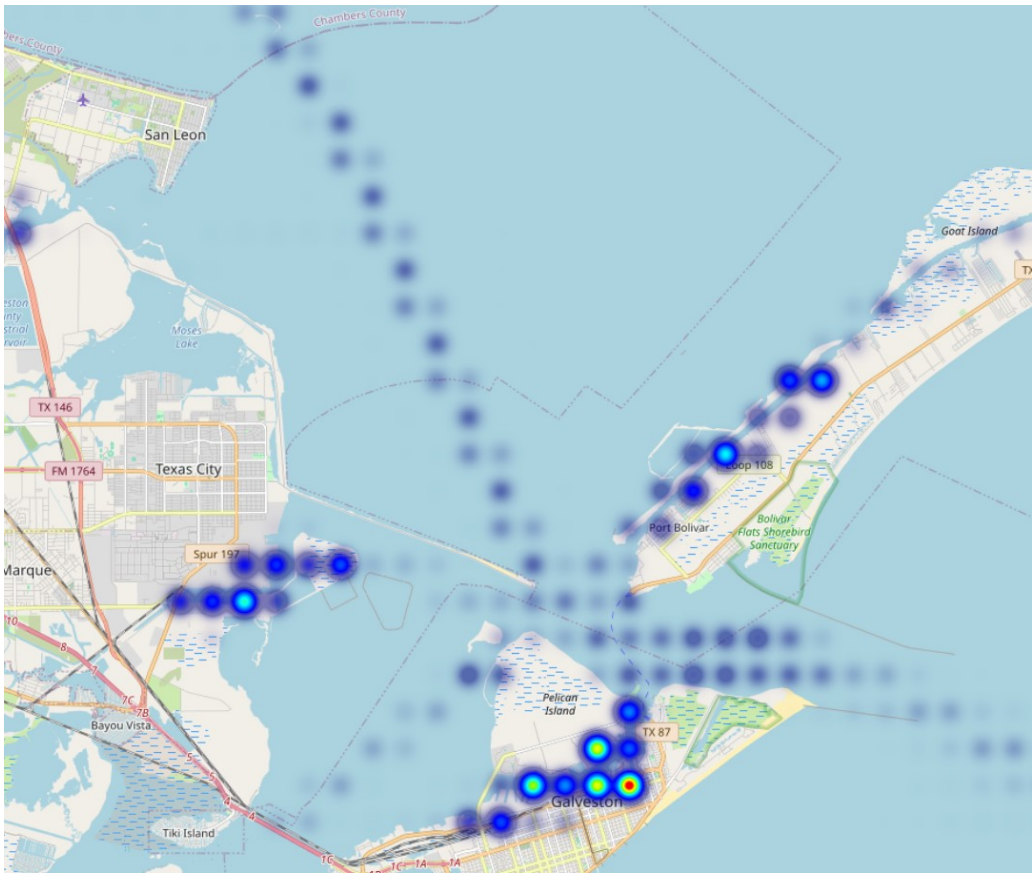


Kuvio 10: Lämpökarttasovelluksen tiedonsiirtoprosessi

Selaimessa luodaan lämpökarttakerros, Deck:n HeatmapLayer -olio, jolle GeoJSON-olioista välitetään koordinaatit sekä näitä vastaavat AIS-lähetysten lukumäärät. Kerros liitetään osaksi Deck-oliota, jolle lämpökarttakerros osoitetaan ja joka lopulta liitetään osaksi HTML-dokumentissa määrättyä karttaelementtiä. Selainikkunan näkymästä on esimerkki kuvassa 4. Riittävän kaukaa tarkasteltaessa datapisteet sulautuvat paikoin yhtenäiseksi lämpökuvioksi. Kuvassa 5 on vastaava näkymä karttasovelluksessa lähennettynä. Tässä datapisteet erottuvat selkeämmin omiksi joukoikseen.



Kuva 4: Lämpökarttakerros näyttää AIS-lähetysten sijainnit viikon aikavälillä



Kuva 5: Kuvaa vastaava näkymä lähempää tarkasteltuna

6 POHDINTA

Opinnäytetyössä valittiin vertailuun neljä toimintaperiaatteiltaan erilaista tietokantajärjestelmää, joiden suoriutumista mitattiin yksinkertaisen datan suodattamisen sekä toisaalta monimutkaisempien analyyttisten kyselyiden näkökulmasta. Datana käytettiin merialusten AIS-lähetyksiä. Työssä kehitettiin lisäksi kaksi web-karttasovellusta, joissa OpenLayers-kirjastojen avulla verkkosivuun liitettiin OpenStreetMap:n maailmankartta. Spatiotemporaalisessa karttasovelluksessa näytettiin laivojen sijainteja kartalla kelattavana aikasarjana. Lämpökarttasovelluksessa näytettiin AIS-lähetysten kertymät toisiinsa sulautuvina tarkastelupisteinä. Sovellusten tetokantajärjestelminä toimivat Apache Arrow ja DuckDB, jotka tarjosivat dataa yksittäin muistiin mahtuvista Parquet-tiedostoista.

6.1 Relaatiotietokantojen vertailu

PostgreSQL suoriutui analyyttisistä kyselyistä DuckDB:a nopeammin pienillä aikajänteillä. Suuremmilla aikajänteillä suorituskyky alkoi heiketä, erityisesti kun järjestelmä lakkasi hyödyntämästä aikaleiman, syväyksen ja aluksen tyypin yhdistävää indeksiä. Jatkokehitystä ajatellen voisi analyyttisten kyselyiden näkökulmasta olla hyödyllistä pohtia seuraavia kysymyksiä:

1. Millaisia ovat karttasovelluksessa hyödylliset analyyttiset kyselyt?
2. Kuinka suuriin datamääriin kyselyt kohdistuvat?
3. Millaiset indeksit tehostavat näitä kyselyitä parhaiten PostgreSQL-järjestelmässä?
4. Missä määrin indeksien viemä tila on suorituskyvyn paranemisen arvoista?
5. Kumman järjestelmän tuoma tasapaino tilankäytön ja suorituskyvyn välillä on arvokkaampi, DuckDB:n vai PostgreSQL:n?

Raasveldt & Mühleisen (2019) ovat tunnistaneet joitakin edellytyksiä prosessinsisäisille OLAP-järjestelmille, joista osa on hyvä huomioida myös

karttasovellusta kehittäessä. Tietokantajärjestelmän tulee ongelmatilanteessa, esimerkiksi muistin loppuessa, voida toipua ajamatta alas isäntäprosessia. Koska järjestelmää tulee voida ajaa isäntäprosessissa tämän ympäristöstä riippumatta, kannattaa sen olla rakenteeltaan keveä ja riippumaton ulkoisista kirjastoista. (Raasveldt & Mühleisen 2019, Introduction.)

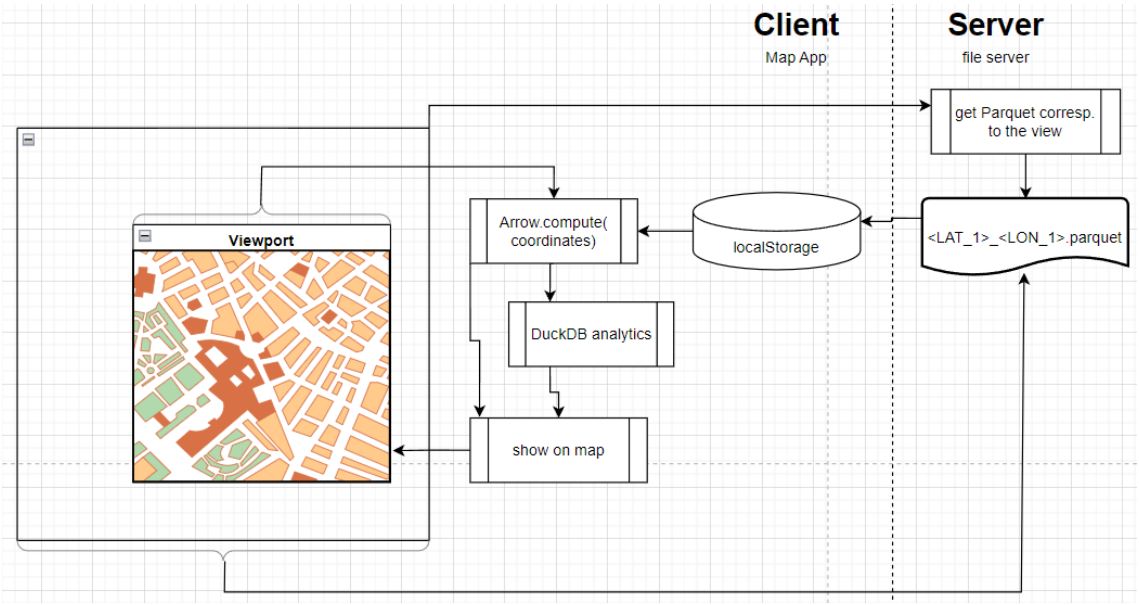
Eräs toimeksiantajan esittämä ongelma liittyy jo olemassa olevien tietokantojen hyödyntämisen mahdollisuuksiin. DuckDB:ltä on saatavilla PostgreSQL-tietokannan lukemisen mahdollistava laajennus, Postgres Scanner (Mühleisen 2022, Introduction). Tämä hyödyntää kyselyissä tietokannan palvelinprotokollan tarjoamaa binääristä tiedonsiirtotapaa (Mühleisen 2022, Implementation). Scanner:n avulla voitaisiin hyödyntää aiemmin PostgreSQL-tietokantaan lisättyä dataa sellaisenaan ilman erillistä kopioimista toiseen tietokantaan. Suorituskyky on edelleen kuitenkin monin verroin parempi DuckDB:n omalla tietokantajärjestelmällä (Mühleisen 2022, Performance). PG-Storm (n.d.) on myös kehittänyt työkalun PostgreSQL-hakutulosten kirjoittamiseksi Arrow:n Record Batches –taulukkomuotoon (PG-Storm n.d., Using Pg2Arrow). Toinen toimeksiantajaa kiinnostanut jatkotutkimuksien aihe on, miten tietokannat suoriutuvat useista samanaikaisista kyselyistä.

6.2 Parquet-tiedostot tietokantana

Apache Arrow:n Compute-rajapinta osoittautui erityisen tehokkaaksi yksinkertaisessa datan suodattamisessa, kun tietueita pyydettiin tietyltä aikaväliltä ja koordinaattialueelta. DuckDB ja PostgreSQL suoriutuivat parhaiten kyselyistä, jotka sisälsivät datan keräämistä ja laskentaa. Tällaiset kyselyiden toteuttaminen oli myös luontevaa SQL-kyselykielellä. Vertailussa sekä Arrow että DuckDB käyttivät tietokantanaan samoja Parquet-tiedostoja ja näin ollen myös samankaltaista ohjelmalogiikkaa tiedostojen kyselyihin sisällyttämisessä. Karttasovelluksen sijaintitietojen data-arkkitehtuurin voisi hyvin rakentaa Parquet-tiedostojen varaan. Arrow ja DuckDB kumpikin sekä suodattavat ja koostavat dataa hyvin, mutta erityisen optimaalisen tuloksen saa hyödynnettäessä kummankin vahvuuksia.

Parquet-tiedostojen suodattaminen loppukäyttäjän selainmoottorilla on Arrow-kirjastojen avulla myös mahdollista. Koordinaattialueittain jaetun datan yhteydessä tästä menetelmästä voi olla hyötyä, kun käyttäjä selailee karttaa tiedostokohtaisen alueen sisällä. Seuraavat kyselyt voidaan tehdä paikallisesti käyttäjän laitteistolla niin kauan, kuin kyseltävä tiedosto vastaa näkymää kartalla. Seuraava palvelinkysely tehdään vasta, kun tarvitaan toista näkymää vastaava tiedosto. Näin voidaan vähentää palvelinkyselyiden määrää. Haittapuolena menettelyssä on, että lähetettävä tiedosto sisältää lähes väistämättä ylimääräistä dataa. Datan suodattaminen tapahtuu loppukäyttäjän selaimessa, jolloin menetetään tietokantapalvelimella tehtävän nopeamman laskennan hyöty. Suodatetulla datalla voidaan tehdä edelleen esimerkiksi analytiikkaa. Ehdotettua prosessia on havainnollistettu kuviossa 11.

Kuvio 11: Tiedonsiirtoprosessi, jossa Parquet-tiedostoja käsitellään loppukäyttäjän selaimessa



6.3 Tietokanta osana asiakasohjelmistoa

Kuvion 11 mukaista selainmoottorin ympärille rakennettua dataekosysteemiä voi vahvistaa entisestään, kun hyödynnetään WebAssembly:n tuomat mahdollisuudet käyttää paikallista laitteistoa tehokkaasti. WebAssembly (Wasm) on modernien verkkoselainten tukema matalan tason ohjelmointikieli. Sen avulla voidaan kääntää sovelluksia eri ohjelmointikielillä verkkoselaimella ajettavaksi.

Eräs merkittävä WebAssembly:n piirre on, että sovelluksia voi ajaa niiden alkuperäisiin ajoympäristöihin verrattavilla nopeuksilla. WebAssembly toimii tiedon välittäjänä selaimen JavaScript-ympäristön ja sovellusten välillä, Wasm-kirjastojen rajapintoja käyttäen. (Mozilla Foundation n.d..)

DuckDB-Wasm on verkkoselaimessa tehtävää datan analysointia varten kehitetty järjestelmä (Kohn, Moritz, Raasveldt, Mühleisen, & Neumann 2022, Introduction). DuckDB-Wasm on DuckDB:n WebAssembly-kielelle käännetty variantti. Järjestelmä hyödyntää Arrow:n yhdenmukaista datansiirtoprotokollaa eri osien välillä. Tietokantahaun tulokset lisätään Record Batch -tietorakenteisiin. DuckDB on kehitetty C++-kielellä, joten DuckDB-Wasm serialisoi datan Arrow IPC -tavuvirtoina C++-kielellä. Tämän jälkeen DuckDB-Wasm taas lukee tulokset Wasm-pinosta JavaScript:lla selainohjelmiston käyttöön. (Kohn ym. 2022, Embedding WebAssembly.) Tietokantaprosesseja voi myös joustavasti jakaa palvelimen ja loppukäyttäjän laitteiston välillä niin, että selaimessa tehtävää tietokantahakua rajataan palvelimelta saatujen hakutulosten perusteella (Kohn ym. 2022, Web filesystem).

Selaimen yhteyteen liitetty tietokantajärjestelmä voi myös olla sijoitus tulevaisuuteen. Kohn ym. (2022) linjaavat, että DuckDB-Wasm:lla on mahdollisuus kehittyä muistissa toimivasta OLAP-tietokantajärjestelmästä kiinteäksi paikalliseksi järjestelmäksi, jonka muistinkäyttö taikka säikeistys ei enää ole selainmoottorin rajoittamaa. (Kohn ym. 2022, Summary.)

6.4 Johtopäätökset sovelluskokeiluista

Sovelluskokeiluissa painotettiin ohjelmiston toiminnan laadullista tarkkailua. Kokeiluilla pyrittiin sekä hahmottelemaan toimeksiantajalle kiinnostavia ominaisuuksia sekä samalla selvittämään näiden toteuttamisen uskottavuutta. Sovellusten todettiin tuottavan mielekästä tilannekuvaa merialusten liikkeistä ja sijainneista suorituskyvyn suuremmin kärsimättä. Spatiotemporaalisessa sovelluksessa datan prosessointia selainmoottorilla on varaa reilusti optimoida. Tietokannat sijaitsivat sovelluksen kanssa samalla tietokoneella, joten suorituskykyä ei tiedonsiirron osalta pystytty uskottavasti arvioimaan.

Spatiotemporaalisen karttasovelluksen tapauksessa merkittävin suorituskyvyn pullonkaula löytyy datan paljoudesta. Siinä missä reaaliaikaisessa sovelluksessa olennaista dataa ovat olioiden viimeisimmät sijaintitietolähetykset, spatiotemporaalisessa sovelluksessa datana on olioiden sijaintihistoria. Lämpökarttasovelluksessa haasteena on määrittää sopiva esitettävien datapisteiden määrä verkkoselainta pyörittävän laitteiston suorituskyky huomioiden. Sovelluskokeiluissa ei huomioitu selaimen välimuistin hyödyntämisen mahdollisuutta. Tämän hyödyntämisellä voitaisiin vähentää tietoliikenteen määrää, käyttäjän laitteiston muistinkäytön kustannuksella kylläkin.

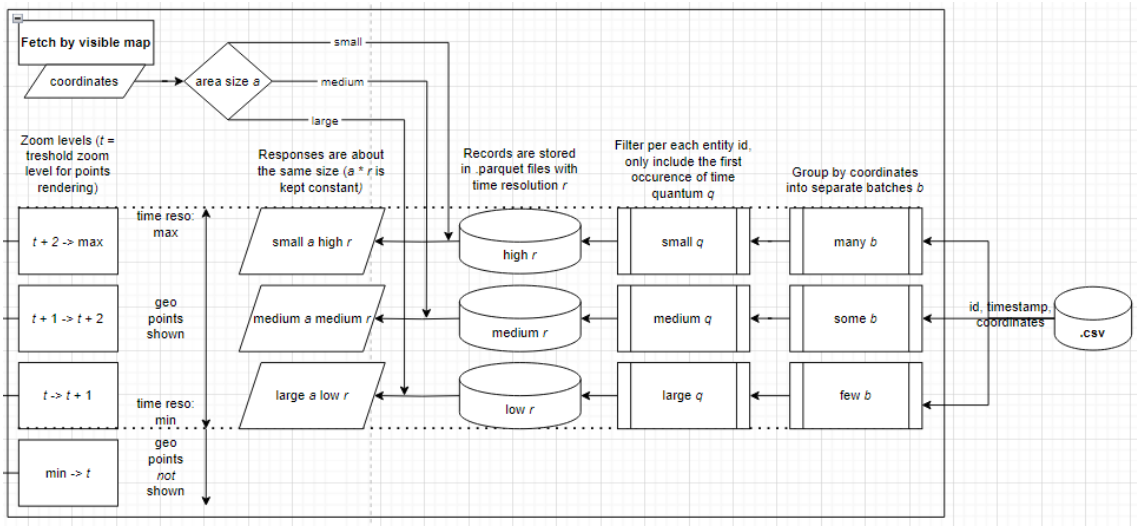
Datanpaljouden äärellä on syytä pohtia eri tapoja minimoida datan määrä eri vaiheissa kyselyprosessia. Tietokantojen indeksoinnit sekä kokonaan erillisten tietokantojen luominen nopeuttavat kyselyitä, mutta kasvattavat tilantarvetta tietokantapalvelimella. Arrow:n työkaluilla palvelimelta lähetettävän datan määrä voidaan minimoida lähettämällä se IPC-virtana, ja data on purettavissa käyttäjän laitteella jälleen Arrow Table -rakenteeseen. Toisaalta data ei ole tällöin karttasovelluksille tyypillisessä GeoJSON-muodossa, jolloin muunnostyö on tarvittaessa tehtävä verkkoselaimen moottorilla. Tietokantapalvelun toteuttaminen Arrow Flight -kehyksen varaan oli Python-ympäristössä luontevaa, mutta JavaScript:lle ei sovellusta kehitettäessä ollut saatavilla vastaavaa toteutusta.

6.5 Käyttäjälle olennainen data

Spatiotemporaalisessa karttasovelluksessa kaikkien näytettävän alueen AIS-lähetysten esittäminen kaikilla kartan etäisyyksillä ei ole järkevää, sillä lähetysten määrä kasvaa karkeasti ottaen karttanäkymän käsittämän alueen neliössä. Tämä sekä kuormittaa järjestelmää että tuottaa loppukäyttäjälle ylimääräistä tietoa: käyttäjä tuskin hyötyy esimerkiksi $(1800 \times 1000)\ \mathrm{km}^2$:n karttanäkymässä aluksen etenemisen historiasta minuutin tarkkuudella, kun kahden peräkkäisen lähetyksen välillä kuljettu matka ei näy muutoksena kartalla. Toisaalta lähetykset kymmenen minuutin välein saattavat jo erottua.

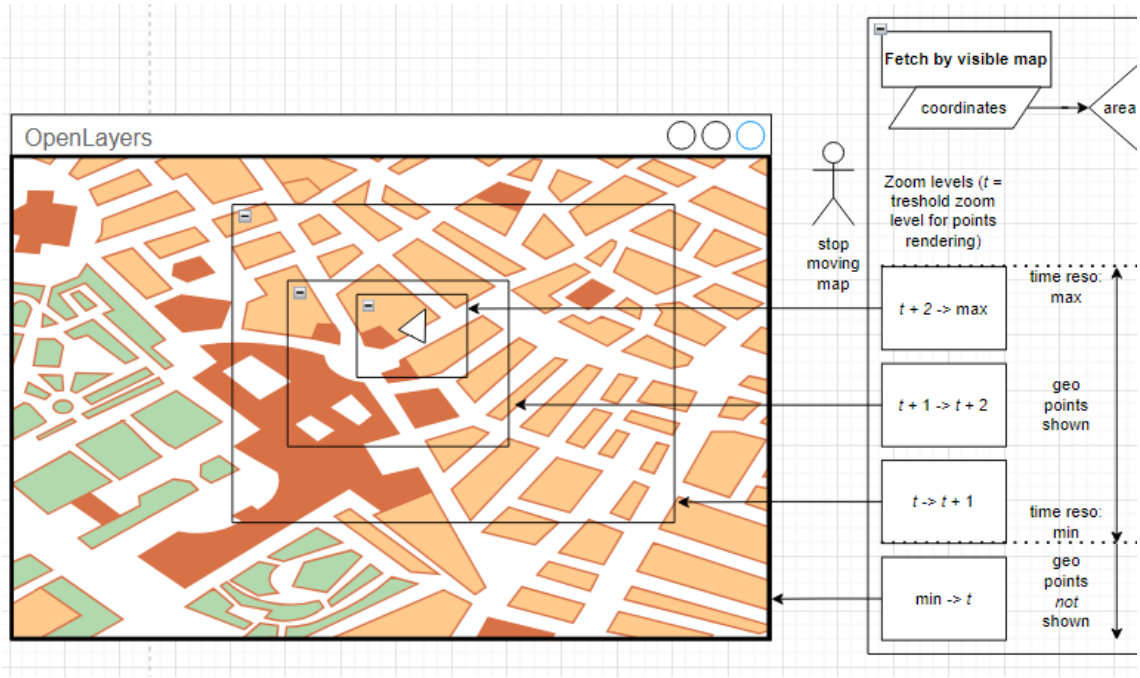
Kun ylimääräiset lähetykset aikaulottuvuudessa karsitaan pois, saadaan datan määrä pysymään aisoissa kartta-alueen kasvaessakin. Tästä syntyi idea muodostaa tietokantoja eri aikaresoluutioilla. Kuviossa 12 näytetään prosessi, jossa raakadatasta muodostetaan Parquet-tiedostojen joukkoja, kukin tiettyä kartan etäisyyttä (zoom level) varten. Lähintä tarkennusta vastaavassa tiedostojoukossa tiedostoja on eniten, ja kukin tiedosto myös sisältää dataa tarkimman sovitun aikaresoluution mukaisesti (AIS-datan tapauksessa enimmäisresoluutio olisi yksi sekunti). Vastaavasti suurinta etäisyyttä vastaava tiedostomäärä on pienin, mutta aikaresoluutio kaikista karkein. Näin voidaan pitää kyseltävien Parquet-tiedostojen koko samassa suuruusluokassa kaikilla kartan etäisyyksillä, mikä lisää tietokantapalvelimen muistinkäytön ennakoitavuutta.

Kuvio 12: Aikaresoluution data-arkkitehtuuri palvelimen näkökulmasta



Kuviossa 13 on havainnollistettu, miten eri etäisyydet kartalla vastaavat eri aikaresoluutioita. Käyttäjän näkymän pysähdyttyä pyydetään palvelinta palauttamaan näytettävää kartta-aluetta vastaavat tietueet. Kyseltävä aikaresoluutiojoukko valitaan näytettävän alueen koon mukaan (Kuviossa 12 area size a). Lähetettävän datan määrä pysyy niin ikään samassa suuruusluokassa kaikilla kartan etäisyyksillä. Datan määrän voidaan esimerkiksi olettaa kasvavan alueen neliössä A. Tällöin tiedostokohtaisen datan määrä k pysyy vakiona, jos kunkin olion lähetyksistä valitaan aina n:nnes lähetys $k = \frac{A}{n}$ mukaisesti.

Kuvio 13: Aikaresoluution data-arkkitehtuuri loppukäyttäjän näkökulmasta



Datan hyödyllisyyttä voidaan pohtia myös ajankohtaisuuden näkökulmasta. Joissakin sovelluskohteissa viimeaikaiset tapahtumat ovat kaukaista menneisyyttä tärkeämpiä. Näin aikaresoluutiota voitaisiin harventaa sitä mukaa, mitä pidemmälle historiaan datassa mennään. Myöskään yksittäisten olioiden liikkeet kartalla eivät välttämättä ole merkityksellisiä kauempaa tarkasteltuna. Näin ollen voitaisiin suuremmilla etäisyyksillä ottaa automaattisesti käyttöön jokin muu näkymä datalle, kuten esimerkiksi työssä esitelty lämpökarttakerros. Tästä olisi myös mahdollista luoda dynaaminen, etäisyyteen reagoiva versio, jossa datapisteiden lukumäärä kartalla pidettäisiin vakiona etäisyydestä riippumatta. Työssä esitellyssä versiossa käytettiin ainoastaan kartan näyttämisen yhteydessä luotua kerrosta, joka lähennettäessä erottui enenevissä määrin datapisteiksi. Tästä ilmiöstä seuraa näkymän vääristymistä (pisteen keskikohta on laivojen tapauksessa esimerkiksi keskellä maata).

Suurilla etäisyyksillä aikaryhmittäiset alukset voidaan myös lisätä kartalla ryppäisiin, klustereihin, jolloin useampi alus tietyltä alueelta korvataan näiden lukumäärän osoittavalla elementillä. Tämä on helppo tapa vähentää selaimen graafista kuormaa. Datan tarkastelun näkökulmasta tämä muistuttaa työn lämpökarttasovellusta, joskin tässä tarkasteltiin viikon kertymiä staattisesti.

6.6 Riskialtis WebSocket

WebSocket:ia käytetään tyypillisesti osakemarkkina- sekä uhkapelisivustoilla, verkkokaupoissa ja sosiaalisessa mediassa sekä chat-viestinnässä. Murley:n ym. (2021) tutkimuksessa arvioitiin, että esimerkiksi tradingview.com-sivustolla käytettynä WS:n käyttö alentaa palvelimen tarvetta kaistanleveydelle 16 Mb/s. Sivustolla oli syyskuussa 2019 lähes 5000 samanaikaista käyttäjää. (Murley, Ma, Mason, Bailey & Kharraz 2021, Measurement.)

Murley ym. (2021) havaitsivat lisäksi, että monilla sivuilla protokollan asettaminen on jätetty puolitiehen. Tämän seurauksena sivustojen suorituskyky heikkenee ja turvallisuus vaarantuu. Jos esimerkiksi hyväksyttäviä HTTP-alkuperän (origin) tietoja ei ole määritetty palvelimella, on vaarana, että tämä hyväksyy kaikki saapuvat yhteydet oletuksena. Toinen mahdollinen haavoittuvuus liittyy yhteyden salaamiseen: Valtaosa WS-yhteyksistä avataan HTTPS-yhteyden kautta, jolloin kyseinen yhteys voidaan muuttaa ainoastaan salatuksi WSS-yhteydeksi. Tutkimuksessa noin 14 % lähes 56:sta tuhannesta sivustosta kuitenkin salli Upgrade-pyynnön välittämisen erillistä salaamatonta kanavaa käyttäen. (Murley ym. 2021, Misconfiguration.)

WebSocket:n mahdollistama reaaliaikainen viestintä ilman kiertokyselyitä soveltuu hyvin tilanteisiin, joissa edestakaista viestintää tapahtuu tiheään. Viestien sisältämä ylimääräinen data on protokollan käytössä myös minimoitu. Nämä ominaisuudet tekevät WS:sta kiinnostavan etenkin sellaista karttasovellusta ajatellen, jossa painotetaan selainohjelmistosta erillisen palvelimen käyttöä. Protokollan käyttöön liittyy kuitenkin riskejä, jotka voivat jäädä kehittäjältä valitettavan helposti huomaamatta.

LÄHTEET

Ahmad, T., Al-Ars, Z. & Hofstee, H. P. 2022. Benchmarking Apache Arrow Flight – A wire-speed protocol for data transfer, querying and microservices. Julkaisu. Viitattu 24.4.2024. https://dl.acm.org/doi/pdf/10.1145/3527199.3527264

Amazon Web Services. n.d. What's the difference between OLAP and OLTP?. Verkkosivu. Viitattu 24.4.2024. https://aws.amazon.com/compare/the-difference-between-olap-and-oltp/

PG-Storm. n.d. Apache Arrow (Columnar Store). Verkkosivu. Viitattu 24.4.2024. https://heterodb.github.io/pg-strom/arrow_fdw/

Apache Parquet. n.d.a. Overview. Verkkosivu. Viitattu 19.3.2024. https://parquet.apache.org/docs/overview/

Apache Parquet. n.d.b. Motivation. Verkkosivu. Viitattu 19.3.2024. https://parquet.apache.org/docs/overview/motivation/

Chodorow, K. 2013. MongoDB: The definitive guide, 2nd Edition. O'Reilly Media, Inc.. Viitattu 23.4.2024. https://learning.oreilly.com/library/view/mongodb-the-definitive/9781449344795/

Comer, D. 1979. The ubiquitous B-Tree. Julkaisu. Viitattu 27.4.2024. https://dl.acm.org/doi/pdf/10.1145/356770.356776

DuckDB Foundation, Amsterdam NL. n.d. Why DuckDB. Verkkosivu. Viitattu 17.3.2024. https://duckdb.org/why_duckdb

Drake, J. D. & Worsley, J. C.. 2002. Practical PostgreSQL. O'Reilly Media, Inc.. Viitattu 23.4.2024. https://learning.oreilly.com/library/view/practical-post-gresql/9781449309770/

Fielding, R., Gettys., J, Mogul, J., Frystyk., H., Masinter, L., Leach, P. & Bern-ers-Lee, T. 1999. Request for Comments: 2616. Verkkosivu. https://data-tracker.ietf.org/doc/html/rfc2616

Fette, I. & Melnikov, A. 2011. Request for Comments: 6455. Verkkosivu. https://datatracker.ietf.org/doc/html/rfc6455

Kohn, A., Moritz, D., Raasveldt, M., Mühleisen, H. & Neumann, T. 2022. DuckDB-Wasm: Fast Analytical Processing for the Web. Julkaisu. Viitattu 25.4.2024. https://duckdb.org/pdf/VLDB2022-kohn-duckdb-wasm.pdf

Medium. 2023. Demystifying HTTP Request vs HTTP Polling vs Long Polling vs WebSocket vs Server sent event vs WebHooks. Verkkosivu. Viitattu 27.4.2024. https://medium.com/@rano3003/demystifying-http-request-vs-http-polling-vs-long-polling-vs-websocket-vs-server-sent-event-vs-aa65d26d1e80

MongoDB, Inc. n.d.a What is MongoDB?. Verkkosivu. Viitattu 17.3.2024. https://www.mongodb.com/what-is-mongodb

MongoDB, Inc. n.d.b. Explaining BSON with Examples. Verkkosivu. Viitattu 17.3.2024. https://www.mongodb.com/basics/bson

MongoDB, Inc. n.d.c. 2d index internals. Verkkosivu. Viitattu 27.4.2024. https://www.mongodb.com/docs/manual/core/indexes/index-types/geospatial/2d/internals/

Murley, P., Ma, Z., Mason, J., Bailey, M. & Kharraz, A. 2021. WebSocket Adoption and the Landscape of the Real-Time Web. Julkaisu. Viitattu 26.4.2024. https://faculty.cc.gatech.edu/~mbailey/publications/www21_websocket.pdf

Mühleisen, H. 30.9.2022. Querying Postgres Tables Directly From DuckDB. Verkkosivu. https://duckdb.org/2022/09/30/postgres-scanner.html

NATO Shipping Centre. n.d. AIS (Automatic Identification System) overview. Verkkosivu. Viitattu 16.3.2024. https://shipping.nato.int/nsc/operations/news/2021/ais-automatic-identification-system-overview

OpenJS Foundation. n.d.a. Deck.gl. Verkkosivu. Viitattu 18.4.2024. https://deck.gl

OpenJS Foundation. n.d.b. Introduction. Verkkosivu. Viitattu 18.4.2024. https://deck.gl/docs

OpenLayers. n.d. Verkkosivu. Viitattu 18.4.2024. https://github.com/openlayers/openlayers/blob/main/README.md

PostGIS PSC & OSGeo. n.d. About PostGIS. Verkkosivu. Viitattu 17.3.2024. https://postgis.net/

Raasveldt, M. & Mühleisen, H. 2019. DuckDB: an embeddable analytical database. Julkaisu. Viitattu 25.4.2024. https://15721.courses.cs.cmu.edu/spring2024/papers/20-duckdb/2019-duckdbdemo.pdf

Richter, T. Database of the year: Postgres. IEEE Software. 20.8.2021. Viitattu 23.4.2024. https://ieeexplore.ieee.org/ielx7/52/9520220/09520330.pdf?tag=1

Sinha, T. 2021. OLAP vs. OLTP: What’s the difference?. Verkkosivu. Viitattu 24.4.2024. https://www.ibm.com/blog/olap-vs-oltp/

The PostgreSQL Global Development Group. n.d.a. What is PostgreSQL?. Verkkosivu. Viitattu 17.3.2024. https://www.postgresql.org/docs/current/intro-whatis.html

The PostgreSQL Global Development Group. n.d.b. 11.2. Index types. Verkkosivu. Viitattu 23.4.2024.

The Apache Software Foundation. n.d.a. What is Arrow?. Verkkosivu. Viitattu 17.3.2024. https://arrow.apache.org/

The Apache Software Foundation. n.d.b. Apache Arrow overview. Verkkosivu. Viitattu 24.4.2024. https://arrow.apache.org/overview/

UnixTime.org. n.d. Epoch & Unix Timestamp Conversion Tools. Verkkosivu. Viitattu 17.3.2024. https://unixtime.org/

Mozilla Foundation. n.d. WebAssembly. Verkkosivu. Viitattu 25.4.2024. https://developer.mozilla.org/en-US/docs/WebAssembly

LIITTEET

Liite 1. Näyte AIS-datasta

MMSI	Timestamp	LAT	LON	SOG	COG	Heading	VesselName
370412000	1578441200	10.51287	145.06672	12.6	152.0	145.0	MILKY WAY II
370412000	1578439390	10.60855	145.01463	12.9	153.0	147.0	MILKY WAY II
370412000	1578440581	10.54495	145.04895	12.8	151.0	145.0	MILKY WAY II
370412000	1578439131	10.62233	145.00738	12.9	152.0	146.0	MILKY WAY II
370412000	1578439560	10.59938	145.01935	13.0	151.0	146.0	MILKY WAY II
370412000	1578439640	10.59525	145.02157	13.0	151.0	146.0	MILKY WAY II
370412000	1578439059	10.6263	145.00522	12.8	151.0	146.0	MILKY WAY II
370412000	1578439251	10.61608	145.01068	12.9	153.0	147.0	MILKY WAY II
370412000	1578440079	10.57138	145.03435	13.0	150.0	145.0	MILKY WAY II
370412000	1578439759	10.58885	145.025	13.1	151.0	146.0	MILKY WAY II
370412000	1578439860	10.58328	145.02808	13.2	151.0	145.0	MILKY WAY II
370412000	1578439939	10.57903	145.03032	13.1	153.0	148.0	MILKY WAY II
370412000	1578441120	10.51702	145.06442	12.6	151.0	144.0	MILKY WAY II
370412000	1578444209	10.3531	145.15202	13.0	153.0	146.0	MILKY WAY II
370412000	1578442009	10.47038	145.09	12.6	153.0	145.0	MILKY WAY II
432969000	1578762761	10.31293	145.41299	12.0	330.4	333.0	CORONA SPLENDOR
432969000	1578764120	10.37949	145.37692	11.9	331.4	333.0	CORONA SPLENDOR
432969000	1578762971	10.32322	145.40744	12.0	331.5	333.0	CORONA SPLENDOR
432969000	1578763980	10.3727	145.38068	12.0	331.2	332.0	CORONA SPLENDOR
432969000	1578765670	10.45496	145.33689	11.9	332.6	334.0	CORONA SPLENDOR
432969000	1578765991	10.47073	145.32895	11.8	334.7	335.0	CORONA SPLENDOR
432969000	1578762700	10.30992	145.41452	12.0	332.1	333.0	CORONA SPLENDOR

(jatkuu)

IMO	CallSign	VesselType	Status	Length	Width	Draft	Cargo	TransceiverClass
IMO9405447	3ESS7	70.0	0.0	199.0	32.0	11.6	70.0	A
IMO9405447	3ESS7	70.0	0.0	199.0	32.0	11.6	70.0	A
IMO9405447	3ESS7	70.0	0.0	199.0	32.0	11.6	70.0	A
IMO9405447	3ESS7	70.0	0.0	199.0	32.0	11.6	70.0	A
IMO9405447	3ESS7	70.0	0.0	199.0	32.0	11.6	70.0	A
IMO9405447	3ESS7	70.0	0.0	199.0	32.0	11.6	70.0	A
IMO9405447	3ESS7	70.0	0.0	199.0	32.0	11.6	70.0	A
IMO9405447	3ESS7	70.0	0.0	199.0	32.0	11.6	70.0	A
IMO9405447	3ESS7	70.0	0.0	199.0	32.0	11.6	70.0	A
IMO9405447	3ESS7	70.0	0.0	199.0	32.0	11.6	70.0	A
IMO9405447	3ESS7	70.0	0.0	199.0	32.0	11.6	70.0	A
IMO9405447	3ESS7	70.0	0.0	199.0	32.0	11.6	70.0	A
IMO9405447	3ESS7	70.0	0.0	199.0	32.0	11.6	70.0	A
IMO9405447	3ESS7	70.0	0.0	199.0	32.0	11.6	70.0	A
IMO9405447	3ESS7	70.0	0.0	199.0	32.0	11.6	70.0	A
IMO9675614	7JRD	70.0	0.0	229.0	38.0	13.9	70.0	A
IMO9675614	7JRD	70.0	0.0	229.0	38.0	13.9	70.0	A
IMO9675614	7JRD	70.0	0.0	229.0	38.0	13.9	70.0	A
IMO9675614	7JRD	70.0	0.0	229.0	38.0	13.9	70.0	A
IMO9675614	7JRD	70.0	0.0	229.0	38.0	13.9	70.0	A
IMO9675614	7JRD	70.0	0.0	229.0	38.0	13.9	70.0	A
IMO9675614	7JRD	70.0	0.0	229.0	38.0	13.9	70.0	A

Liite 2. Attribuuttien datatyypit eri tietokantajärjestelmissä

Attribute	Parquet	PostgreSQL	PostgreSQL
MMSI	`pa.int32()`	`integer`	`int32`
Timestamp	`pa.int32()`	`bigint`	`int32`
LAT	`pa.float32()`	`real`	`double`
LON	`pa.float32()`	`real`	`double`
SOG	`pa.float32()`	`real`	`double`
COG	`pa.float32()`	`real`	`double`
Heading	`pa.float32()`	`real`	`double`
VesselName	`pa.string()`	`character varying(255)`	`string`
IMO	`pa.string()`	`character varying(50)`	`string`
CallSign	`pa.string()`	`character varying(50)`	`string`
VesselType	`pa.float32()`	`real`	`double`
Status	`pa.int16()`	`smallint`	`double`
Length	`pa.int16()`	`smallint`	`double`
Width	`pa.int16()`	`smallint`	`double`
Draft	`pa.float32()`	`real`	`double`
Cargo	`pa.int16()`	`smallint`	`double`
TransceiverClass	`pa.string()`	`character varying(8)`	`string`
<location>	-	`geometry(Point, 4326)`	`coordinates, string`
_id	-	-	`objectId`

Liite 3. MongoDB:n indeksit

Name and Definition	Type	Size	Properties
_id_	REGULAR	87.9 MB	UNIQUE
location_2dsphere	GEOSPA-TIAL	104.1 MB	
Timestamp_1	REGULAR	67.6 MB	
IMO_1	REGULAR	40.1 MB	
CallSign_1	REGULAR	47.6 MB	
Cargo_1	REGULAR	38.2 MB	
COG_1	REGULAR	57.5 MB	
Draft_1	REGULAR	36.4 MB	
Heading_1	REGULAR	41.5 MB	
Length_1	REGULAR	42.2 MB	
MMSI_1	REGULAR	49.5 MB	
SOG_1	REGULAR	38.4 MB	
Status_1	REGULAR	38.8 MB	
TransceiverClass_1	REGULAR	35.8 MB	
VesselName_1	REGULAR	45.3 MB	
VesselType_1	REGULAR	38.2 MB	
Width_1	REGULAR	41.0 MB	
Timestamp_1_Draft_1	REGULAR	88.2 MB	COMPOUND
Timestamp_1_Draft_1_VesselType_1	REGULAR	92.1 MB	COMPOUND

Liite 4. PostgreSQL:n indeksit

Index Name	Index Type	Column(s) Indexed
idx_imo	btree	imo
idx_records_callsign	btree	callsign
idx_records_cargo	btree	cargo
idx_records_cog	btree	cog
idx_records_draft	btree	draft
idx_records_geom	gist	geom
idx_records_heading	btree	heading
idx_records_length	btree	length
idx_records_mmsi	btree	mmsi
idx_records_optimized	btree	"timestamp", draft, vesseltype
idx_records_sog	btree	sog
idx_records_status	btree	status
idx_records_timestamp	btree	"timestamp"
idx_records_timestamp_draft_vesseltype	btree	"timestamp", draft, vesseltype
idx_records_transceiverclass	btree	transceiverclass
idx_records_vesselname	btree	vesselname
idx_records_vesseltype	btree	vesseltype
idx_records_width	btree	width