PROJEKTIRAPORTTI

Sisällysluettelo

Tiivistelmä **2**

1 Johdanto **3**

1.1 Tausta 3

1.2 Menetelmät 3

2 Rajapintaratkaisun toteuttaminen **4**

3 Historiadatan käyttöönotto **5**

4 Kohti tietomallipohjaisia toiminnan ohjaus- ja seurantaratkaisuja **6**

5 Johtopäätökset **9**

Tiivistelmä

Tampereen Infra Oy kerää GPS-tietoja kunnossapitopalvelukalustonsa liikkumisesta kaupunkiympäristössä. Mahdollisuus analysoida tätä dataa paikkatietojärjestelmissä koettiin tärkeäksi tavoitteeksi toiminnan kehittämisen kannalta. Tiedonhallinnan kokonaistavoitteet huomioon ottaen kokeilussa pyrittiin luomaan datankeruutehtävän suorittamiseen kykenevä rajapintaratkaisu asiakkaalle.

Kunnossapitokaluston sijainnit ovat tähän asti olleet saatavilla JSON-rajapinnan kautta. Gispo Oy toteutti JSON-rajapinnan päälle jatkuvasti päivittyvän PostgreSQL Foreign Data Wrapper ratkaisun, jonka avulla tietojen hyödyntäminen esimerkiksi QGIS-työpöytäohjelmistossa tuli huomattavasti helpommaksi.

Tiedonkeruuprosessin koko dokumentaatio on julkaistu avoimesti GitHubissa osoitteessa https://github.com/GispoCoding/snowplow_fdw. GitHubiin on dokumentoitu tarkemmat rajapintaprosessin replikoimiseen, käynnistämiseen ja muokkaamiseen liittyvät ohjeet.

1 Johdanto

Tämän loppuraportin tarkoituksena on kuvata kyseessä olevan kokeilupalveluhankinnan tulokset hankinnan edellyttämällä tavalla eli julkisesti.

1.1 Tausta

Tarve ratkaisulle pohjautuu ajatukseen hakea sovelluksen avulla data olemassa olevasta API-rajapinnasta mahdollisimman monikäyttöiseen muotoon tilaajaorganisaation omaan tietokantaympäristöön. Näin tavoitellaan tiedon laaja-alaista rikastamista ja analysointia.

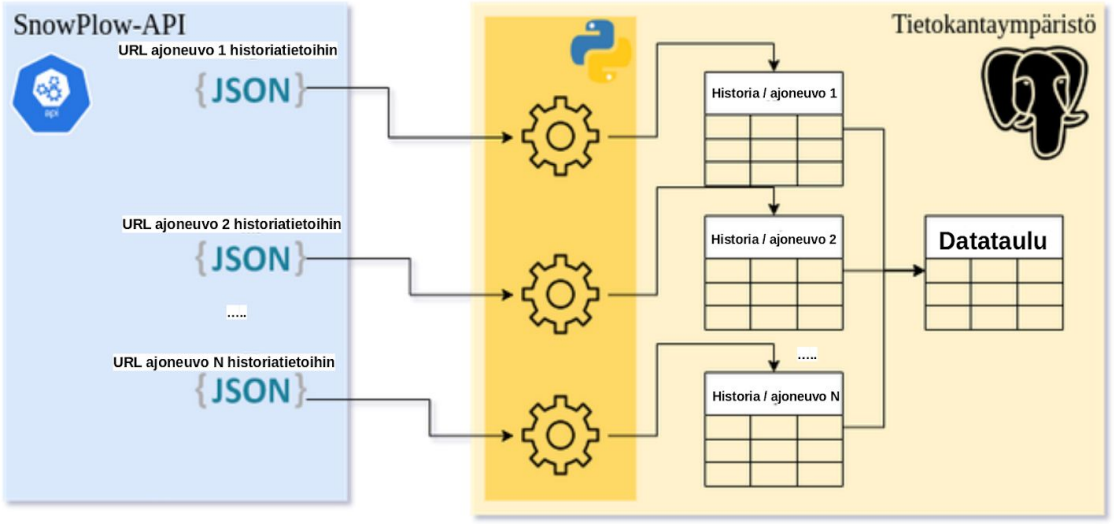
1.2 Menetelmät

Tavoitteiden saavuttamisen kannalta nähtiin tehokkaaksi rakentaa tilaajaorganisaatiolle sovellus, joka mahdollistaa Infra Oyn työntekijöille suoran yhteyden ottamisen dataan tietokannan kautta. Työn alkuperäisestä taulukkopohjaisen datan ratkaisusta päädyttiin rakentamaan ratkaisu tietokantaympäristöön.

Paikkatiedon hyödyntäminen tietokantapohjaisesti mahdollistaa erittäin tehokkaan ja monipuolisen datan käytön. Lisäksi tietokanta mahdollistaa myös natiivisti datan samanaikaisen hyödyntämisen usean käyttäjän puolesta niin datan lukemiseen, muokkaamiseen ja tietojen lisäämiseen (read-/write-oikeuksilla).

Teknisenä ratkaisuna hyödynnettiin avoimen lähdekoodin relaatiotietokantajärjestelmä PostgreSQLää ja sen paikkatietolaajennus PostGISia. Lisäksi PostgreSQLän päälle räätälöintiin sen multicorn- ja pg_cron-laajennusten avulla ajastettu API-rajapintaratkaisu, jossa PostgreSQLän Foreign Data Wrapper implementaatio kerää tasaisin väliajoin JSON-pohjaiseen APIin kertyvää dataa suoraan tilaajaorganisaation tietokantaympäristöön.

2 Rajapintaratkaisun toteuttaminen



Rajapintaratkaisun lopputuotteena on luotu automatisoitu prosessi, joka tasaisin väliajoin käy hakemassa SnowPlow APIn rajapinnasta sinne kertyneen uuden datan. Prosessi on tällä hetkellä implementoitu Gispon AWS-instanssille, mutta tullaan tulevaisuudessa siirtämään tilaajaorganisaation pilvipalveluympäristöön. Automatisoitu prosessi on kaksivaiheinen.

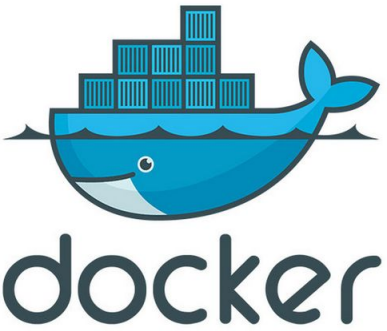
Ensimmäinen ajastettu tehtävä hakee APIsta tiedon siitä, mitkä ajoneuvot ovat olleet aktiivisina edellisen datankeruuskriptin ajamisen jälkeen. Käytännössä sovellus käy hakemassa tiedon kuhunkin ajoneuvotunnisteeseen yhdistettävästä viimeisestä aikaleimasta ja vertaa, onko se tapahtunut edellisen datankeruutehtävän suorittamisen jälkeen. Jos on, kyseisen auton viimeisimmät lokaatiotiedot päivitetään idtable-nimiseen PostGIS-tauluun.

Kun idtable-nimisen taulun tiedot on päivitetty, sovellus siirtyy automatisoidun prosessin toiseen vaiheeseen, jossa yksitellen käydään hakemassa APIin kertyneet uudet GPS-pistetiedot jokaisen edellisen datankeruuprosessin jälkeen aktiivisena olleen ajoneuvon kohdalta. Aktiivisten ajoneuvojen lista saadaan vaiheessa yksi päivitetystä idtable-taulusta.

APIin kerääntynyttä dataa täytyy hieman jalostaa ennen kuin se on valmis lisättäväksi datatable-nimiseen PostGIS-tauluun. Suurin syy tähän on se, että API pystyy palauttamaan kerrallaan vain yhteen ajoneuvotunnisteeseen liittyvää historiatietoa ja usein olisi informatiivisempaa tarkastella esimerkiksi käsillä olevan vuorokauden aikana kerääntynyttä dataa. Niinpä datatable-tauluun on APIssa olevien aikaleima-, koordinaatti- ja tapahtumatiedon lisäksi luotu sarake, joka pitää sisällään tiedon siitä, mihin ajoneuvotunnisteeseen kyseinen GPS-piste liittyy.

Kahden ajastettujen datankeruutehtävien kautta päivittyvien taulujen lisäksi rajapintaratkaisuun kuuluu APIn sisältämien metatietojen tuominen tilaajan tietokantaympäristön PostgreSQLään. Metatietotaulut sisältävät nimen ja tunnisteinformaation mahdollisista työkoneiden tyypeistä (https://tampere.fluentprogress.fi/KuntoTampere/v1/snowplow/mt), operaatiotyypeistä (https://tampere.fluentprogress.fi/KuntoTampere/v1/snowplow/op) ja pääasiallisista operaatiotyypeistä (https://tampere.fluentprogress.fi/KuntoTampere/v1/snowplow/mo).

Rajapintaratkaisu on toteutettu Docker Composella. Docker on sujuvampaan sovellusten luomiseen, kehittämiseen ja ajamiseen suunniteltu työkalu. Dockerin avulla sovellus ja kaikki sen toiminnan kannalta oleelliset kirjastot ja riippuvuudet voidaan pakata samaan säiliöön. Sen lisäksi, että Docker helpottaa sovelluksen käyttöönottoa säilömällä kaikki sen osaset samaan pakettiin, Docker myös takaa, että kehitetty sovellus toimii jokaisessa Linux-koneessa samalla tavalla riippumatta mahdollisista käyttäjäkohtaisista tietokoneen kustomoinneista. Dockeria voi siis pitää tietyn tyyppisenä avoimen lähdekoodin virtuaalikoneena. Kuitenkin normaalista virtuaalikoneesta poiketen Docker ei luo kokonaan uutta virtuaalista käyttöjärjestelmää vaan operoi tietokoneen systeemin päällä.



Suurimmat haasteet rajapintaratkaisun kehittämisessä liittyivät Dockerin ja PostgreSQLän pg_cron-laajennuksen yhteensovittamiseen tarvittavan ympäristön luomiseen ja konfigurointiin sekä url osoitteen käyttämiseen rajapinnan yli välitettävänä ajonaikaisesti muokkautuvana avaimena.

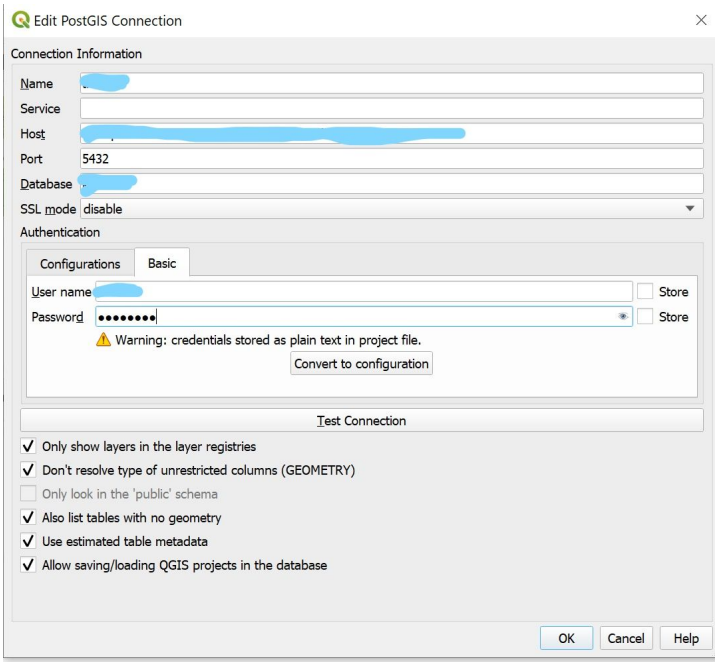
3 Historiadatan käyttöönotto

Uuden datan keräämisen lisäksi analyyseihin saatiin käyttöön myös kaikki 2017-2020 aikana kerätyt kunnossapitolaitteiston sijaintitiedot. Tätä dataa hyödynnettiin spatiotemporaalisten analyysien pohjana.

Aineisto saatiin kahtena PostgreSQL tietokantadumppina, jotka palautettiin Amazon RDS tietokantaan. Tietokannan palautus toteutettiin yksinkertaisesti pg_restore ohjelmistolla komentoriviltä. Historiadata sisältää uutta kerättävää dataa enemmän informaatiota. Koko historia-aineisto indekseineen oli PostGIS-tietokannassa noin 150 gigan kokoinen.

4 Kohti tietomallipohjaisia toiminnan ohjaus- ja seurantaratkaisuja

Datan kerääminen Tampereen Infran omaan tietokantaan auttaa huomattavasti työn seurannassa ja työprosessien suunnittelussa. Seuraamalla aktiivisesti työn toteutumista pystytään kasvattamaan työn tuottavuutta ja esimerkiksi validoimaan toteutunutta ajoa. Myös datan integrointi PostgreSQL-tietokannasta muihin sovelluksiin on helpompaa.

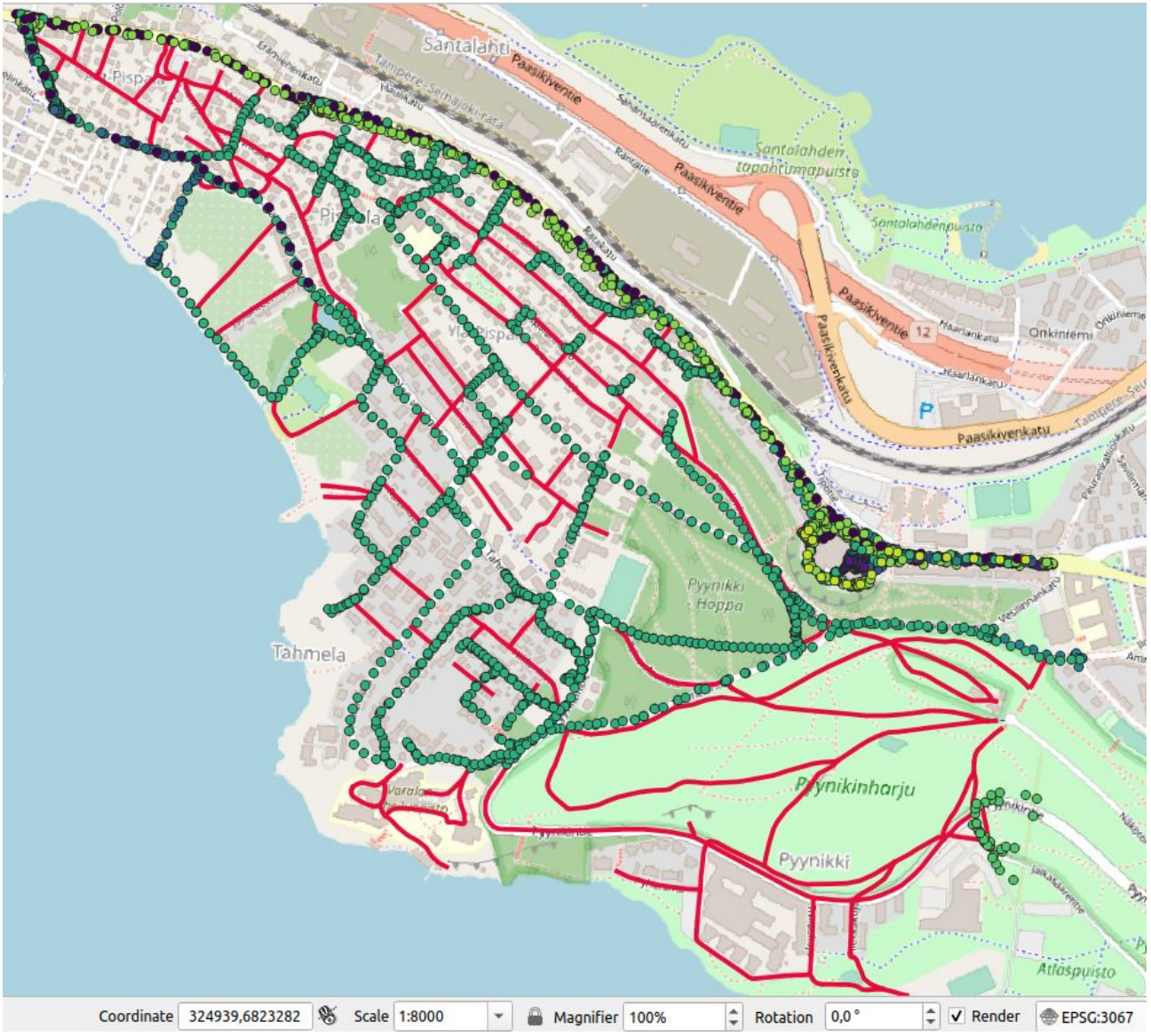


Datan helpompi käyttömuoto mahdollistaa moninaisten analyysien toteuttamisen työpöytäohjelmistolla (esim. QGIS) ilman ohjelmointiosaamista. QGISista voi ottaa suoraan yhteyden PostGIS-tietokantaan viereisen kuvankaappauksen mukaisesti. Onnistuneen yhteyden muodostamisen jälkeen kiinnostavat tasot voi raahata QGIS-näkymään suoraan Selain-ikkunasta. Mikäli käyttäjä napsauttaa viereisessä kuvankaappauksessa näkyviin Store-laatikkoihin myös rastit, QGIS-projekti muistaa PostGIS-yhteyden tunnukset ja osaa automaattisesti päivittää QGIS-projektiin PostGIS-tietokannasta raahatun tason sisältöä mikäli sinne ilmestyy automaattisen ajastetun datankeruuprosessin kautta lisää sisältöä.

Analyysien ja datan hyödyntämisen helpottamiseksi tietokantaan (analysis-skeeman alle) luotiin valmiiksi kaksi näkymää. Toinen näistä näyttää edellisen 24 tunnin sijainnit pistegeometrioina ja toinen viivasegmentteinä. Näitä näkymiä hyödyntämällä voidaan helposti tarkastella visuaalisesti mitä alueita viimeisen vuorokauden aikana on ajettu ja mitkä ovat kenties vielä käymättä läpi.

Kuvassa 1 on nähtävissä millaista näkymän sisältämä viivamainen tieto on. Kuvassa 2 on otettu Pispalan alue lähempään tarkasteluun ja visualisoitu sen alueelta näkymän sisältämää pistemäistä tietoa. Sekä kuvan 1 viivasegmentit että kuvan 2 pisteet on väritetty ajoneuvon yksilöivän tunnisteen mukaan. Kuvassa 2 on lisäksi väritetty punaisella sellaiset tiet, joilla ei olla edellisen 24 tunnin aikana käyty lainkaan.

Kuva 1. Yhden 24 tunnin jakson aikana kertyneet reitit visualisoituna QGISissä.



Kuva 2. Pispalan alueelle yhden 24 tunnin jakson aikana osuneet GPS-pisteet ja “koskemattomat” tiet.

Staattisten visualisointien lisäksi SnowPlow APIin kerääntyneen datan tarkastelu QGISin Temporal Controllerin avulla voi olla erityisen havainnollistavaa. Esimerkinomaisesti tuotetussa rajapinta1-nimisessä GIFissä (löytyy Tulosdata-nimisestä Google Drive kansioista) on tehty animaatio, joka kuvaa ajoneuvojen liikettä keskiviikkona 30.12 klo 06-08. GIF-animaatiossa eri värit kuvastavat ajoneuvon yksilöivää tunnistetta ja symbolin muoto pitää sisällään tiedon siitä, mitä operaatioita kyseinen ajoneuvo on sillä hetkellä suorittanut (“au” ympyrä, “hi” neliö, “pj” timantti, “Siirtymäajo” kolmio, “su” viisikulmio).

Animaation tuottaminen etenee kutakuinkin seuraavasti:

1. Napsauta tason nimen päällä hiiren oikealla painikkeella ja valitse Filter. Query Builder ikkuna avautuu. Provider Spesific Filter Expression kohdassa määritellään millaista tietoa halutaan päästää läpi tasoon (kaikki muu suodattuu pois näkyvistä), esimerkiksi "timestamp" >= '2020-12-30 06:00:00' and "timestamp" <= '2020-12-30 08:00:00'.
2. Napsauta hiiren oikeaa painiketta uudelleen tason päällä ja valitse Ominaisuudet.
3. Avautuvasta ikkunasta tulee navigoida Temporal-välilehdelle ja napsauttaa rasti Temporal-laatikkoon. Configuration-kohtaan tulee valita Single Field with Date/Time. Field-kohtaan spesifioidaan sen sarakkeen nimi, josta aikaleimatieto löytyy. Mikäli haluat, että karttapohjalle ilmestyneet symbolit jäävät näkyviin animaation edetessä, napsauta rasti Accumulate features over time -laatikkoon. Mikäli käyttäjä ei halua kerryttää symboleita karttapohjalle animaation edetessä, käyttäjän tulee asettaa arvo sille kauanko hän haluaa kunkin symbolin pysyvän näkyvissä (Event duration, esim. 2 minuuttia). Lopuksi tulee muistaa napsauttaa ikkunan alareunasta Apply ja OK.
4. Itse animaatiota pääset tarkastelemaan avaamalla Temporal Controllerin QGIS-paneelin kellotaulukuvakkeesta. Avautuvasta ikkunasta pääset säätämään sekä animoitavaa aikaväliä (voit asettaa tämän samaksi kuin miltä aikaväliltä suodatimme datatable-taulua) että animaatiokehysten välissä otettavan aika-askeleen pituutta (Step, esim. 1 minuutti). Tarkempia ohjeita Temporal Controllerin käyttöön löytyy esimerkiksi Gispo Oy:n blogista[1].

5 Johtopäätökset

Tiedonkeruuprosessin koko dokumentaatio on julkaistu avoimesti GitHubissa osoitteessa https://github.com/GispoCoding/snowplow_fdw. Kehitetty datankeruuprosessi on luotu helposti muokattavaksi. Tilaajaorganisaatio voi hallita prosessia pgAdminin (tai vastaavan PosgreSQL hallinta- ja kehitysalustaohjelmiston) kautta. Ajastetut tehtävät luodaan SQL-lauseilla. Käyttäjä pystyy säätelemään sekä sitä, mihin kellonaikaan hän haluaa ajastetut tehtävät ajettavan että kuinka usein hän haluaa dataa Foreign Data Wrapperin avulla SnowPlow APIsta haettavan. Käyttäjä voi myös halutessaan lopettaa automatisoidun ajastetun tehtävän tai tarkastella tehtävien sujumista SQL-lauseiden avulla.

Tällä hetkellä prosessi on automatisoitu siten, että idtable-taulun päivityksestä vastaava idupdate-funktio käynnistyy aina kun kello lyö tasan. Sen lisäksi aina viisi minuuttia yli tasan käynnistyy datatable-taulun päivittämisestä vastaava dataupdate-funktio. Mikäli käyttäjä haluaa muuttaa datankeruuprosessin suoritusfrekvenssiä, hänen tulee muokata muutamaa kohtaa PostGIS-tietokannan

[1] https://www.gispo.fi/en/open-software/geogiffery-in-2020-with-qgis-temporal-controller/

cron-skeeman alta löytyvistä funktioista. Mitään muutoksia itse multicornin päälle rakennettuun rajapintasovellukseen ei tarvitse tehdä; psql-kielisen dataupdate-funktion muokkaaminen riittää.

Käyttäjän tulee ottaa huomioon, että mikäli hän haluaa kerätä dataa harvemmin, kerralla APIsta haettavien rivien lukumäärää kannattaa kasvattaa. Tällä hetkellä rivimäärä on 1000. Arvio perustuu siihen, että SnowPlow API palauttaa GPS-pisteen korkeintaan joka viides sekunti, joten teoriassa pisteitä voi tunnissa kertyä 720. Haettavien rivien lukumääräksi kannattaa valita hieman teoreettista maksimimäärää suurempi luku, sillä dataupdate-funktion suorittamiseen kuluu jonkin aikaa ja niinpä viimeisten ajoneuvojen url-osoitteeseen päästessä edellisestä datankeruuprosessin suorituksesta on saattanut kulua hieman yli tunti. Eikä muutaman ylimääräisen rivin hakemisesta ole mitään haittaa sillä funktiomäärittely osaa itse huolehtia siitä, ettei datatable-tauluun tule lisättyä duplikaattirivejä.

Sen lisäksi, että käyttäjän tulee ajastetun prosessin suoritusfrekvenssiä muokatessaan muokata dataupdate-funktion haettavien rivien lukumäärää, hänen tulee muokata myös funktiossa esiintyviä aikaintervalleja tavoitteisiinsa sopivaksi (ks. GitHubin dokumentaatiosta lisätietoja).