Anni Kärkkäinen

Netezza-tietokannan taustajärjestelmän kehittäminen dbplyr-paketille

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

3.4.2023

Tiivistelmä

Tekijä:	Anni Kärkkäinen
Otsikko:	Netezza-tietokannan taustajärjestelmän kehittäminen dbplyr-paketille
Sivumäärä:	41 sivua
Aika:	3.4.2023
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Ohjelmistotuotanto
Ohjaajat:	Lehtori Simo Silander Analyytikko Johannes Turunen

Nykypäivänä dataa on paljon tarjolla. Datan määrän sijaan ongelmana on enemmänkin sen säilöntä ja analysointi. Netezza on tietokantajärjestelmä, joka on suunnattu erityisesti suurille tietokannoille. dbplyr on taustajärjestelmä dplyrille, joka on R-kielelle saatava paketti. dplyr sisältää useita paljon käytettyjä funktioita datan hallintaan ja analysointiin. Se on erittäin paljon käytetty paketti R:llä.

dbplyrille on saatavilla taustajärjestelmiä lukuisille eri tietokannoille, mutta ei Netezzalle. dbplyriä tarvitaan, jotta dplyr ja tietokanta pystyvät kommunikoimaan keskenään. Tässä opinnäytetyössä on tehty taustajärjestelmä Netezzalle. Tässä esityksessä tutustutaan ensin R-kieleen, dplyriin, dbplyriin ja Netezzaan. Sen jälkeen esitetään, miten taustajärjestelmää lähdettiin tekemään ja miten se toteutettiin. Itse taustajärjestelmä toteutettiin R:llä. Lopuksi esitellään jatkotoimenpiteet työlle ja mahdolliset vaihtoehdot käytetyille järjestelmille, koska käytettävissä olevassa ajassa valmiiksi saadusta taustajärjestelmästä ei löydy ratkaisuja kaikille funktioille dplyrissä.

Avainsanat: R, Netezza, dbplyr, IBM Netezza

Abstract

Author:	Anni Kärkkäinen
Title:	Development of the Background System for the Netezza Database for the dbplyr Package
Number of Pages:	41 pages
Date:	3 April 2023
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communication Technology
Professional Major:	Software Engineering
Supervisors:	Simo Silander, Senior Lecturer Johannes Turunen, Analyst

---

Nowadays there is lot of data on available. The problem is where to store all that data. IBM Netezza is a data warehouse system, which is planed specially for large databases. dblyr is one backend solution to dplyr which is itself a package to R. dplyr includes many heavily used functions for data managing and analyzation.It is highly used package in R.

In dbplyr there is backend solutions available for countless amount of databases, but not for IBM Netezza. In this thesis, we have tried to make a backend to Netezza. This thesis starts with an introduction to the R language, to dplyr, to dbplyr and to IBM Netezza. After introduction, we continue to implementation part of the backend and we go through how it was made. Backend part was made with using R. In the end of this thesis work, we go through what will be happening next regarding this thesis work and the alternative solutions because the completed backend within the time frame is missing few solutions for functions found in dblyr.

Keywords: R, Netezza, dbplyr, IBM Netezza

Sisällys

1 Johdanto 1

2 Datan analysointi 2

3 Teknologiat 4

3.1 IBM Netezza 4
3.1.1 Netezzan arkkitehtuuri 6
3.1.2 NZPLSQL 8
3.2 R-kieli 9
3.3 dplyr-paketti 11
3.4 dbplyr-paketti 13
3.5 Stringr-paketti 14
3.6 DBI 15
3.7 Olio-ohjelmointi R-kielellä 16
3.8 Paketit R:ssä 16
3.8.1 Paketin rakenne 17
3.8.2 Paketin tasot 18
3.9 DBeaver 19

4 Toteutus 19

4.1 Paketin luonti 19
4.2 Tarvittavat funktiot 20
4.3 Funktioiden testaus 20
4.4 Testien tulokset 25
4.5 Testeissä havaittujen ongelmien ratkaisu 27
4.6 Ongelmat, joita ei ratkaistu 28
4.7 Paketin dokumentointi 30
4.8 Vastaan tulleet ongelmat 30
4.8.1 Käytetty yhteys ei ole DBI-yhteensopiva 31
4.8.2 dplyr ei ole yhteensopiva NZPLSQL:n ja DBI:n kanssa 32

5 Jatkokehitys 32

5.1 Paketin saaminen CRANiin 32
5.2 Paketin lisenssi 35
5.3 Vaihtoehdot Netezzalle 35

5.4 Vaihtoehdot R:lle 36
5.5 Ratkaisu on olemassa, mutta se on maksullinen 36
6 Yhteenveto 37
Lähteet 38
Liite1: Testeissä havaitut taustajärjestelmää tarvitsevat funktiot 1

Lyhenteet

AMPP: *Asymmetric Massively Parallel Processing*. Tietokannan arkkitehtuuri Netezzassa. Suunniteltu erityisen massiivisille kyselyille ja lukuisille käyttäjille.

CRAN: *The Comprehensive R Archive Network*. R:n säätiön ylläpitämä ja tukema R-pakettien jakelualusta.

DBMS: *Database Management System.* Tietokantaohjelma. Tässä työssä se on Netezza.

FPGA: *Field-programmable Gate Array*. Ohjelmoitava porttimatriisi.

MPP: *Massively Parallel Processing*. Datan varastointirakenne, joka on kehitetty toimimaan useilla prosessoreilla.

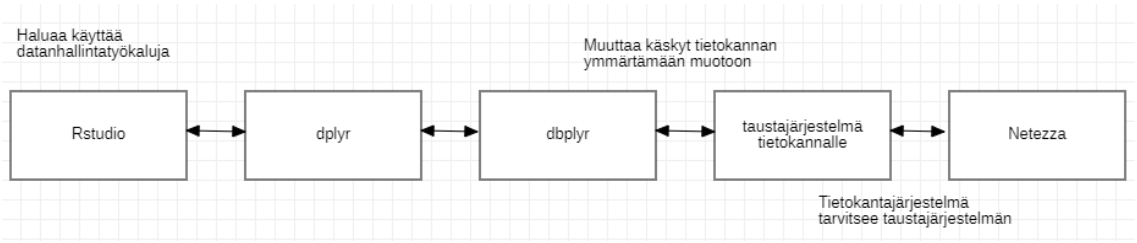
NZPlSQL: *IBM Netezza Structured Query Language*. Netezzan sql-kieli.

ODBC: *Open Database Connectivity*. Standardoitu ohjelmointirajapinta tietokantajärjestelmille. Riippumaton ohjelmistokielestä, tietokantajärjestelmästä tai käyttöliittymästä.

SMP: *Symmetric Multiprocessing*. Symmetrinen moniprosessointi tai jaetun muistin moniprosessointi.

1 Johdanto

R-kieltä (myöhemmin tässä työssä R) käytetään datan analysointiin, manipulointiin ja datasta tehtäviin kyselyihin. Käytettävä data on säilöttynä IBM Netezzan (myöhemmin työssä Netezza) tietokantajärjestelmässä. Netezza on IBM:n tietokantaratkaisu erityisen suurille tietokannoille. Tarvittavien toimintojen toteuttamiseen tarvitaan R:ssä dplyr-pakettia. Paketti sisältää paljon hyödyllisiä funktioita datan mallintamiseen, muuntamiseen ja visualisointiin. Tätä pakettia käytetään paljon, ja se kuuluu niin sanottuihin R:n ydinpaketteihin, jotka ovat tidyverse-paketteja. dplyr tarvitsee dbplyrin tarjoamaa taustajärjestelmää, kun halutaan päästä käyttämään tietokantajärjestelmää. dbplyr muuttaa dplyrin komennot SQL-koodiksi ja ajaa nämä komennot tietokannassa. dbplyr tarvitsee taustajärjestelmän sille tietokantajärjestelmälle, jonka kanssa se toimii. Suosituimmilla järjestelmillä tämä taustajärjestelmä on olemassa, mutta Netezzalle sitä ei ole saatavilla. Kuvassa 1 on hahmoteltu prosessia.

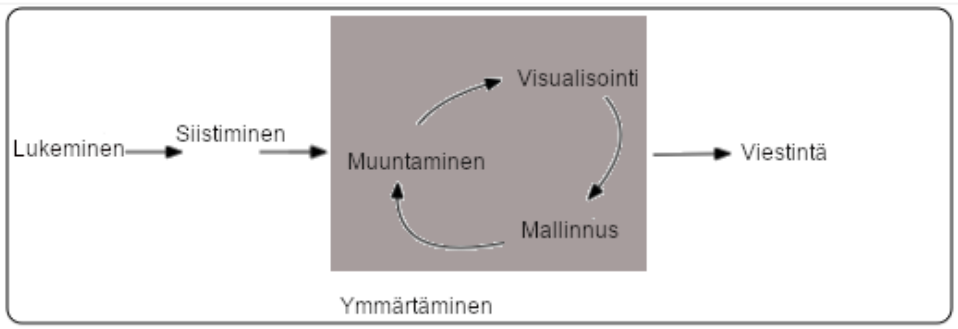


Kuva 1. R:n ja Netezzan vaikutus toisiinsa

Lähtötilanteessa on tiedossa, että osa dplyrin funktioista toimii Netezzalla, mutta osa ei toimi tai toimii väärin. Tässä toteutuksessa lähdetään ensimmäisenä selvittämään funktioita, jotka tarvitsevat taustajärjestelmää, ja sen jälkeen tehdään järjestelmä. Sitä ennen tässä työssä esitellään taustaa.

2 Datan analysointi

R:llä voidaan hoitaa datan analysoinnin kaikki tekniset vaiheet. Ennen analysointia on syytä miettiä kysymyksiä, joihin halutaan saada vastauksia datan perusteella, ja tämän jälkeen voidaan siirtyä hyödyntämään R:ää. Kuvassa 2 esitetään kaikki vaiheet, jotka kuuluvat datan analysointiin. Vaiheet ovat lukeminen, siistiminen, muuntaminen, visualisointi, mallinnus ja viestintä. R:ään on saatavilla lukemattomia paketteja, joiden käyttö helpottaa datan analysointia.



Kuva 2. Datan analysointi

R osaa lukea dataa monista eri tiedostomuodoista, mm. csv:stä, xlsx:stä, xls:tä, txt:stä, json:stä, html:stä sekä SPSS-, STATA- ja SAS-tiedostoja. Sen lisäksi dataa voidaan hakea API-rajapinnan kautta tai lukemalla suoraan nettisivulta. (Import, Export, and Convert Data Files 2021.)

Datan lukemisen jälkeen datan siistiminen onnistuu myös R:llä. Koska kaikki siistimistä kaipaavat datat ovat omalla tavallaan sotkussa, ei niiden siistimistä voi tehdä vain yhden kaavan mukaan, vaan jokaisessa tilanteessa tarvitaan erilaisia paketteja, funktioita ja työtapoja. R:ään on saatavilla tidyr-paketti, joka on erityisesti suunniteltu datan siistimiseen, mutta tilanteen mukaan voidaan siistimistä tehdä myös muillakin paketeilla tai funktioilla (Wickham & Grolemund 2022, s. 12.1 Introduction).

Siistimisen jälkeen dataa yleensä muunnetaan. Siitä voidaan suodattaa käyttöön haluttuja tietoja, esimerkkisi tiedot vain yhden kaupungin osalta tai tietyn

ikäisten tuloksia. Muuntovaiheessa voidaan käyttää työkaluna dplyr-pakettia. dplyr-paketti on yksi R:n suosituimmista paketeista. Toinen paketti, jonka funktioille tehdään ratkaisuja tähän samaan taustajärjestelmään, on stringr-paketti. Stringr -paketin funktiot on suunniteltu erityisesti merkkijonotyyppisten tietojen muuntamiseen ja siistimiseen.

Muuntamisen jälkeen datalle tehdään visualisointi. Visualisointi voi olla graafinen esitys tai taulukko. Visualisointivaiheessa pystytään näkemään, ovatko alkuperäiset kysymykset oikein asetettuja. Visualisointivaihe voi paljastaa ongelmia kysymysten asettelussa ja nostaa esille uusia kysymyksiä. Voi tulla myös ilmi, että data on valituille kysymyksille väärän kaltainen. (Wickham & Grolemund 2022, s. 1 Introduction.) R:ssä on tarjolla monia paketteja datan visualisointiin; ggplot2 on yksi suosituimmista vaihtoehdoista.

Mallinnusvaihe täydentää visualisointia. Kun dataan kohdistetut kysymykset on saatu riittävän tarkoiksi, tehdään niistä mallinnus. Mallit ovat yleensä matemaattisia tai laskennallisia työkaluja. Useissa tapauksissa visualisointi- tai mallinnusvaiheessa havaitaan, että dataa voitaisiin muuntaa toisella tavalla, jotta alkuperäisiin kysymyksiin saataisiin vastaukset tai myöhemmin esiin tulleisiin kysymyksiin saadaan vastaukset. Tällöin palataan takaisin datan muuntovaiheeseen ja sen jälkeen suoritetaan visualisointi ja mallinnus uusilla tuloksilla. Tätä toistetaan niin kauan, kunnes koetaan, että visualisointi ja mallinnus antavat vastaukset datalle asetettuihin kysymyksiin.

Viimeinen vaihe data-analysoinnissa on viestintä. Tässä vaiheessa kerrotaan eteenpäin analysoinnin tulokset. Tulokset voi toimittaa eteenpäin, vaikka sähköpostin tai nettisivujen välityksellä. R tarjoaa myös tähän työkaluja. Shiny-paketilla voi tehdä shiny-nettisivuja. Näille sivuille on helppoa laittaa näkyville visualisointeja. Toinen paljon käytetty vaihtoehto R:ssä on R Markdown. Sillä voidaan luoda muistikirjatyyppisiä tiedostoja, jotka tallentuvat myös .html-muotoon.

R:ään voi kirjoittaa suoraankin SQL-koodia, mutta dplyrillä saa kirjoitettua käyttäjäystävällisemmin koodia, jonka dbplyr voi kääntää SQL-koodiksi. Koodi 1 on

esimerkki dplyrillä kirjoitetusta kyselystä, ja koodi 2 on sama kysely SQL-koodilla kirjoitettuna (Kojima 2017). Koodi hakee tiedot flights_db-tietokannasta, listaa Yhdysvaltojen osavaltiot lentojen lähtöjen viivästymisen mukaan ja näyttää kymmenen pisimmistä myöhästymisistä kärsivää osavaltiota.

```
flights_db %>%
  group_by(CARRIER, ORIGIN_STATE_ABR) %>%
  summarize(DEP_DELAY_AVG = mean(DEP_DELAY)) %>%
  top_n(10, DEP_DELAY_AVG) %>%
  collect()
```

Esimerkkikoodi 1. dplyrillä toteutettu SQL-kysely.

```
SELECT "CARRIER", "ORIGIN_STATE_ABR", "DEP_DELAY_AVG"
FROM (SELECT "CARRIER", "ORIGIN_STATE_ABR", "DEP_DELAY_AVG", rank()
OVER (PARTITION BY "CARRIER" ORDER BY "DEP_DELAY_AVG" DESC) AS "zzz2"
FROM (SELECT "CARRIER", "ORIGIN_STATE_ABR", AVG("DEP_DELAY") AS
"DEP_DELAY_AVG"
FROM " flights_db"
GROUP BY "CARRIER", "ORIGIN_STATE_ABR") "akblwapghp") "xgqydqtjzm"
WHERE ("zzz2" <= 10.0)
```

Esimerkkikoodi 2. Esimerkkikoodi 1:n koodi kirjoitettuna SQL:llä.

Esimerkistä pystyy havaitsemaan, että dplyrillä voidaan ilmaista SQL-kysely huomattavasti lyhyemmin ja selkeämmin kuin kirjoittamalla se suoraan SQL:llä. Yksinkertaisissa kyselyissä erot eivät ole yhtä radikaaleja dplyrin ja SQL:n välillä. Ikävä kyllä dplyr ei selviydy ihan kaikenlaisten SQL-kyselyiden luomisesta.

3 Teknologiat

3.1 IBM Netezza

Netezza on datan varastointiin ja analysointiin suunniteltu järjestelmä. Netezza pohjautuu PostgreSQL 7.2 -tietokantojen hallintajärjestelmään, mutta ei ole yhteensopiva sen kanssa. Se eroaa PostgreSQL-järjestelmästä siten, että Netezzassa voidaan räätälöidä niin tietokannan ohjelmistoa kuin fyysistäkin palvelinta. Ensimmäinen versio Netezzan datan varastointi- ja tietokantajärjestelmästä tuli markkinoille 2002 (Netezza Corporation History). Se ei ollut alun perin IBM:n kehittämä järjestelmä, vaan sen on kehitellyt Netezza Corporation.

Vuonna 2010 IBM osti Netezzan (Tibken 2010). IBM:llä Netezza kuului PureData-tuoteperheeseen, mutta vuodesta 2019 lähtien se on kuulunut omaan Netezza Performance Server under Cloud Pak for Data -tuoteryhmään. Tällä hetkellä Netezza on melko vähän käytettty tietokantajärjestelmä, eikä siitä ole saatavilla ilmaisversiota.

Netezzan pohjalla ovat tietokannat. Tietokannat koostuvat yhdestä tai useammasta skeemasta. Skeema tarkoittaa tietokannassa käytettyä rakennetta. Se määrittää, miten tieto tallennetaan ja järjestetään tietokantaan. Oletusarvoisesti käytetään yhtä skeemaa, mutta versiosta 7.0.3 lähtien on Netezzassa ollut mahdollista käyttää yhdessä tietokannasta useita skeemoja. Skeeman sisällä olevaan rakenteeseen kuuluvat tiedot tauluista, näkymistä, säilötyistä proseduureista, käyttäjien määrittelemistä toiminnoista ja synonyymeistä. Haittapuolena usean skeeman käytössä on, että kokematon käyttäjä voi helposti luoda todella vaikean ja hankalakäyttöisen rakenteen. (Netezza database schema overview 2021.)

Toinen mahdollisesti käyttäjää sotkeva ominaisuus Netezzassa on, ettei se tarkista taulujen yhteyksiä toisiinsa esim. PRIMARY KEY – FOREIGN KEY -mallilla, vaan vastuu yhteyksien toimivuuksien tarkastuksesta jää käyttäjälle. Tämä johtuu siitä, ettei Netezza tue tietokannoissa yleistä indeksointia, vaan Netezzalla on oma ratkaisu tähän. Sen ratkaisu on käyttää karttoja (zone maps). Yhdessä kartassa on tietoja yhdestä tietokannan osasesta: siinä on tietoja alueella olevien tietojen maksimi- ja miniarvoista ja muita tietoja kuvaavia tietoja. Kun Netezza sitten alkaa etsiä tietoja tietokannasta, se katsoo ensin kartoista, minkä kartan alueelta etsitty tieto löytyy, ja tutkii tämän osasen sitten tarkemmin. Menetelmä säästää aikaa isojen tietokantojen kohdalla verrattuna indeksointiin (Leishman 2018).

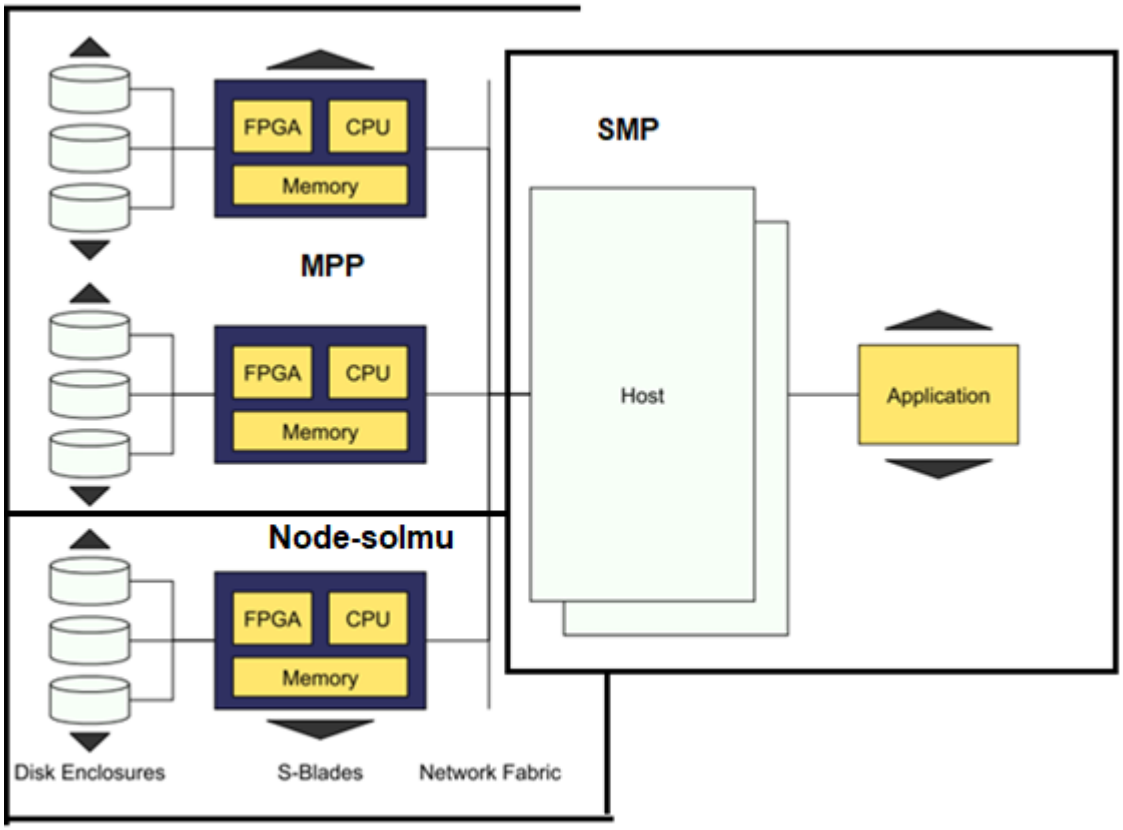
Netezza tukee SQL:lle kaksia erilaisia tunnisteita: säännöllisiä ja rajattuja. Tunnisteet ovat taulukkojen nimiä ja taulukoiden sarakkeiden nimiä. Säännölliset tunnisteet eivät ole tarkkoja isoista ja pienistä kirjaimista. Niiden täytyy alkaa kirjaimella, mutta ne voivat sisältää kirjainten lisäksi numeroita, dollarimerkkejä,

desimaalilukuja ja ala- ja väliviivoja. Välilyönnit ja SQL:ssä käytetyt komennot ovat tunnisteessa kiellettyjä. Rajattu tunniste on tarkka isoista ja pienistä kirjaimista. Rajatuissa tunnisteissa ovat erikoismerkit sallittuja, samoin SQL:ssä käytetyt komennot ja välilyönnit. (Netezza identifiers 2021.)

R:lle Netezzan käyttöä tukevista paketeista on ODBC-paketti asennettu käyttämääni järjestelmään. ODBC on standardoitu ohjelmointirajapinta tietokantajärjestelmille, ja se mahdollistaa eri ohjelmistojen (tässä tapauksessa Netezza) käytön eri sovelluksissa ja kielissä. ODBC tarjoaa yhtenäisen tavan käyttää tietokantaa. Tämä tapahtuu rajapinnan kautta.

3.1.1 Netezzan arkkitehtuuri

Netezzan arkkitehtuuri perustuu AMPP (asymmetric massively parallel processing) -rakenteeseen. Tässä rakenteessa frontend-puolen muodostaa SMP (Symmetric Multiprocessing) -palvelin, jossa kyselyt kommunikoivat taustajärjestelmän MPP (Massively Parallel Processing) kanssa. Kuvassa 3 on kuvattu Netezzan rakennetta.



Kuva 3. Netezzan rakenne (Laskov 2017: 6).

SMP on symmetrinen tai jaetun muistin moniprosessipalvelin. Siinä kaksi tai useampi keskenään identtistä prosessoria on kytketty yhteen keskusmuistiin. Niillä kaikilla on yhteys sisään- ja ulostulolaitteisiin. Koko systeemiä ohjaillaan yhden järjestelmän kautta, joka kohtelee kaikkia osia tasapuolisesti. (Patterson & Hennessy 2018: 506.) Netezzassa SMP-palvelimen käyttöjärjestelmä on Red Hat Enterprise Linux® (RHEL) (Linux host operating system requirements). SMP-palvelin muuttaa SQL-kyselyt ajettaviksi pätkiksi (snippets), luo optimoituja kyselysuunnitelmia ja jakaa koodisegmentit MPP:lle ajoa varten.

MPP on datan varastointirakenne, jotka on kehitetty toimimaan useilla prosessoreilla. Järjestelmässä ohjelman eri osia pyörittävät omat prosessorit, joilla on omat ohjausjärjestelmät ja muistit. MPP-rakenteen ansiosta voidaan käsitellä suuriakin määriä tietoa nopeasti suuresta tietokannasta (Indicative Team). Snippet blades (s-blades) -tietokantaprosessorit ovat MPP:n moottori. Jokaisessa snippet bladessa on prosessori, muistia ja mikropiiri.

Netezzan arkkitehtuurissa toteutetaan tietojenkäsittelyn perusperiaatteetta: suurta dataa käsiteltäessä ei dataa siirretä, jollei ehdottomasti ole pakko. Tätä periaatetta Netezza toteuttaa MPP:n puolella käyttäen hyväksi FPGA- (Field-programmable gate array) moniytimistä mikropiiriä, jolla voidaan ajaa useaa sovellusta yhtä aikaa. Sen sisältämä logiikka on helppo ohjelmoida uudestaan. FPGA suodattaa ylimääräiset tiedot aikaisessa vaiheessa pois datasta. Se toimii yhtä nopeasti kuin dataa saadaan luettua levyltä. Prosessin ansiosta saadaan poistettua I/O-pullonkaulat ja vapautetaan komponentit (CPU, muisti yms.) turhan tiedon käsittelystä. Tämän seurauksena komponentit saavat käsiteltyä nopeasti tarpeellisen datan. (Francisco 2014: 2.)

Netezza-järjestelmä koostuu node-solmuista, joissa Netezza-tietokannanhallintajärjestelmä on asennettuna. Näissä node-solmuissa sijaitsevat s-bladet, jotka ovat tietokantaprosessoreja. Ne vastaavat tietojen käsittelystä ja hallinnasta. Tieto tallentuu node-solmujen kiintolevyille. Jokaisella kiintolevyllä on tallennettuna yksi osa tietokantataulukosta. Nämä kaikki on yhdistetty toisiinsa verkon välityksellä. Verkko on optimoitu skaalatumaan jopa tuhanteen node-solmuun saakka. Tämä mahdollistaa suuret tiedonsiirrot kaikkiin nodeihin samanaikaisesti. Sen lisäksi järjestelmä tarvitsee käyttöliittymän, jonka avulla käyttäjä pääsee käsiksi tietoihin. Järjestelmä on kuvattu kuvassa 3. (Francisco 2014: 4–5.)

3.1.2 NZPLSQL

NZPLSQL eli IBM Netezza Structured Query Language on Netezzan käyttämä sql-kieli. Se perustuu Postgres PL/pgSQL -kieleen. Se on proseduraalinen ohjelmointikieli. Proseduraalisessa ohjelmointikielessä ohjelma jaetaan aliohjelmiin ja se suoritetaan ajamalla proseduureja eli aliohjelmia. Aliohjelmien lisäksi NZPLSQL:llä ovat käytössä haara- ja silmukkarakenteet. Aliohjelmissa käytettyjä muuttujia ja argumentteja voidaan kutsua muista saman Netezza-palvelimen tietokannoista. (NZPLSQL language 2022.)

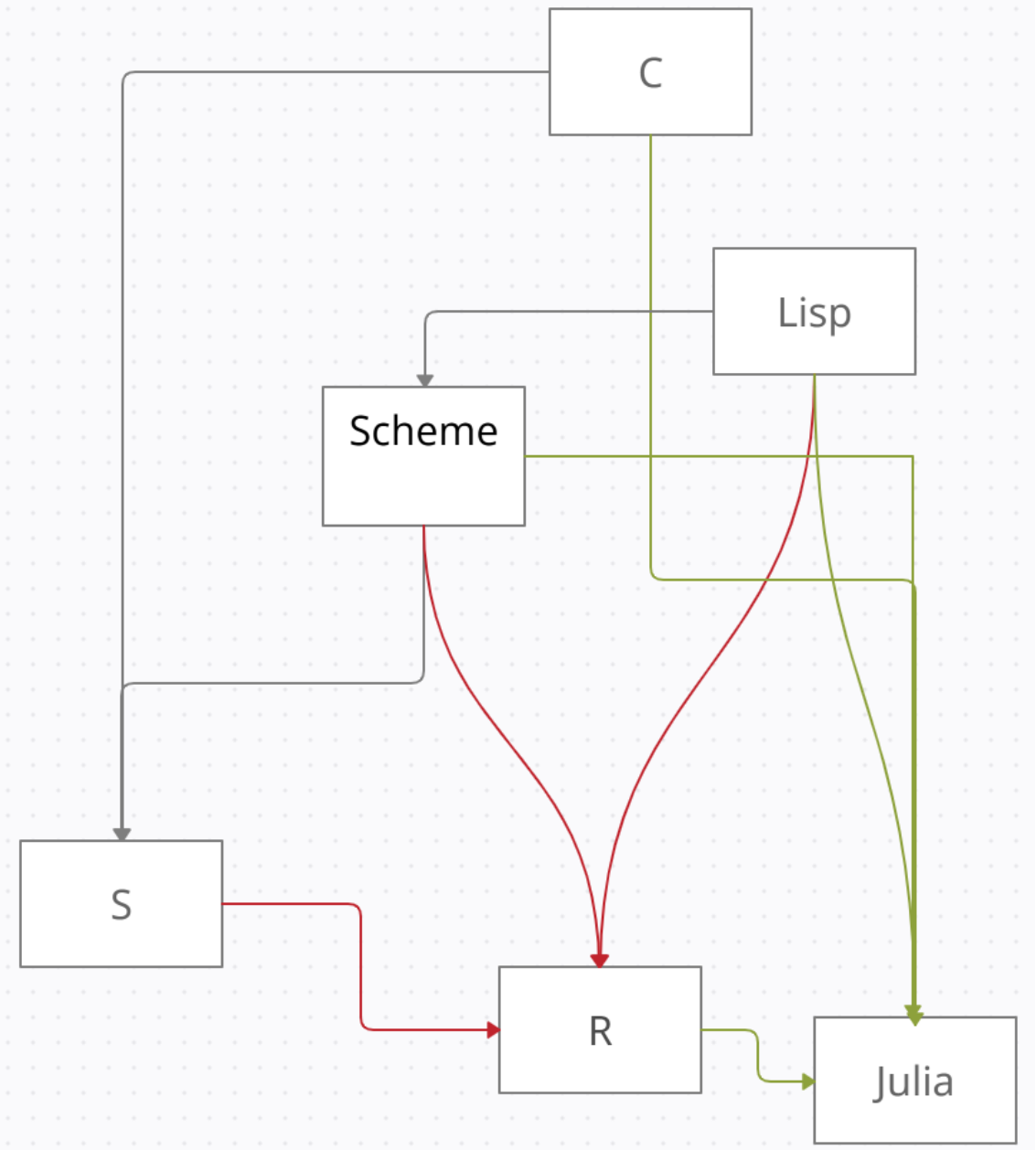
3.2 R-kieli

R-kieli on tarkoitettu datan hallintaan, manipulointiin sekä graafisten esitysten valmistukseen. Se onkin paljon käytetty kieli datan analysoinnissa, visualisoinnissa ja datatieteen parissa. Kieli on julkaistu vuonna 1993. (Peng 2020: 2.4 Back to R.) TIOBE-indeksin mukaan se on 16:nneksi suosituin ohjelmointikieli maailmalla maaliskuussa 2023 (TIOBE).

R-kieli on avoimella lähdekoodilla tehty kieli. Se perustuu S-kieleen. S-kielen juuret ovat data-analysoinnissa, eikä sillä ole perinteistä ohjelmistokielitaustaa. Se suunniteltiin ajatellen, miten tehdä datan analysoinnista helppoa, ja sen kehittelyssä oli ajatuksena palvella sekä käyttäjiä että kehittäjiä (Peng 2020: 2.2 What is S?). Myös R:ssä ajatus on sama. Kehittäjät voivat kehitellä uusia asioita, esim. dbplyr-taustajärjestelmän Netezzalle, tai sitten käyttäjät voivat analysoida tietokannasta löytyviä datoja ja tehdä informaatiota antavia visualisointeja niille.

S-kielen ongelma oli monille käyttäjille sen maksullisuus. Tämän vuoksi Aucklandin yliopiston lehtorit Ross Ihaka ja Robert Gentleman kehittelivät R:n vuonna 1991. Vuonna 1995 R:lle annettiin GNU-hankkeen yleinen lisenssi (Peng 2020: 2.4 Back to R). Tämä lisenssi tarkoittaa sitä, että kenellä tahansa on oikeus käyttää, kopioida, muuttaa ja jakaa edelleen ohjelmaa ja sen lähdekoodia. Koodiin pohjautuvissa muunnelluissa teoksissa säilyvät vielä nämä vapaudet (GNU General Public License, version 2).

R:n syntaksi oli aluksi lähes identtinen S:n kanssa. Sen semantiikka vaikuttaa samankaltaiselta kuin S:n, mutta todellisuudessa se on lähempänä schema-kieltä kuin S, joka pohjautuu myös tähän kieleen (Peng 2020: 2.5 Basic Features of R). Sen lisäksi R:n syntaksiin on vaikuttanut Lisp. Kuvassa 4 on esitetty, miten R:ään liittyvät kielet ovat vaikuttaneet toisiinsa. Nuolenpää osoittaa sitä kieltä, johon toinen kieli on vaikuttanut.



Kuva 4. R:n syntaksiin vaikuttaneet kielet

Suurin ero S:n ja R:n käytön välillä on se, että R:ään on saatavilla valtavasti erilaisia paketteja. Tämä on seurausta siitä, että R on ilmainen ja sille on hankittu jo 1990-luvulla GNU-projektin lisenssi. Näin ollen innokkaat kehittäjät ovat voineet jo vuosikymmeniä kehittää R:ää eteenpäin.

R on tulkattava ohjelmointikieli, eli koodia ajettaessa järjestelmä suorittaa ohjelman suoraan lennosta. R tukee proseduraalista ohjelmointia funktioiden avulla

sekä olio-ohjelmointia. Kehittyneet ohjelmoijat voivat ohjelmoida R-objekteja suoraan C:llä, C++:lla, Javalla ja Pythonilla. (Wickham 2021: s. S4, s. 25 Re-writing R code in C++; Jozef's Rblog 2018; Velasquez 06.12.)

R:ää voidaan laajentaa erilaisilla paketeilla. Tällä hetkellä CRANissa on jaossa 18 985 pakettia R:lle (CRAN). CRAN on avoimen lähdekoodin R-pakettien jakelualusta. Lisää paketteja on saatavilla Bioconductorin, Omegahatin, GitLabin ja GitHubin kautta.

R-kieltä kirjoitetaan yleisesti RStudiolla tai Jupyter Notebookilla, mutta sitä on mahdollista kirjoittaa monilla muillakin editoreilla. Tässä projektissa käytettiin ilmaisen avoimen lähdekoodin versiota RStudiosta.

Käyttötarkoituksesta riippuen R:n käytön korvaamiseen on useita eri vaihtoehtoja. Datan analysointiin, laskentaan ja graafisten esitysten tekemiseen vaihtoehdot ovat esim. SAS, SPSS ja Stata (Grieder & Steiner 2021). Vaihtoehtoisia kieliä sille ovat Python ja Julia.

3.3 dplyr-paketti

dplyr on yksi tidyverse-pakkaukseen kuuluvista ydinpaketeista (tidyverse packages). Tidyverse-pakettikokoelmaan kuuluu työkaluja datan mallintamiseen, muuntamiseen ja visualisointiin. dplyr sisältää työkaluja datan manipulointiin. Se on suosituimpia paketteja R:ssä, ja ilman sitä on datan manipulointi paljon haastavampaa.

dplyr-paketissa on kokoelma funktiota. Käytetyimmät ovat filter(), arrange(), select(), mutate(), summarise() ja group_by() (dplyr). Taulukossa 1 esitellään käytetyimpien funktioiden käyttötarkoitukset esimerkein ja esitetään esimerkin koodi sql:nä.

Taulukko 1. dplyr-paketin käytetyimmät funktiot (Wickham).

Funktion nimi	Käyttötarkoitus	Esimerkki	Vastine sql:ssä
filter()	Suodattaa datasta tuloksia sanallisilla tai numeraalisilla ehdoilla.	`fiter(TEST_TB2, COL_DBL1 > 0 | COL_INT1 > 30)`	`SELECT * FROM TEST_TB2 WHERE (COL_DBL1 > 0.0 OR COL_INT1 > 30.0)`
arrange()	Järjestää datan tiettyjen ehtojen mukaisesti.	`arange(TEST_TB2, COL_DBL2, date1))`	`SELECT * FROM TEST_TB2 ORDER BY COL_DBL2, date1;`
select()	Valitsee datasta tuloksia sanallisilla perusteilla.	`select (TEST_TB2, COL_INT1)`	`SELECT COL_INT1 FROM TEST_TB2;`
mutate()	Muodostaa uusia muuttujia dataan ja säilyttää vanhat.	`TEST_TB2%>% select (COL_DBL2)%>% mutate( COL_INT22 = as.integer (COL_DBL2)`	`SELECT COL_DBL2, CAST(COL_DBL2 AS INTEGER) AS COL_INT22 FROM TEST_TB2;`
summarise()	Muodostaa uuden datakehyksen.	`TEST_TB2 %>% summarise (sd_int=sd(COL_INT1), sd_DBL = sd(COL_DBL1))`	`SELECT STDDEV(COL_INT1) AS sd_int, STDDEV(COL_DBL1) AS sd_DBL FROM TEST_TB2;`

group_by()	Järjestelee ja ryhmittelee data-taulua.	`TEST_TB2 %>%`
		
		`group_by(round(COL_DBL1)) %>%`
		
		`summrise(mean(COL_INT1))`
			`SELECT round(COL_DBL1), AVG(COL_INT1) AS mean(COL_INT1) FROM (SELECT COL_INT1, COL_INT2, COL_DBL1, COL_DBL2, date1, date2, date3, c, c2, ROUND(COL_DBL1, 0) AS round(COL_DBL1) FROM TEST_TB2)`

dplyriin on olemassa kolme virallista taustajärjestelmää: dtplyr, dbplyr ja sparklyr. dtplyr kääntää dplyrin koodin data.tablen koodiksi. Sparklyr kääntää koodin yhteensopivaksi Apache Sparksin kanssa. dbplyr muuttaa dplyr-koodin SQL-koodiksi. (Wickham.)

3.4 dbplyr-paketti

dbplyr-paketti tarjoaa dplyriin tietokantatoiminnallisuudet, joita tarvitaan sovelluksen taustajärjestelmätoiminnallisuuden toteuttamisessa. Se kääntää dplyr-koodia SQL:ksi ja mahdollistaa etätietokantataulujen lukemisen, kuin ne olisivat muistissa olevia datatauluja. dbplyriä ei tarvitse erikseen ladata, sillä dplyr osaa tehdä sen, kun sen tarvitsee työskennellä tietokantojen kanssa. (Wickham.)

SQL-kyselyjen kääntämisessä on yksinkertaisesti ajatellen kaksi osaa. On olemassa dplyrissä olevat ilmaisujen kääntäminen sekä muut ilmaisut, jotka esitetään dplyrin ilmaisujen pohjalta, jotta nämä voidaan kääntää. Kaikki dplyr-komennot dbplyr kääntää SELECT-lauseiksi SQL:ssä. Kaikkea ei ole mahdollista kääntää, eikä tuloksia ole mahdollista saada kaikissa tapauksissa R:n ja SQL:n välillä identtisiksi. Esim. R käyttää suurempaa numerotarkkuutta pyöristämisessä kuin SQL. (Wickham a ym.)

SQL-käännös tehdään tbl_lazy-komennon päälle. Funktiolla tbl_lazy voidaan testata tietokantayhteyksiä vaativia SQL-lauseita ilman, että ollaan todellisuudessa yhteydessä tietokantaan. SQL-kysely muodostetaan R-koodista kolmessa eri vaiheessa. (Wickham a ym.)

1. Funktio sql_build() rakentaa kyselyrakenteen kyselyfunktiolla, jolla voidaan muodostaa SELECT-kysely (esim. select_query(), join_query(), set_op_query())

2. Funktio sql_optimise() optimoi SQL-funktiota. Käytännössä se pystyy vain poistamaan alikyselyjä.

3. Funktio sql_render() kutsuu SQL:n luontifunktioita (sql_query_select(), sql_query_join(), sql_query_semi_join(), sql_query_set_op()), jotka tuottavat itse SQL-lauseen.

Vaikka dbplyrin avulla SQL-kyselyiden tekemistä voidaan pitää helppona tapana, ei se selviä todella erikoisista ja harvinaisista SQL-kyselyiden tekemisestä. Tällaisissa tilanteissa on mahdollista kirjoittaa SQL-koodia R:n sekaan, jolloin dbplyr osaa yhdistää nämä koodit keskenään. (Wickham c ym.)

3.5 Stringr-paketti

Työssä keskityttiin kehittämään toimivat ratkaisut dplyr-paketin funktioille. Nämä testattavat funktiot sain selville dplyrin taustajärjestelmän tiedostosta, joka oli dplyrin lähdekoodin sisältävässä Github-projektissa. Tässä tiedostossa oli lueteltu myös stringr-paketin funktioita, minkä vuoksi työssä päädyttiin tekemään ratkaisuja myös näille funktioille.

Stringr-paketti sisältää funktioita, joita käytetään merkkijonomuotoisen tiedon muokkaamiseen. Käytännössä R:ssä käytetään stringr-pakettia yleensä datan siistimiseen. Stringr-paketti kuuluu dplyrin kanssa tidyverse-kokonaisuuteen eli on osa R:n ydinpaketteja (tidyverse packages).

3.6 DBI

DBI (Database Interface) määrittelee rajapinnan R:n ja relaatiotietokannan hallintajärjestelmän (DBMS) välillä (Package ’DBI’). Tässä työssä tietokannan hallintajärjestelmä on Netezza. DBI:n avulla voidaan tehdä seuraavia toimintoja: yhdistää tai katkaista yhteys DBMS:ään, luoda ja suorittaa lausekkeita DBMS:ssä, kerätä tuloksia, tehdä virhe- tai poikkeusmenettely, hallita tapahtumia ja saada tietoja tietokantaobjekteista (Wickham & Müller). DBI:n suhdetta muihin komponentteihin esitetään kuvassa 5.

DBI:tä on alun perin kehitetty S:llä. Sitä alettiin kehitellä S:n ja SQLiten välille. Historian saatossa DBI:tä on kehitetty niin S- kuin R-kielelläkin. Tarve DBI:n kehittämiseen lähti siitä, kun ymmärrettiin, ettei suurten ulkoisten tietokantojen hallinta ollut erityisen helppoa S:llä. (David.)

DBI-objektin voi jakaa kolmeen pääluokkaan: DBIDriveriin, DBIConnectioniin ja DBIResultiin. Näissä luokissa on määrittelyt eri tietokantaratkaisuille. Itse DBI-objekti on luokka, joka kokoaa alleen kaikki DBI-luokat. Ne ovat kaikki virtuaaliluokkia. DBIDriver-luokassa on DBMS-ajurit. DBI tuottaa ja generoi dbDriverin(driverName), joka taas herättää funktion driverName, joka sitten vahvistaa driver-objektin. DBIConnection kapseloi yhteyden DBMS:ään. Sen avulla päästään käsiksi dynaamisiin kyselyihin, tulosjoukkoihin ja DBMS-yhteyksien hallintaan. DBIResult esittää DBMS-kyselyn tulokset. (Wickham & Müller 2003.)

DBI-paketin kaltainen ratkaisu on saatavilla myös C:lle (ODBC), Pythonille (dbapi), Javalle (JDBC) ja Perlille (DBI/DBD).

Kuva 5. DBI:n suhde muihin komponentteihin

3.7 Olio-ohjelmointi R-kielellä

R-kieli taipuu olio-ohjelmointiin. R:n rakenteessa on tiettyjä haasteita olio-ohjelmoinnin suhteen. R:ssä voidaan ajatella kaiken olemassa olevan olevan objekteja. Ikävä kyllä olio-ohjelmoinnin näkökulmasta kaikki eivät ole objekteja. Objektit voidaan luokitella perusobjekteiksi ja olio-ohjelmointiobjekteiksi. Ne R-objektit, joilla on luokkaominaisuudet, ovat sopivia olio-ohjelmointiin. R:ssä tärkein ohjelmoinnin toiminnallisuus on funktiot ja olio-ohjelmointi eikä ole läheskään niin tärkeää kuin esim. Javassa. (Advanced R.) R:n olio-ohjelmoinnissa on myös poikkeuksellista se, että luokka ja rakenne määrittyvät ajon yhteydessä, toisin kuin esim. Javassa, joissa luokan määritys tapahtuu käännöshetkellä ja objektin rakenne määrittyy myöhemmin suorituksen aikana (Wickham: s. S4). R:n kohdalla olio-ohjelmoinnista on monia eri versioita. Käytännössä eniten käytetyt olio-ohjelmointiversiot ovat S3, R6 ja S4. dplyr-paketti ymmärtää S3:a ja DBI taas S4:ää.

Tässä työssä ei tarvittu olio-ohjelmointia, vaan toteutus on funktioita. Olio-ohjelmointia tarvittiin DBI:n taustajärjestelmän tekemiseen, jotta Netezzan taustajärjestelmä saatiin toimimaan, mutta minä en toteuttanut tätä pakettia.

3.8 Paketit R:ssä

Paketit ovat R:ssä tapa kerätä yhteen kuuluvia funktioita yhteen ja jakaa niitä helposti käyttäjille. dplyr-paketissa on kerätty yhteen monikäyttöisiä laaja-alaisesti datan manipulointiin käytettäviä funktioita. Tässä työssä kerätään tarvittavia funktioita pakettiin, ja niiden avulla dbplyriin saadaan taustajärjestelmä, joka mahdollistaa Netezzan käytön helposti dplyristä käsin.

R:n käyttäjät ovat tottuneet käyttämään paketteja, ja paketit ovat siisti ja yhteneväinen tapa jakaa koodia (Peng ym. 2020: s. 3.2 R Packages). CRAN on R:ään saatavilla olevien pakettien jakeluverkosto. Siellä olevat paketit ovat ajan tasalla, ja ne on dokumentoitu vaatimusten mukaisesti (The Comprehensive R

Archive Network). Paketteja on myös tarjolla Bioconductorissa, Omegahatissa, GitLabissa ja GitHubissa.

3.8.1 Paketin rakenne

R-paketin rakenteelle on vaatimuksia. Minimivaatimus on, että paketissa on vähintään kansiot R ja man. Näiden kanssa samassa tasossa ovat vielä tiedostot DESCRIPTION ja NAMESPACE. R-kansiossa on R-koodia ja man-kansiossa dokumentaatiotiedostoja. (Peng ym. 2020: s. 3.2.1 Basic Structure of an R Package.)

DESCRIPTION-tiedosto sisältää metadatan paketista. Tiedostossa on tiedot paketin nimestä, sen versionumerosta, tekijöiden tiedot, yhteys ylläpitäjään, tietoa lisenssistä ja yhteydet muihin paketteihin. (Peng ym. 2020: s. R: 3.2.2 DESCRIPTION File.)

NAMESPACE-tiedostossa määritellään, miten paketti näkyy käyttäjälle. Tämä toteutetaan sarjalla export()-kutsuja, joilla kerrotaan, mitkä funktiot paketista ovat saatavilla käyttäjälle. Jos jotain funktiota paketista ei mainita export()-listauksessa, ei käyttäjä voi kutsua sitä suoraan paketista. NAMESPACE -tiedostossa luetellaan import()- tai importFrom()-kutsuilla, mitä funktioita tai paketteja paketti itse käyttää. (Peng ym. 2020: s. 3.2.3 NAMESPACE File.)

Periaatteessa ei ole mitään rajaa siihen, kuinka paljon paketti tarvitsee muita paketteja ja funktioita toimiakseen. Käytännössä jos paketti on riippuvainen muista paketeista ja funktioista, vaatii se myös näiden muiden funktioiden ja pakettien ylläpitoa. Toisaalta voi olla haastavaa yrittää tehdä paketti, joka ei tarvitse muita paketteja. Helpoin ratkaisu tähän on käyttää avuksi paketteja, jotka ovat laajalti käytössä ja joiden ylläpidosta vaikuttavat vastaavan useat ihmiset tai tahot. Pakettien funktioiden nimet voivat aiheuttaa myös konflikteja. Eri paketeissa voi olla saman nimisiä funktioita, ja tämä voi aiheuttaa konflikteja pakettiin. (Peng ym. 2020: s. 3.2.3 NAMESPACE File.)

3.8.2 Paketin tasot

Paketeilla voidaan ajatella olevan R:ssä viisi erilaista tasoa. Peruskäytössä käytetään valmiita paketteja. Nämä paketit voivat olla kahdella eri tasolla: joko asennettu tai muistissa. Asennettu-tasolla olevat paketit on asennettu kehitysympäristöön, mutta kyseisessä projektissa ne eivät ole käytössä. Muistissa olevat paketit on asennettu ja ne ovat käytettävissä käytössä olevassa R-projektissa. R:ssä paketit voi asentaa install.packages()-komennolla ja asennetut paketit voi ottaa käyttöön library()-komennolla.

Paketteja kehittäessä tutustuu kolmeen muuhun pakettien tasoon: lähtötasoon, kokoavaan tasoon ja binääritasoon (Wickham & Bryan 2022: s. 4.1 Package states).

Lähtötason paketissa on lähinnä vain paketin vaatimat komponentit eikä mitään muuta, eli käytännössä paketissa ovat R- ja man-kansiot, sekä NAMESPACE- ja DESCRIPTION-tiedostot.

Kokoavalla tasolla on pakettista tehty yksi .tar-päätteinen tiedosto, ja tämä tiedosto on vielä pakattu .gz-tiedostoksi. Jos paketti on tarkoitettu vain lokaaliin käyttöön, ei siitä tarvitse tehdä .tar- tai .gz-tiedostoja. Näitä tiedostoja tarvitaan paketin siirtelyyn eri käyttäjien ja työasemien välillä. Paketti pakataan devtools-pakettiin kuuluvalla build()-funktiolla. Tämä funktio pakkaa paketin ja tekee .Rbuildignore-tiedoston. (Wickham & Bryan 2022: s. 4.3 Bundled package.)

Kokoavalla tasolla tehdyt .tar- ja .gz-tiedostot on tarkoitettu käyttäjälle, jolla on paketin kehitystyökaluja käytössä, esim. Rstudio. Jos taas paketti halutaan jakaa käyttäjälle, jolla näitä ei ole, tehdään binääripaketti. Paketin pakkaaminen binääripaketiksi tekee siitä yhden tiedoston. Jos halutaan tehdä paketti Windows-ympäristöön, tehdään zip-paketti, ja jos macOS-ympäristöön, tehdään tgz-paketti.

3.9 DBeaver

Työn tekemisen apuna käytettiin SQL-editorina DBeaveria. DBeaver on avoimen lähdekoodin SQL-editori, ja se tukee tietokantoja, jotka käyttävät JDBC-ajureita, eli myös Netezzaa (DBeaver Community). DBeaverista on olemassa maksullinen Enterprise Edition -versio, mutta tässä työssä on käytetty Community Editionia.

4 Toteutus

Työtä lähdettiin toteuttamaan dbplyrin nettisivuilla olevilla ohjeilla (https://dbplyr.tidyverse.org/articles/new-backend.html). Ohjeissa opastetaan alkuun, miten voi tehdä oman DBI-taustajärjestelmän dbplyrille.

4.1 Paketin luonti

Lopputuloksesta on tarkoitus tulla paketti. Jotta paketista saa tehtyä kerralla toimivan, käytin apuna usethis-pakettia. Usethis-paketti on pakettien tekemistä varten. Se automatisoi toistuvat tehtävät, jotka syntyvät projektin määrityksen ja kehityksen aikana. Tämä paketti on kiinteänä Rstudiossa, joten minun ei tarvitse edes kutsua kirjoittamalla konsoliin komentoa usethis::create_package(). Voin vain valita Rstudion valikosta File > New Project > New Directory > R Package, ja Rstudio kutsuu automaattisesti usethis::create_package()-funktion.

Funktio Usethis() luo valmiiksi uuteen projektiin R-kansion ja pohjat DESCRIPTION- ja NAMESPACE-tiedostoille. Se tekee myös tiedoston pkgname.Rproj Rstudiossa, mikä helpottaa projektin tallentamista ja käyttämistä Rstudiossa.

4.2 Tarvittavat funktiot

Työtä toteutettaessa lähdettiin testaamaan, mitkä PostgreSQL:n DBI-taustajärjestelmän funktioista toimivat suoraan Netezzalla ja mille joudumme kirjoittamaan funktiot, jotta ne toimivat. PostgreSQL:n DBI-taustajärjestelmä valittiin pohjaksi, koska Netezza pohjautuu PostgreSQL:ään ja on odotettavissa, että jotkin sen ratkaisusta toimisivat suoraan myös Netezzan kanssa. Sen lisäksi dbplyrin DBI:n taustajärjestelmän teko-ohjeen sivuilta löytyi lista funktioita, joiden toiminta pitää testata uuden taustajärjestelmän kanssa.

4.3 Funktioiden testaus

Lähtötilanteessa oli tiedossa, etteivät kaikki funktiot toimi oikein Netezzan ja dplyrin välillä. Kukaan ei ollut selvittänyt, mitkä kaikki funktiot eivät toimi. Oli vain sekalaisia havaintoja funktioista, jotka eivät toimineet. Ensimmäinen tehtävä olikin testata kaikki dbplyrin taustajärjestelmässä luetellut funktiot dplyrin omien funktioiden kanssa. dbplyrin taustajärjestelmästä löytyi testattavia funktioita yli sata. dplyrin omista funktioista kehotettiin taustajärjestelmän ohjeen mukaan testaamaan funktiot summarise(), mutate(), filter(), left_join(), inner_join(), semi_join(), anti_join(), union(), intersect() ja setdiff().

Tein suurimman osan testeistä kolmella taululla. Kaikkien käyttämieni taulukoitten koko oli 100 riviä.Tauluissa oli negatiivisia ja positiivisia int- ja double-tyyppisiä lukuja. Tauluissa oli pieniä ja suuria kirjaimia string-tyyppisinä muuttujina, yksittäisinä merkkeinä ja merkkijonoina, jotka olivat yhteen kirjoitettuja tai joiden välissä oli välilyöntejä. Merkkijonot alkoivat yleensä kirjaimella, mutta muutamat alkoivat välilyönnillä. Mukana oli myös päivämääriä date- ja POSIXct-tyyppisinä. POSIXct-tyyppiset aikamääreet sisältävät R:ssä tiedon kellonajasta ja päivämäärästä. POSIXct perustuu Unixin standardi- tai määrittelykokonaisuusteen, ja sen ajanlasku alkaa 1.1.1970. R:ssä jokaisella päivällä on järjestysnumero ja numero yksi kuuluu päivälle 1.1.1970, tätä vanhemmilla päivämäärillä on negatiivinen järjestysnumero (Jones ym. 2021). Päivämäärien testauksessa testasin

erityisesti ajankohtia, joissa on jotain erityistä. Testasin karkausvuoden vaikutusta päivämääriin, päivämäärän 1.1.1970 vaikutusta ja sitä, tapahtuuko jotain ongelmallista, jos päivä ja kuukausi ovat samoja, mutta vuosi vaihtelee sadalla. Taulukossa 2 näkyy yhden testaukseen käytetyn taulukon sisältöä.

Taulukko 2. Näkymä yhdestä testaukseen käytetystä taulusta.

COL_INT4	COL_INT5	date4	date5	date6	h	h2	COL_DBL3	COL_DBL4
26	-6	12.21.1969 8:20:00	12.26.2019 8:20:00	3.27.2016	a	Y	-0,285	99,4522
62	0	1.26.1970 8:20:00	1.31.2020 8:20:00	3.1.2016	c	M	0,5727	100,6893
100	46	3.5.1970 8:20:00	3.9.2020 8:20:00	3.22.2016	d	E	-1,1187	101,8157
72	1	2.5.1970 8:20:00	2.10.2020 8:20:00	3.28.2016	b	Y	0,6425	98,051
30	24	12.25.1969 8:20:00	12.30.2019 8:20:00	3.11.2016	y	R	-0,3793	102,0741
29	-34	12.24.1969 8:20:00	12.29.2019 8:20:00	3.21.2016	l	M	-1,3334	101,2107
67	41	1.31.1970 8:20:00	2.5.2020 8:20:00	2.24.2016	f	H	2,5576	98,3702
59	25	1.23.1970 8:20:00	1.28.2020 8:20:00	1.26.2016	s	R	-0,2446	99,6254
76	-12	2.9.1970 8:20:00	2.14.2020 8:20:00	1.10.2016	m	C	1,0388	100,119
50	-11	1.14.1970 8:20:00	1.19.2020 8:20:00	1.14.2016	d	H	-0,2714	100,4471

Kolmannessa taulussa minulla oli string-tyyppisiä merkkijonoja. Merkkijonot oli luotu Lorem Ipsum -generaattorilla. Lisäilin isoja kirjaimia, välilyöntejä ja samoja sanoja eri sarakkeisiin. Taulukossa 3 on nähtävillä näkymä tästä taulukosta.

Taulukko 3. Ote sanoja sisältävästä testitaulukosta.

text1	text2	text3	COL_INT6	COL_INT7
Pellentesque	Morbi maximus	et	76	-12
nisl	conubia nostra	ornare	87	36
facilisis	pharetra libero	quis	83	-21
sollicitudin	at mollis	facilisis	90	62
sollicitudin	id lacinia	risus	72	63
turpis	Cras vitae	justo	96	13
eros	quis consequat	litora	119	-11
eleifend	habitant morbi	amet	98	25
eleifend	eget orci	ullamcorper	131	59
pretium	mi Cras	imperdiet	120	33

Testasin funktioitten toimintaa muuttujilla, joiden pitäisi toimia funktiossa, ja muuttujilla, joiden ei pitäisi toimia funktioissa. Esim. yritin suorittaa matemaattisia laskutoimituksia string-tyyppisille muuttujille ja yritin laskea neliöjuurta päivämäärästä.

Apuna käytin Netezza Database users guide -ohjeessa olevaa taulukkoa, josta on nähtävissä, miten eri tietotyypit on mahdollista muuttaa toiseksi (taulukko 4).

Taulukko 4. Netezzassa tietotyyppien muuttaminen toiseksi.

	explicit and implicit casts									
	...to this data type									
		Char	Varchar	Boolean	Integer	Numeric	Real	Double	Date	Time	Nvarc
Cast from this data type	Char		x	x	x	x	x	x	x	x	x
	Varchar	x		x	x	x	x	x	x	x	x
	Boolean										x
	Integer	x	x			x	x	x			x
	Numeric	x	x		x		x	x			x
	Real	x	x		x	x		x			x
	Double	x	x		x	x	x		x	x	x
	Date	x	x								x
	Time	x	x								x
	Nvarc	x	x	x	x	x	x	x	x	x	

Säilytin luomiani tauluja Netezzan tietokannassa. Testasin ensin haluamaani asiaa Rstudion puolella. Kuvassa 6 näkyy between()-funktioiden testaus mutate()-funktion yhteydessä. Funktiolla between() voidaan testata, onko arvo haluttujen rajojen sisä- vai ulkopuolella. Se palauttaa arvon TRUE tai FALSE. Jos arvo on rajalla, se palauttaa TRUE-arvon. Funktioon mutate() yhdistettynä se luo taulukkoon väliaikaiset an1- ja an2-sarakkeet, joissa kullakin rivillä on sen rivin between()-funktion tulos. Sarakkeeseen an1 tulee tulos, ovatko sarakkeen COL_DBL1 double-arvot välillä –0,5 ja 0,5. Sarakkeeseen an2 tulee tulos, ovatko sarakkeen COL_INT1-arvot välillä 35 ja 55. Rstudio tulostaa 10 rivin näytteen 100 rivin taulusta.

```
> con <- dbConnect(Kazam())
> tbl(con, "TEST_TB2") %>%
+   mutate(
+     an1 = between(COL_DBL1, -0.5, 0.5),
+     an2 = between(COL_INT1, 35, 55)
+
+     )
# Source:   lazy query [?? x 11]
# Database: KazamSQL
   COL_INT1 COL_INT2 COL_DBL1 COL_DBL2 date1      date2      date3      c     c2    an1   an2
      <int>    <int>    <dbl>    <dbl> <date>     <date>     <date>     <chr> <chr> <lgl> <lgl>
 1       14     1014     1.20     101.  1999-11-10 1916-01-10 2016-01-10 w     W     FALSE FALSE
 2       90     1090     1.05      97.4 2000-01-25 1916-03-26 2016-03-26 d     D     FALSE FALSE
 3       99     1099    -0.728     99.7 2000-02-03 1916-04-04 2016-04-04 c     C     FALSE FALSE
 4       50     1050     0.285     99.2 1999-12-16 1916-02-15 2016-02-15 r     R     TRUE  TRUE
 5       24     1024     1.19     101.  1999-11-20 1916-01-20 2016-01-20 w     W     FALSE FALSE
 6       28     1028     0.926     99.8 1999-11-24 1916-01-24 2016-01-24 c     C     FALSE FALSE
 7       75     1075    -0.812     99.4 2000-01-10 1916-03-11 2016-03-11 m     M     FALSE FALSE
 8       45     1045    -0.515     99.8 1999-12-11 1916-02-10 2016-02-10 w     W     FALSE TRUE
 9       19     1019    -0.562     99.5 1999-11-15 1916-01-15 2016-01-15 s     S     FALSE FALSE
10       44     1044     1.60     100.  1999-12-10 1916-02-09 2016-02-09 k     K     FALSE TRUE
# ... with more rows
```

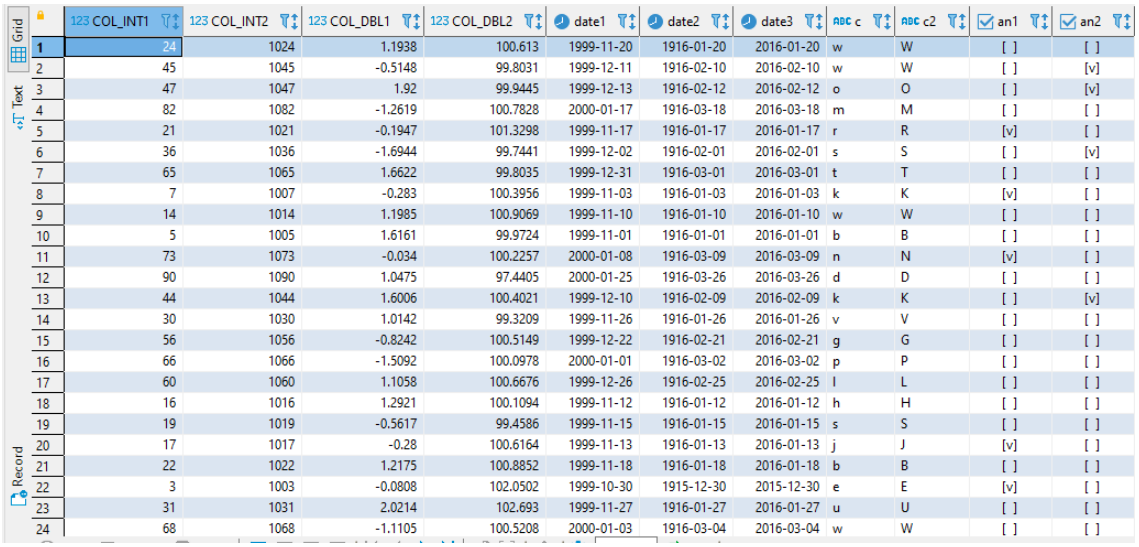
Kuva 6. Funktion between() testi.

Tämän jälkeen arvioin, vastaako tulos haluttua. Kirjaan havainnot ja teen SQL-kutsun. Tämän kutsun saan tehtyä Rstudiossa funktiolla show_query(). Yhdistän show_query()-funktion %>%-operaattorilla alkuperäiseen koodiin. %>% muodostaa putken komennoista. Rstudio tulostaa konsoliin SQL-koodin r-koodin pohjalta (kuva 7). SQL-koodia tarvitsen, sillä haluan testata hakua myös SQL-editorin kautta ja tarkistaa, ovatko SQL-editorin ja Rstudion tulokset toisiaan vastaavia.

```
> tbl(con, "TEST_TB2") %>%
+   mutate(
+     an1 = between(COL_DBL1, -0.5, 0.5),
+     an2 = between(COL_INT1, 35, 55)
+     ) %>% show_query()
<SQL>
SELECT "COL_INT1", "COL_INT2", "COL_DBL1", "COL_DBL2", "date1", "date2", "date3", "c", "c2", "COL_DBL
1" BETWEEN -0.5 AND 0.5 AS "an1", "COL_INT1" BETWEEN 35.0 AND 55.0 AS "an2"
FROM "TEST_TB2"
>
```

Kuva 7. SQL-koodi between()-funktion testaamiseen.

Tämän jälkeen vaihdan DBeaver-ohjelman puolelle ja ajan siellä saadun SQL-koodin. Kuvassa 8 näemme DBeaverin tuloksen saadulla SQL-koodilla. DBeaver-käyttöliittymässä boolean-arvoja ei ilmoiteta termeillä TRUE tai FALSE, vaan se esittää boolean-arvot valintaoptio, jossa [] tarkoittaa FALSE ja [v] TRUE.



	COL_INT1	COL_INT2	COL_DBL1	COL_DBL2	date1	date2	date3	c	c2	an1	an2
1	24	1024	1.1938	100.613	1999-11-20	1916-01-20	2016-01-20	w	W	[ ]	[ ]
2	45	1045	-0.5148	99.8031	1999-12-11	1916-02-10	2016-02-10	w	W	[ ]	[v]
3	47	1047	1.92	99.9445	1999-12-13	1916-02-12	2016-02-12	o	O	[ ]	[v]
4	82	1082	-1.2619	100.7828	2000-01-17	1916-03-18	2016-03-18	m	M	[ ]	[ ]
5	21	1021	-0.1947	101.3298	1999-11-17	1916-01-17	2016-01-17	r	R	[v]	[ ]
6	36	1036	-1.6944	99.7441	1999-12-02	1916-02-01	2016-02-01	s	S	[ ]	[v]
7	65	1065	1.6622	99.8035	1999-12-31	1916-03-01	2016-03-01	t	T	[ ]	[ ]
8	7	1007	-0.283	100.3956	1999-11-03	1916-01-03	2016-01-03	k	K	[v]	[ ]
9	14	1014	1.1985	100.9069	1999-11-10	1916-01-10	2016-01-10	w	W	[ ]	[ ]
10	5	1005	1.6161	99.9724	1999-11-01	1916-01-01	2016-01-01	b	B	[ ]	[ ]
11	73	1073	-0.034	100.2257	2000-01-08	1916-03-09	2016-03-09	n	N	[v]	[ ]
12	90	1090	1.0475	97.4405	2000-01-25	1916-03-26	2016-03-26	d	D	[ ]	[ ]
13	44	1044	1.6006	100.4021	1999-12-10	1916-02-09	2016-02-09	k	K	[ ]	[v]
14	30	1030	1.0142	99.3209	1999-11-26	1916-01-26	2016-01-26	v	V	[ ]	[ ]
15	56	1056	-0.8242	100.5149	1999-12-22	1916-02-21	2016-02-21	g	G	[ ]	[ ]
16	66	1066	-1.5092	100.0978	2000-01-01	1916-03-02	2016-03-02	p	P	[ ]	[ ]
17	60	1060	1.1058	100.6676	1999-12-26	1916-02-25	2016-02-25	l	L	[ ]	[ ]
18	16	1016	1.2921	100.1094	1999-11-12	1916-01-12	2016-01-12	h	H	[ ]	[ ]
19	19	1019	-0.5617	99.4586	1999-11-15	1916-01-15	2016-01-15	s	S	[ ]	[ ]
20	17	1017	-0.28	100.6164	1999-11-13	1916-01-13	2016-01-13	j	J	[v]	[ ]
21	22	1022	1.2175	100.8852	1999-11-18	1916-01-18	2016-01-18	b	B	[ ]	[ ]
22	3	1003	-0.0808	102.0502	1999-10-30	1915-12-30	2015-12-30	e	E	[v]	[ ]
23	31	1031	2.0214	102.693	1999-11-27	1916-01-27	2016-01-27	u	U	[ ]	[ ]
24	68	1068	-1.1105	100.5208	2000-01-03	1916-03-04	2016-03-04	w	W	[ ]	[ ]

Kuva 8. Näkymä DBeverin tuloksesta between()-funktiolla.

Tutkin toistamiseen saatuja tuloksia ja tarkistan, että ne näyttävät halutuilta.

En automatisoinut testejä, koska ne olivat keskenään erilaisia. Ei riittänyt, että laitoin testattavaan funktioon muuttujan aina samaan kohtaan testifunktiota ja katsoin lopputuloksen, vaan jouduin tutustumaan jokaiseen funktioon ja miettimään jokaisen kohdalla erikseen, minkälaisen kokonaisuuden tarvitsen funktion testaamiseen. Koin, että testien automatisointiin olisi mennyt enemmän aikaa

kuin testien tekemiseen manuaalisesti meni. Lopulliseen pakettiin on tarkoitus lisätä automaattiset testit paketin yksikkötesteihin.

4.4 Testien tulokset

Suurin osa testatuista funktioista ja matemaattisista operaatioista toimi oikein Netezzalla. Kaikki eivät kumminkaan toimineet. Testauksessa tuli ilmi vain yksi tilanne, jossa testi, jonka ei olisi pitänyt toimia, toimi. Tämä oli second()-funktio. Xor-funktio oli ainut, joka antoi vääriä tuloksia. Kaikkien muiden kohdalla tuli vain virheilmoitus Netezzalta. Jotkin testatuista funktioista toimivat DBeaverin puolella sql-kyselynä, vaikka eivät Rstudion puolella saaneet tulosta aikaiseksi.

Liitteessä 1 on listattu kaikki funktiot, joiden toiminnassa havaittiin ongelmia testauksessa. Esittelen seuraavaksi tärkeimpiä funktioita, joiden testauksessa havaittiin ongelmia sekä mielenkiintoisia tapauksia testauksesta.

Netezza ei tunnista R:n puolella ^-merkkiä potenssimerkiksi. DBeaverin puolella se tunnisti ^-merkin potenssimerkiksi. Sen sijaan potenssifunktion pow() Netezza tunnisti ja osasi laskea sillä potenssilaskuja; tämä toimi myös DBeaverin puolella.

Funktio log10 on Netezzalle tuntematon; log10():n pitäisi olla funktio 10-logaritmin laskeminen, mutta se ei toiminut. Funktiolla log() voidaan laskea luonnollinen logaritmi arvoista. Tämä toimii int- ja double-tyyppisille muuttujille. Funktiolla log() pitäisi pystyä laskemaan myös logaritmeja, joissa valitaan logaritmi. Tällöin funktio kirjoitetaan muodossa log(muuttuja, haluttu logaritmi) tai log(muuttuja, base=haluttu logaritmi). Kumpikaan näistä tavoista ei toimi Netezzalla.

Funktion as.character() pitäisi muuttaa merkkijono charcter-tyyliseksi merkkijonoksi. R:ssä stringin voi ymmärtää useaksi character-tyyppiseksi muuttujaksi peräjälkeen. Testauksessa as.character()-funktio ei toiminut integer-, double-, POSIXct- ja date-tyyppisille muuttujille. Se ei myöskään osannut muuttaa jo val-

miiksi character-tyyppistä muuttujaa characteriksi, vaan antoi siitäkin virheilmoituksen. Tämä tosin on oikein toimimista, sillä Netezzassa ei charctaeria voi muuttaa charteriksi.

Funktio as.logical() palauttaa TRUE- ja FALSE-arvoja. Jos muuttuja on integer- tai double-tyyppinen, se antaa arvon TRUE. Se palauttaa arvon TRUE myös string-tyyppisille TRUE- ja T-arvoille. Jos muuttujan arvo on 0, F tai FALSE, se antaa arvon FALSE. NA-arvo on as.logical()-funktiolla NA. Muun tyyppisille muuttujille as.logicalin ei pitäisi antaa tulosta. Testeissä havaittiin, ettei as.logical() anna tuloksia integer-, double-, string- tai date-tyyppisille muuttujille. Netezzan dokumenteista selviää, ettei Netezzassa ole ollenkaan olemassa as.logical()-tyyppistä muutosta.

Logiikkafunktiot toimivat Netezzassa. Ongelma on vain, ettei xor()-funktio toimi oikein. Xor eli eksklusiivinen disjunktio on tosi, kun lauseilla on eri totuusarvot. Testasin xor()-funktiota R:ssä filter()-funktion kanssa (esimerkkikoodi 3).

```
#1. testi
tbl(con, "TEST_TB") %>%
 filter (xor(COL_DBL1 > 0, COL_INT1 > 30))
#2. testi
tbl(con, "TEST_TB") %>%
  filter (xor(COL_DBL1 > 0, c == "x"))
#3. testi
tbl(con, "TEST_TB") %>%
  filter (xor(COL_DBL1 > 0, c > "p"))
```

Esimerkkikoodi 3. Funktion xor() testejä.

Tulokset testeistä olivat keskenään ristiriitaisia, eikä missään testissä xor() toiminut oikein. Testeillä 1 ja 2 poisti xor tuloksista ne rivit, joissa kumpikaan ehdoista ei ole totta. Testissä 3 tuloksesta oli poistettu rivit, joissa ensimmäinen ehto ei ole totta, mutta toinen on. Yhdelläkään testillä ei siis saatu oikeaa tulosta.

Funktiot paste() ja paste0() keräävät yhteen niille annetut muuttujat ja tekevät niistä yhden kokonaisuuden. Lähinnä tätä ominaisuutta voi hyödyntää merkkijonoille. Funktio paste0() eroaa paste():sta siten, että se erottelee yhteen kerätyt

muuttujat toisistaan joillain välimerkillä. Kumpikaan funktioista ei toiminut. Niistä tuli virheilmoitus, joka ilmoittaa, ettei funktiota ole olemassa.

Funktio second() palauttaa merkkijonon toisen merkin. Se on tarkoitettu aika-tyyppisille muuttujille (POSIXct, POSIXlt, Date, Period, chron, yearmon, yearqtr, zoo, zooreg, timeDate, xts, its, ti, jul) (RDocumentation: second: Get/set seconds component of a date-time). Niiden kanssa se toimii aivan oikein. Testeissä tuli ilmi, että se ei anna virheilmoitusta, jos sille annetaan int-tyyppisiä arvoja. Int-tyyppisten arvojen kanssa se palauttaa hyvinkin outoja tuloksia. Funktio second() on testin ainut funktio, joka ”toimi” silloin, kun sen ei olisi pitänyt toimia.

Tidyversen ydinpaketteihin kuuluvan stringr-paketin funktioiden kanssa oli myös ongelmia. Stringr-paketti sisältää funktiota, joilla voidaan manipuloida charcter-tyyppisiä merkkijonoja. Ne ovat erityisen hyödyllisiä datan siistimiseen. Paketista toimivat str_length()-, str_to_lower()-, str_to_upper()-, str_trim()- ja str_sub()-funktiot. Loppujen funktioiden kohdalla tuli sama vikailmoitus, joka ilmoitti, ettei funktio ole saatavilla tässä SQL-variantissa.

4.5 Testeissä havaittujen ongelmien ratkaisu

Kun korjattavat ongelmat oli saatu selville, lähdin korjaamaan niitä. Kirjoitin dbplyrin taustajärjestelmään jokaiselle korjattavalle funktiolle korvaavan funktion. Kun käyttäjä sitten kutsuu jotain dplyrin funktiota, osaa dbplyr hakea korvaavan funktion, joka toimii Netezzan kanssa. Niille funktioille, jotka toimivat jo valmiiksi Netezzassa, en tehnyt mitään: dbplyr ymmärtää käyttää alkuperäisiä funktioita näiden funktioiden kohdalla.

Taustajärjestelmään luotiin sql_variant-funktion sisään sql_translator-funktion avulla funktioita. Näissä olioissa oli sisällä tiedot siitä, kuinka jokainen korjattava funktio tulisi kääntää sql:ksi R:ssä. Jokaiselle virheellisesti toimivalle funktiolle pyrittiin luomaan oma funktio, jossa on tiedot uudesta tavasta kääntää funktio.

Helpoimmissa tapauksissa toimimattoman funktion sai korjattua kutsumalla Netezzan vastaavaa funktiota, jolla oli eri nimi kuin testaamallani funktiolla. Nämä olivat helpoimpia korjattavia funktioita.

Esim. Netezzan puolella kymmenkantaisen logaritmin saa laskettua log()-funktiolla. Netezza ei tunne log10()-funktiota, joka on dbplyrissä kymmenkantaisen logaritmin funktio, joten kirjoitin taustajärjestelmään funktion, joka kutsuu log10():n kohdalla Netezzan log()-funktion. Funktio on nähtävissä koodissa 4.

```
log10 = function(x) {
        dbplyr::sql_expr(LOG(!!x))
      }
```

Esimerkkikoodi 4. Funktion log10 kutsun korjaus dbplyrin taustajärjestelmässä.

Ne funktiot, joille ei ollut Netezzassa tarjolla vastaavaa funktiota, vaativat uuden funktion kirjoittamista. Uuden funktion kirjoittamiseen pystyi käyttämään Netezzassa toimivia funktioita ja logiikkaoperaattoreita. Koodissa 5 on nähtävissä ratkaisu, kuinka saadaan xor-funktio toimimaan Netezzassa.

```
xor = function(x,y) {
        dbplyr::sql_expr((!!x %OR% !!y) %AND NOT%(!!x %AND% !!y))
      }
```

Esimerkkikoodi 5. Funktion xor korjaus.

Koodissa 5 funktio ottaa vastaan kaksi parametria: x:n ja y:n. Näitä käsitellään logiikkaehdoilla: (!!x %OR% !!y) tarkoittaa x tai y, %AND NOT% tarkoittaa negatiivista ja operaattoria ja %(!!x %AND% !!y) tarkoittaa x ja y.

4.6 Ongelmat, joita ei ratkaistu

Kaikille funktioille ei löytynyt ratkaisua. Ratkaisematta jäi lopulta lähinnä vain stringr-paketin funktioita. Niiden ratkaisujen löytäminen osoittautui haastavaksi. Useissa tapauksissa todettiin funktio vähän käytetyksi, eikä sen ratkaisuun koettu mielekkääksi uhrata rajattomasti aikaa. Esim. str_locate-funktion kohtalo oli juuri tällainen. Funktiolla str_locate voidaan etsiä tekstistä, merkkijonosta yms.

haluttuja sanoja tai merkkijonoja. Se palauttaa matriisin, josta selviää etsityn merkkijonon aloitus- ja lopetuspisteet. Tässä suurimmaksi hankaluudeksi nousi funktion palautteen muuttaminen matriisiksi.

Taustajärjestelmän ratkaisuissa pystyi käyttämään if-else-rakennetta, mutta se toimii vain, jos if-valinta koskettaa koko aineistoa. If-rakenne ei toimi, jos sen pitää ottaa informaatiota silmukan valinnasta jokaisen manipuloitavan arvon kohdalla erikseen, eli ei voida tehdä rakennetta, jossa aineiston esim. positiiviset luvut kerrottaisiin kahdella ja negatiiviset jaettaisiin kahdella. Esimerkkikoodissa 6 näkyy, miten if-rakenne toimii str_pad-funktiolla. Str_pad-funktio pidentää string-muotoisia merkkijonoja haluttuun pituuteen. Se lisää haluttuja merkkejä merkkijonon alkuun tai loppuun, jotta merkkijonosta tulee halutun pituinen.

```
str_pad = function(string, length, side = "left", pad = ' '){
       if(side == "left"){
         dbplyr::sql_expr(lpad(!!string, !!length, !!pad))
       }
       else if(side == "right"){
         dbplyr::sql_expr(rpad(!!string, !!length, !!pad))
       }
```

Esimerkkikoodi 6. If-else-rakenne.

Str_pad tekee samankaltaisen muutoksen koko joukkoon. Sen sijaan muutos, jonka lopputulos vaihtelee esim. sen mukaan, ovatko manipuloitavat numerot negatiivisia vai positiivisia, ei onnistu, koska olion pitäisi tarkastella jokaista riviä erikseen ja tehdä manipulaatio niiden tietojen mukaan.

Funktio as.logical() jäi myös vaille korjausta. Funktio as.logical() muuttaa int- ja double-tyyppisiä arvoja TRUE- ja FALSE-arvoiksi. Jos luku on eri suuri kuin nolla, sen arvo on as.logical()-funktiolla TRUE. Nolla saa taas arvoksi FALSE. Sen lisäksi as.logical()-funktio osaa muuttaa arvot ”F” ja FALSE arvoksi FALSE. NA-arvo on as.logical()-funktion muunnoksen jälkeen edelleenkin NA. Muille numeroarvoille ja logiikkamuuttujille se antaa arvon TRUE.

Netezzasta ei löytynyt vastaavaa funktiota as.logical()-funktiolle. Netezzalla on oma boolean-tyyppisiä muutoksia tekevä funktio, mutta se toimii vain merkkijonotyyppisille muuttujille. Uskon, että funktio voidaan saada toimimaan tulevaisuudessa, mutta tässä työssä se ei ajallisesti onnistunut.

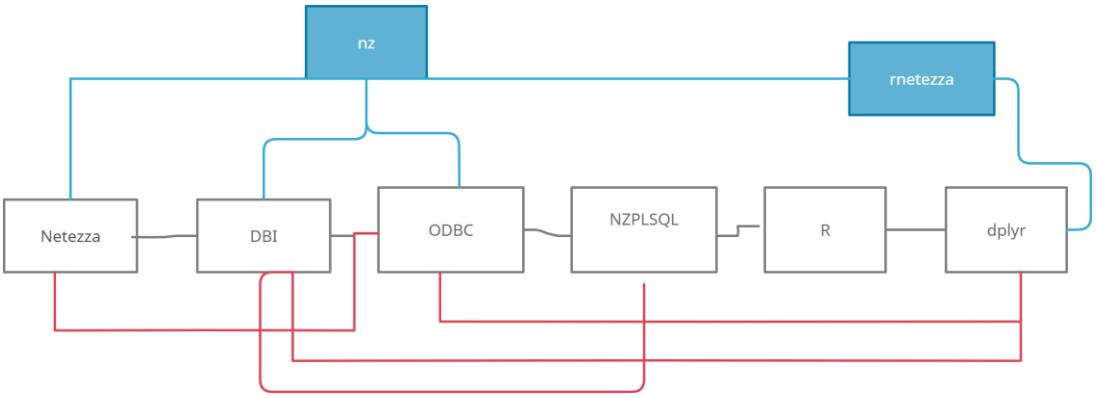
4.7 Paketin dokumentointi

Kun paketin koodi on valmis, on viimeistään hyvä hetki dokumentoida sen sisältö. Tämä on tärkeää, jotta muutkin osaavat käyttää pakettia. Dokumentoinnin avuksi on olemassa valmis paketti ja siinä funktioita. Roxygen2-paketissa on roxygenise()-funktio, jonka avulla voi saada dokumentoinnin alkuun.

roxygenise()-funktiota varten täytyy funktiot kommentoida koodissa. Roxygeniä varten kommentit pitää aloittaa merkkiyhdistelmällä #'. Kommenteissa pitää kertoa @param-ilmaisulla funktioon menevät muuttujat ja niiden tyypit. Ilmaisu @return kuvailee, minkä tyyppinen on funktion palautusarvo. Ilmaisun @examples kohtaan voi laittaa esimerkkejä siitä, miten käyttää funktiota. Kun funktiot on kommentoitu tämän tyyppisesti, voidaan ajaa roxygenise()-funktio. Funktio roxygenise() luo man-kansioon jokaisen funktion mukaan nimetyn .Rd-tiedoston. Näitten tiedostojen pohjalta R osaa tuottaa komennolla ?funktion_nimi() pyynnöllä näytettävät ohjeet. (Introduction to roxygen2.)

4.8 Vastaan tulleet ongelmat

Valitettavasti tämä projekti ei edennyt oikeasti niin suoraviivaisesti, kuin tästä työstä on voinut ymmärtää. Ennen kuin päästiin testaamaan funktioita, oli selvitettävä useita ongelmia, että prosessin eri osaset alkoivat ymmärtää toisiaan. Kuvassa 9 on esitetty kaikki komponentit, jotka vaikuttavat toisiinsa ja ovat oleellisia tässä työssä. Punaisella viivalla on yhdistetty komponentit, jotka eivät toimi hyvin yhteen tai eivät ymmärrä toisiaan. Harmaalla viivalla on merkitty komponenttien suorat yhteydet toisiinsa. Sinisellä on merkitty, mitä yhteyksiä tarvittaisiin, jotta Netezza ja dplyr alkaisivat ymmärtää toisiaan. Siniset laatikot ovat paketteja, joita ei ole vielä olemassa mutta jotka pitää luoda.



Kuva 9. Eri komponenttien yhteydet, tarvittavat yhteydet, uudet komponentit ja kommunikaatioon häiriöitä tuottavat yhteydet. Punainen viiva kuvastaa häiriötä toisiinsa aiheuttavia yhteyksiä, sininen puuttuvia komponentteja ja niiden yhteyksiä ja musta olemassa olevia suoria yhteyksiä.

4.8.1 Käytetty yhteys ei ole DBI-yhteensopiva

Lähtötilanteessa oleva yhteydenmuodostusfunktio ei ole täysin DBI-yhteensopiva. Rstudio tarjoaa vähemmän käytetyille tietokannoille samoja ODBC-ajureita. Nämä ajurit ovat perustasoiset, ja niiden toimivuus vaihtelee eri tietokantojen välillä. Ikävä kyllä ne eivät toimineet kovin hyvin Netezzalla. Tämän vuoksi ei voitu alkaa tekemään vain pakettia dbplyrille, vaan ensin piti saada DBI:n taustajärjestelmä toimimaan. Lähtötilanteessa yhteys toimi ODBC:n kautta, mutta se tuotti S4-luokan connection-olioon. Tarvitaan siis taustajärjestelmä DBI:lle. Monille tietokannoille on olemassa R:lle oma taustajärjestelmä, jonka sisällä on toimiva DBI-ratkaisu. Ikävä kyllä Netezzalle ei ole tällaista ilmaisversiota käyttämällemme yhteydelle tarjolla. (Müller, Wickham.)

Netissä on valmis pohja, jonka pohjalta voi tehdä oman DBI-ratkaisun Netezzalle (https://github.com/r-dbi/RKazam/). Tämä pohjan avulla onnistui työssäni paljon auttanut henkilö työstämään toimivan DBI-taustajärjestelmän tähän projektiin.

4.8.2 dplyr ei ole yhteensopiva NZPLSQL:n ja DBI:n kanssa

NZPLSQL on vähän käytetty SQL-ratkaisu, mutta siitä ei ole opensource-versiota. Tällä hetkellä R ei suoraan käytä Netezzaa, vaan välissä pyörivät NZPLSQL, ODBC ja DBI. ODBC-ajuri ei toimi hyvin Netezzalla. NZPLSQL toimii R:ssä, mutta ei toimi kunnolla DBI:n kanssa.

dplyr on kirjoitettu S3:lla ja DBI S4:llä. S4 perustuu S3:een, mutta ne eivät ole suoraan yhteensopivia. Tähän ongelmaan auttoi, kun sain käyttöön uuden DBI-taustajärjestelmän.

Kuusi vuotta sitten Philippe Chataignon ja Lasse R lisäsivät GitHubiin dplyrnz-projektin, jossa on dplyrin SQL-taustajärjestelmä Netezzalle (https://github.com/philippechataignon/dplyrnz/). Tarkoitukseni oli ottaa siitä vaikutteita tähän työhön, mutta sen kylkeen kehitelty DBI-ratkaisu ei toimi käyttämälläni yhteydellä. En voinut muutenkaan hyödyntää vanhaa ratkaisua, koska sen toteutus oli muiltakin osilta erilainen kuin omani. En saanut siitä edes apua sellaisten funktioiden etsimiseen, jotka eivät toimi Netezzassa.

5 Jatkokehitys

Projekti jäi insinöörityön puitteissa kesken. Kaikille havaituille toimimattomille funktioille ei kerennyt insinöörityön aikataulun rajoissa kehittämään toimivaa ratkaisua. Jatkan näiden ratkaisujen kehittelyä. Lähitulevaisuudessa olisi tarkoitus jakaa työ GitHubin kautta muille sitä tarvitseville maailmalla ja mahdollisesti saada työ jakoon CRANiin.

5.1 Paketin saaminen CRANiin

Koska tavoitteet on hyvä asettaa korkealle, oli jo alussa tavoitteena saada dplyrin taustajärjestelmä Netezzalle toimimaan ja saada paketti CRANiin. Ikävä kyllä oman paketin saaminen CRANiin ei ole mikään nopeasti toteutettava prosessi, ja tämän toteutuminen jää odottamaan tulevaisuutta.

Paketin saamiseksi CRANiin täytyy sille antaa versionumero. Numeroinnin voi tehdä hyvinkin mielivaltaisesti, mutta esim. Hadley Wickham ja Jennifer Bryan suosittelevat R Packages -kirjassa käyttämään muotoa merkittävä muutos.vähäinen muutos.korjaus.pieni parannus ja aloittamaan numerointi arvosta 0.0.0.9000. Tämän jälkeen voi numerointia jatkaa sen mukaan, miten suuria parannuksia tekee aiempaan versioon.

Versionnumeroinnin asettaminen järkeväksi on helppo osuus paketin saamisesta CRANiin. Haastavampi kohta on saada ajettua R CMD check ja saada testit menemään läpi. R CMD check testaa metadatan paketista, NAMESPACE-tiedoston, itse koodia, paketin dokumentaatiota ja monia muita asioita. R CMD checkin ajaminen ja sen läpi pääseminen ei tarkoita sitä, etteikö paketissa voisi olla vikoja ja että koodi toimisi täydellisesti, mutta se on hyvä tapa testata, onko paketti järkevässä muodossa ylipäätään ennen kuin sitä voi alkaa jakaa ulkopuolisille. Jos testistä löytyy vikoja tai varoituksia, on ne korjattava, sillä RUN CMD check pitää päästä läpi, jos haluaa saada paketin CRANiin. (Peng ym. 2020: s. Passing CRAN checks.)

RUN CMD check tekee myös NOTE-nimisiä havaintoja paketista. Jos paketti on tehty vain omaan käyttöön tai pienen yhteisön käyttöön, ei NOTE-huomioille ole pakko tehdä mitään. Mutta jos haluaa saada paketin CRANiin, olisi syytä yrittää päästä eroon niin monesta NOTE-huomiosta kuin mahdollista. Jokainen NOTE vaatii CRANin tarkastajilta tutustumista tilanteeseen, ja tämä hidastaa vapaehtoisten voimin toimivaa prosessia. Ne NOTE:t, joita ei saa poistettua paketista, pitää dokumentoida cran-comments.md-nimiseen tiedostoon. Tässä tiedostossa pitää kuvailla yksitellen jokainen NOTE-huomio ja kuvata syy, miksi se huomio on turha. Ensimmäistä pakettia CRANiin tarjottaessa löytyy paketista aina vähintään yksi NOTE, sillä ensikertalaisille tulee aina NOTE, joka kertoo CRANiin kyseessä olevan ensimmäistä pakettiaan CRANiin tarjoava taho. Näille paketeille tehdään ylimääräisiä tarkastuksia, joita ei kokeneimpien paketin tuottajien paketeille tehdä. (Wickham & Bryan 2022: s. 20.3.2 Check results.)

CMD-tarkastustestien lisäksi on erittäin järkevää tehdä muitakin testejä paketin ja funktioiden toimimisen testaamiseksi kehitysvaiheessa. Ajatus on kumminkin saada paketti siihen kuntoon, ettei heti julkaisun jälkeen tarvitsisi alkaa korjata koodista löytyviä virheitä. CMD-tarkastustestien läpi pääsy ei tarkoita sitä, että paketti ja funktio toimisivat halutulla tai oikealla tavalla, joten omiakin testejä tarvitaan.

Kun R CMD check menee läpi, luodaan tai päivitetään README.md- ja NEWS.md-tiedostot ajan tasalle. README.md-tiedostossa kerrotaan, mikä on paketin käyttötarkoitus, ja mahdollisesti, kenelle se on tarkoitettu. Tilannetta voidaan avata kertomalla, millaisen ongelman ratkaisuun paketti on suunniteltu. Kerrotaan ohjeita siihen, miten paketit voidaan ottaa käyttöön ja miten ne saa asennettua itselleen. GitHubissa ja GitLabissa README.md-tiedosto tulee näkyville repositoryn aloitussivulle. (Wickham & Bryan 2022: s. 20 Releasing a package.)

NEWS.md-tiedostossa kerrotaan, mitä muutoksia pakkaukseen on tullut viimeisimmän päivityksen mukana. Tiedostoon on tarkoitus jättää tiedot myös vanhemmista päivityksistä. Uusimmat päivityksen tiedot tulevat tiedoston alkuun vanhempien päivitysten yläpuolelle. (Wickham & Bryan 2022: s. 20 Releasing a package.)

Tämän jälkeen paketin voikin yrittää saada julkaistuksi CRANissa. Paketin voi ladata kirjoittamalla konsoliin devtools::release(). Julkaisufunktio ajaa R CMD checkin ja esittää muutaman kysymyksiä, joihin vastataan kyllä tai ei. Sen jälkeen täytyy vain odottaa, sillä CRANia pyöritetään vapaaehtoisjoukoin, joten paketin hyväksyntä vie aikaa. Jos pakettia ei hyväksytä, voi sitä parannella ja yrittää saada se hyväksytyksi uudemman kerran. (Wickham & Bryan 2022: s. 20.6 Release.)

Kun paketti on julkaistu, ei sitä voi vain jättää oman onnensa nojaan, vaan sitä täytyy muistaa päivittää, tai sitten voi yrittää saada jotain muuta tahoa vastaamaan päivityksistä. Tarvetta päivityksille tulee luultavasti ainakin, kun koodista löytää virheitä tai Netezza päivittyy.

5.2 Paketin lisenssi

CRANiin vaaditaan lisenssi paketille. Lisenssi kannattaa valita jo siinä vaiheessa, kun paketti laitetaan jakoon GitHubiin. Periaatteessa avoimen lähdekoodin tuotoksien lisenssille vaihtoehtoja on kaksi: General Public License (tunnetaan myös nimillä GPL, the GNU GPL tai GPL-3) ja MIT.

GPL-lisenssi antaa kenelle tahansa oikeuden käyttää, kopioida, muuttaa sekä jakaa ohjelmaa ja sen lähdekoodia. Lisenssi säilyy myös, jos koodia muutellaan ja sitä levitetään eteenpäin eli se on copyleft-lisenssi. (Peng ym. 2020: s. 3.7.1 The General Public License.)

MIT-lisenssillä olevaa koodia voidaan käyttää ohjelmaan, joka ei ole avoimen lähdekoodin järjestelmä. MIT-lisenssillä varustettua koodia voi vapaasti kopioida, muokata ja käyttää omassa projektissa. Ehtona on, että sen lisenssiteksti säilyy lähdekoodissa. (Peng ym. 2020: s. 3.7.2 The MIT License.)

5.3 Vaihtoehdot Netezzalle

Työssä isona ongelmana oli, että Netezza on todella vähän käytetty tietokantajärjestelmä ja kohdattuihin ongelmiin oli hankala löytää apua. Vaihtoehtoja ei ole kauheasti tarjolla, koska tietokanta halutaan pitää omalla fyysisellä palvelimellaan pilven sijasta. Toisaalta tietokantajärjestelmän vaihtaminen ei ole ihan pieni juttu ja toisessa järjestelmässä olisi luultavasti toisenlaisia ongelmia.

Parhaiten Netezzan saisi korvattua Teradatalla tai Verticalla. Teradata ja Vertica voisivat korvata Netezzan, koska ne tarjoavat datan tallennusta fyysiselle palvelimelle pilven sijasta. Kaikki niistä on suunniteltu erityisen suurille datamäärille. Kaikista on myös saatavilla pilviversiot, ja ne tarjoavat työkaluja datan analysointia varten. Teradata on näistä kolmesta suosituin, Vertica ja Netezza ovat suurin piirtein yhtä suosittuja. Kaikkien pitäisi ymmärtää R-kieltä. (DB-Engines; Netezza Performance Server under Cloud Pak for Data.) Teradata on ai-

nut näistä kolmesta, jolla on olemassa taustajärjestelmä dbplyrillä. Tämä taustajärjestelmä kuuluu jo valmiiksi dbplyr-pakettiin. Luultavasti Teradatan voisi siis olettaa alkavan toimimaan hyvinkin helposti R:n ja Netezzan välillä.

Suuria eroja tietokantajärjestelmien väliltä löytyi vain muutama. Netezza käyttää karttarakennetta indeksoinnin sijasta. Teradata käyttää systeemissään useita palvelimia, kun taas Netezza käyttää vain yhtä.

5.4 Vaihtoehdot R:lle

R:lle vaihtoehtoja ovat Julia ja Python. Julia on tällä hetkellä huomattavasti R:ää vähemmän käytetty kieli. Maaliskuussa 2023 TIOBE-listauksessa se on sijalla 33, ja kaikesta maailmalla kirjoitetusta koodista on 0,29 % kirjoitettu Julialla. R:llä on kirjoitettu 0,93 %, eli se on yli kolme kertaa suositumpi. Julialla on mahdollista toteuttaa samoja toiminnollisuuksia kuin dplyrillä. Julia on saanut vaikutteita R:stä, joten siinä on tuttuja elementtejä R:n käyttäjille.

Python on tällä hetkellä suosituin ohjelmointikieli, TIOBE-listauksessa se on tällä hetkellä ykkönen. Sen suosio on ollut selkeässä nousussa vuoden 2017 lopusta lähtien. Toisin kuin Julia ja R, Pythonia ei ole tehty datan analysointikieleksi, vaan se on yleiskieli, johon on kuitenkin saatavilla paljon datan analysointiin ja muokkaukseen sopivia paketteja.

Käytetyn kielen vaihtaminen ei ole helppo ja nopea työ. Odotettavaa on, että tulee joitain toisia ongelmia Netezzan ja uuden kielen välillä. Työntekijät eivät myöskään hetkessä opi uutta kieltä. R taustalla Julian tai Pythonin oppiminen ei ole mikään mahdoton tehtävä, mutta siihenkin menee aikaa ja tämä aika on pois muista työtehtävistä.

5.5 Ratkaisu on olemassa, mutta se on maksullinen

Viisauden ”pyörää ei pidä keksiä uudelleen” mukaan on turha keksiä ratkaisua ongelmaan, joka on jo ratkaistu. Koko tämän työn ongelma on jo ratkaistu, ja

ratkaisu on saatavilla Rstudion maksullisessa RStudio Desktop Pro -versiossa (RStudio). Käyttämäni budjetin vuoksi maksullista versiota ei ollut mahdollista hankkia, joten jouduin ”keksimään pyörän uudelleen”.

Tieto siitä, että ratkaisu on olemassa, vei välillä motivaatiota työltä, mutta toisaalta se oli myös paluu R:n syntymään. Kielihän kehiteltiin, kun haluttiin käyttää S-kieltä, mutta ei ollut rahaa maksaa siitä.

6 Yhteenveto

Työn tarkoituksena oli tuottaa dbplyriin taustajärjestelmä Netezzalle. Valmiin taustajärjestelmän olisi tarkoitus helpottaa dbplyr-paketin käyttöä Netezzan ja Rstudion välillä. Käyttäjä voi hakea tietoja Netezzan tietokannasta käyttäen dplyr-funktioita, eikä hänen tarvitse miettiä vaihtoehtoisia toimintatapoja funktioille, kun Netezza ei tunnista komentoja. Käyttäjälle ei ollut lähtötilanteessa tietoa, toimivatko hänen käyttämänsä dplyr-funktiot, kun hän käytti niitä datan analysointiin ja muokkaukseen Netezzassa.

Työn lähtötilanteessa oli ilmassa epäilys, että tämä saattaisi olla parhaimmillaan todella helposti toteuttavissa. Kaikki vaan alkaisi toimia ihan pienellä koodailulla. Tietenkään asia ei mennyt niin. Taustajärjestelmää varten piti tehdä uusi taustajärjestelmä DBI:lle. En tehnyt uutta taustajärjestelmää DBI:lle, vaan sen teki minua suunnattomasti avustanut henkilö.

Kun DBI:n taustajärjestelmä oli tehty, edessä oli pitkä ja uuvuttava testausvaihe, jossa testattiin kaikki dbplyrin taustajärjestelmässä olevat funktiot. Toimimattomille funktioille tehtiin toiminnallisuudet dbplyrin taustajärjestelmään Netezzalle. Moni funktio toimii tällä hetkellä siellä, mutta ikävä kyllä kaikkia ei tämän työn rajoissa saatu toimimaan.

Tämä taustajärjestelmä tulee aidosti käyttöön, ja sen kehitys jatkuu. Tulen itsekin käyttämään tätä taustajärjestelmää työtehtävissäni, ja se helpottaa tehtävien tekemistä.

Lähteet

CRAN. Contributed Packages. Verkkoaineisto. <https://cran.r-project.org/> Luettu 11.3.2022.

DBeaver Community. About. Verkkoaineisto. <https://dbeaver.io/about/> Luettu 8.3.2022.

DB-engines. 2022. Netezza Performance Server under Cloud Pak for Data. Verkkoaineisto. <https://db-engines.com/en/system/Netezza%3BTeradata%3BVertica> Luettu 22.6.2022.

David, James. History of DBI. Verkkoaineisto. <https://dbi.r-dbi.org/articles/dbi-history> Luettu 14.3.2022.

Francisco, Phil. 2014. IBM PureData System for Analytics Architecture A Platform for High Performance Data Warehousing and Analytics. Verkkoaineisto. <https://www.redbooks.ibm.com/redpapers/pdfs/redp4725.pdf> Luettu 20.6.2022.

GNU General Public License, version 2. Verkkoaineisto. http://www.gnu.org/licenses/old-licenses/gpl-2.0.html Luettu 19.4.2022.

Grieder Silvia, Steiner Markus D. 2021. Algorithmic jingle jungle: A comparison of implementations of principal axis factoring and promax rotation in R and SPSS. Behavior Research Methods volume 54, 2022. s. 54–74.

Import, Export, and Convert Data Files. 8.11.2021. <https://cran.r-project.org/web/packages/rio/vignettes/rio.html> Luettu 27.7.2022.

Indicative Team. What Is Massively Parallel Processing (MPP). Verkkoaineisto. <https://www.indicative.com/resource/what-is-massively-parallel-processing-mpp/> Luettu 20.6.2022.

Introduction to roxygen2. <https://cran.r-project.org/web/packages/roxygen2/vignettes/roxygen2.html> Luettu 15.4.2022.

Kojima, Hide. 28.9.2017. Using dplyr to query databases directly instead of using SQL <https://blog.exploratory.io/using-dplyr-to-query-databases-directly-instead-of-using-sql-fed43f059ed6> Luettu 23.6.2022.

Jones Megan A, Guarinello Marisa, Soderberg, Courtney, Wasser, Leah A. 13.5.2021. Time Series 02: Dealing With Dates & Times in R - as.Date, POSIXct, POSIXlt. Verkkoaineisto. <https://www.neonscience.org/resources/learning-hub/tutorials/dc-convert-date-time-posix-r> Luettu 18.4.2022.

Jozef's Rblog. 23.6.2018. A primer in using Java from R – part 1 <https://www.r-bloggers.com/2018/06/a-primer-in-using-java-from-r-part-1/> Luettu 27.4.2022.

Laskov, Boris. 21.4.2017. Tool for metadata extraction from database Netezza, Czech Technical University in Prague Faculty of Information Technology Department of Software Engineering. Bachelor's thesis. s. 6.

Linux host operating system requirements. 31.3.2021. Verkkoaineisto. <https://www.ibm.com/docs/en/psfa/7.2.1?topic=appliance-linux-host-operating-system-requirements> Luettu 15.4.2022.

Leishman, Ross. 28.05.2018. Netezza Zone Maps — They're The Same As Indexes, Right? Verkkoaineisto. <https://medium.com/@rleishman/netezza-zone-maps-theyre-the-same-as-indexes-right-fb3249f01cea> Luettu 21.6.2022.

McKnight, William. 04.2005. 80, Introducing the Data Warehouse Appliance, Part 2. DM Review; New York Vol. 15, Iss. 4.

Netezza Corporation History. Verkkoaineisto. <http://www.fundinguniverse.com/company-histories/netezza-corporation-history/> Luettu: 29.4.2022.

Netezza database schema overview. 30.3.2021. Verkkoaineisto. <https://www.ibm.com/docs/en/psfa/7.2.1?topic=tables-netezza-database-schema-overview> Luettu 1.6.2022.

Netezza identifiers. 5.3.2021. Verkkoaineisto. <https://www.ibm.com/docs/en/iis/9.1?topic=connector-netezza-identifiers>, Luettu 1.6.2022.

NZPLSQL language. 12.5.2022. Verkkoaineisto. <https://www.ibm.com/docs/en/netezza?topic=sp-nzplsql-language-2> Luettu 6.6.2022.

Package 'DBI'. 2021. Verkkoaineisto. <https://cran.r-project.org/web/packages/DBI/DBI.pdf> Luettu 20.3.2022.

Patterson, David, Hennessy, John. 2018. Computer Organisation and Design: The Hardware/Software Interface (RISC-V ed.). Cambridge, United States: Morgan Kaufmann. s. 509.

Peng, Roger D, Kross, Sean, Anderson, Brooke. 2020. Mastering Software Development in R. Verkkoaineisto <https://bookdown.org/rdpeng/RProgDA/>. 20.12.2020. Luettu 1.3.2022.

Peng, Roger D. 3.9.2020. R Programming for Data Science: 2 History and Overview of R. Verkkoaineisto. <https://bookdown.org/rdpeng/rprogdatascience/history-and-overview-of-r.html> Luettu 2.3.2022.

RDocumentation. Get/set seconds component of a date-time. Verkkoaineisto. <https://www.rdocumentation.org/packages/lubridate/versions/1.8.0/topics/second> Luettu: 24.4.2022.

RStudio. Verkkoaineisto. <https://www.rstudio.com/products/rstudio/> Luettu 18.3.2022.

Tibken, Shara. 20.9.2010. IBM Scoops Up Data Firm Netezza. The Wall Street Journal. Verkkoaineisto. <https://www.wsj.com/articles/SB10001424052748703989304575503554245091876> Luettu 29.4.2022.

Tidyverse packages. Verkkoaineisto <https://www.tidyverse.org/packages/> Luettu 12.4.2022.

TIOBE Index for March 2023. Verkkoaineisto <https://www.tiobe.com/tiobe-index/> Luettu 27.3.2023.

The Comprehensive R Archive Network. Verkkoaineisto. <https://cran.r-project.org/> Luettu 30.3.2022.

Velasquez, Isabella. 12.6. Three Ways to Program in Python With RStudio. Verkkoaineisto. <https://www.rstudio.com/blog/three-ways-to-program-in-python-with-rstudio/> Luettu 27.4.2022.

Wickham, Hadley. 2021. Advanced R- Object-oriented programming Introduction. Verkkoaineisto. <https://adv-r.hadley.nz/oo.html> Luettu 3.3.2022.

Wickham, Hadley. dblyr. Verkkoaineisto. <https://dbplyr.tidyverse.org/> Luettu 12.3.2022.

Wickham, Hadley, Grolemund, Garrett. 21.6.2022. R for Data Science. Verkkoaineisto < https://r4ds.had.co.nz/index.html> Luettu 27.7.2022.

Wickham Hadley, Bryan Jennifer. 2022. R Packages. Verkkoaineisto. <https://r-pkgs.org/index.html> Luettu 12.3.2022.

Wickham, Hadley, François, Romain, Henry, Lionel, Müller Kirill. dplyr – Overview. Verkkoaineisto. <https://dplyr.tidyverse.org//> Luettu 12.3.2022.

Wickham Hadley, Girlich Maximilian, Ruiz Edgar. a. Function translation. Verkkoaineisto. <https://dbplyr.tidyverse.org/articles/translation-function.html> Luettu 22.6.2022.

Wickham Hadley, Girlich Maximilian, Ruiz Edgar. b. Verb translation. Verkkoaineisto. <https://dbplyr.tidyverse.org/articles/translation-verb.html> Luettu 20.4.2022.

Wickham Hadley, Girlich Maximilian, Ruiz Edgar. c. Writing SQL with dbplyr. Verkkoaineisto. <https://dbplyr.tidyverse.org/articles/sql.html> Luettu 30.4.2022.

Wickham, Hadley, Müller, Kirill. R Consortium, DBI. Verkkoaineisto. <https://dbi.r-dbi.org/> Luettu 11.3.2022.

Wickham, Hadley, Müller, Kirill. 16.6.2003. A Common Database Interface (DBI). Verkkoaineisto. <https://dbi.r-dbi.org/articles/dbi-1> Luettu: 1.5.2022.

Liitteet

Liite1: Testeissä havaitut taustajärjestelmää tarvitsevat funktiot

%/%
^
log10
as.character()
as.logical()
pmin()
pmax()
xor()
quantile()
paste()
paste0()
substring()
substr()
cov()
cor()
sd()
nth()
second()
str_to_title()
str_c()
str_conv()
str_count()
str_detect()
str_dup()
str_extract()
str_extract_all()
str_flatten()
str_glue()
str_glue_data()
str_interp()
str_locate()
str_locate_all()
str_match()
str_match_all()
str_order()
str_pad()
str_remove()
str_remove_all()
str_replace()
str_replace_all()
str_replace_na()
str_sort()

Liite1 (2)

str_split()
str_split_fixed()
str_squish()
str_subset()
str_trunc()
str_view()
str_view_all()
str_which()
str_wrap()