jamk.fi

Datamigraation testauksen automatisointi

Eetu Niskanen

Opinnäytetyö
Toukokuu 2021
Tietojenkäsittely ja tietoliikenne
Insinööri (AMK), Tieto- ja viestintätekniikka

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Niskanen, Eetu	Opinnäytetyö, AMK	Toukokuu 2021
	Sivumäärä 53	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi
Datamigraation testauksen automatisointi

Tutkinto-ohjelma
Insinööri (AMK), Tieto- ja viestintätekniikka

Työn ohjaaja(t)
Esa Salmikangas

Toimeksiantaja(t)
Pinja Digital Oy

Tiivistelmä

Työn toimeksiantajana toimi Pinja Digital Oy. Toimeksiantajalla oli tarve selvittää datamigraatioiden jälkeisen testauksen automatisoinnin mahdollisuuksia ja soveltaa käyttöön työkalu, jolla testausta voitaisiin toteuttaa automaattisesti ja kustannustehokkaasti. Näiden tarpeiden pohjalta alettiin tutkimaan datamigraatioiden testausstrategioita ja automaattisen testauksen työkaluja sekä sovellettiin toimeksiantajan käyttöön työkalu migraation automaattitestausta varten.

Työn teoriaosuudessa on kuvailtu datamigraatiota projektina, käyty läpi datamigraation testausmenetelmiä sekä vertailtu erilaisia ETL-työkaluja, joilla datamigraation testausta voidaan automatisoida. Työkalujen vertailun jälkeen päädyttiin lopputulokseen, että Talend Open Studio for Data Integration -sovellus sopisi parhaiten toimeksiantajan tarpeisiin.

Suunnittelu- ja toteutusvaiheessa aloitettiin suunnittelemaan tarkempien tarpeiden pohjalta kolmea eri testaustapaa, joilla jokaisella saatiin erityyppistä informaatiota datamigraatiosta ja datan siirtymisestä lähde- ja kohdejärjestelmän välillä. Testausmenetelmillä haluttiin varmistaa, että data siirtyy ehjänä ja kohdejärjestelmän tietokantarakenteen mukaisena, data on kohdejärjestelmän logiikan mukaista, ja siirrettävä data vastaa migraation määrityksiä.

Lopputuloksena saatiin sovellettua tarvittavat testausmenetelmät, joilla todettiin tarpeiden mukaisten testausten olevan mahdollista toteuttaa. Jokainen datamigraatio on toki määrityksiltään erilainen ja täten myös testausmenetelmät vaativat yksityiskohtaisia määrityksiä, mutta kokonaisuutena opinnäytetyön tavoitteisiin päästiin.

Avainsanat (asiasanat)
Datamigraatio, testaus, automatisointi, ETL, Talend Open Studio

Muut tiedot (Salassa pidettävät liitteet)

jamk.fi

Description

Author(s)	Type of publication	Date
Niskanen, Eetu	Bachelor's thesis	May 2021
		Language of publication: Finnish
	53	Permission for web publication: x

Title of publication
Automatization in data migration testing

Degree programme
Information and Communication Technology

Supervisor(s)
Salmikangas, Esa

Assigned by
Pinja Digital Oy

Abstract

The research was assigned by Pinja Digital Oy. The client needed to find out the possibility to automate the post-data migration testing and to apply a tool that could be used to perform testing automatically and cost-effectively. Based on these needs, data migration testing strategies and automated testing tools were studied, and a tool for automated migration testing was deployed for the client.

The theoretical part of the research was to describe the data migration as a project and the data migration testing methods. Various ETL-tools that can be used to automate data migration testing was also compared. After comparing the tools, it was concluded that the Talend Open Studio for Data Integration would be the application that would be the best suited for the client's use.

In the design and implementation phase of the research, three different test methods were started to be designed based on more specific needs. Each of these methods were to provide different types of information about the data migration and data transfer between the source's system and the destination's system. The test methods were intended to ensure that the data is transferred intact and in accordance with the target system's database structure, that the data is in accordance with the target system's logic and finally that the data that is to be transferred corresponds to the migration specifications.

As a result, the necessary testing methods were applied, which showed that it was possible to carry out all the tests according to needs. Each data migration is different in its core definitions and thus the testing methods require further and detailed determinations, but as a whole the goals of the thesis were achieved.

Keywords/tags (subjects)
Data migration, testing, automatization, ETL, Talend Open Studio

Miscellaneous (Confidential information)

Sisältö

1 Johdanto 4

1.1 Toimeksiantaja 4

1.2 Työn tavoitteet 4

1.3 Tutkimusmenetelmä 5

2 Datamigraatio 5

2.1 Datamigraatio projektina 5

2.2 Datamigraatioprojektin vaiheet 6

2.3 Datamigraation testausstrategia 12

3 Työkalut 16

3.1 ETL-työkalut 16

3.2 RightData 16

3.3 Datagaps ETL Validator 17

3.4 Informatica Data Validation 17

3.5 iCEDQ 17

3.6 QuerySurge 18

3.7 Talend Open Studio for Data Integration 18

3.8 Yhteenveto työkaluista 19

4 Nykytilanne 19

4.1 Migraatioiden toteutus 19

4.2 Migraatioiden testaus 21

5 Toteutus 21

5.1 Aloitus 21

5.2 Schema Compliance Test 24

5.3 Target Logicality Test 33

5.4 Data Matches Spec Test 40

6 Tulokset 46

7 Pohdinta 47

Lähteet 49

Kuviot

Kuvio 1. IBM:n ehdottama kolmivaiheinen migraatioprosessi 6
Kuvio 2. Matthes & Schulzin kuvaama datamigraation prosessimalli 8
Kuvio 3. Datamigraatioalustan toiminnallisuus 9
Kuvio 4. Suositukset testausstrategian laatimisen avuksi 13
Kuvio 5. Työtilan perustaminen 22
Kuvio 6. Uuden projektin perustaminen 22
Kuvio 7. Talend Open Studion oletusnäkymä 23
Kuvio 8. Uuden SchemaComplianceTest -tehtävän lisääminen 25
Kuvio 9. CSV-tiedoston määrittäminen 25
Kuvio 10. CSV-tiedoston määrittäminen Input-komponenttina 26
Kuvio 11. SchemaComplianceCheck-komponentin lisääminen 26
Kuvio 12. Komponenttien yhdistäminen Row-Main liitoksella 27
Kuvio 13. SchemaComplianceCheck-komponentin määritykset 28
Kuvio 14. SchemaComplianceCheck-komponentin ulostulon valinta 29
Kuvio 15. LogRow-komponentin lisääminen tehtävälle 29
Kuvio 16. SchemaComplianceTest-tehtävän suorittaminen 30
Kuvio 17. tMap-komponentin lisääminen tehtävälle 30
Kuvio 18. tMap-komponentin määritykset datan suodatusta varten 31
Kuvio 19. SchemaComplianceTest-tehtävän tulos 32
Kuvio 20. Valmis SchemaComplianceTest-tehtävä 32
Kuvio 21. MySQL-tietokantayhteyden muodostaminen 34
Kuvio 22. tDBInput-komponentin tarkemmat määritykset 35
Kuvio 23. tJavaRow-komponentin lisääminen tehtävälle 36
Kuvio 24. tJavaRow-komponenttiin määritetty logiikkatarkastelun koodi 37
Kuvio 25. TargetLogicalityTest-tehtävän tMap-komponentin määritykset 38

Kuvio 26. tFileOutPutExcel-komponentin määritykset 39
Kuvio 27. TargetLogicalityTest-tehtävän tulos 39
Kuvio 28. Valmis TargetLogicalityTest-tehtävä..... 40
Kuvio 29. Vertailtavat datakomponentit DataMatchesSpecTest-tehtävällä..... 41
Kuvio 30. DataMatchesSpecTest-tehtävän ensimmäinen tMap-komponentti.... 42
Kuvio 31. Vertailtavien taulujen yhdistäminen tMap-komponentissa..... 43
Kuvio 32. tMap-komponentin määritykset liitosta varten 44
Kuvio 33. DataMatchesSpecTest-tehtävän tulos..... 45
Kuvio 34. Valmis DataMatchesSpecTest-tehtävä 45

1 Johdanto

1.1 Toimeksiantaja

Opinnäytetyön toimeksiantajana toimi jyväskyläläinen Pinja Digital Oy, joka tuottaa palveluita IT-konsultoinnin ja -palveluiden sekä sovellusten ja ohjelmistojen toimialoilla. Toimeksiantajalla oli tarve tutkia ja kehittää datamigraatioiden jälkeistä testausta ja tavoitteena oli selvittää menetelmiä, joilla toimeksiantaja voi suorittaa testauksen automaattisesti migraatioprojekteissa. Nykytilanteessa datamigraation jälkeistä testausta tehdään manuaalisesti vertailemalla satunnaisjoukkojen osalta dataa kohde- ja lähdejärjestelmän välillä.

Pinja Digital on osa Pinja-konsernia, joka tuottaa asiakkailleen ylläpito-, pilvialusta-, ICT-, tietoturva- ja tukipalveluita tuotantokriittisiin ympäristöihin. Pinja syntyi maaliskuussa 2020, kun Protacon ja siihen kasvun myötä aiempina vuosina mukaan tulleet ARROW, SWD, Descal, Netwell, Vision Systems sekä Powen yhdistyivät yhden nimen alle. Pinjalla työskentelee Suomessa yli 550 henkilöä, jotka palvelevat suomalaisia teollisuus, ja yritysasiakkaita sekä kansainvälisiä organisaatioita jopa yli 30 eri maassa. (Me olemme Pinja n.d.)

1.2 Työn tavoitteet

Työn tavoitteena oli tutkia ja soveltaa käyttöön menetelmä, jolla datamigraation jälkeinen testaus voitaisiin suorittaa automaattisesti. Toimeksiantajan aikaisemmat käytännöt datamigraation jälkeisessä testauksessa eivät tuottaneet absoluuttista varmuutta migraation onnistumisesta ja koska testaus suoritettiin manuaalisesti, olivat menetelmät työmäärällisesti kuormittavammat. Opinnäytetyön tavoitteena oli tutkia datamigraatioiden testausstrategioita ja testaustyökaluja sekä soveltaa toimeksiantajan käyttöön kustannustehokkain automaattisen testauksen työkalu.

1.3 Tutkimusmenetelmä

Opinnäytetyö oli rakenteeltaan tutkimuksellista kehittämistyötä, koska tavoitteena oli selvittää ja soveltaa automaattisen testauksen tapoja toimeksiantajan käyttöön datamigraatioita varten. Työssä keskityttiin datamigraatioon prosessina, migraation testausmenetelmiin ja testaustyökaluihin. Työllä pyrittiin ratkaisemaan toimeksiantajan ongelma ja antaa vastaus kysymykseen: Kuinka datamigraation testaus voidaan toteuttaa automaattisesti? Tutkimuksessa ja työssä hyödynnetään koulusta opittuja tietoja ja taitoja, töiden ohessa opittuja käytänteitä sekä aiheesta tehtyjä tutkimuksia, artikkeleita ja kirjallisuutta.

2 Datamigraatio

2.1 Datamigraatio projektina

Datamigraatio on tietojärjestelmien uudistuksiin liittyvä prosessi, jossa dataa siirretään vanhasta järjestelmästä uuteen järjestelmään. Datamigraatiossa dataa ei kuitenkaan pelkästään siirretä järjestelmästä toiseen, vaan data puretaan ensin lähtöjärjestelmästä, jonka jälkeen dataa valmistellaan siirtoa varten. Dataa on yleensä tarve käsitellä ja konvertoida, jotta se sopii kohdejärjestelmään. Datan käsittelyn jälkeen konvertoitu data viedään kohdejärjestelmään.

Datamigraatio on yleensä tarpeellinen seuraavissa tapauksissa:

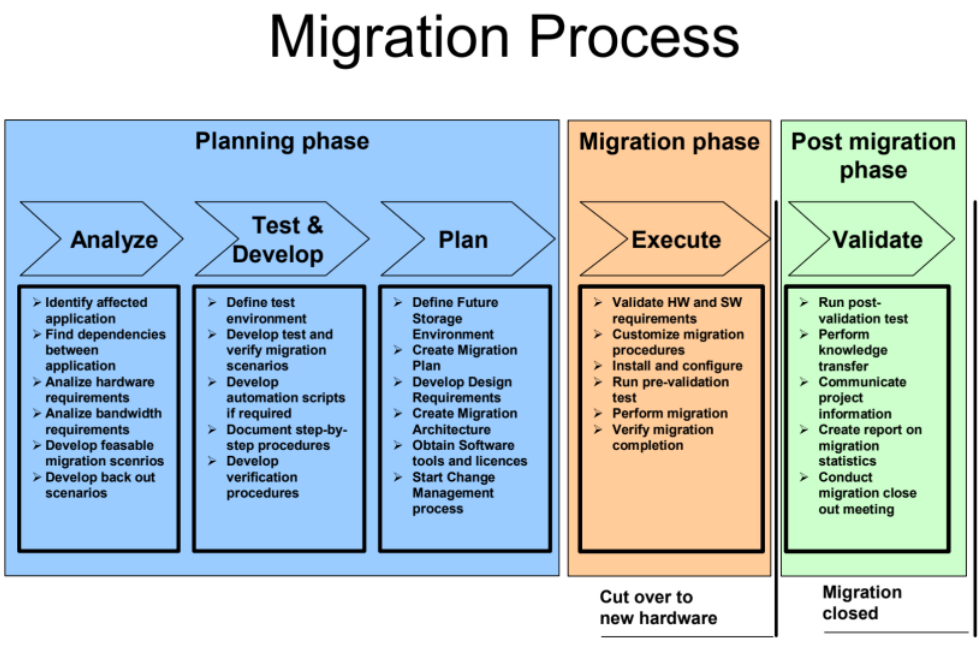
1. Yksi tai useampi tietovarasto täytyy korvata. Korvaus saattaa olla tarpeellinen, mikäli nykyisen tietovaraston kapasiteetti loppuu tai suorituskyky heikkenee.
2. Nykyisen tietovaraston fyysinen jalanjälki on saatava pienemmäksi. Uudet tietojärjestelmämallit mahdollistavat vanhempien tietojärjestelmien yhdistämisen yhdeksi järjestelmäksi, joka tarjoaa suuremman kapasiteetin, parempaa suorituskykyä ja jopa vähemmän virrankulutusta

3. Fyysinen siirtyminen uuteen järjestelmään edellyttää tietojen siirtämistä ja migraatiota uuteen tietojärjestelmään. Fyysinen siirtyminen on tarpeellista, kun uusi tietojärjestelmä otetaan käyttöön, kun nykyinen järjestelmä suljetaan tai kun nykyisen järjestelmän käyttö on estetty huomattavan pitkän ajan ylläpidon takia.

Nämä esimerkit kertovat, että datamigraatio itsessään on vain yksi osa suurempaa projektia. Oikein suunniteltu datamigraatio on prosessina kuitenkin merkittävä ja vaatii erityisiä tietoja ja taitoja, jotta migraatio voidaan saattaa loppuun onnistuneesti. (Dufrasne, Warmuth, Appel, Bauer, Douglass, Klee, Pura, Wells, & Wesselbaum 2017, 2.)

2.2 Datamigraatioprojektin vaiheet

Lähestulkoon jokainen tavanomainen datamigraatioprosessi noudattaa Kuviossa 1 esitettyä kolmivaiheista perusrakennetta. Teknologiayritys IBM:n mukaan datamigraatio koostuu yleensä kolmesta vaiheesta: suunnitteluvaiheesta, migraatiovaiheesta ja migraation jälkeisestä vaiheesta (Dufrasne ym. 2017, 4).



Kuvio 1. IBM:n ehdottama kolmivaiheinen migraatioprosessi (Dufrasne ym. 2017, 4.)

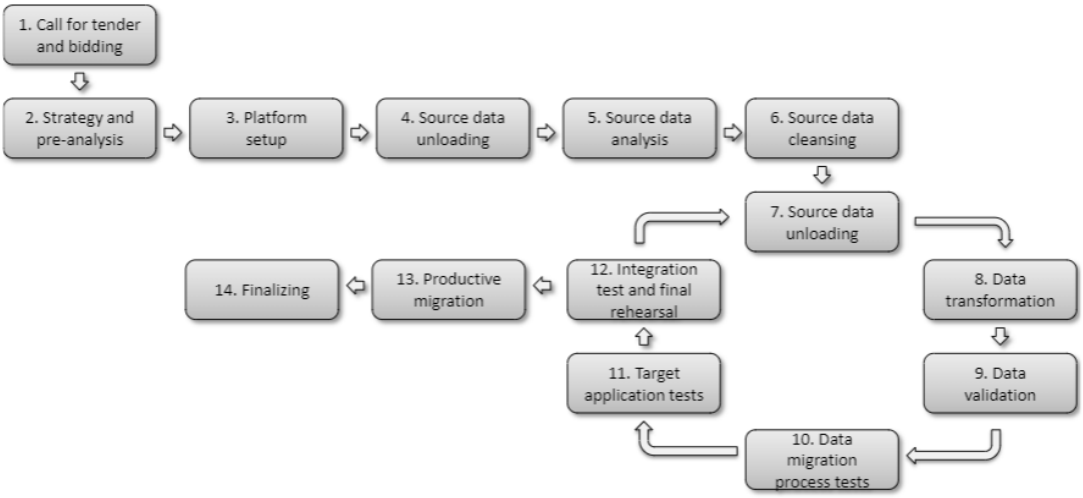
Migraatioprojektin suunnitteluvaihe jakautuu vielä kolmeen pienempään osaan; analysointiin, testaukseen ja tuotantoon, sekä suunnitteluun. Analysoinnissa kerätään mahdollisimman paljon tietoa lähdejärjestelmästä ja kohdejärjestelmästä sekä tässä vaiheessa yleensä myös saadaan jo haltuun ja tarkasteltavaksi data, jota ollaan siirtämässä. Testaus ja tuotanto-osiossa luodaan testausympäristöt, määritellään mahdolliset konversiot ja automatisoidaan vaiheita mahdollisimman paljon. Suunnitteluosiossa määritellään vielä migraation läpivientisuunnitelma ja varmistetaan että kaikki on valmiina itse seuraavaa migraatiovaihetta varten. Suunnitteluvaiheessa siis analysoidaan nykyistä tilannetta ennen varsinaista migraatiota. Tässä vaiheessa määritetään mihin sovelluksiin ja osa-alueisiin migraatio tulee vaikuttamaan, määritellään vaatimukset ja rajoitukset. Tämän vaiheen lopputuloksena päästään tilanteeseen, jossa migraatio voidaan toteuttaa. (Dufrasne ym. 2017, 4.)

Varsinaisessa migraatiovaiheessa datamigraatio toteutetaan edellisessä vaiheessa määritetyillä työkaluilla, konversioilla ja suunnitelmilla. Ennen kuin migraatio toteutetaan tuotantoon, tehdään vielä viimeiset validoinnit ja testaukset. Kun kaikki on valmiina tuotantoon siirtymistä varten, migraatio ja yliheitto uuteen järjestelmään voidaan toteuttaa.

Migraation jälkeisessä vaiheessa varmistetaan, että datamigraatio on valmistunut ja että kaikki toiminnallisuudet toimivat oikein kohdejärjestelmässä. Migraation jälkeisissä testauksissa havaitaan, mikäli data ei ole oikeassa muodossa kohdejärjestelmässä tai mikäli muita ongelmia kuten suorituskyvyllisiä ongelmia järjestelmässä ilmenee. Mikäli migraatio on ollut onnistunut eikä ongelmia ilmene, voidaan sanoa, että uusi järjestelmä on ottanut haltuunsa datan onnistuneesti. On tärkeää, että ongelmatapauksissa on mahdollista perääntyä tilanteeseen ennen migraatiota. Jos datassa tai järjestelmässä havaitaan virheitä migraation jälkeen, voidaan ongelmat korjata joko saman tien tai perääntyä kokonaan migraatiota aikaisempaan lähtötilanteeseen. Mikäli peruutetaan tilanteeseen ennen migraatiota, voidaan ongelman aiheuttanut asia korjata ja toteuttaa migraatiovaihe uudelleen. (Dufrasne ym. 2017, 4–15.)

Kuviossa 2 kuvataan vielä tarkemmin, kuinka datamigraatioprosessi etenee vaihe vaiheelta. Kuvattu malli on suuntaa antava, ja antaa esimerkkimenetelmiä, joiden avulla

datamigraatio voidaan toteuttaa. Mallista voidaan hyvin jättää joitain vaiheita suorittamatta, mikäli ne katsotaan tarpeettomaksi. Seuraavaksi myös käydään läpi tämän prosessimallin tärkeimmät vaiheet.



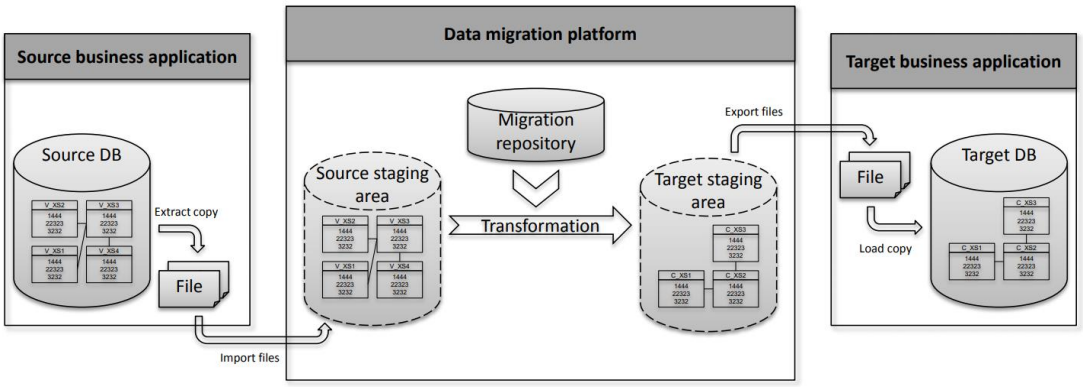
Kuvio 2. Matthes & Schulzin (2011, 18) kuvaama datamigraation prosessimalli

Datamigraatiot alkavat usein siitä, että asiakas ilmoittaa markkinoille tarpeestaan tehdä datamigraatio järjestelmästä toiseen. Tämä vaihe vastaa kuviossa 2 olevaa ensimmäistä kohtaa (Call for tender and bidding) ja tässä vaiheessa usein joko asiakkaan kumppanina jo toimiva järjestelmätoimittaja antaa tarjouksen datamigraatioprojektin toteuttamisesta, tai jokin toinen datamigraatiopalvelua tarjoava osapuoli antaa kyseisestä projektista tarjouksen. Ensimmäisen vaiheen lopputuloksena on saada datamigraatioprojektia tarjoavalle toimittajalle alustava ymmärrys datamigraation laajuudesta, lähdejärjestelmän datasta ja rajoituksista, sekä konvertoitavan datan määrityksistä. Lopputuloksena on lisäksi asiakkaalle toimitettava tarjous projektista. (Matthes & Shulz 2011, 18–19.)

Kun tarjous on laadittu ja hyväksytty, siirrytään strategia ja esianalyysi -vaiheeseen (Strategy and preanalysis). Tässä vaiheessa huomio keskittyy yhden pääkysymyksen ympärille: ”Kuinka datamigraatio tullaan toteuttamaan?”. Vaiheessa tehdään tarkempaa analyysiä siitä, mitä dataa käsitellään ja miten sitä tullaan käsittelemään, jotta siirto järjestelmästä toiseen voidaan toteuttaa, sekä siitä, mitä organisatorisia,

teknisiä ja liiketoiminnallisia rajoituksia datamigraatiolla on. Vaiheessa keskitytään myös projektiteknisiin asioihin, kuten esimerkiksi aikatauluun ja projektiorganisaatioon. (Matthes & Shulz 2011, 19–20.)

Strategia-vaiheesta siirrytään prosessimallin kolmanteen vaiheeseen, joka on teknisen alustan pystyttäminen (Platform setup). Tässä vaiheessa määritellään työkalu, jolla migraatiotekniset osat, kuten datan käsittely ja konversiot sekä datan sisäänajo tullaan pääosin toteuttamaan. Matthes & Shulz:n (2011) mukaan datamigraatioalusta toimii lähdejärjestelmän ja kohdejärjestelmän välillä kuvion 3 mukaisesti. Määritetty alusta voi olla joko täysin manuaalinen tai se voi tarjota osittain automaattisia työkaluja helpottamaan migraatioprosessia. (Matthes & Shulz 2011, 28–29.)



Kuvio 3. Datamigraatioalustan toiminnallisuus. (Matthes & Shulz 2011, 29.)

Kun datamigraatioalusta on todettu toimivaksi työkaluksi toteuttaa datamigraatio, voidaan edetä seuraavaan vaiheeseen prosessissa, joka on lähtödatan purkaminen (Source data unloading). Lähtödatan purkamisessa otetaan lähdejärjestelmästä vain migraatioon liittyvä data ulos käsittelyä varten. Riippuen migraation toteutustavasta, laajuudesta ja monimutkaisuudesta, lähtödataa saatetaan joutua purkamaan useampaan kertaan, jotta voidaan työskennellä ajantasaisella datalla ja varmistaa, että migraatio toteutetaan oikein. Dataan on mahdollista päästä käsiksi suoraan tietokantayhteyksiä hyödyntämällä, tai data voidaan ottaa lähtöjärjestelmästä ulos ns. flat-

tiedostona, joka tunnetaan myös nk. tekstitietokantana, joka pitää sisällään datan tekstimuodossa. (Matthes & Shulz 2011, 32–34.)

Kun lähtödataa on mahdollista tarkastella, voidaan edetä prosessimallissa esiteltyihin vaiheisiin 5 – lähtödatan analyysi (Source data analysis) ja 6 – lähtödatan puhdistus (Source data cleansing). Lähtödatan analyysi -vaiheessa tarkastellaan dataa ja analysoidaan sen rakennetta ja laatua. Tämä vaihe kulkee käsikädessä seuraavana tehtävän lähtödatan puhdistus -vaiheen kanssa, sillä mikäli analyysivaiheessa havaitaan jotain virheitä tai puutteita, voidaan ne puhdistusvaiheessa korjata. Datan puhdistus ja korjaus on tärkeää, koska virheellinen data on korjattava sellaiseen muotoon, jonka kohdejärjestelmä hyväksyy. (Matthes & Shulz 2011, 35–41.)

Yleensä on hyvä tapa säilyttää asiakkaan toimittama lähtödata täydellisessä alkuperäisessä muodossaan ja käytettävänä mahdollisimman pitkään ja mielellään koko projektin ajan. Mikäli alkuperäinen, koskematon data on käytettävissä, mahdollistaa se konvertoidun datan tyhjentämisen ja tarkastelu- ja puhdistusvaiheiden aloittamisen, tai pahimmassa tapauksessa koko projektin aloittamisen kokonaan uudelleen, mikäli jotain merkittäviä ongelmia ilmenee projektin aikana. (Dufrasne ym. 2017, 4–15.)

Lähtödatan puhdistuksen jälkeen koittaa iteratiiviset vaiheet 7–12. Koska seitsemäs vaihe, eli lähtödatan purkaminen on jo kuvattu aikaisemmassa kappaleessa, siirrytään tarkastelemaan vaihetta 8, joka on datan transformaatio (Data transformation). Datan transformaatio -vaihe toistetaan useampaan kertaan ja niin pitkään, kunnes data saadaan konvertoitua konversiomääritysten mukaisesti. Jokaisella kerralla, kun tämä vaihe iteroidaan, konversiomääritykset sekä ohjelma, jolla datan käsittely toteutetaan, paranevat, joka tarkoittaa sitä, että päästään lähemmäksi sitä lopputulosta ja dataa, joka voidaan ajaa kohdejärjestelmään (Matthes & Shulz 2011, 41–42).

Kun data on transformoitu, on tälle datalle suoritettava validointia (Data validation). Tämä tarkoittaa käytännössä sitä, että käsiteltyä dataa tarkastellaan ja verrataan alkuperäiseen lähtödataan joko manuaalisesti tai automaattisesti. Aikaisemman vaiheen tavoin, myös tämä vaihe toistetaan iteroiden aina kun datan transformaatio -

vaihe on tehty. Mikäli datan validointia ja vertailua tehdään tässä vaiheessa automaattisesti, mahdollistaa se suurempien datajoukkojen tarkastuksen pienemmässä ajassa. Automaattisella vertailulla varmistetaan datan ja rakenteen oikeellisuus, täydellisyys ja johdonmukaisuus. Manuaalisessa vertailussa puolestaan käydään käsin läpi datajoukkoja ja yksittäisiä tietueita, joka on paljon työläämpää kuin automaattinen validointi. Manuaalista validointia voidaan tehdä tarvittaessa myös asiakkaan yhteyshenkilön kanssa, koska asiakas tietää aina enemmän ja paremmin sitä koskevasta datasta. Manuaalisessa tarkastuksessa voidaan hyödyntää myös tähän sopivia työkaluja, mikäli valittu datamigraatioalusta tukee näitä. (Matthes & Shulz 2011, 46–47.)

Datan validoinnin jälkeen seuraa datamigraatioprosessin testaus (Data migration process tests). Tässä vaiheessa tarkoituksena on tarkastella aikaisempien vaiheiden lopputuloksia ja selvittää sekä testata koko prosessin läpivientiä. Tätä vaihetta voidaan pitää ns. kuivaharjoitteluna migraation läpiviennille ja tarkoituksena on saada käsitys siitä, miten ja kuinka tehokkaasti tai virheettömästi prosessi voidaan suorittaa. Vaiheessa testataan käytännössä datan purkaminen ja lukeminen lähdejärjestelmästä, datan transformaatio ja testaus, sekä viimein käsitellyn datan vieminen testijärjestelmään. (Matthes & Shulz 2011, 50.)

Kun varsinainen datamigraatioprosessi on testattu ja todettu toimivaksi, siirrytään kohdejärjestelmän testaukseen (Target application tests), jossa testataan testijärjestelmän puolella, kuinka juuri sisään ajettu data käyttäytyy (Matthes & Shulz 2011, 51). Järjestelmän kokonaisvaltainen toimivuus ja toiminnot tulisi testata ja varmistaa että ainakin sisään ajettuun dataan liittyvät toiminnallisuudet toimivat ongelmitta.

Kun varsinainen kohdejärjestelmän testaus on suoritettu ja todettu ettei virheitä ole, voidaan edetä testaamaan kohdejärjestelmään liitettyjen sovellusten ja integraatioiden toimivuus uudella datalla sekä viimeiseen harjoitukseen (Integration tests and final rehearsal). Integraatiotestauksen tarkoituksena on varmistaa, että kohdejärjestelmä ja sen yhteydessä toimivat integraatiot ovat edelleen yhteensopivia ja toimivat myös uuden datan kanssa virheettömästi. Tässä yhteydessä voidaan suorittaa myös viimeinen harjoitus, jota voidaan ajatella ikään kuin migraatioiden kenraaliharjoituksena. Tässä harjoituksessa suoritetaan jokainen kuviossa 2 esitetyn prosessimallin

vaihe, ja toteutetaan migraatioprosessi alusta loppuun testijärjestelmään. Harjoituksessa siis testataan käytännössä koko prosessin toimivuus ja mikäli viimeisessä harjoituksessa ei havaita enää ongelmakohtia, voidaan todeta, että migraatio voidaan toteuttaa prosessin mukaisesti tuotantoon. (Matthes & Shulz 2011, 53–55.)

Kun kenraaliharjoitus on suoritettu onnistuneesti, käsittelyssä oleva data on todettu olevan mahdollista siirtää uuteen järjestelmään ja kaikki prosessin aikana tehdyt testit eivät ole aiheuttaneet ongelmia, migraatiosta vastuussa oleva tiimi voi viimein tehdä päätöksen toteuttaa migraation tuotannon kohdejärjestelmään (Productive migration). Mikäli tuotantoon siirtymiselle on annettu aikataulu, pyritään tietenkin noudattamaan suunniteltua aikataulua. Tuotantoon siirtyminen aiheuttaa mahdollisesti katkoa tai seisokkiaikaa kohdejärjestelmässä, joten olisi suositeltavaa, että tuotantoon siirtyminen toteutetaan viikonloppuisin tai pyhäpäivinä, jolloin vaikutetaan mahdollisimman vähän järjestelmää käyttävien henkilöiden tekemiseen (Matthes & Shulz 2011, 55).

Suoraan tuotannon migraation yhteydessä tai sen jälkeen alkaa projektin viimeistelyvaihe (Finalizing). Kun todetaan, että migraatio on toteutettu lähdejärjestelmästä kohdejärjestelmään onnistuneesti, voidaan yhdessä asiakaan kanssa sopia, että projekti on päättynyt ja toimitettu. Tämän jälkeen datan hallinta ja vastuu järjestelmästä siirtyy migraatiotiimiltä asiakkaan ja kohdejärjestelmän IT-tiimille. Projektin lopuksi on suositeltavaa, että migraatiotiimi dokumentoi kokemuksensa ja oppinsa, jotta he kehittyvät seuraavia migraatioprojekteja varten. (Matthes & Shulz 2011, 56.)

2.3 Datamigraation testausstrategia

Migraation testaus on erillinen prosessi datamigraatioprojektissa. Testauksessa varmistetaan, että data on siirtynyt lähdejärjestelmästä kohdejärjestelmään onnistuneesti, minimaalisella häiriö/seisokkiajalla ja ilman että yhtään tärkeää tietoa ei ole menetetty. Testauksessa myös saadaan varmistus siitä, että kohdejärjestelmässä kaikki toimivuuteen liittyvät näkökulmat ja vaatimukset täyttyvät. (Data Migration Testing Tutorial: A Complete Guide 2020.)

Syitä, miksi migraation testaus olisi hyvä toteuttaa varsinaisen migraation yhteydessä ovat Data Migration Testing Tutorial: A Complete Guide (2020) -artikkelin mukaan esimerkiksi:

1. Migraation takia aiheutuneet häiriöt käyttäjälle tulisi välttää tai minimoida
2. Tulisi varmistaa, että käyttäjä voi jatkaa järjestelmän käyttöä normaalisti sekä migraation aikana, että sen jälkeenkin
3. Tulisi pystyä ennakoimaan ja rajaamaan pois kaikki mahdolliset häiriöt, joita saattaa ilmaantua migraation aikana

Jotta voidaan minimoida migraation aiheuttamat häiriöt ja ongelmat, on hyvä suunnitella etukäteen testausstrategia. Hyvin suunniteltu testausstrategia mahdollistaa myös sen, että testaus voidaan suorittaa tehokkaasti jokaisessa vaiheessa. Kuviossa 4 on esitetty Katzoffin (n.d.) suosittelemat näkökulmat, jotka auttavat hyvän testausstrategian suunnittelussa.

Kuvio 4. Suositukset testausstrategian laatimisen avuksi (Katzoff n.d.)

Hyvänä käytäntönä on suorittaa testausta ennen migraatiota ja sen jälkeen (Data Migration Testing Tutorial: A Complete Guide 2020). Migraation testauksen vaiheiksi voidaan esittää kolmea vaihetta: testaus ennen migraatiota (Pre-Migration Testing), migraation testaus (Migration Testing) ja testaus migraation jälkeen (Post-Migration Testing) (Data Migration Testing Tutorial: A Complete Guide 2020; Katzoff n.d.).

Ennen migraatiota tehtävässä testauksessa on muutamia tärkeitä asioita, jotka tulisi ottaa huomioon. Tässä vaiheessa migraatiotiimin tulisi varmistaa ja ymmärtää migraation ja datan laajuus, eli mitä dataa siirretään ja millä rajoitteilla (Considerations When Planning To Test Data Migration n.d.). Kun datan laajuus on varmistettu, tulisi muodostaa datamappaukset eli datan kartoitukset. Jokaiselle siirrettävälle datatyypille ja objektille tulisi olla vastaavuus uudessa kohdejärjestelmässä, ja datamappauksilla tarkoitetaan sääntöjä, joiden mukaan data osataan ohjata oikeaan paikkaan. Tämän yhteydessä olisi hyvä myös varmistaa mitä sellaisessa tilanteessa tehdään, jos kohdejärjestelmän puolella datalle on asetettu pakollisuus, mutta lähtöjärjestelmässä vastaava tieto ei ole ollut pakollinen ja on täten esimerkiksi tyhjänä (Katzoff n.d.). Kun nämä asiat on varmistettu, voidaan lopuksi vielä testata, että jokaiselle siirrettävälle kentälle on varmasti oikea linkitys uuden kohdejärjestelmän kenttään, eli varmistetaan vielä, että data siirtyisi oikeaan paikkaan ja virheettömästi. Tässä vaiheessa voidaan yleensä jo konfiguroida migraation määritysten ja speksien mukaisesti testaustyökalu, jota voidaan käyttää ennen varsinaista datamigraatiota ja sen jälkeen (Considerations When Planning To Test Data Migration n.d; Katzoff n.d.).

Juuri ennen varsinaista datamigraatiota ja tuotantoajoja toteutetaan vielä migraation testaus. Tällä testauksella on pääasiallisesti tarkoituksena varmistua siitä, että varsinainen migraatio ja tuotantoon siirtyminen onnistuu ongelmitta. Tässä vaiheessa myös varmistuu, onko migraatio dokumentoitu hyvin ja onko prosessi selkeä ja helposti toteutettavissa (Data Migration Testing Tutorial: A Complete Guide 2020). Tämä vaihe liittyy luvussa 2.2 esitettyyn ja kuviossa 2 näkyvään ”Integration tests and final rehearsal” -vaiheeseen. Kun migraation testauksessa on saatu varmuus siitä, että migraatio voidaan onnistuneesti toteuttaa, voi migraatiotiimi edetä varsinaiseen migraatiovaiheeseen ja sen jälkeiseen testaukseen.

Kun datamigraatio on toteutettu kohdejärjestelmään onnistuneesti, koittaa migraation jälkeisen testauksen vaihe. Migraation jälkeinen testaus voidaan ja yleensä myös halutaan toteuttaa viimeisten testiajojen ja niin kutsutun kenraaliharjoituksen jälkeen testijärjestelmän puolella varmistuksena siitä, että tuotantoon ajetaan data varmasti halutussa muodossa. Testauksessa tarkastellaan migraation läpivientiin kuluvaa aikaa ja varmistetaan, että datan siirto on mahdollista suunnitellussa seisokkiajassa. Tässä vaiheessa myös oleellista on testata, vastaako sisäänajettu data migraation määrityksiä, eli datan on lähtökohtaisesti siirryttävä sen tyyppejä ja arvoja muuttamatta, ellei kyseisiin kenttiin ole määritetty konversioita. Samalla sisäänajettua dataa tulee testata järjestelmän käytön kautta ja on seurattava, että data käyttäytyy järjestelmän mukaisesti. Tämä tarkoittaa käytännössä sitä, että uusi järjestelmä kykenee lataamaan ja esittämään datan oikeassa muodossa ja että järjestelmän kautta dataa pystytään muokkaamaan. Lisäksi migraation jälkeisessä testauksessa vertaillaan vanhaa lähtöjärjestelmän dataa kohdejärjestelmään siirrettyyn dataan ja varmistetaan, että ne vastaavat toisiaan ja että liitokset datan välillä ovat säilyneet. (Data Migration Testing Tutorial: A Complete Guide 2020; Katzoff n.d.)

Migraation jälkeisessä testauksessa lähde- ja kohdejärjestelmien datan vertailua voidaan tehdä manuaalisesti niin, että otetaan satunnaisotanta datasta, jota vertaillaan. Tämä vertailu voidaan myös toteuttaa mahdollisesti automaattisella testaustyökalulla, jolloin saadaan testattua 100 % siirretystä datasta. Mikäli automaattiseen testaukseen ei ole mahdollisuutta, tulisi manuaalista vertailua tehdä riittävän suuren satunnaisjoukon osalta. Tämän yhteyteen on suositeltavaa soveltaa myös käyttäjän hyväksyntätestausta, sillä datan alkuperäinen käyttäjä usein tietää paremmin omaan dataan liittyviä asioita ja havaitsee poikkeavuudet herkemmin.

3 Työkalut

3.1 ETL-työkalut

Markkinoilla on tarjolla paljon erilaisia työkaluja, joilla voidaan taata datamigraatioiden onnistuminen. On paljon myös ratkaisuja, joilla datamigraatio ja testaus voidaan suorittaa samalla työkalulla. Tässä raportissa keskitytään ja tarkastellaan kuitenkin vain niitä työkaluja, jotka tarjoavat datamigraation testaukseen sopivia ratkaisuja. Työkaluja, joilla datamigraation testausta voidaan suorittaa, kutsutaan usein ETL (Extract, Transform, Load) testaustyökaluiksi.

Työssä tarkasteltiin Timothy Kingin kahta verkkojulkaisua (2019a; 2019b), joissa listattiin sekä maksullisia, että ilmaisia avoimen lähdekoodin sovelluksia, jotka tarjoavat datamigraation testaukseen sopivia ETL-työkaluja. Tämän lisäksi vertailtiin Kingin listaamia vaihtoehtoja Top 10 ETL Testing Tools In 2021 (2020) -verkkojulkaisun listaamiin ja arvostelemiin ETL testaussovelluksiin.

3.2 RightData

RightData tarjoaa intuitiivisen, joustavan ja skaalautuvan datan testaus- ja validointisovelluksen, jonka avulla voidaan tunnistaa datan laatuun liittyviä ongelmia. Sovellus tarjoaa automatisoidun ETL-testaustyökalun, joka auttaa tiimejä parantamaan esimerkiksi migraatioprojektien menestystä ja pienentää kyseisten projektien kokonaiskustannuksia (King 2019a). Se on suunniteltu toimimaan tehokkaasti monimutkaisten ja isojen datamäärien testauksessa (Top 10 ETL Testing Tools In 2021 2020). RightDatan tuotetta ei ole mahdollista kokeilla suoraan, vaan demo- tai kokeiluversiota täytyy pyytää erillisellä lomakkeella palvelun verkkosivuilta.

3.3 Datagaps ETL Validator

Datagaps ETL Validator on työkalu, joka on suunniteltu ETL testaukseen ja massadatan testaukseen. Se auttaa integraatioiden ja migraatioiden projekteissa, joissa käytetään suuria datamääriä ja eri datalähteitä, käyttämällä hyödykseen automaatiota, joka puolestaan helpottaa vähentämään projektien kustannuksia. ETL Validator tarjoaa uniikin ja visuaalisen testausalustan, joka tukee vedä ja pudota-ominaisuuksia, sekä toiminnon, joka mahdollistaa testien määrittelyn ilman koodaamista tai manuaalista kirjoittamista. Datagaps tarjoaa ilmaisen 30 päivän kokeilujakson, joka on tämäntyyppisen työkalun käyttöönoton harkinnassa erittäin hyödyllistä. (King 2019a; Top 10 ETL Testing Tools In 2021 2020.)

3.4 Informatica Data Validation

Informatican tarjoama Data Validation on ETL testaustyökalu, jolla on miellyttävä graafinen käyttöliittymä ja se tarjoaa ratkaisuja, jotka automatisoivat datamigraation testausta sekä testi-, että tuotantoympäristöissä. Tämä tarkoittaa sitä, että työkalulla voidaan suorittaa testausta ja datajoukkojen vertailua toistuvia kertoja testijärjestelmässä ja tuotannon järjestelmässä. Informatica haluaa testaustyökalullaan taata sen, että data on siirtynyt oikein järjestelmästä toiseen ja että se on oikeassa formaatissa kohdejärjestelmässä (ETL Testing n.d.). Datagapsin tavoin myös Informatica tarjoaa ratkaisuistaan 30 päivän kokeiluversion, jonka jälkeen tuotteen käytöstä on maksettava.

3.5 iCEDQ

iCEDQ tarjoaa alustan automaattiselle ETL testaukselle, ja se on kehitelty datakeskeisiä projekteja, kuten datamigraatioita varten tarjoamaan menetelmiä datan validointiin ja testaukseen kohde- ja lähdejärjestelmien välillä. Työkalu helpottaa löytämään tarkat rivit, jotka sisältävät virheitä tai eroavat järjestelmien välillä. Työkalulla pystytään vertailemaan miljoonia rivejä eri tietokannoista tai tiedostoista, joka on oleellinen osa datamigraation jälkeistä testausta ja validointia. Työkalu on maksullinen,

mutta iCEDQ:n sivustolta on mahdollista pyytää 30 päivän kokeiluversiota työkalun testausta varten. (Top 10 ETL Testing Tools In 2021 2020.)

3.6 QuerySurge

QuerySurge on älykäs datan testauksen ratkaisu, jolla voidaan automatisoida testaus, validointi ja muut ETL testauksen prosessit. QuerySurge on aloittelijaystävällinen, sillä työkalulla datan validointi onnistuu nopeasti Query Wizard-avustajan avulla. Kokeneimmillekin tehokäyttäjille työkalu mahdollistaa edistyneempiä menetelmiä kuten oman koodien ja sääntöjen kirjoittamisen. Työkalu säästää aikaa ja vaivaa automatisoimalla testausta ja testien suorittaminen on mahdollista aikatauluttaa tarkasti. QuerySurge on pienten ja suurten yritysten käyttöön suunniteltu maksullinen tuote, mutta tuotetta on mahdollista kokeilla ilmaiseksi heti 15 päivän kokeilujakson ajan. Pidempää, 30 päivän kokeilujaksoa tai 45 päivän Proof-Of-Concept-jaksoa on mahdollista myös pyytää QuerySurgen verkkosivuilta.

3.7 Talend Open Studio for Data Integration

Talend Open Studio for Data Integration on avoimeen lähdekoodiin perustuva ETL työkalu, joka myös helpottaa ETL testausta. Työkalun testaustoiminnoilla voidaan verrata lähtödataa ja tuotantojärjestelmään siirtynyttä dataa ja täten varmistaa onko siirron aikana tapahtunut virheitä. Työkalussa on helppokäyttöinen graafinen käyttöliittymä, joka yksinkertaistaa ETL prosessien suunnittelua ja määrittämistä. Talend tarjoaa sekä maksullista, että ilmaista avoimen lähdekoodin työkalua. Maksullisessa versiossa on hieman edistyneempiä toimintoja, mutta ilmaisessa versiossa on kaikki ETL testaukseen liittyvät toiminnot. Talendin tarjoamaa ilmaista työkalua on ladattu miljoonia kertoja vuoden 2006 jälkeen. (King 2019a; Top 10 ETL Testing Tools In 2021 2020.)

3.8 Yhteenveto työkaluista

Tässä raportissa mainittujen ETL-työkalujen lisäksi markkinoilla on lukuisia eri sovelluksia ja ratkaisuja, joilla samantyylistä testausta voidaan toteuttaa. Tässä työssä ei ollut mahdollista eikä järkevää käydä kaikkia markkinoilla olevia vaihtoehtoja läpi, vaan fiksuinta oli lähteä vertailemaan mainittuja tuotteita, sillä ne ovat yleisimpiä ja suosituimpia ETL testauksen työkaluja (Top 10 ETL Testing Tools In 2021 2020).

Tässä työssä päädyttiin käyttämään Talendin tarjoamaa Open Studio for Data Integration -työkalua. Kyseiseen ETL-työkaluun päädyttiin lähinnä työkalun helpon käytön ja saatavuuden takia. Vaikka esimerkiksi maksullisista vaihtoehdoista myös QuerySurge olisi ollut helppokäyttöinen ja aloittelijaystävällinen, testauksissa havaittiin Talend Open Studion monikäyttöisyys ja mahdollisuudet luoda erittäin yksilöllisiä ja räätälöityjä testausmenetelmiä, joka vaikutti valintaan. Koska Talend Open Studio for Data Integration on ilmainen avoimen lähdekoodin työkalu, olisi se myös toimeksiantajalle kustannustehokkain vaihtoehto toteuttaa datamigraatioiden testausta.

4 Nykytilanne

4.1 Migraatioiden toteutus

Toimeksiantaja tuottaa tuki- ja asiantuntijapalvelua suurelle toiminnanohjausjärjestelmälle. Asiakkaalta tulee aika-ajoin toimeksiantoja, joissa tehtävänä on toteuttaa datamigraatio toisesta järjestelmästä omaan hallinnassa olevaan järjestelmään.

Lähtökohtaisesti datamigraatioprojektit alkavat neuvotteluilla, johon osallistuvat loppuasiakkaan edustajat, asiakkaan yhteyshenkilöt sekä migraatioprojektia vetävät henkilöt. Neuvotteluita ja tapaamisia voi olla useampia eri sidosryhmien välillä, ja näillä keskusteluilla tarkoituksena on saada kokonaiskuva projektin laajuudesta, työmääräarvioista, aikataulusta, sekä lähtödatasta, joka tulisi viedä kohdejärjestelmään.

Kun asiakkaalta vastaanotetaan tarvittava lähtödata ja ollaan tietoisia migraation määrityksistä kuten transformaatiosäännöistä, voidaan alkaa työstämään tarvittavia datan konversioita. Tällä hetkellä datamigraatioalustana käytetään Microsoft Visual Studio 2019 -ohjelmistoa, johon on yhdistetty SQL Server Integration Services (SSIS) -ominaisuudet. Käytössä oleva datamigraatioalusta yhdistettynä monipuolisilla SSIS-paketeilla mahdollistaa datamigraatiolle oleelliset ETL-prosessit. Pääosin kaikki datan transformaatiot ja käsittelyt tehdään SQL-lauseita hyväksikäyttäen.

Kun lähtödata on saatu transformoitua kohdejärjestelmän tietokannan taulujen mukaiseen muotoon, ajetaan lähtödata ensin testikantaan. Testikanta replikoi loppuasiakkaan kohdejärjestelmää ja käsitelty data voidaan ajaa turvallisesti testattavaksi testipuolelle. Kun data ajetaan ensin testikantaan, voidaan tässä vaiheessa havaita mahdollisia virheitä, joita esiintyy sisäänajojen yhteydessä. Kun data on siirretty onnistuneesti testikantaan, tulee testijärjestelmän sisällä vielä tarkastella, miten tuore data käyttäytyy ja aiheuttaako se toiminnallisia häiriöitä.

Tämänkaltaisia testiajoja ja testauksia tehdään mahdollisesti lukuisia kertoja migraatioprojektin aikana ja testauksissa havaittuja virheitä ja datan transformaatioita korjataan testauksissa havaittujen tulosten perusteella. Kun viimeiset iteraatiokierrokset datan käsittelyssä ja testauksessa on tehty, suoritetaan vielä viimeinen harjoitus testijärjestelmään. Tätä voidaan pitää migraation kenraaliharjoituksena ja siinä testataan koko datamigraatioprosessi alusta loppuun. Mikäli kenraaliharjoituksessa ei havaita mitään ongelmakohtia, voidaan varsinainen datamigraatio toteuttaa asiakkaan tuotantojärjestelmään.

Kun tuotantoon siirtymisen ajankohta koittaa, datamigraatio toistetaan kenraaliharjoitusten mukaisesti, mutta tällä kertaa kohteena on asiakkaan tuotantojärjestelmä. Tuotantoajot pyritään suorittamaan aina niin, että ajot vaikuttaisivat mahdollisimman vähän tuotantojärjestelmän toimintaan tai suorituskykyyn. Mikäli ajot vaativat järjestelmän huoltokatkoa tai seisokkiaikaa, pyritään nämä aina minimoimaan, jotta ajoilla olisi mahdollisimman pieni vaikutus loppuasiakkaiden toimintaan. Kun tuotantoajot on suoritettu onnistuneesti, alkaa datamigraation viimeistelyvaihe. Viimeiste-

lyssä tarkastellaan kohdejärjestelmän toimintaa ja ollaan valmiina, mikäli järjestelmän käytössä havaitaan vielä datan käyttäytymiseen liittyviä ongelmia. Tässä vaiheessa havaitut ongelmat ovat yleensä niin pieniä, että niiden korjaus tapahtuu nopeasti ja normaalien tukiprosessien mukaisesti. Viimeistelyssä tehdään vielä mahdollisia dokumentointeja ja lopuksi, mikäli asiakas katsoo datamigraation toimitetuksi ja päättyneen, voidaan projekti päättää.

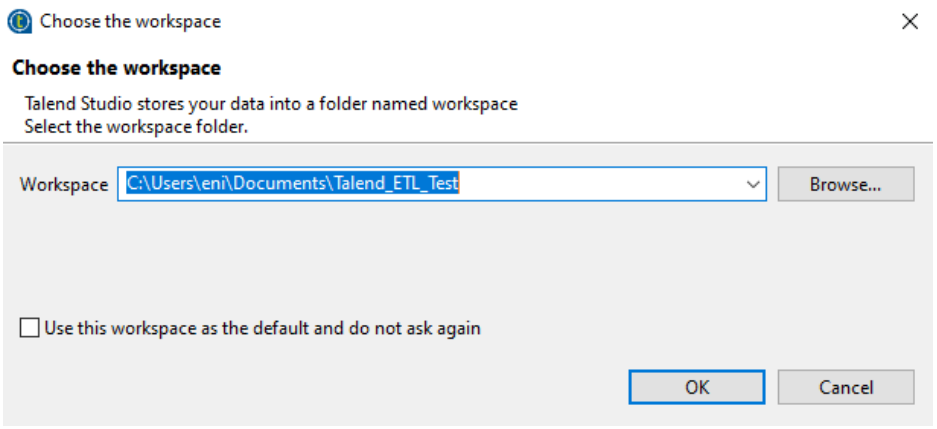
4.2 Migraatioiden testaus

Tällä hetkellä datamigraatioiden yhteydessä ei tehdä mitään automaattisia ETL-testaamisen menetelmiä. Kaikki datamigraatioiden testaukset tehdään pääosin silloin, kun dataa ajetaan testijärjestelmään. Testiajoissa havaitaan varsinaiset tekniset rajoitukset ajettavan datan ja tietokannan taulujen määritysten välillä. Tämän lisäksi testipuolen järjestelmää testataan käytännössä ja tällä selvitetään, aiheuttaako tuore siirretty data toiminnallisia virheitä. Testiajojen, niin kuin myös tuotantoajojen jälkeen datan vertailua tehdään niin kutsutulla field-by-field-menetelmällä, eli vertaillaan datajoukkoja ja niiden kenttiä manuaalisesti lähtöjärjestelmän ja kohteessa olevan datan välillä.

5 Toteutus

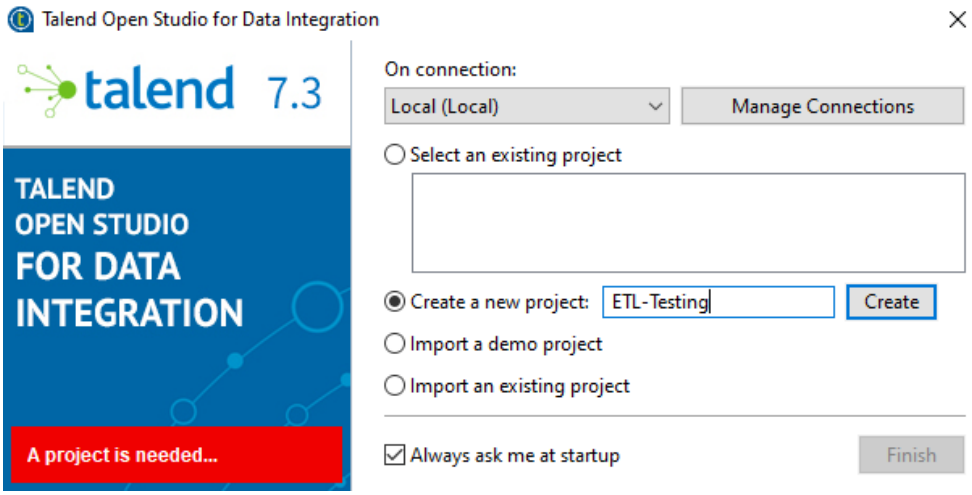
5.1 Aloitus

Datamigraatioiden automaattista testausta varten päädyttiin käyttämään Talend Open Studio for Data Integration -sovellusta. Kyseinen sovellus on monipuolinen ETL-testauksen työkalu, jolla toimeksiantajan tarvitsemat testausmenetelmät voidaan toteuttaa. Työtä varten ladattiin uusin versio 7.3.1. Talend Open Studio for Data Integration -ohjelmistosta suoraan Talendin virallisilta verkkosivuilta. Sovelluksen asentamisen jälkeen ensimmäisellä käynnistyskerralla sovellus pyytää muodostamaan työtilan, johon kaikki projektit perustetaan. Kuviossa 5 muodostetaan uusi työtila projektia varten.



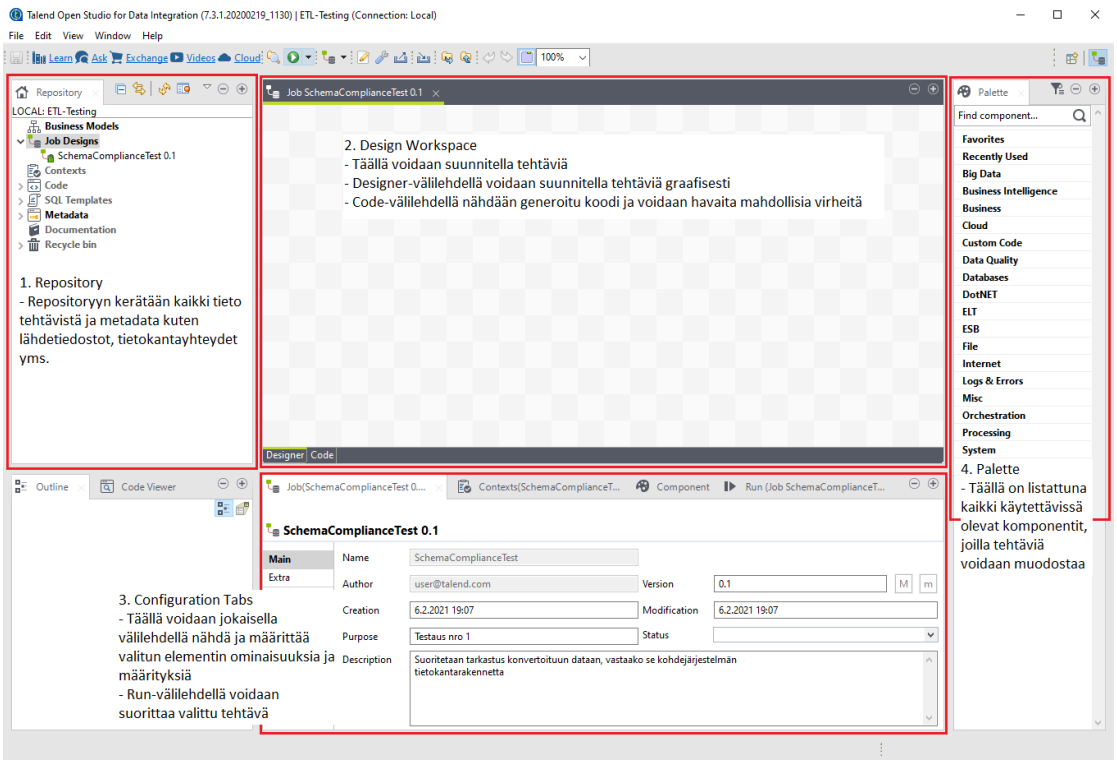
Kuvio 5. Työtilan perustaminen

Työtilan perustamisen jälkeen sovellus pyytää muodostamaan uuden projektin, mikäli olemassa olevia projekteja ei löydy. Kuviossa 6 perustetaan testauksia varten uusi projekti nimeltään ETL-Testing.



Kuvio 6. Uuden projektin perustaminen

Kun projekti on perustettu, on kaikki valmiina testausmenetelmien määrittämistä varten. Talend Open Studiossa muodostettavia tehtäviä kutsutaan nimellä “Job”. Kuviossa 7 näkyy perusnäkymä, jossa erilaisia tehtäviä voidaan muodostaa, ja kuviossa on selitetty myös tehtävien muodostamisessa tarvittavat osiot.



Kuvio 7. Talend Open Studion oletusnäkymä

Kun työtila ja projekti oli perustettuna, aloitettiin testausmenetelmien suunnittelu. Toimeksiantajalle tarpeellisinta oli toteuttaa testausmenetelmät seuraavien kolmen pääongelman ympärille ja näillä testauksilla halutaan varmistaa, että:

1. Data on kohteen tietokantarakenteen mukaista ja ehjää
    a. Eli data sisältää pakolliset kentät eikä data siirry väärällä datatyypillä
2. Data on kohteen logiikan mukaista
    a. Eli esimerkiksi projektidatassa alkamispäivät ovat ennen päättymispäiviä, projektipäällikkötiedot löytyvät, sekä muut pakollisuudet
3. Data vastaa migraatioiden määrityksiä ja speksiä
    a. Eli data siirtyy konvertointimääritysten mukaisesti

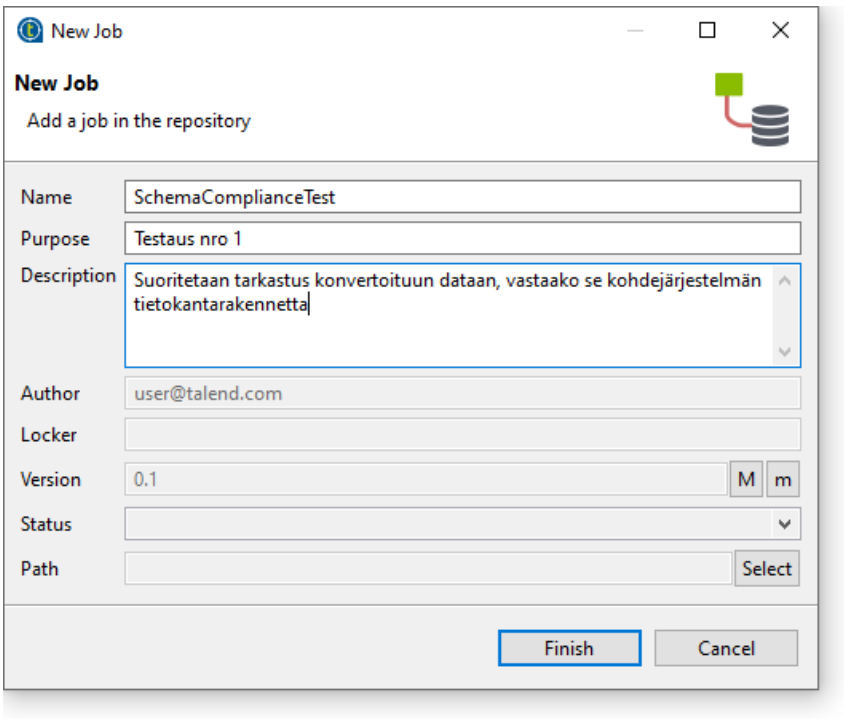
Näihin tarpeisiin perustuen muodostettiin 3 erilaista tehtävää, joissa todistettiin kyseisten testausten ja tarkasteluiden olevan mahdollista toteuttaa. Työssä perustettiin tehtävät nimeltään Schema Compliance Test, Target Logigality Test, ja Data Matches

Spec Test. Schema Compliance Test tarkastaa nimensä mukaisesti, vastaako se kohdejärjestelmän tietokantarakennetta. Target Logigality Test sen sijaan tarkastaa hieman yksityiskohtaisemmin, onko syötettävä data kohteen logiikan mukaista. Tässä menetelmässä voidaan määrittää hyvinkin yksityiskohtaisesti rajoituksia ja logiikkaa, jonka mukaista datan täytyy olla. Kolmannessa testissä, Data Matches Spec Test -tehtävässä tarkastetaan vastaako data ennalta sovittuja konvertointimäärityksiä. Tämän työn mallinnusta varten muodostettiin esimerkkidataa Mockaroo.com -verkkosivustolla. Seuraavaksi käydään läpi jokaisen testausmenetelmän muodostaminen.

5.2 Schema Compliance Test

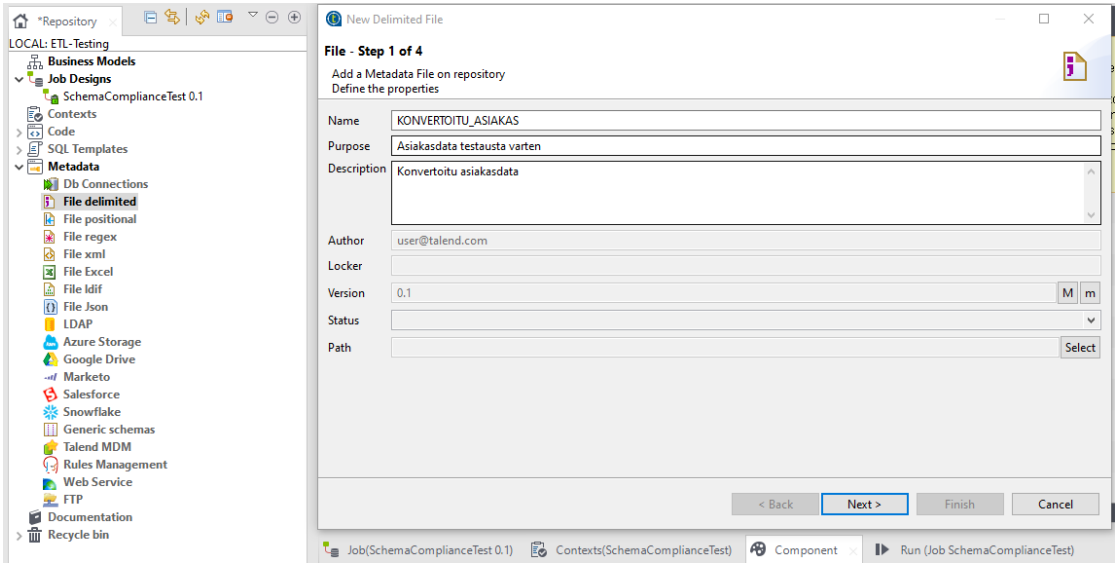
Schema Compliance Test -tehtävässä tarkoituksena on tarkistaa konvertoidussa muodossa oleva data vastaako se kohdejärjestelmän tietokannan taulun rakennetta. Lähtödataa käsitellään ja konvertoidaan SQL-muodossa ja kun lähtödata on saatu SQL:llä käsiteltyä haluttuun muotoon, voidaan se tässä tehtävässä testata ja varmistaa, että kohdejärjestelmän tietokanta hyväksyy sen teknisten rajoitusten mukaisesti. Tässä tehtävässä ajetaan SQL:llä konvertoitu data CSV-muotoon, ja syötetään se Talendin valmiiseen tSchemaComplianceCheck-komponenttiin. Tähän tSchemaComplianceCheck-komponenttiin voidaan määrittää kohdejärjestelmän tietokannan taulun skeeman mukaisesti rajoitukset kuten datatyypit ja koot. Mikäli komponentti havaitsee, että datan kenttä ei läpäise tarkistusta, otetaan tieto virheellisestä rivistä ylös esimerkiksi erilliseen tiedostoon.

Testausta varten muodostettiin uusi tehtävä nimeltään SchemaComplianceTest. Uusi tehtävä voidaan lisätä esimerkiksi Repositoryn kautta, valitsemalla Job Designs-otsikon alta avautuvasta valikosta “Create job”. Kuviossa 8 on esiteltynä määritysikkuna, jossa tehtävän tiedot voidaan määrittää ja luoda tehtävä.



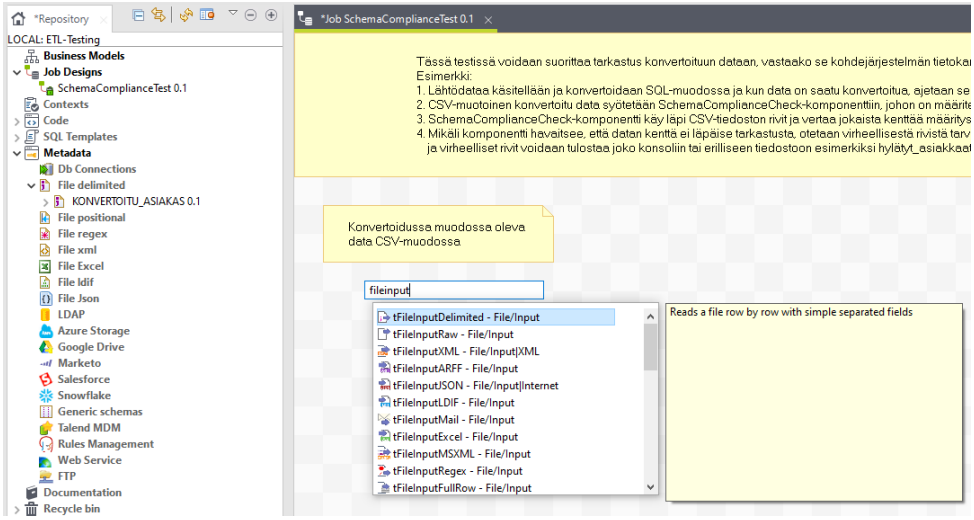
Kuvio 8. Uuden SchemaComplianceTest -tehtävän lisääminen

Seuraavaksi lisättiin Metadata-osion alle uusi CSV-tiedosto, joka piti sisällään konvertoidun asiakasdatan. Uusi CSV-tiedosto voidaan määrittää kuvion 9 mukaisesti File delimited kohdasta ja syöttämällä tarvittavat tiedot sekä seuraamalla määritysikkunan ohjeistusta.



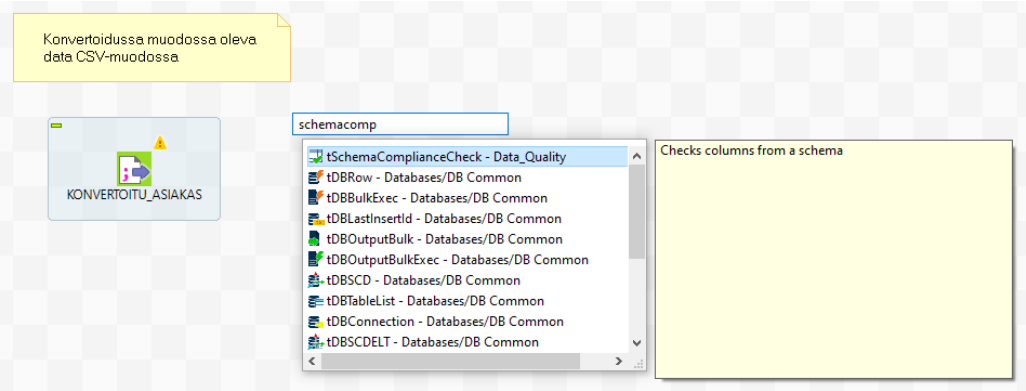
Kuvio 9. CSV-tiedoston määrittäminen

Kun konvertoitu asiakasdata on saatu määritettyä CSV-muodossa projektin metadataosioon, voidaan sitä käyttää tehtävässä esimerkiksi lukemalla sen sisältö rivi kerrallaan. Tässä tehtävässä käytetään määritettyä CSV-tiedostoa inputkomponentin sisällä, ja sen lisääminen onnistuu joko kirjoittamalla suoraan tehtävän suunnitteluosiossa hakusanalla esimerkiksi “FileInputDelimited” kuvion 10 mukaisesti, tai raahaamalla sen suoraan metadataosiosta suunnitteluosioon, jolloin komponentin määritykset luodaan automaattisesti. Tässä vaiheessa raahattiin CSV-tiedosto suoraan suunnitteluosioon.



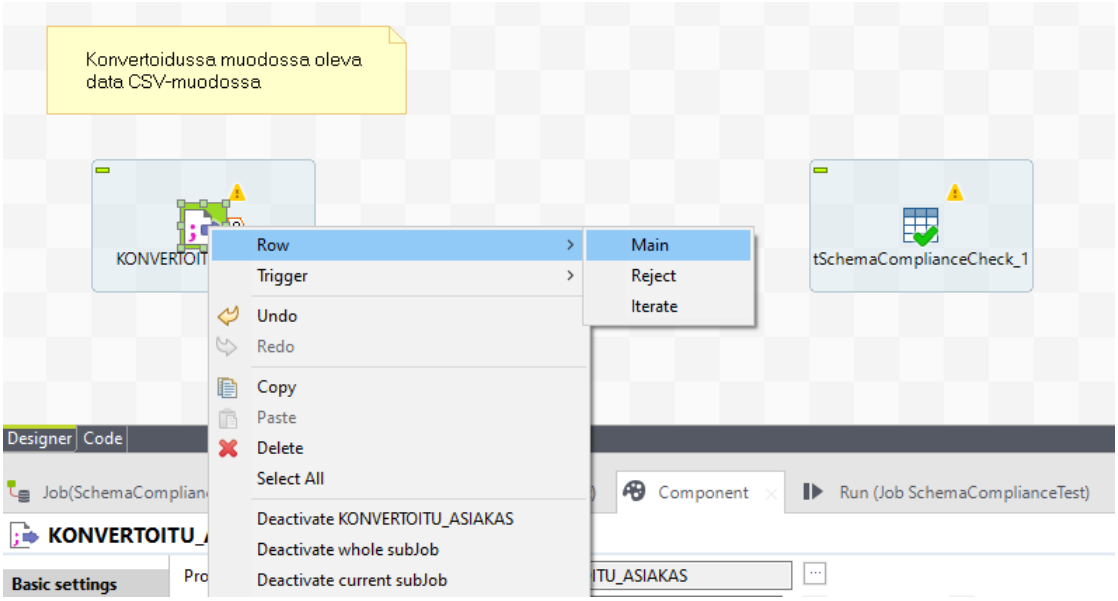
Kuvio 10. CSV-tiedoston määrittäminen Input-komponenttina

Tämän jälkeen lisättiin tehtävälle SchemaComplianceCheck-komponentti, kirjoittamalla ja hakemalla kätevästi suunnitteluosiossa komponentin nimellä, ja valitsemalla se tehtävälle kuvion 10 esimerkin mukaisesti.



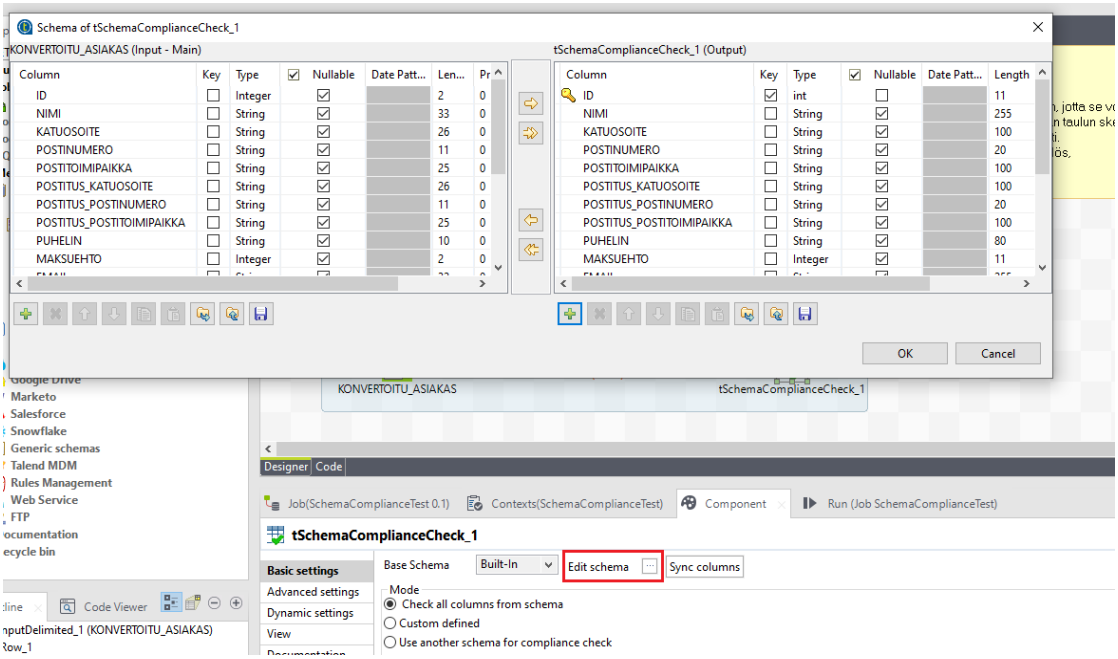
Kuvio 11. SchemaComplianceCheck-komponentin lisääminen

Tämän jälkeen yhdistettiin komponentit kuviossa 12 esitetyllä tavalla, jolloin jokainen CSV-tiedoston rivi luetaan SchemaComplianceCheck-komponentin sisällä.



Kuvio 12. Komponenttien yhdistäminen Row-Main liitoksella

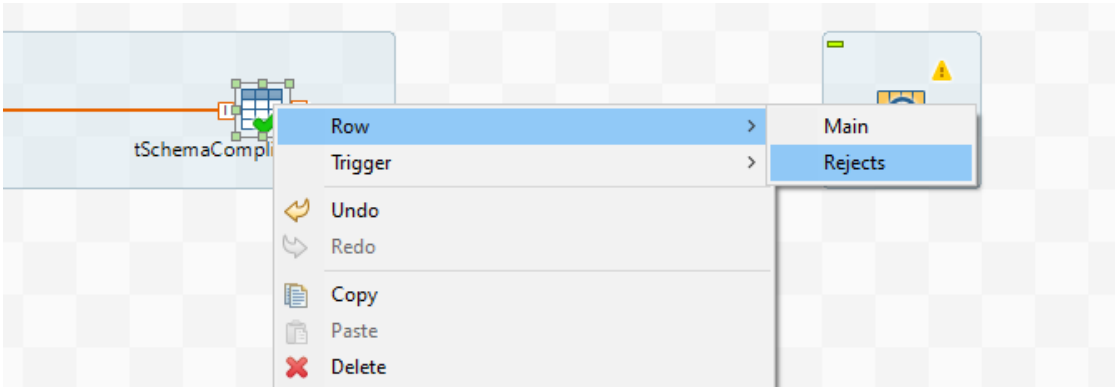
Jotta konvertoitua dataa voidaan tarvittavalla tasolla vertailla kohdejärjestelmän rakenteen mukaisesti, tulee SchemaComplianceCheck -komponentissa tehdä tarkat määritykset. Seuraavaksi tehtiin nämä tarvittavat määritykset, klikkaamalla SchemaComplianceCheck-komponentin Component-välilehdeltä “Edit schema”-painiketta (kuvio 13).



Kuvio 13. SchemaComplianceCheck-komponentin määritykset

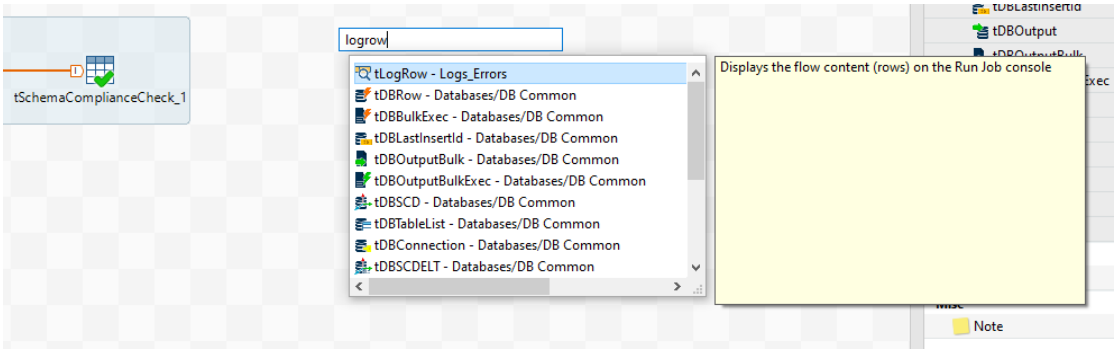
SchemaComplianceCheck-komponentin määrityksissä vasemman puolen sarake vastaa komponenttiin sisään tulevan datan skeemaan, ja tämän määrityksiin ei lähtökohtaisesti tarvitse koskea, mikäli skeema ja sarakkeiden tiedot on määrittyneet automaattisesti oikein. Oikealla puolella olevaan Output-kohtaan määritetään kohdetaulun skeeman asetukset ja rajoitteet. Tässä kohtaa siis katsotaan kohdetaulun mukaisesti kenttien tyypit ja koot, ja määritetään ne kuviossa 13 näkyvän määritysikkunan oikealla puolella. Komponentin asetuksissa on myös varmistettava, että sarakkeiden nimet ja järjestys täsmäävät.

Kun komponentin määritykset on asetettu, voidaan komponenttiin määrittää, että se syöttää ulos vain hylätyt rivit. Tämä onnistuu kuvion 14 esimerkin mukaisesti, valitsemalla komponentin ulostuloon “Rejects”. Hylätyt rivit voidaan joko tulostaa suoraan Talendin konsoliin tai kirjoittaa rivit erilliseen tiedostoon.



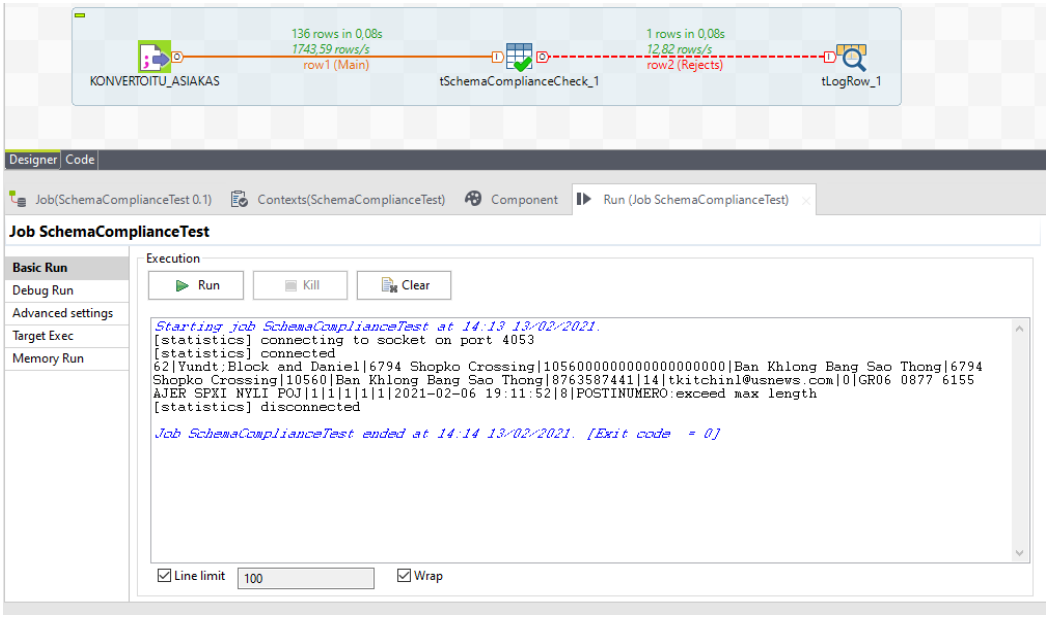
Kuvio 14. SchemaComplianceCheck-komponentin ulostulon valinta

Tehtävälle voidaan valita erillinen tLogRow-komponentti, joka kirjoittaa sisään tulevat rivit suoraan sovelluksen konsoliin. Tälle tehtävälle lisättiin tLogRow-komponentti, jotta virheellisistä riveistä saatiin heti tieto, ilman että tietoa tarvitsisi katsoa erillisestä tiedostosta. Kuviossa 15 lisätään tehtävälle uusi tLogRow-komponentti.



Kuvio 15. LogRow-komponentin lisääminen tehtävälle

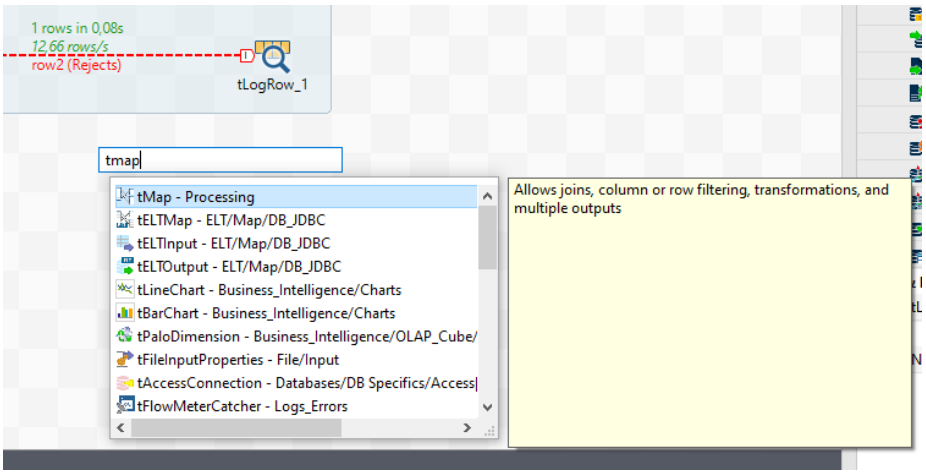
SchemaComplianceCheck-komponentista otetaan Rejects-ulostulo ja tehdään sillä liitos LogRow-komponenttiin. Jos tehtävä suoritetaan tällaisenaan, konsoli tulostaa kaikki rivit, joita eivät ole SchemaComplianceCheck-komponentissa määritetyn skeeman mukaisia. Demoamistarkoituksessa lähtödataan muokattiin yksi kenttä tarkoituksella liian pitkäksi, jotta voidaan nähdä käytännössä millä tavalla tieto virheellisestä rivistä esitetään. Kuviossa 16 on tässä tapauksessa suoritetun tehtävän tulos.



Kuvio 16. SchemaComplianceTest-tehtävän suorittaminen

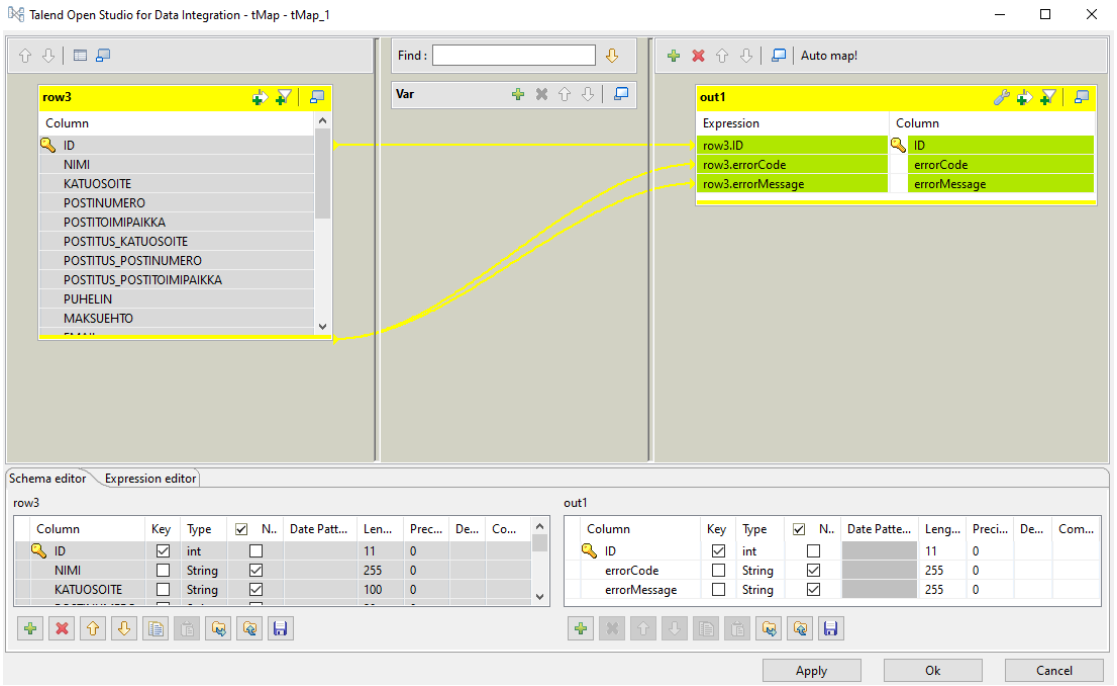
Tehtävää voidaan vielä hieman parantaa esimerkiksi siten, että pelkästään hylättyjen rivien virheilmoitukset ja rivin kohdistustieto kuten ID tulostetaan erilliseen tiedostoon. Tämä on hyödyllinen siinä tapauksessa, kun dataa on hyvin paljon eikä testin tulosta ole järkevää lukea konsolista.

Tehtävään lisättiin seuraavaksi tMap-komponentti, joka mahdollistaa hyvin paljon erilaisia yhdistämiseen, transformointiin ja suodattamiseen liittyviä toimintoja. Tässä tapauksessa komponentilla suodatetaan ulostulosta vain rivien ID:t ja virheilmoituksen sisältö eteenpäin. Kuvion 17 esimerkissä lisätään tehtävälle tMap-komponentti.



Kuvio 17. tMap-komponentin lisääminen tehtävälle

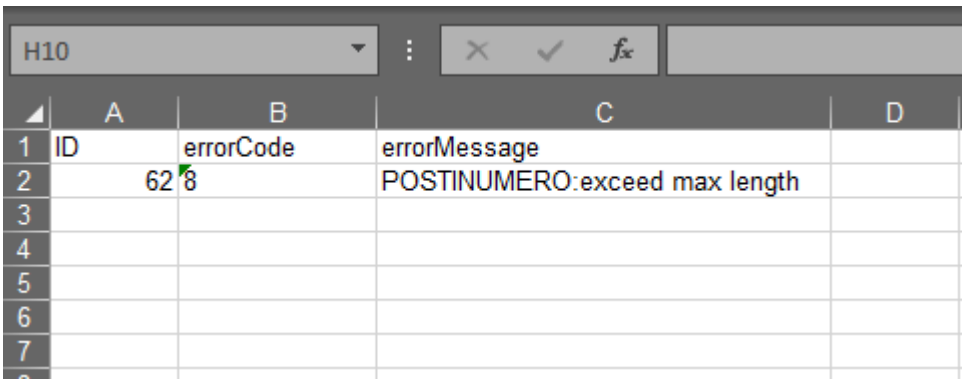
Seuraavaksi yhdistettiin tmap-komponentti edelliseen tLogRow-komponenttiin suoralla row-main liitoksella, ja tMap-komponentissa tehtiin kuvion 18 mukaiset määritykset. Komponentissa lisättiin uusi ulostulo, johon voitiin yksinkertaisesti raahata halutut sisääntulon sarakkeet eteenpäin viemiseksi. Komponentin ulostulo-lohkoon valittiin sarakkeet ID, errorCode ja errorMessage.



Kuvio 18. tMap-komponentin määritykset datan suodatusta varten

tMap-komponentin ulostulo voidaan kirjoittaa Talendin tehtävällä moneen eri tiedostoon, mutta tässä tapauksessa valitut sarakkeet kirjoitettiin erilliseen Excel-tiedostoon. Tämä onnistui valitsemalla tehtävälle komponentti nimeltään tFileOutputExcel, joka kirjoittaa rivin tiedot Excel-taulukkoon. Komponentin lisäämisen jälkeen tMap-komponentin out1-ulostulo yhdistettiin tFileOutputExcel-komponenttiin.

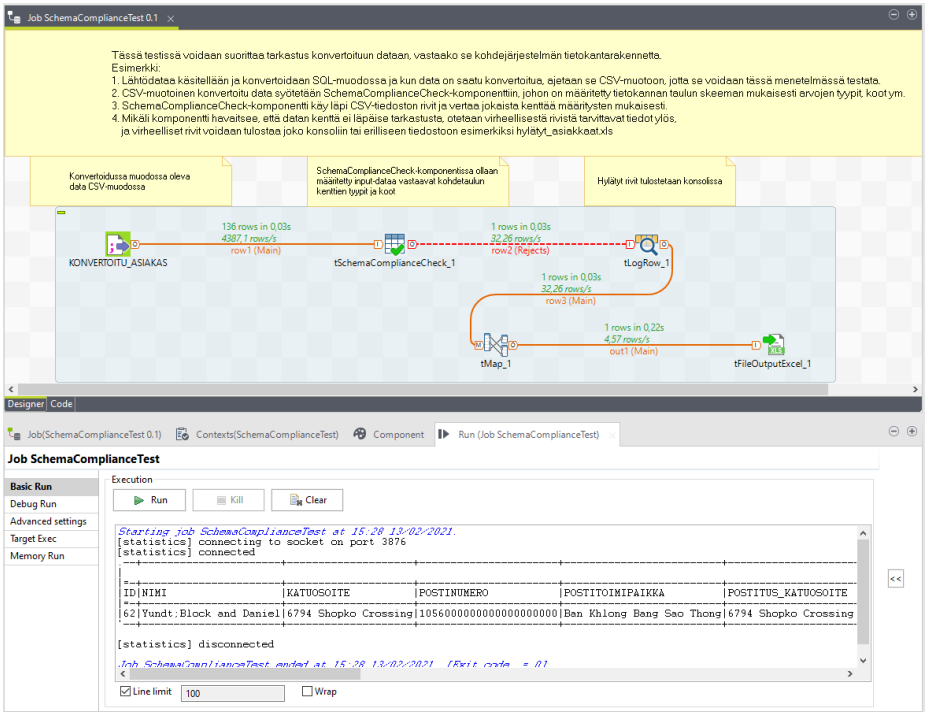
Kun tehtävä suoritetaan, sen lisäksi, että konsolissa esitetään tieto hylätyistä riveistä, kirjoitetaan tieto erilliseen excel-taulukkoon, jonka sisältö on kuvattuna vielä kuvissa 19.



	A	B	C	D
1	ID	errorCode	errorMessage	
2	62	8	POSTINUMERO:exceed max length	
3				
4				
5				
6				
7				

Kuvio 19. SchemaComplianceTest-tehtävän tulos

Tällä testauksella saadaan riittävä tieto siitä, vastaako konvertoitu data kohdejärjestelmän ja kohdetaulun teknisiä rajoituksia. Nykyisillä toimintatavoilla, kun konvertoitu data ajetaan ensin testijärjestelmän tauluun, saadaan toki tieto siitä, mikäli data on virheellistä. Mikäli syötettävässä datassa on virheitä datatyyppien ja kokorajoituksien kanssa, ajo tietokantaan keskeytyy ensimmäiseen virheeseen, eikä muista mahdollisista virheistä saada tietoa. Tässä esitetyn SchemaComplianceTest-tehtävän avulla saadaan yhdellä testauskerralla tieto jokaisesta rivistä ja datakentästä, joka ei ole kohdejärjestelmän taulun rajoitusten mukainen. Valmis SchemaComplianceTest -tehtävä on kuvattu kokonaisuudessaan vielä kuviossa 20.



Kuvio 20. Valmis SchemaComplianceTest-tehtävä

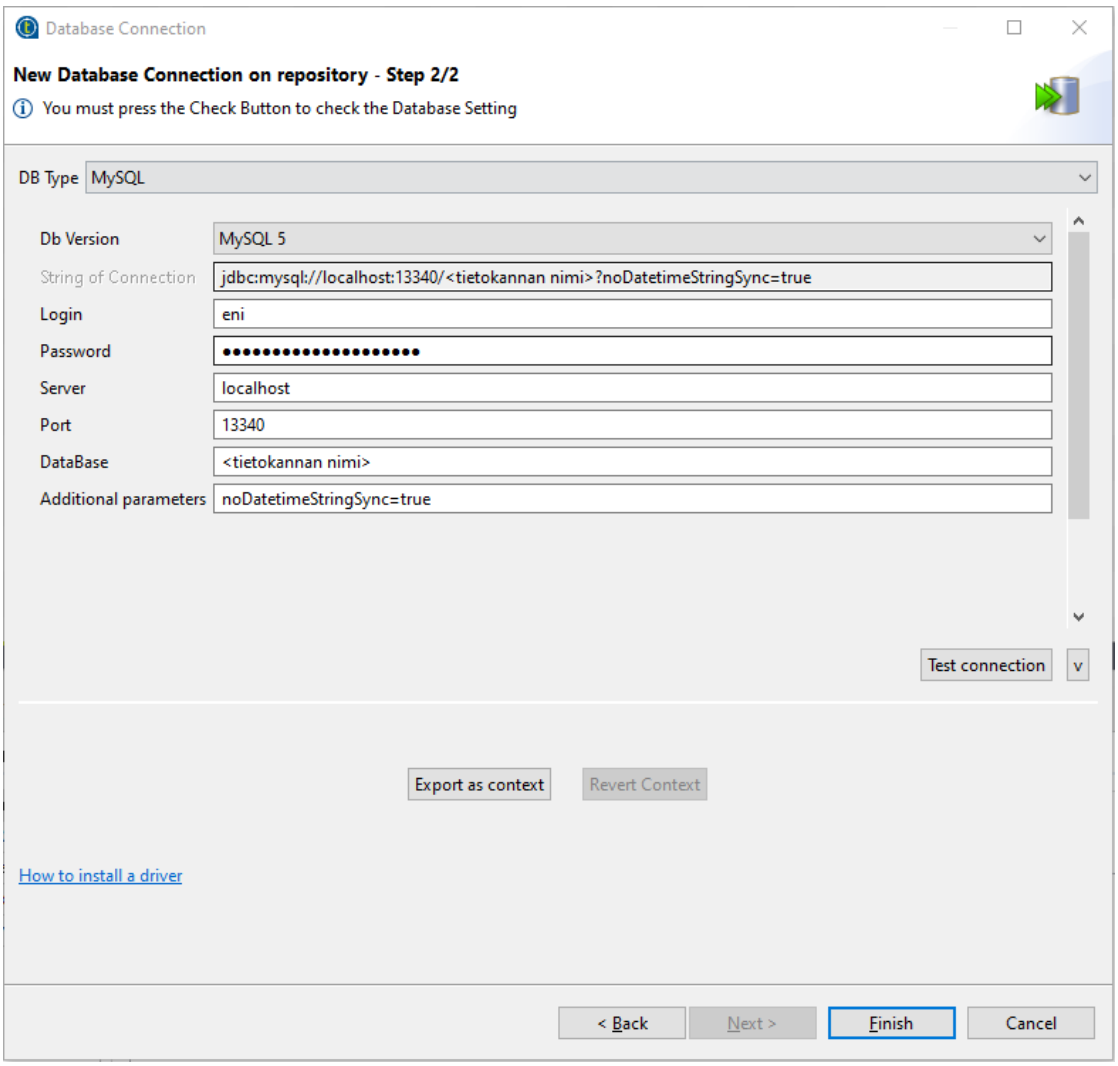
5.3 Target Logicality Test

Target Logicality Test -tehtävän tarkoituksena on tarkastaa vastaako konvertoitu data tai sisään ajettu data kohdejärjestelmän logiikkaa. Tämä logiikka voi olla tarkoin kohdejärjestelmän mukaisesti määriteltyä. Tässä tehtävässä haetaan testi- tai kohdejärjestelmään ajettu data SQL-lauseella ja tehdään siihen itse ohjelmoitavilla testauksilla tarkastuksia. Tämän tehtävän tavoitteena on saada ulos raportti tarkasteltavasta datasta, vastaako se itse määritettyä logiikkaa.

Uusi TargetLogicalityTest-niminen tehtävä lisättiin kuvion 7 esimerkin tavoin. Tätä testausta varten tarvittiin yhteys kohdejärjestelmän tietokantaan. Koska tarkasteltavan tuotannonohjausjärjestelmän tietokantaan ei voida luoda Talendin kautta suoraa yhteyttä, tuli yhteyden muodostamista varten avata ensin SSH-tunneli. SSH-tunnelista käytetään myös nimitystä SSH portin välitys ja sen avulla voidaan luoda oman koneen ja palvelimen välinen suojattu SSH yhteys, jonka kautta puolestaan erilaiset palvelut pääsevät palvelimeen käsiksi (How to Set up SSH Tunneling (Port Forwarding) 2019). Toimeksiantajalla käytetään yhteyden muodostamisen apuna Plink-sovellusta, ja SSH tunneli voidaan sen avulla muodostaa esimerkiksi kirjoittamalla komentoriville seuraava komento, jonka Philippe Cambien (2018) on artikkelissaan avannut.

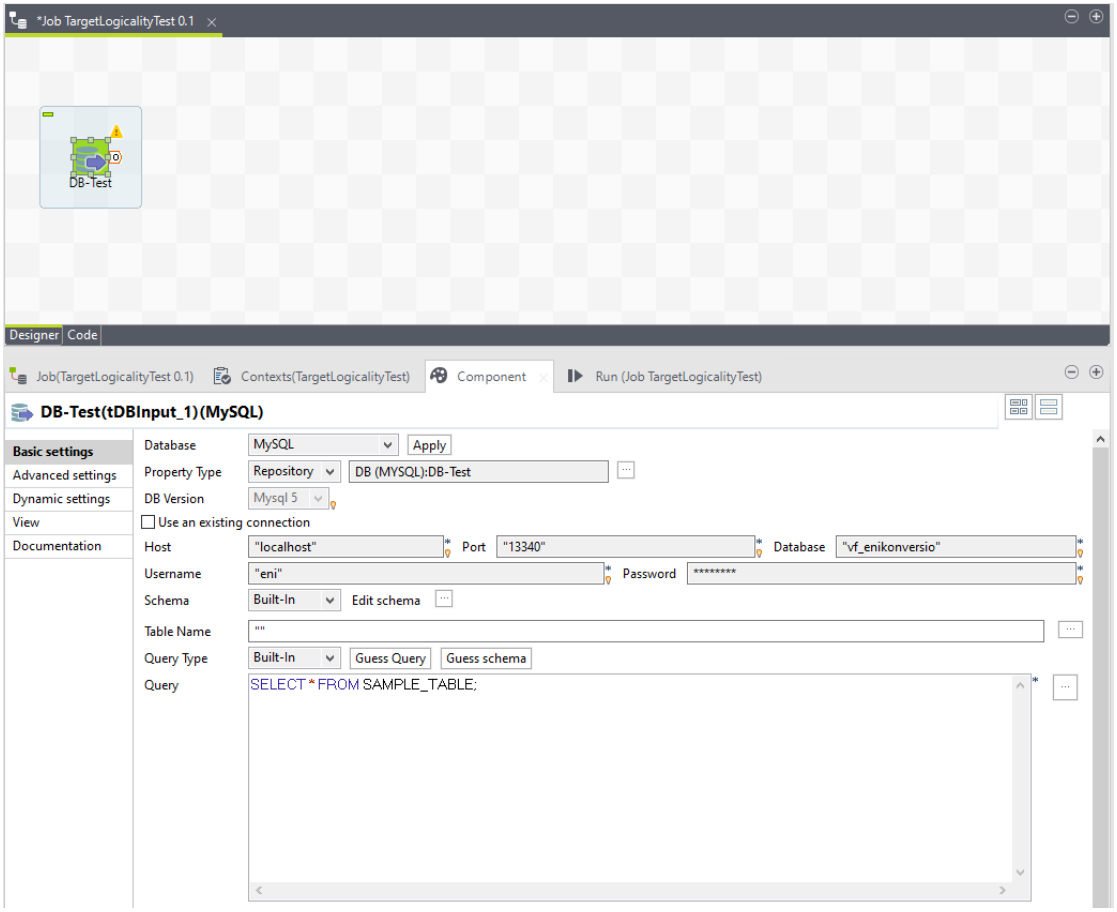
```
plink -agent -L <localPort>:<remoteAddress>:<remotePort> <username>@<re-
moteHost>
```

Kun SSH tunneli oli avattu, voitiin yhteys muodostaa Talendin kautta valitsemalla Metadata-osiossa “Db Connections”-kohdasta Create connection. Kuvion 21 mukaisessa määritysikkunassa määritettiin yhteydelle tarkemmat asetukset.



Kuvio 21. MySQL-tietokantayhteyden muodostaminen

Kun tietokantayhteys oli muodostettu onnistuneesti, pystyttiin tehtävällä hakemaan dataa suoraan tietokannasta erilaisilla SQL-kyselyillä. Tehtävälle lisättiin seuraavaksi tDBInput-komponentti, jonka avulla data voidaan lukea tietokannasta. DBInput-komponentti määritettiin käyttämään aikaisemmin luotua tietokantayhteyttä, ja komponentin Query-kentässä voitiin kirjoittaa oma SQL-kysely (Kuvio 22).

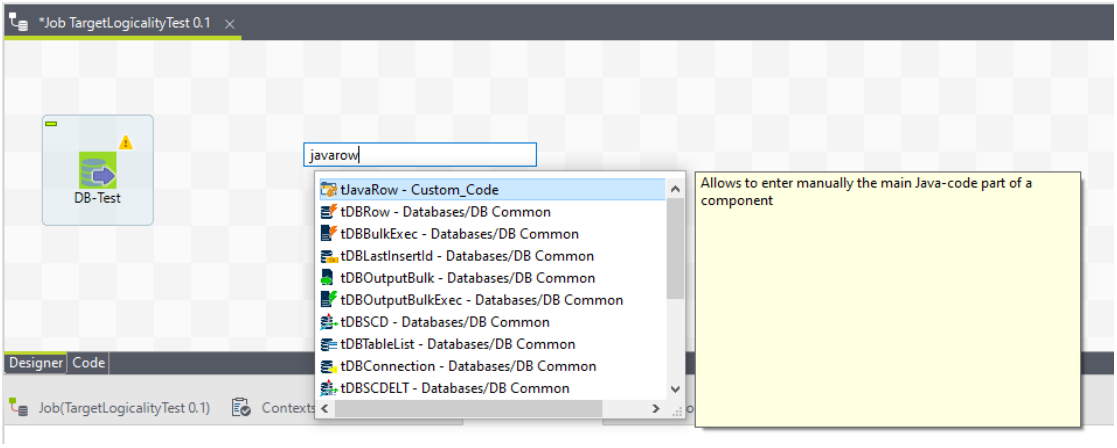


Kuvio 22. tDBInput-komponentin tarkemmat määritykset

Koska tässä opinnäytetyössä ei haluttu paljastaa kohdejärjestelmän tietokantarakennetta, muodostin tämän tehtävän mallintamista varten mallidatan mockaroo.com -sivuston kautta ja perustin tietokantapalvelimelle tämän mallidatan mukaisesti uuden taulun. Tässä tehtävässä siis voitiin kirjoittaa komponentin Query-kenttään SQL-kysely, jolla haettiin kaikki data SAMPLE_TABLE-nimisestä taulusta.

Seuraavaksi lisättiin tälle tehtävälle oleellisin komponentti, jonka sisälle määritettiin tärkeimmät logiikkatarkastelut. Talendin tJavaRow-komponentti mahdollistaa oman Java-koodin sisällyttämisen tehtävän suoritukseen. JavaRow-komponentin voidaan ajatella suorittavan sen määrityksissä olevan koodipätkän jokaisen rivin läpikäynnin yhteydessä. Koska tässä tehtävässä halutaan määrittää omia, hyvin yksilöllisiä kohdejärjestelmän mukaisia logiikkatarkasteluja, sopii JavaRow-komponentti todella hyvin

tähän tarkoitukseen. Kuviossa 23 lisätään tehtävälle kyseinen tJavaRow-komponentti.

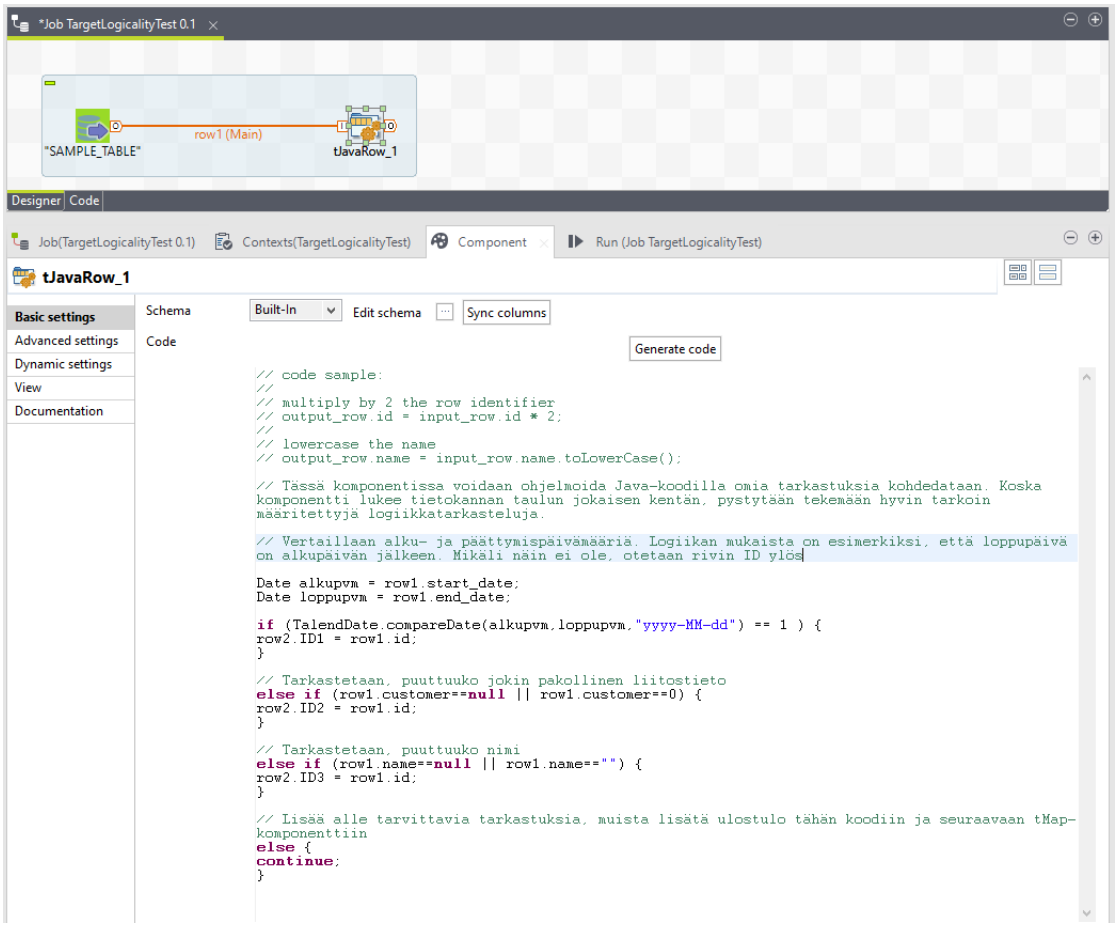


Kuvio 23. tJavaRow-komponentin lisääminen tehtävälle

Kun JavaRow-komponentti lisättiin, voitiin DBInput komponentti yhdistää siihen normalilla row-main liitoksella. Tämän jälkeen JavaRow-komponentin Component-välilehdellä voitiin lisätä tarvittava Java-koodi, joka tekisi dataan tarkastuksia. Tässä vaiheessa ei ruvettu ohjelmoimaan kovin yksityiskohtaista logiikkaa, vaan lisättiin muutamia hyvin yksinkertaisia tarkastuksia. Tehtävälle määritettiin esimerkiksi tarkastus, jossa katsottiin, onko objektin alkupäivämäärä ennen päättymispäivää. Tämä onnistui käyttämällä hyväksi Talendin omaa päivämäärien vertailun funktiota `TalendDate.CompareDate()`, joka palauttaa arvon 1, jos parametrien ensimmäinen päivämäärä on toista päivämäärää suurempi. Funktio puolestaan palauttaa arvon 0, jos päivämäärät ovat samat ja arvon -1, jos ensimmäinen on pienempi kuin toinen. Päivämäärävertailun koodi on esiteltynä kuviossa 24.

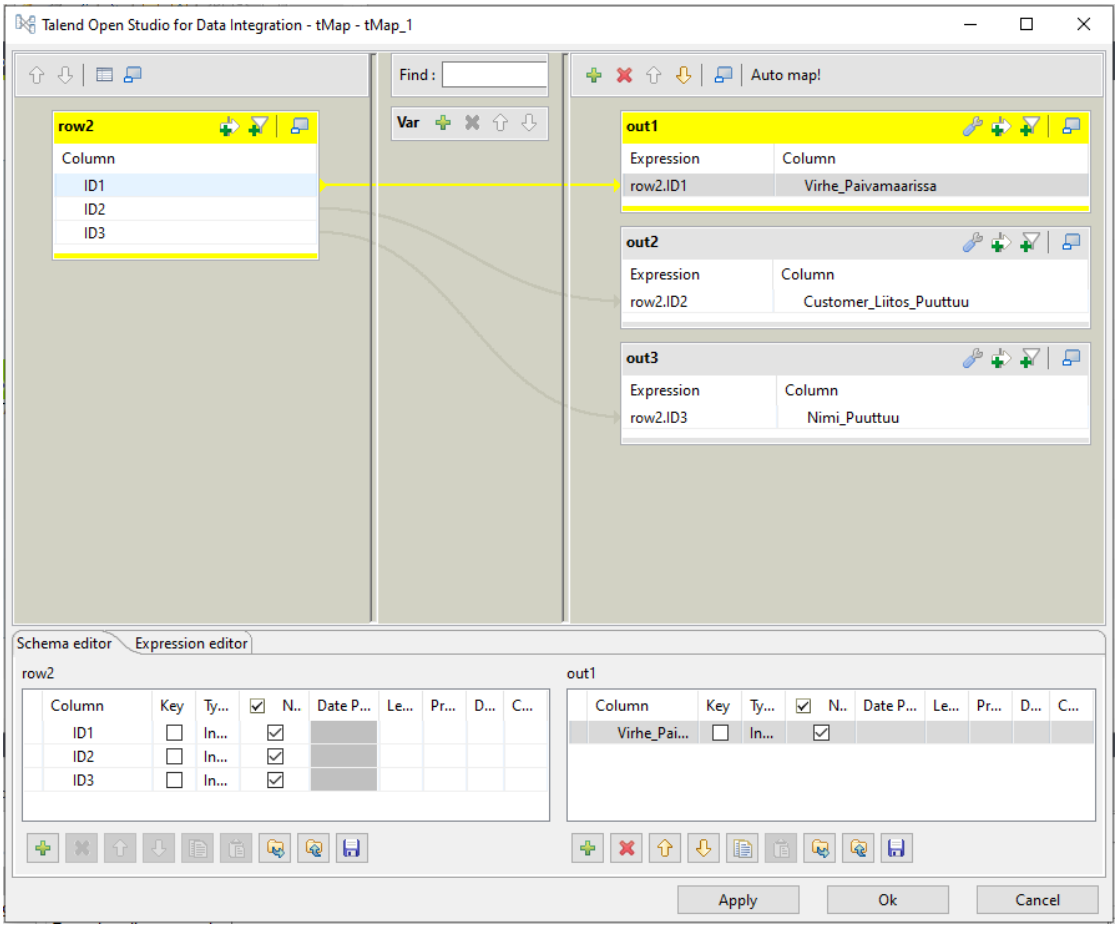
Tämän lisäksi komponenttiin lisättiin tarkastus, puuttuuko jokin ennalta määritetty liitostieto, ja tämän tehtävän mallinnuksessa käytettiin liitostietona SAMPLE_TABLE-datan customer-kenttää. Tässä tarkastuksessa käytettiin ensimmäisen päivämäärävertailunkin mukaisesti yksinkertaista if-ehtoa. Käytännössä siis tarkastettiin, onko kyseisen kentän arvo null tai 0. Mainittujen tarkastusten lisäksi määritin komponentille vielä ehdon, joka tarkistaisi puuttuuko objektilta nimi. Kaikki komponenttiin

määritetyt koodit näkyvät kuviossa 24. Tehtävälle olisi voitu jatkaa tarkempien logiikoiden määrittämistä, mutta tässä opinnäytetyössä riitti todistaa, että tämäntyyppisten logiikkatarkastusten teko on mahdollista ja helposti toteutettavissa.



Kuvio 24. tJavaRow-komponenttiin määritetty logiikkatarkastelun koodi

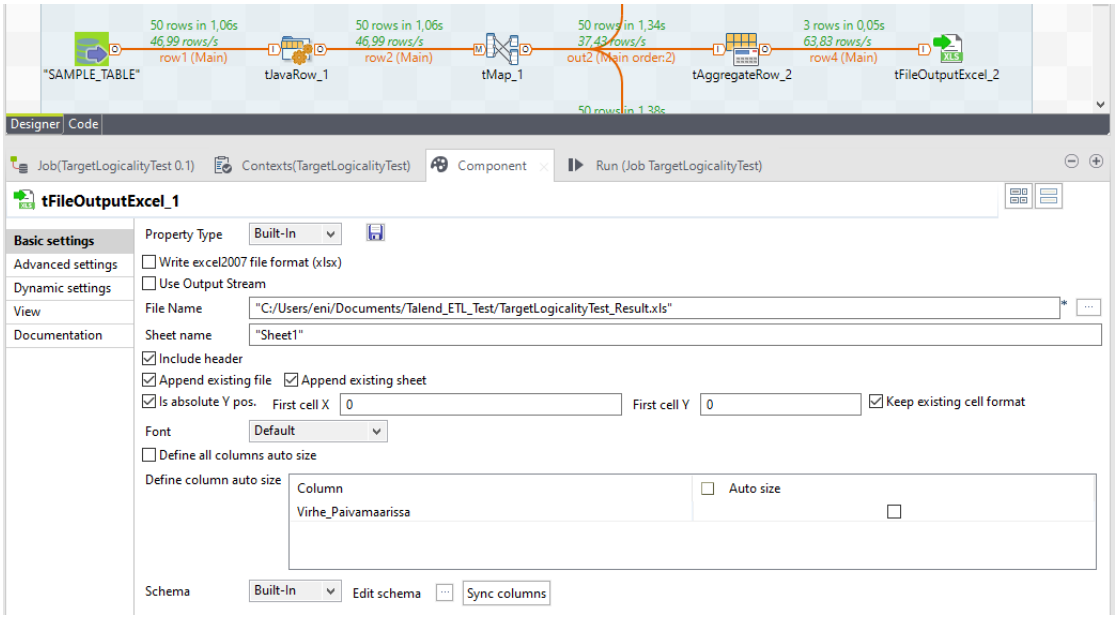
Mikäli JavaRow-komponentissa jokin logiikkatarkastus ei mene läpi, otetaan kyseisen rivin ID ylös asian selvittämistä varten. Tämä onnistui määrittämällä ehtoihin kuviossa 24 näkyvä koodirivi: row2.ID1 = row1.id. Tällä määritettiin siis row1-sisääntulon id-kenttä row2-ulostulon kentäksi nimellä ID1. Jotta tieto voitiin esittää helpolla tavalla luettavaksi, lisättiin tMap-komponentti, joka käsittelisi kentät eteenpäin. tMap-komponenttiin tehtiin kuvion 25 mukaiset määritykset. Määrityksillä saatiin yhdestä ulostulosta kolme eri ulostuloa jokaiselle eri kentälle, ja tämä oli tarpeellinen vaihe, jotta ID-tiedot saatiin kerättyä ja ryhmiteltyä seuraavaa vaihetta varten.



Kuvio 25. TargetLogicalityTest-tehtävän tMap-komponentin määritykset

Koska tMap-komponentista saadaan nyt jokaiselle kentälle oma ulostulo, joudutaan jokaisen ulostulon sisältö ryhmitellä ja koota yhteen. Tämä onnistuu liittämällä ketjuun tAggregateRow-komponentti, jonka toiminnot vastaavat SQL-muotoista group by-toimintoa. tAggregateRow-komponentilla voidaan tehdä muitakin operaatioita kuten esimerkiksi maksimi- ja minimiarvojen valitsemista ja summaamista. Jokaisen kentän ulostulon ketjuun liitetään mainittu tAggregateRow-komponentti, joka tekee ryhmittelyoperaation tuon kentän arvojen mukaisesti. tAggregateRow-komponentista voitiin ottaa tulos ylös jo helposti luettavassa muodossa joko konsoliin tai erilliseen tiedostoon. Tässä tehtävän ketjuun lisättiin edellisen tehtävän tapaan tFileOutPutExcel-komponentti, joka kirjoittaisi tuloksen Excel-taulukkoon. Jokaisen haarautuvan ketjun päässä oleva tFileOutPutExcel-komponentti määritettiin niin, että tieto tallennettiin samaan tiedostoon, jottei tietoja tarvitsisi lukea monelta eri taulukolta. Jokaisen ulostulon tulos kirjattiin siis ylös samaan tiedostoon, eri lohkoihin, ja tämä onnistui määrittämällä tFileOutPutExcel-komponentin asetukset esimerkiksi kuten

kuviossa 26 nähdään. Valinnoilla “Append existing file” ja “Append existing sheet” saatiin data jo olemassa olevalle tiedostolle. Valinnan “Is absolute Y pos” kohdalta määritettiin mihin kohtaan mikäkin tulos listataan taulukkoon.



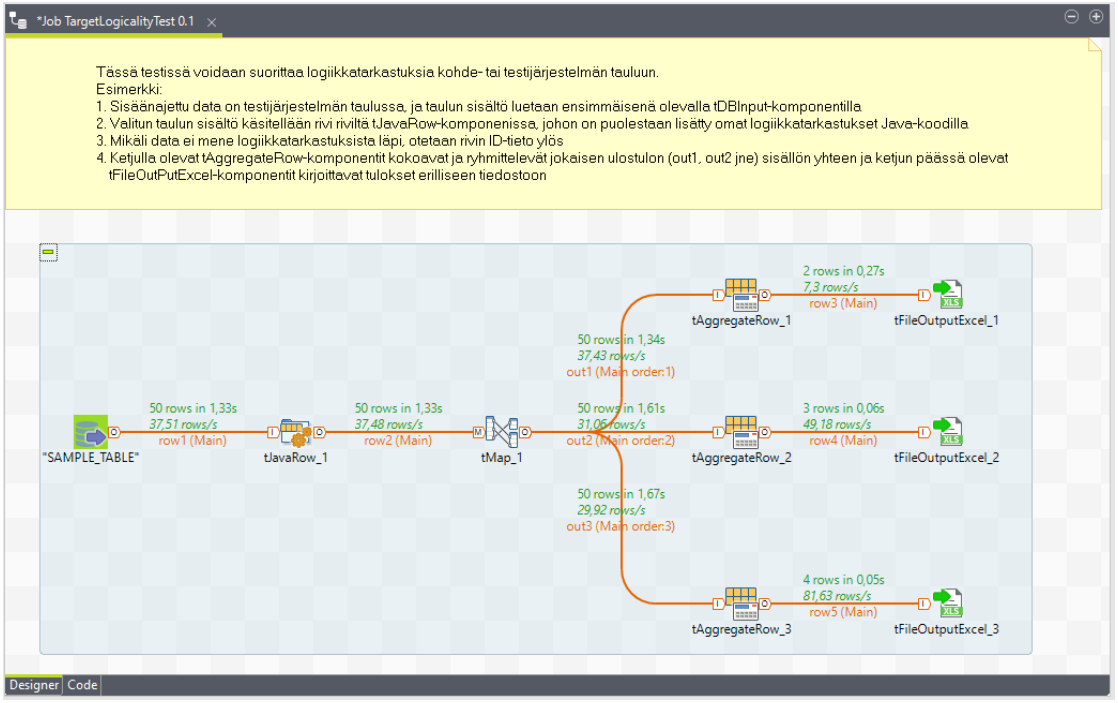
Kuvio 26. tFileOutPutExcel-komponentin määritykset

Myös tämän tehtävän mallinnusta varten muokattiin tarkastelun alla olevaa SAMPLE_TABLE-taulun sisältöä, jotta tehtävä tuottaisi jotain virhetietoa. Tauluun muokattiin esimerkiksi alkupäivämäärä ennen päätttymispäivää sekä poistettiin customer-liitostietoja ja nimiä. Kun tehtävä suoritettiin, logiikkatarkasteluissa kiinni jääneiden rivien ID-tiedot otettiin ylös ja tulos on nähtävillä kuviossa 27.

	A	B	C	D
1	Virhe_Paivamaarissa	Customer_Liitos_Puuttuu	Nimi_Puuttuu	
2	18	38	5	
3		26	7	
4			26	
5				
6				
7				
8				
9				

Kuvio 27. TargetLogicalityTest-tehtävän tulos

Tällä testausmenetelmällä saadaan riittävä tieto siitä, vastaako valitun taulun data määriteltyä logiikkaa. Vaikka tässä vaiheessa tehtävälle ei lisätty kovin monimutkaisia logiikkatarkasteluja, todistaa se kuitenkin sen, että tämän tyyppisten tarkastuksien tekeminen valittuun dataan on mahdollista tehdä ja tulos saadaan koottua helposti luettavaan muotoon. Valmis TargetLogicalityTest-tehtävä on vielä esiteltynä kuviossa 28.

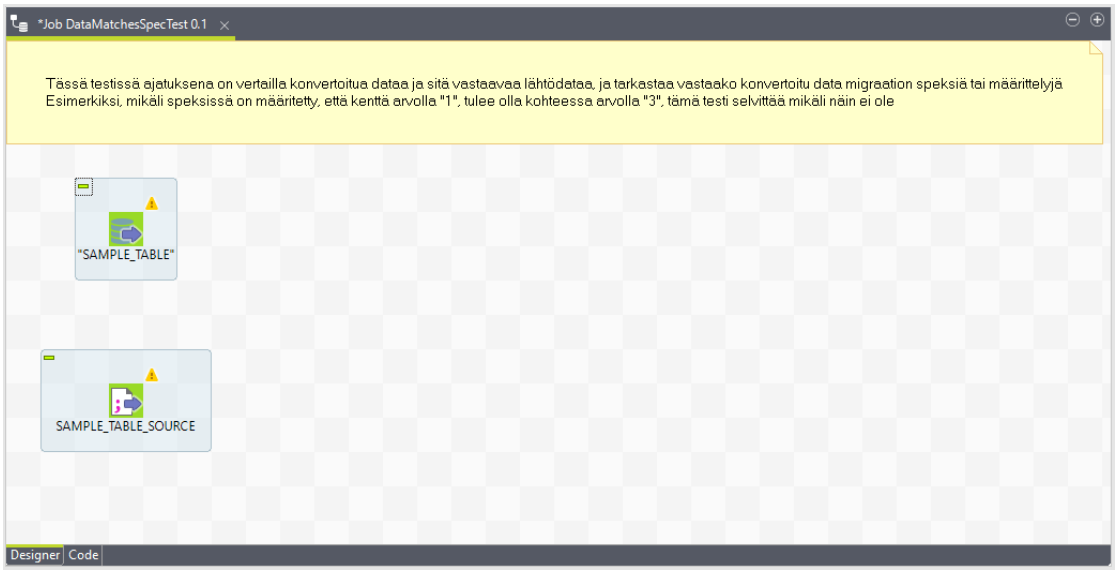


Kuvio 28. Valmis TargetLogicalityTest-tehtävä

5.4 Data Matches Spec Test

Toimeksiantajalla oli tarve myös testaukselle, joka tarkastaisi vastaako konvertoitu tai sisäänajettu data migraation speksiä, eli tarkempia ja yksityiskohtaisia määrityksiä. Tässä vaiheessa suunniteltu tehtävä vertailisi lähtödataa ja kohde- tai testijärjestelmässä olevan taulun dataa, ja huomioisi poikkeavat tiedot perustuen näihin migraation määrityksiin.

Työssä lisättiin uusi tehtävä nimeltään DataMatchesSpecTest ja koska aikaisemmassa vaiheessa tarvittava tietokantayhteys oli jo muodostettu, voitiin lisätä tehtävälle suoraan tDBInput-komponentti, joka lukisi dataa kohde- tai testijärjestelmän tietokannan taulusta. Tämä komponentti määritettiin ja kohdistettiin SAMPLE_TABLE-tauluun. Koska tässä tehtävässä tarkoituksena on vertailla konvertoitua dataa ja lähtödataa, täytyi Metadata-osioon määrittää vastaava CSV-muotoinen lähtödata. Lähtödata määritettiin vastaavalla tavalla kuin aikaisemmin kuviossa 9 on esitetty. Tehtävälle lisättiin tämän jälkeen tFileInputDelimited-komponentti ja se kohdistettiin juuri määritettyyn lähtödataan. Tässä vaiheessa tehtävän rakenne näytti kuvion 29 mukaiselta.

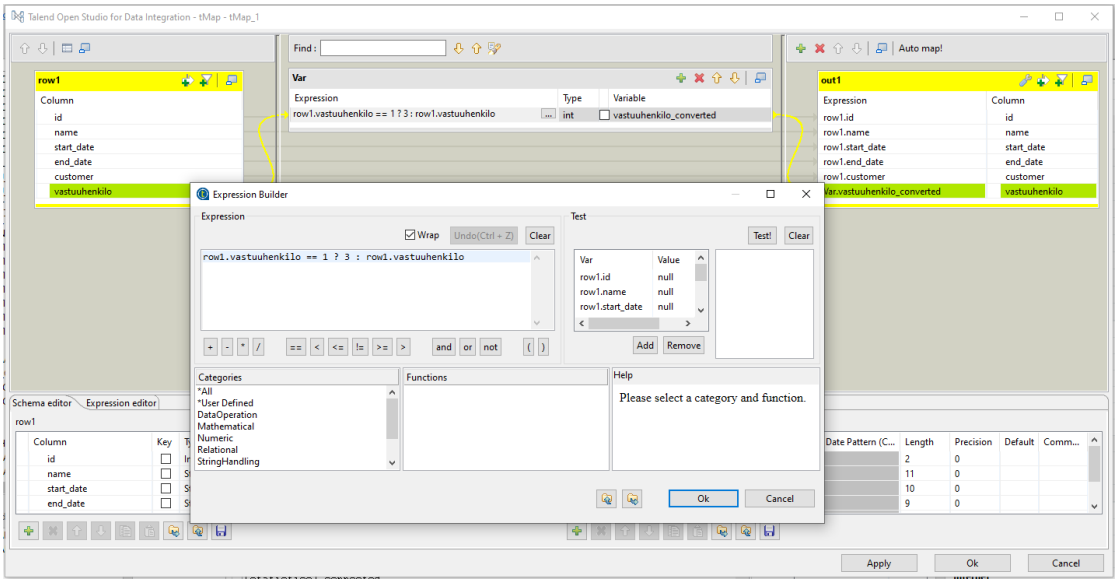


Kuvio 29. Vertailtavat datakomponentit DataMatchesSpecTest-tehtävällä

Seuraavaksi lisättiin lähtödatan komponentin jatkoksi tMap-komponentti, jonka sisälle tehtiin tarkemmat konversiomääritykset, joita sitten myöhemmässä vaiheessa peilattaisiin konvertoituun dataan testi- tai kohdejärjestelmässä.

Tämän tehtävän mallintamisen vuoksi tehtävällä tehtiin tarkastus yhteen lähtödatan kenttään, joka oli vastuuhenkilo-kenttä. Lähtödatan jatkoksi lisätyllä tMap-komponentille lisättiin transformaatiomääritys, joka voidaan sanella suoraan migraation

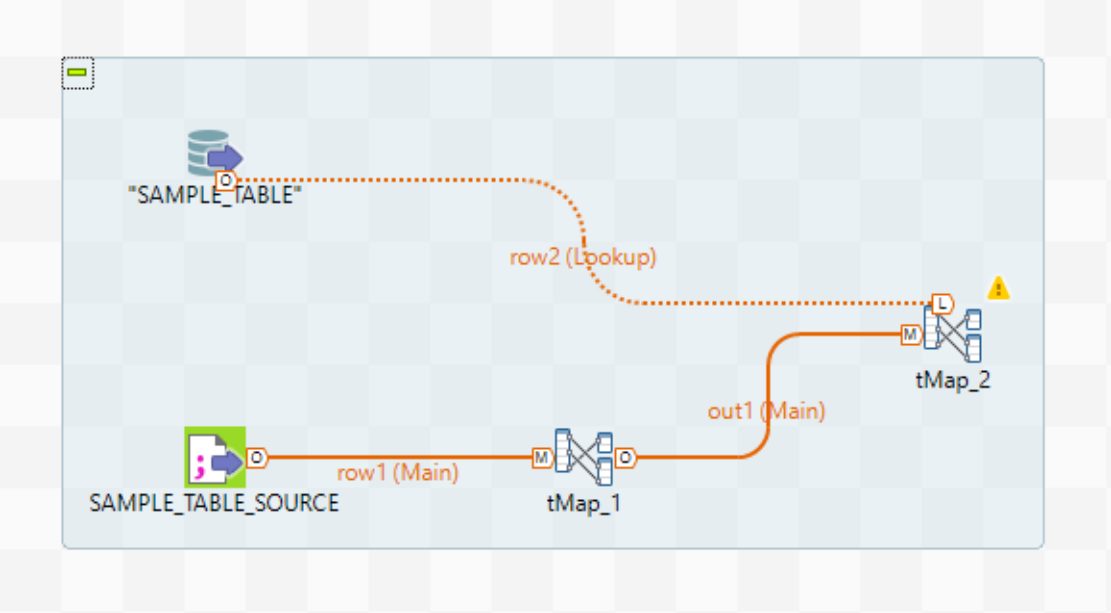
tarkempien määritysten mukaiseksi. Tässä tehtävässä esimerkiksi haluttiin, että vastuuhenkilo-kenttä tulee olla kohdejärjestelmässä arvolla 3, mikäli se lähtödatassa on arvolla 1. tMap-komponentin määritysikkunassa voidaan keskimmäisessä osiossa asettaa transformaatiosääntöjä. Tämän tehtävän esimerkissä käytetty transformaatiosääntö ja ensimmäisen tMap-komponentin määritykset ovat nähtävissä kuviosta 30. Transformaatiosäännöksi muodostui kolmivaiheista operaattoria käyttävä lauseke: `row1.vastuuhenkilo == 1 ? 3 : row1.vastuuhenkilo`. Tällä lausekkeella katsotaan, onko row1-sisääntulon vastuuhenkilo-kenttä arvolla 1, ja jos on, ulostuloksi syötetään arvo 3. Muussa tapauksessa ulostuloksi syötetään vastuuhenkilo-kentän alkuperäinen arvo.



Kuvio 30. DataMatchesSpecTest-tehtävän ensimmäinen tMap-komponentti

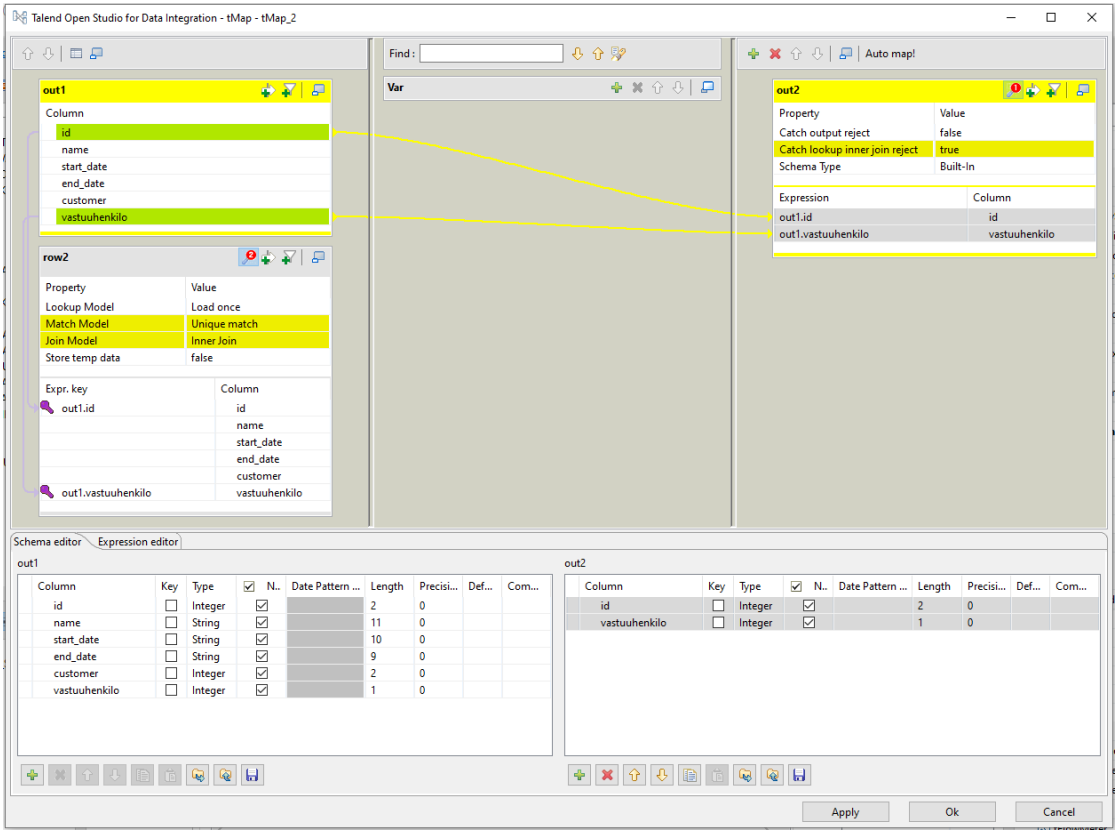
Tämän jälkeen transformaatiosäännöillä muunneltu lähtödata ja konvertoitu data voitiin yhdistää vielä toisella tMap-komponentilla ja liittää taulut yhdistävillä tekijöillä. Tehtävälle lisättiin siis toinen tMap-komponentti, johon liitettiin ensimmäisenä aikaisemman tMap-komponentin ulostulo main-liitoksella. Tämän jälkeen tarkastettavana oleva konvertoitu data liitettiin myös samaan tMap-komponenttiin main-liitoksella, mutta liitos muuttuu automaattisesti lookup-muotoon. Kun tMap-kompo-

nenttiin lisätään toinen sisääntulo niin sanotuksi lookup-tauluksi, voidaan tMap-komponentissa yhdistää nämä taulut, mikäli taulujen välillä löytyy yhdistävä tekijä. Tässä vaiheessa tehtävän rakenne vastasi kuviota 31.



Kuvio 31. Vertailtavien taulujen yhdistäminen tMap-komponentissa

Kun taulut yhdistettiin toiseen tMap-komponenttiin, pystyttiin komponentin määrityksissä asettamaan liitosta varten yhdistävät tekijät. Kuviossa 32 on nähtävissä kyseiset asetukset, joilla taulut liitettiin toisiinsa. Liitoksen muodostamisessa yhdistävinä tekijöitä käytettiin id-, ja vastuuhenkilo-kenttiä. Liitos tehtiin käytännössä siten, että out1-lohkosta raahattiin yhdistävä tekijä row2-lohkon kentän kohdalle, esimerkiksi out1-lohkosta id-kenttä raahattiin row2-lohkon id-kentän kohdalle. Sama tehtiin myös vastuuhenkilo-kentän osalta. Tämän jälkeen liitosmuoto (Join Model) valittiin muotoon “Inner Join”. Inner Join -liitoksella otetaan ulos ainoastaan ne rivit, joilla yhdistävän tekijän arvo on sama molemmissa tauluissa.



Kuvio 32. tMap-komponentin määritykset liitosta varten

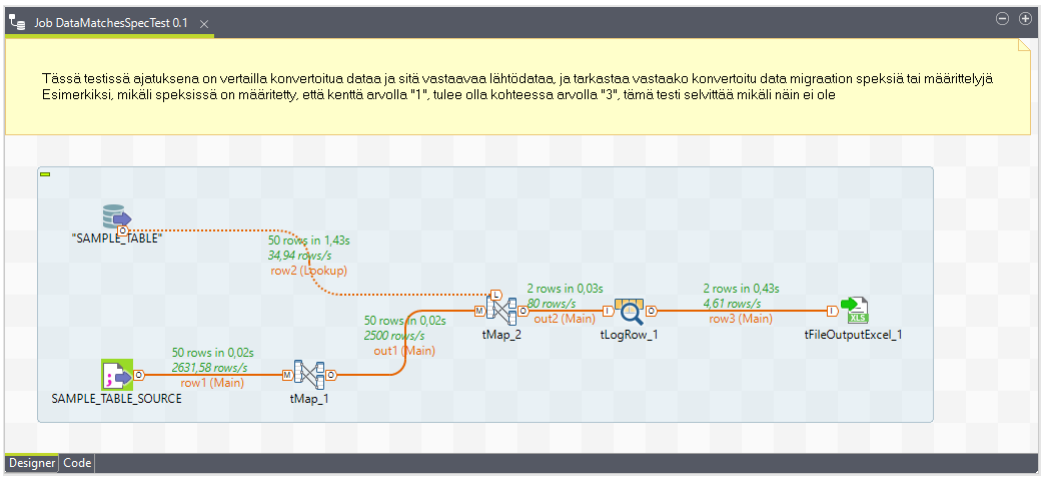
Näiden määritysten jälkeen täytyi vielä vaihtaa ulostulon asetuksista “Catch lookup inner join reject” -asetus muotoon true. Tämä tarkoittaa sitä, että komponentista viedään eteenpäin vain ne rivit, joilla ei yhdistävää tekijää löytynyt.

Tämän jälkeen voitiin tehtävälle lisätä ulostulon tulostusta varten tarvittavat komponentit, ja tässä vaiheessa lisättiin sekä konsoliin kirjoittava tLogRow-komponentti että tFileOutPutExcel-komponentti, joka muodostaisi tehtävän tulokset Excel-taulukkoon. Tehtävän mallinnusta ja testausta varten testattavaa ja konvertoitua data muokattiin tarkoituksella virheelliseksi, ja kahdelle riville muokattiin käsin vastuuhenkilo-kenttään arvo 4. Kun valmis tehtävä suoritettiin, saatiin Excel-taulukkoon tieto virheellisesti konvertoiduista riveistä, joiden vastuuhenkilo-kenttä tarvitsisi vielä tarkastelua. Tässä vaiheessa Excel-tiedostoon muodostunut tulos on kuvattuna kuviossa 33. Tuloksen id-sarakkeeseen nousee virheellisen rivin ID-tieto ja vastuuhenkilo-sarakkeeseen arvo, joka rivillä pitäisi olla vastuuhenkilo-kentässä.

	A	B	C	D	E
1	**Rivit, jotka eivät vastaa migraation speksiä**				
2					
3	id	vastuuhenkilo			
4	28	3			
5	36	5			
6					
7					
8					

Kuvio 33. DataMatchesSpecTest-tehtävän tulos

Tämän tehtävän mallintamisen vuoksi käytettiin yhtä tarkasteltavaa kenttää, mutta muiden määritysten ja kenttien tarkastaminen on mahdollista samalla tavalla, kuin tässä työssä ja tässä vaiheessa on kuvattu. Tällä testauksella saadaan riittävä tieto siitä, onko konvertoitu data migraation tarkempien määritysten mukaista. Käytännössä testillä tarkastetaan ainoastaan se, vastaako tarkasteltava kenttä erikseen määritettyä transformaatiomääritystä. Testi tuottaa testituloksena tiedon, mikä rivi vaatii vielä tarkempaa tarkastelua. Valmis tehtävä ja sen rakenne on nähtävissä kuvio-sta 34.



Kuvio 34. Valmis DataMatchesSpecTest-tehtävä

6 Tulokset

Työn tavoitteena oli tutkia ja selvittää millä tavoin datamigraation testausta voidaan toteuttaa automaattisesti. Työssä lähdettiin kattavien alkuselvitysten jälkeen tutkimaan markkinoilla olevia vaihtoehtoja automaattisen ETL-testauksen työkaluista. Kun markkinoiden suosituimpia datamigraatiotestauksen työkaluja oli vertailtu ja osaa jopa kokeiltu, päädyttiin lopulta Talend Open Studio for Data Integration -työkaluun.

Koska datamigraatioita toteutetaan hyvin räätälöitävään toiminnanohjausjärjestelmään, on usein myös datamigraatioiden määritykset ja vaatimukset erittäin yksityiskohtaisia ja tarkoin määritettyjä. Talendin tarjoamaa työkalua kokeillessa havaittiin, että se soveltuu vertailtavista vaihtoehdoista parhaiten toimeksiantajan käyttöön, sillä Open Studio for Data Integration -työkalulla voitiin määrittää lopulta erittäin spesifisiä testausmenetelmiä. Koska tarkoin määritettyjen testausmenetelmien määrittäminen on tarpeellista räätälöitävissä olevan toiminnanohjausjärjestelmän kanssa, oli se yksi syy miksi Talendin tarjoamaan työkaluun päädyttiin. Toinen syy valinnalle oli sovelluksen helppo saatavuus, sillä työkalu on vapaasti ja ilmaiseksi ladattavissa, koska se perustuu avoimeen lähdekoodiin. Tämän lisäksi sillä saatiin parhaiten muodostettua tarvittavia tietokantayhteyksiä, jotka oleellisesti vaikuttivat valintaan.

Työkalun valinnan jälkeen lähdettiin suunnittelemaan toimeksiantajan tarpeiden perusteella kolmea eri testimenetelmää. Toimeksiantajalla oli tarve testata datamigraatioissa sisään ajettavaa dataa varmistaen, että data olisi kohteen tietokantarakenteen mukaista ja ehjää. Toisena tarpeena oli testata, että data on kohteen järjestelmän logiikan mukaista, ja kolmantena haluttiin testata data vielä niin, että saataisiin tieto siitä, vastaako data migraation sovittuja määrittelyjä. Näiden tarpeiden perusteella muodostettiin kolme erilaista ratkaisua, joilla todistettiin tarvittavien testausten olevan mahdollista toteuttaa.

Talendissa testimenetelmiä lähdettiin suunnittelemaan sillä tietämyksellä, joka itselläni jo entuudestaan kyseisen työkalun komponenteista oli, ja tämän lisäksi turvauduttiin hyvin vahvasti Talendin help-sivustolla olevaan pikaoppaaseen, jossa yleisimmistä komponenteista oli ohjeistuksia ja esimerkkejä. Tässä opinnäytetyössä ja sen teknisessä toteutuksessa tarvittiin hieman ohjelmoinnin tietämystä, tietokantaosaamista sekä ValueFrame-toiminnanohjausjärjestelmän asiantuntemusta.

Tässä opinnäytetyössä saatiin todettua, että datamigraation testausta on mahdollista toteuttaa automaattisesti monen eri työkalun avulla. Tämän lisäksi sovellettiin toimeksiantajan käyttöön yksi mahdollinen työkalu, jolla todettiin toimeksiantajan tarpeiden mukaisten testausten olevan mahdollisia. Keskeisenä tuloksena saatiin siis ratkaistua toimeksiantajan ongelma ja vastattua alkuperäiseen tutkimuskysymykseen.

7 Pohdinta

Mielestäni opinnäytetyölle asetetut tavoitteet saavutettiin hyvin ja tutkimusongelmaan löydettiin ratkaisu. Testausmenetelmien suunnittelussa ja määrittämissä tuli vastaan monenlaisia haasteita, mutta lopulta saatiin muodostettua tarvittavia ja toimivia ratkaisuja. Eniten aikaa ja haasteita ilmeni Talendin erilaisten komponenttien tutkimisessa ja sovittaessa näitä yhteen toimiviksi. Tämän lisäksi, vaikka java-ohjelmoinnista oli hyvää perustietämystä, Talend Open Studio for Data Integration -työkalussa on paljon sisäistä toteutusta, joka poikkeaa hieman perinteisestä java-kielestä, ja tämä puolestaan tuotti haasteita tehtävien ongelmien debuggaamisessa ja korjaamisessa.

Tämän työn selvitykset ja tutkimustulokset ovat hyödyllisiä datamigraatioiden ja muiden ETL-prosessien parissa työskenteleville, jotka etsivät ratkaisuja manuaalisen datamigraatiotestauksen tilalle. Vaikka tässä työssä käyttöön sovellettu Talend Open Studio for Data Integration -työkalu ei soveltuisi muiden käyttöön, on työssä esitelty muitakin vaihtoehtoja ja ne voivat auttaa eri ratkaisujen ja työkalujen vertailussa.

Vaikka tässä työssä esitellyt testimenetelmien esimerkit eivät välttämättä ole yksinkertaisimpia ja täysin loppuun hiottuja, antavat ne silti riittäviltä osin vastaukset siitä, kuinka toimeksiantajan tarpeisiin peilaten datamigraation testaus voidaan suorittaa. Koska työssä muodostetut testimenetelmät ovat mallintavia esimerkkejä, vaativat ne vielä jonkun verran jatkojalostusta varsinaista käyttöönottoa varten. Tämän lisäksi jokaista testimenetelmää ja sen rakennetta voidaan jatkokehittää hyvinkin pitkälle, jolloin testauksilla saadaan erittäin spesifejä testituloksia. Talendin kattava komponenttikirjasto mahdollistaa sen, että tehtävistä saadaan muodostettua loppujen lopuksi hyvin monimutkaisia. Yksi mahdollinen ja hyödyllinen jatkokehitysehdotus olisi esimerkiksi jokaisen erillisen tehtävän sitominen yhteen, jolloin yhdellä testiajolla saataisiin jokaiset testitulokset luettavaksi yhdestä tiedostosta.

Lähteet

Cambien, P. 2018. Setting up an SSH-tunnel using plink. Viitattu 15.2.2021. https://medium.com/@incubusattax/setting-up-an-ssh-tunnel-using-plink-7d8dacfd4014

Considerations When Planning To Test Data Migration. N.d. Artikkeli Qualitest verkkosivuilla. Viitattu 4.4.2021. https://www.qualitestgroup.com/white-papers/considerations-planning-test-data-migration/

Data Migration Testing Tutorial: A Complete Guide. 2020. Artikkeli datamigraation testauksesta Software Testing Help sivustolla. Viitattu 2.1.2021. https://www.softwaretestinghelp.com/data-migration-testing/

Dufrasne, B., Warmuth, A., Appel, J., Bauer, W., Douglass, S., Klee, P., Pura, M., Wells, M. & Wesselbaum, B. 2017. DS8870 Data Migration Techniques. IBM Redbooks. Viitattu 20.12.2020. https://play.google.com/books/reader?id=3fHDCAAAQBAJ&hl=en&pg=GBS.PP1

ETL Testing. N.d. ETL testauksen ja Data Validation -ratkaisun esittely Informatican verkkosivuilla. Viitattu 14.1.2021. https://www.informatica.com/etl-testing.html

How to Set up SSH Tunneling (Port Forwarding). 2019. Artikkeli SSH tunneloinnista Linuxize verkkosivuilla. Viitattu 15.2.2021. https://linuxize.com/post/how-to-setup-ssh-tunneling/

Katzoff, D. N.d. Data Migration Testing Strategy: A Complete Guide to Data Migration Testing Success. Ohjeistus Data Migration Pro sivustolla. Viitattu 3.1.2021. https://www.datamigrationpro.com/data-migration-testing-strategy/

King, T. 2019a. The 9 Best ETL Testing Tools for Data Integration Success. Verkkojulkaisu Solutions Review -sivustolla. Viitattu 14.1.2021. https://solutionsreview.com/data-integration/the-best-etl-testing-tools-for-data-integration-success/

King, T. 2019b. Top 12 Free and Open Source ETL Tools for Data Integration. Verkkojulkaisu Solutions Review -sivustolla. Viitattu 14.1.2021. https://solutionsreview.com/data-integration/top-free-and-open-source-etl-tools-for-data-integration/

Matthes, F. & Schulz, C. 2011. Towards an integrated data migration process model - State of the art & literature overview. Viitattu 27.12.2020. http://docplayer.net/52249166-Towards-an-integrated-data-migration-process-model-state-of-the-art-literature-overview-florian-matthes-christopher-schulz.html

Me olemme Pinja. N.d. Pinja Group Oy:n verkkosivut. Viitattu 16.12.2020. https://www.pinja.com/me-olemme-pinja/

Top 10 ETL Testing Tools In 2021. 2020. Arvostelu ja listaus ETL testaustyökaluista Software Testing Help -sivustolla. Viitattu 14.1.2021. https://www.softwaretestinghelp.com/top-4-etl-testing-tools/