LAPIN AMK
Lapland University of Applied Sciences

DATAPUTKEN SUUNNITTELU AZURE-YMPÄRISTÖSSÄ

Raportoinnin automatisointi

Varjonen Minna

Opinnäytetyö

Tietojenkäsittelyn koulutus
Tradenomi (AMK)

2023

LAPIN AMK
Lapland University of Applied Sciences

Opinnäytetyön tiivistelmä

Tietojenkäsittelyn koulutus
Tradenomi (AMK)

Tekijä	Minna Varjonen	**Vuosi**	2023
Ohjaaja(t)	Pekka Reijonen		
Toimeksiantaja	CGI Suomi Oy		
Työn nimi	Dataputken suunnittelu Azure-ympäristössä: Raportoinnin automatisointi		
Sivumäärä	48		

Tämän opinnäytetyön tarkoituksena oli kehittää ETL-/ELT-dataputki määriteltyä käyttötapausta varten. Tavoitteena oli automatisoida asiakkaiden tarvitsemien tietojen tuottaminen sekä yrityksen sisäisen raportoinnin tarvitsemien tietojen toimittaminen.

Opinnäytetyössä tutkittiin tapaustutkimuksen ja toimintatutkimuksen keinoin, miten dataputki kannattaa toteuttaa määritellyssä käyttötapauksessa. Työhön sisältyi prosessien suunnittelu, teknisten ratkaisujen valinta ja itse toteutustyö. Toteutus oli ennalta rajattu Azure-alustalle.

Työn tuloksena syntyi tekninen toteutus, joka siirtää tarvittavan datan automaattisesti lähdejärjestelmästä haluttuihin kohteisiin halutuissa tietomuodoissa. Ratkaisu tehtiin tietylle käyttötapaukselle, joten tutkimus rajattiin sen mukaisesti. Tarve oli kuitenkin varsin tyypillinen, joten työn tuloksia voidaan hyödyntää soveltuvilta osin myös muissa vastaavissa tarpeissa.

Avainsanat	data, SQL, tietokannat, ohjelmointirajapinnat, tietämyksenhallinta
Muita tietoja	Työhön liittyy toimeksiantajalle kehitetty ETL-/ELT-ratkaisu

LAPIN AMK
Lapland University of Applied Sciences

Abstract of thesis

Business Information Technology
Bachelor of Business Administration

Author	Minna Varjonen	**Year**	2023
Supervisor(s)	Pekka Reijonen		
Commissioned by	CGI Suomi Oy		
Title	Designing data pipeline in Azure environment: Automation of reporting		
Number of pages	48		

The purpose of this thesis was to develop an ETL/ELT data pipeline for a pre-defined use case. The goal was to automate the production of the information needed by customers and the provision of the information needed for internal reporting within the company.

Through case study and action research, the thesis examined how to implement a data pipeline in a pre-defined use case. The study included process design, selection of technical solutions and the implementation work of the solution itself. The implementation was predefined for the Azure platform.

The result of the work was a technical implementation that automatically transfers the necessary data from the source system to the desired targets in the desired data formats. The solution is made for a specific use case and the study is limited accordingly. However, the need was quite typical, and based on which the results of the study can be applied to other similar needs.

Keywords	data, SQL, databases, application programming interfaces, business intelligence
Special remarks	The thesis includes an ETL/ELT solution developed for the commissioner

SISÄLLYS

1 JOHDANTO

1.1 Taustaa aiheen valintaan

”Data on uusi öljy”, totesi matemaatikko Clive Humby vuonna 2006. Öljyn tavoin data on arvokasta, mutta vaatii jalostamista ollakseen käyttökelpoista ja muuttuakseen tiedoksi. (TowardsDataScience 2022.) Sanalla data tarkoitetaan tietojärjestelmiin tallennettua koneellisesti käsiteltävissä olevaa raakatietoa. Informaatio puolestaan on tulkittavissa olevaa rakenteista dataa, josta voidaan jalostaa tietoa. Tästä lopulta päästään tietoon: sanalla tieto viitataan kollektiivisesti informaatiosta jalostettuun ja sisäistettyyn tietämykseen ja ymmärrykseen. (Helsingin kaupunki 2023.)

Nykyään puhutaan paljon datavetoisesta päätöksenteosta, jolla tarkoitetaan faktojen, mittareiden ja tietojen käyttämistä liiketoimintapäätösten tekemiseen (Nelson 2022). Jotta jalostunutta tietoa ja raportteja olisi saatavilla päätöksentekoa varten, dataa tulee kerätä ja muokata eri lähteistä tietovarastoihin. Kun järjestelmiä ja rajapintoja tulee yhä lisää, datan siirtäminen paikasta toiseen muuttuu yhä monimutkaisemmaksi (Reeve 2013, 3).

Datan siirtäminen nopeasti paikasta toiseen ja muuntaminen hyödynnettävään muotoon on usein monimutkainen tehtävä, johon tarvitaan tehokkaita työkaluja, käytäntöjä ja osaamista. Tämän alueen ammattilaisia ovat datainsinöörit, joiden työtä on datan valmistelu raportointia tai operatiivista käyttöä varten. Datainsinöörit rakentavat dataputkia, joissa dataa kerätään eri lähdejärjestelmistä, muunnetaan, puhdistetaan, yhdistetään ja viedään tietovarastoihin. (Lutkevich 2021.) Dataputki, josta usein puhutaan sen englanninkielisellä nimellä data pipeline, tarkoittaa automaattista ketjua, jossa suoritetaan datalle operaatioita (Dutrée 2021). Se on hallittu toimintokokonaisuus datan jalostukseen ja liiketoiminta-arvoa tuottavien datatuotteiden, esimerkiksi raporttien, luontiin. Yhteistä kaikille dataputkille on, että ne ovat automatisoituja ja toistettavissa. Lisäksi dataputki sisältää ja yhdistää useita komponentteja kattaen lähdedatan lukemisen, datan muokkauksen ja analysoinnin, tallennuksen sekä datan aktivoinnin jalostetun datatuotteen kautta. (Tietoevry 2020.)

Opintojeni aikana olen saanut datainsinöörin tehtävissä tarvittavaa perustietoa, josta on apua osaamisen syventämisessä; olen perehtynyt esimerkiksi pilvipalveluihin, tietokantoihin, liiketoimintatiedon analysointiin ja low code -ohjelmointiin. Painoarvoa oli myös sillä, että tällaiselle osaamiselle on kysyntää: data engineering on yksi nopeimmin kasvavista tietotekniikka-alan tehtävistä (Dice 2020). Opinnäytetyöllä on arvoa paitsi yksittäisenä kehitystehtävänä, myös tietojen ja taitojen kartuttamisessa tulevia projekteja varten.

1.2 Kehittämistarve

Tämä kehittämispainotteinen opinnäytetyö toteutetaan laadullisena tutkimuksena. Aiheena on ETL-/ELT-prosessin eli dataputken suunnittelu ja toteutus ennalta määritellylle käyttötapaukselle. ETL tarkoittaa datan siirtämistä, muokkaamista ja lataamista: tiedot haetaan (Extract) lähdejärjestelmistä, niitä muokataan (Transform) ja ladataan (Load) lopulta tietovarastoon (Hovi 2018). ELT (Extract, Load, Transform) puolestaan siirtää datan lähdejärjestelmistä suoraan kohteeseensa, ja tarvittavat muokkaukset tehdään vasta kohdetietovarastossa (IBM 2023b). ETL ja ELT ovat yleisesti käytettyjä dataputkityyppejä vaativien datankäsittelytehtävien suorittamiseen (Dutrée 2021). Tässä työssä käytetään molempia lähestymistapoja.

Tehtävänä on noutaa lähdejärjestelmästä API-rajapinnan avulla tietoja, jotka muutetaan a) asiakkaille toimitettavaan muotoon ja b) sisäisen raportoinnin vaatimaan muotoon. API (application programming interface) on ohjelmointirajapinta, jonka avulla eri osapuolet voivat vaihtaa tietoa ja tehdä pyyntöjä eri järjestelmien ja palveluiden välillä (Tietoevry 2020). Se on joukko määriteltyjä sääntöjä, joiden avulla järjestelmät kommunikoivat ja siirtävät tietoa toistensa kanssa (IBM 2023a).

Kehittämistarve on siis kaksiosainen. Ensimmäinen osa on tarvittavan datan noutaminen lähdejärjestelmästä ja muuntaminen asiakkaan kanssa sovittuun muotoon. Toinen osa on datan noutaminen lähdejärjestelmästä ja muuntaminen sisäisen raportoinnin vaatimaan muotoon. Etukäteen on päätetty, että raportointi tullaan rakentamaan Microsoft Power BI:llä, joka on datan visualisointiin ja liiketoimintatiedon analysointiin ja hyödyntämiseen (business intelligence)

tarkoitettu työkalu. Sen avulla dataa voidaan noutaa erilaisista lähdejärjestelmistä ja muuntaa se monipuolisiksi visualisoinneiksi ja raporteiksi. Kehitystyö tehdään Power BI Desktopilla, jonka jälkeen raportit julkaistaan Power BI -palveluun käyttöä varten. (Tutorials Point 2023.)

Tässä tapauksessa dataa ei saada ladattua Power BI:hin suoraan lähdejärjestelmästä, vaan se tulee noutaa API-rajapinnan kautta. Datan siirtämiseksi rakennetaan dataputket. Etukäteen on päätetty, että dataputket toteutetaan Azure-alustalla. Käytännössä työ tehdään datan integrointityökalu Azure Data Factorylla (ks. Microsoft 2023a). Työn raportoinnissa keskitytään siihen, miten dataa kannattaa liikuttaa tai muuntaa parhaan lopputuloksen saavuttamiseksi, ja rajataan itse Azure-ympäristön rakentaminen ulkopuolelle.

Opinnäytetyön päätutkimuskysymys on, miten dataputki on järkevintä ja tehokkainta toteuttaa määritellyssä käyttötapauksessa. Työn tulos on toimiva ETL-/ELT-prosessi, joka a) toimittaa asiakkaan tarvitsemat tiedot halutussa muodossa ja b) mahdollistaa datan hyödyntämisen Power BI:llä. Koodi tallennetaan ja prosessi dokumentoidaan Azure DevOpsiin, jotta sen ylläpito jatkossa on sujuvaa. Azure DevOps on ympäristö, jonka avulla voidaan hallinnoida sovelluksen koko elinkaari. Se tarjoaa esimerkiksi versionhallinnan, projektinhallinnan, automatisoidut koontiversiot ja testaus- ja julkaisuhallintaominaisuudet. (Bigelow 2023.)

2 AIEMPIA ETL-AIHEISIA OPINNÄYTETÖITÄ

ETL- ja ELT-ratkaisuja on tehty jo pitkään, joten aihealue on laajasti tunnettu asiantuntijoiden keskuudessa. Työkaluja dataputkien tekemiseen on tarjolla runsaasti (Datacamp 2023), ja luonnollisesti kaikilla johtavilla pilvitoimittajilla (Microsoft, AWS ja Google) on tarjolla omat ratkaisunsa, kuten myös runsas valikoima toteutuksiin kiinteästi liitettäviä tiedon varastointiratkaisuja ja erilaisia tietokantoja. Datan hyödyntämiseen ja käsittelyyn tarkoitetut työkalut kehittyvät nopeasti, koska koko ajan kasvaville datamäärille tarvitaan yhä nopeampaa ja tehokkaampaa käsittelyä. Lähteiden ajantasaisuus on näin ollen tärkeää tarkistaa.

Dataputkien kehittämistä on aiemmissa opinnäytetöissä käsitelty jonkin verran eri näkökulmista. Miika Pekkarisen (2022) opinnäytetyö “ETL-prosessin toteuttaminen Azure Data Factoryllä” käsittelee tietovarastointiarkkitehtuureja ja tietovarastoinnin eri vaiheita, ja sen tuotos on perehdytysmateriaali ETL-prosessin toteuttamiseen toimeksiantajayritykselle. Myös Janne Kemppaisen (2015) opinnäytetyö “ETL-prosessin suunnittelu” keskittyy ETL-prosessin käsitteisiin, vaiheisiin ja kehittämisessä huomioon otettaviin asioihin. Case study -tyyppisiä opinnäytetöitä ovat puolestaan Pham Phuongin (2020) “A case study in developing an automated ETL solution : concept and implementation” ja Jenna Mustosen (2020) “On-premises-datan käsittely pilviraportoinnissa: case CGI Suomi Oy”. Näissä lopputuloksena on käytännön ratkaisu datan liikuttamiseen ja muuntamiseen käyttökelpoiseen muotoon.

Microsoftin dokumentaatiossa on avattu datan käsittelyyn liittyviä peruskäsitteitä, mutta erityisesti se antaa luotettavaa ja ajantasaista tietoa Azure Data Factoryn toiminnan ja ominaisuuksien selvittämisessä. Teknistä kirjallisuutta Data Factorysta on varsin vähän, mutta esimerkiksi “Hands-On Data Warehousing with Azure Data Factory” esittelee Data Factoryä käytännön näkökulmasta (Cote, Kamrat Gutzait & Ciaburro 2018). Laajempaa teoriapohjaa dataintegraatioon antaa kirja “Managing Data in Motion: Data Integration Best Practice Techniques and Technologies” (Reeve 2013).

3 OPINNÄYTETYÖN TARKOITUS, TAVOITE JA TUTKIMUSKYSYMYKSET

Opinnäytetyön tarkoitus on kehittää ennalta määriteltyä käyttötapausta varten luotettava ja hyväksi todettujen käytäntöjen mukainen ETL/ELT-dataputki. Tavoitteena on automatisoida asiakkaiden tarvitsemien tietojen tuottaminen sekä yrityksen sisäisen raportoinnin tarvitsemien tietojen toimittaminen. Työn tarve on syntynyt järjestelmämuutoksesta, ja samalla datan käsittelystä on tarkoitus tehdä aiempaa nopeampaa ja tehokkaampaa. Kyse on varsin tyypillisestä datankäsittelytarpeesta, joka sisältää teknisiä haasteita ja vaatii ratkaisujen pohtimista. Azure Data Factory on näissä tehtävissä paljon käytetty ympäristö. Onkin hyvin todennäköistä, että työstä saaduille opeille on jatkossa käyttöä sekä opinnäytetyön tekijälle itselleen että kollegoille.

Opinnäytetyön päätutkimuskysymys on, miten dataputki on järkevintä ja tehokkainta toteuttaa määritellyssä käyttötapauksessa. Toimeksiantaja on määritellyt tarpeet tarkasti jo etukäteen. On etukäteen tiedossa, mitä lähdejärjestelmästä saadaan ja mikä on haluttu lopputulos. Vaikka toimintaympäristö on etukäteen määritelty, toteutukselle on useita vaihtoehtoja. Tavoitteena on löytää mahdollisimman luotettava ja tehokas ratkaisu.

Ensimmäinen apututkimuskysymys on, mitä toimeksiantajan määrittelemään lopputulokseen pääseminen vaatii. Tulee selvittää, miten data noudetaan lähdejärjestelmästä. Lisäksi tulee selvittää, millaisia muunnoksia datalle tulee tehdä, jotta se on tietovarastossa käytettävissä halutussa muodossa. Lisäksi tulee selvittää, missä prosessin vaiheessa datan muunnokset kannattaa tehdä.

Toinen apututkimuskysymys on, mitä Azuren tai muita komponentteja työssä kannattaa käyttää ja miksi. Kun tiedetään, mitä vaiheita prosessissa tarvitaan, tulee valita sopivimmat tekniset ratkaisut. Tutkitaan, voidaanko käyttää Data Factoryn natiiveja komponentteja vai tuleeko komponenttivalikoimaa laajentaa ja millaisia tietovarastoratkaisuja tarvitaan.

4 MENETELMÄLLINEN TOTEUTUS

4.1 Tutkimus- ja kehittämismenetelmät

Kehittämistehtävä suoritetaan suuren IT-konsultointiyrityksen data- ja analytiikkapalveluja tarjoavan yksikön alaisuudessa. Tekniseksi toimintaympäristöksi on ennalta määritelty Microsoft Azure, joka tarjoaa laajan työkaluvalikoiman ja hyvät integraatiorajapinnat. Ympäristö tarjoaa erinomaiset mahdollisuudet tarkoituksenmukaisten käytännön ratkaisujen tekemiseen toimeksiannon toteutuksessa.

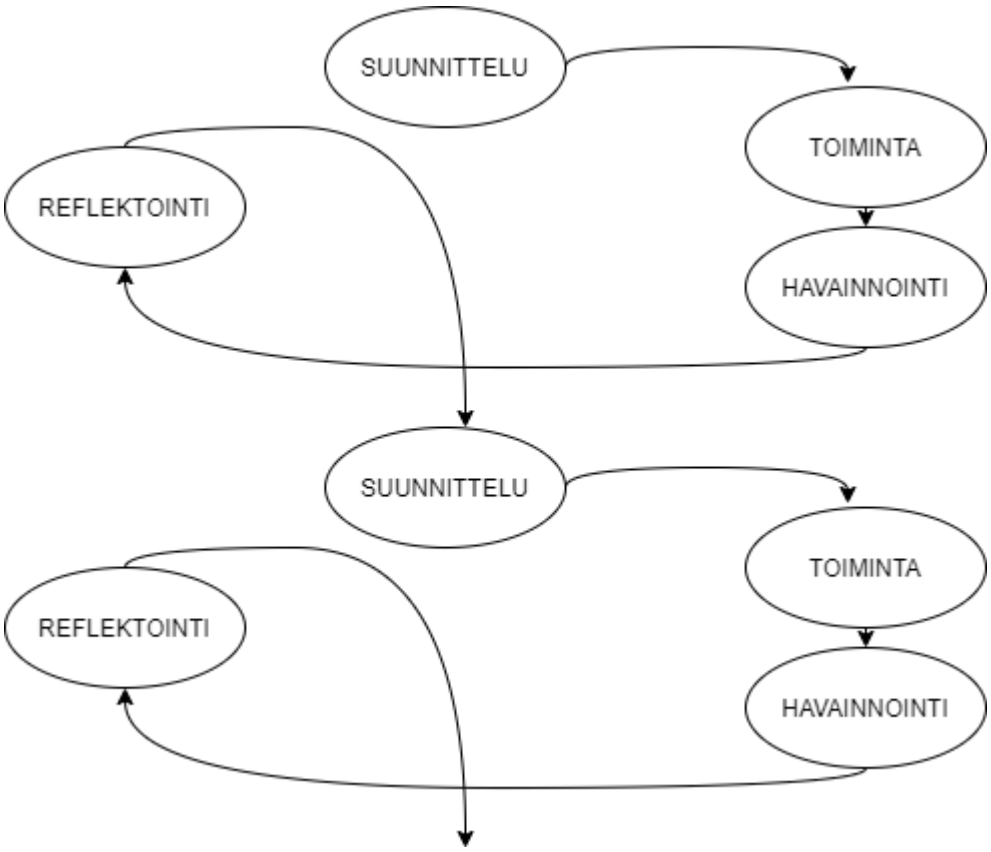
Opinnäytetyö on kehittämispainotteinen, ja se toteutetaan laadullisena tutkimuksena. Laadullisen tutkimuksen ominaispiirre on, että se perustuu ihmisten subjektiivisten kokemusten ja näkemysten tarkasteluun. Näin ollen tutkimuksessa tarkastellaan teorian, empirian ja käytännön yhteyttä. Teoria käsittelee yleisiä lainalaisuuksia, kun taas kokemuksellinen ja käytännön tieto ovat erilaisia eri tilanteissa. (Juuti & Puusa 2020, 56–57.) Kokemuksellinen tieto on tärkeässä asemassa toimivan ratkaisun luomisessa, sillä vaikka teoriatietoa aiheesta on erinomaisesti saatavilla, keskeinen kysymys on, miten tämä kannattaa tehdä. Niin sanottu hiljainen tieto, joka on hankittu kokemuksen kautta (Anttila 2014), on oman kokemukseni mukaan usein oleellisen tärkeää myös teknisissä kehittämistehtävissä.

Laadullisen tutkimuksen lähestymistapana opinnäytetyössä käytetään ensisijaisesti tapaustutkimusta. Tapaustutkimukselle tunnetaan useita määritelmiä, mutta sen voi kiteyttää tutkimusstrategiaksi, joka mahdollistaa aiheen tarkastelun sen omassa kontekstissa käyttäen useita tietolähteitä, tavoitteena tuoda teoria kosketuksiin empiirisen maailman kanssa (Juuti & Puusa 2020, 199). Tapaustutkimuksella pyritään tuottamaan tietoa valitusta, selkeästi rajatusta tapauksesta kontekstissaan eli osana tiettyä ympäristöä (Eriksson & Koistinen 2014, 7). Koska tarkoituksena on pureutua tietyn, yksittäisen ETL-/ELT-prosessin kehittämiseen Azure-ympäristössä, menetelmä sopii tähän luontevasti. Tarkoituksena on vastata tässä nimenomaisessa toimeksiannossa esille tuleviin kysymyksiin. Tapaustutkimuksessa keskeisiä kysymyksiä ovat kuinka ja miksi (Saarela-Kinnunen & Eskola 2010, 191).

Tarkoitukseni on muodostaa tutkimusraportti ns. tunnustuksellisena kertomuksena, jossa esitän näkemykseni tapauksen muodostumisesta ja sisällöstä. Perusajatus on, että empiirinen aines, teoreettiset käsitteet ja keskustelut sekä aiempi tutkimus esitetään vuoropuhelunomaisesti, ei omissa luvuissaan. Raportissa seurataan nimenomaisesti tutkittavaa tapausta; teoreettisia käsitteitä ja aiempaa empiiristä kokemusta tuodaan mukaan vain tapauksen kautta. (Eriksson & Koistinen 2014, 41.)

Lopputulosta voidaan osittain soveltaa myös muihin ETL-/ELT-prosesseihin, joten myös toimintatutkimuksen menetelmät soveltuvat aiheeseen. Toimintatutkimukselle ei ole pystytty muodostamaan yksiselitteistä määritelmää (Juuti & Puusa 2020, 256), mutta Pirkko Anttila (2014) on kiteyttänyt sen tarkoituksen osuvasti: ”Toimintatutkimuksen tarkoituksena on kehittää uusia taitoja tai uutta lähestymistapaa johonkin tiettyyn asiaan sekä ratkaista ongelmia, joilla on suora yhteys johonkin käytännölliseen toimintaan.”. Toimintatutkimuksen keskeisimpiä piirteitä ovat ongelmakeskeisyys, käytäntöön suuntautuminen sekä muutoksen yrittäminen (Kuula 1999, 219).

Toimintatutkimus etenee sykleissä, joita voidaan kuvata spiraalina (kuvio 1). Sen vaiheet ovat ongelman kartoitus ja suunnittelu, suunnitelman toteuttaminen ja vaikutusten havainnointi sekä tilanteen arviointi ja opitun tunnistaminen. Syklejä toistetaan, kunnes haluttu tulos on saavutettu. (Business Research Methodology 2023.) Ajatuksena on käyttää iteratiivista prosessia, jossa lopputulos jalostuu tiedon kertyessä. Toimintatutkimuksen syklinen luonne mahdollistaa etenemistavan, jossa voidaan tutkia, kokeilla ja arvioida useita kertoja, kunnes lopputulos tyydyttää (Kuula 1999, 218).



Kuvio 1. Toimintatutkimuksen spiraali

4.2 Teorian yhteys käytäntöön

Tavoitteena on kerätä tarvittava teoreettinen ja empiirinen tieto aiemmin mainituilla menetelmillä sovitun palvelun toteutustyötä varten. Toimin projektissa raporttikehittäjänä, vastuullani raportointiympäristön ja raportointitietokannan suunnittelu. Karkeasti kuvattuna tehtävinä ovat datan nouto API-rajapinnasta, sen lataaminen tietokantaan, tarvittavien muunnosten tekeminen sekä raporttien rakentaminen.

Tutkimus aloitetaan perehtymällä ETL-/ELT-prosessin toteutukseen Azure-ympäristössä lähdemateriaaleja käyttäen sekä harjoittelemalla. Seuraavaksi määritellään casen toteutukseen soveltuvat työkalut. Työn aikana käydään vuoropuhelua kokeneiden asiantuntijoiden sekä työn tilaajan kanssa parhaiden ratkaisujen löytämiseksi. Lopputulos rakentuu iteratiivisesti, kehittyen tiedon lisääntyessä. Toimeksiantaja on määritellyt halutun lopputuloksen tarkasti jo etukäteen, mutta on saatavilla tarkennuksia ja lisäkysymyksiä varten. Näin ollen ratkaistavaksi jää, millä toteutustavalla lopputulos parhaiten saavutetaan.

4.3 Eettiset lähtökohdat ja luotettavuus

Opinnäytetyössä käsitellään vain tietoja, jotka ovat työtä tehtäessä välttämättömiä. Ihmisten yksityisyyden suojaaminen on tärkeä eettinen lähtökohta (Vilkka 2021, 80). Tässä työssä ei ole tarkoitus tehdä varsinaisia tutkimushaastatteluja, eikä työssä käsitellä henkilöihin liittyviä tietoja. Asiakasyrityksiin liittyviä tietoja tai itse dataa ei tuoda esille.

Lähdekritiikki on tärkeä eettinen lähtökohta ja keskeinen tekijä myös opinnäytetyön luotettavuudessa, sillä lähteiden ja aineiston laatu vaikuttaa suoraan opinnäytetyön laatuun ja luotettavuuteen (Vilkka 2021, 84). Tässä työssä opinnäytetyön luotettavuuteen vaikuttavat tarkoituksenmukaisten lähteiden käyttö sekä kokemusperäinen tieto siitä, miten asioita kannattaa toteuttaa. Työhön pyritään löytämään mahdollisimman ajantasaiset lähteet, jotta vältytään vanhentuneen tiedon käyttämiseltä. Tämä on erityisen tärkeää valitussa aihepiirissä, jossa teknologia ja menetelmät kehittyvät nopeasti. Lisäksi johdonmukainen ja järjestelmällinen työskentelytapa sekä tehtyjen valintojen perustelu lisäävät opinnäytetyön luotettavuutta (Vilkka 2021, 132).

5 TOIMINTAYMPÄRISTÖ

5.1 Eriytetyt kehitys- ja tuotantoympäristöt

Kehitys- ja tuotantoympäristö ovat erillisiä: kummallakin on omat Azure-komponentit, joista keskeisiä tässä toteutuksessa ovat aiemmin mainittu Data Factory ja SQL-tietokanta. SQL-tietokanta on relaatiotietokanta, joka koostuu jäsennellyistä taulukoista ja niiden välisistä suhteista. Taulukon jokainen rivi kuvaa datakokonaisuutta ja jokainen sarake määrittelee tietyn tietokentän. SQL-tietokanta rakennetaan käyttämällä strukturoitua kyselykieltä (Structured Query Language) tietojen luomiseen, tallentamiseen, päivittämiseen ja hakemiseen. (Solarwinds 2023.)

Kehitys- ja testaustyö tehdään kehitysympäristössä, josta ratkaisut siirretään tuotantoon. Prosessi hoidetaan Azuren DevOps-ympäristön CI/CD (Continuous Integration ja Continuous Deployment) pipelinen avulla. CI/CD on DevOpsin selkäranka: automatisoitu prosessi, joka ohjaa ohjelmistokehitystä tarjoamalla työkalut koodin rakentamiseen, testaukseen ja käyttöönottoon. Prosessi auttaa minimoimaan inhimilliset virheet ja dokumentoimaan tehdyt muutokset automaattisesti sekä saamaan muutokset tuotantoon erittäin nopeasti. (RedHat 2022.)

Versionhallintatyökaluna käytetään gitiä. Git on laajasti käytetty avoimen koodin hajautettu versionhallintajärjestelmä, joka pitää kirjaa kaikista koodimuutoksista (Pubudu 2021). Esimerkiksi Data Factoryssa rakennan dataputket graafisella käyttöliittymällä, mutta taustalla toimii git, joka tallentaa koodin ja siihen tehdyt muutokset. Koodi siirretään tuotantoympäristön Data Factoryyn CI/CD pipelinen avulla.

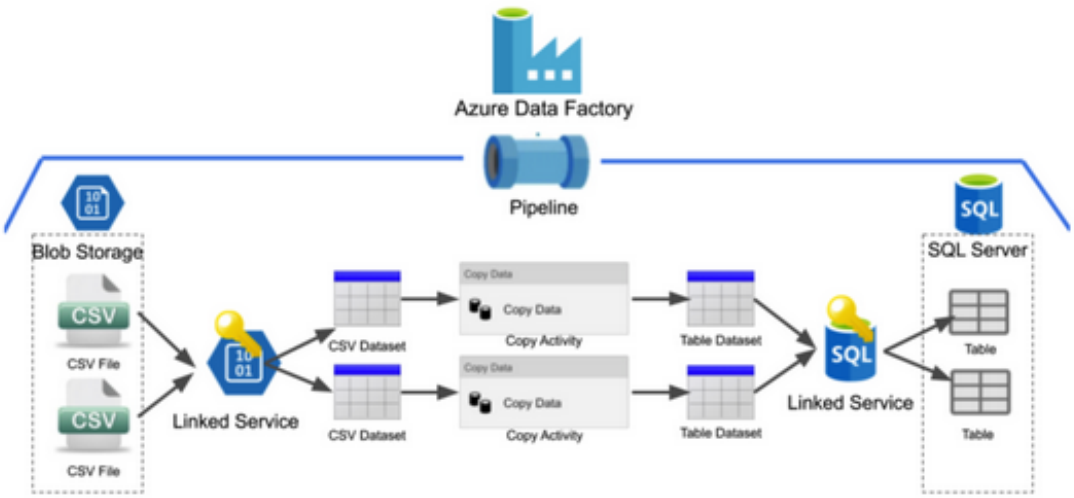
5.2 Azure Data Factory

Dataputkien rakentamisessa keskeinen työväline on datan integrointityökalu Azure Data Factory (Microsoft 2023a), jonka toiminnallisuus muodostuu useista komponenteista. Data Factoryn moottori on dataputki, jonka avulla data saadaan liikkumaan. Dataputkeen määritellään työnkulku; mitä vaiheita kokonaisuus

sisältää, missä järjestyksessä ja miten dataa siirretään ja muunnetaan. Dataputki koostuu erilaisista aktiviteeteista ja niiden välisistä yhteyksistä. (Cote ym. 2018, 11.)

Dataputken ja tietovaraston välissä on dataset, tietojoukko, joka määrittelee käytettävän tiedon rakenteen. Jotta tietovarastoa voidaan käyttää, tarvitaan lisäksi linked service. Sen avulla luodaan tarvittavat yhteydet, esimerkiksi tietokantoihin, tietovarastoihin tai API-rajapintoihin. Linked service mahdollistaa tietojen siirron paikasta toiseen. Datasettiä luodessa valitaan, mihin tietovarastoon otetaan yhteys linked servicen avulla. Yhteyden yksityiskohdat puolestaan määritellään linked servicessä. (Cote ym. 2018, 24–25.)

Dataputken prosessoinnin hoitaa Integration runtime, joka tarjoaa prosessointivoiman (Cote ym. 2018, 24–25). Kuviossa 2 on kuvattu yksinkertainen dataputki, jossa komponentit on havainnollistettu. Esimerkissä kopioidaan data csv-tiedostoista SQL-tietokantaan Copy-aktiviteetin avulla.



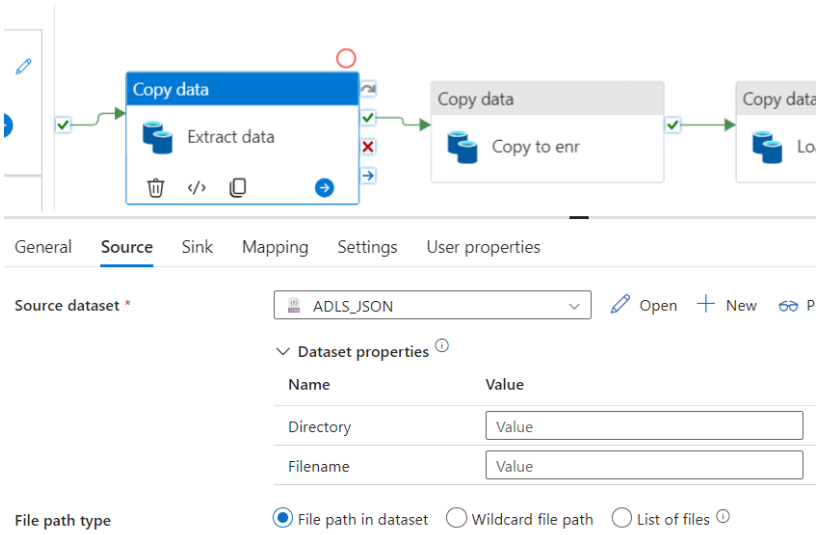
Kuvio 2. Esimerkki Azure Data Factoryn dataputkesta (D'Souza 2020)

6 TEHTÄVÄ 1: TIEDOSTOMUODON MUUNTAMINEN

Tuotoksen ensimmäinen osa-alue on tiedostomuodon muuntaminen xml-muodosta Excel-muotoon. Volyymidata saadaan lähdejärjestelmästä ulos xml-muodossa, mutta asiakas tarvitsee datan Excel-muodossa. Ratkaistava kysymys on, miten muunnos on järkevää toteuttaa. Potentiaalisten toteutusvaihtoehtojen määrä rajautuu neljään.

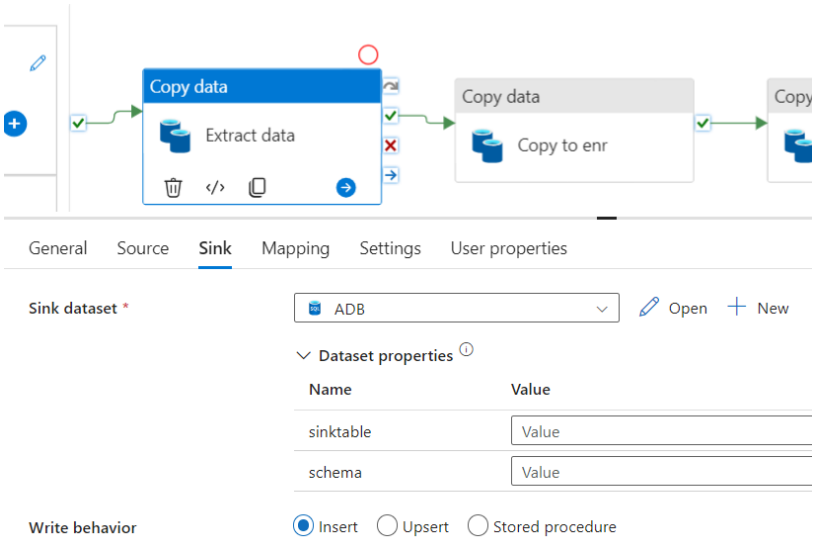
6.1 Tutkitut vaihtoehdot

Ensimmäinen tutkittu vaihtoehto on, voiko xml-tiedoston muuntaa Excel-muotoon Data Factoryn Copy-aktiviteetilla, joko suoraan tai esimerkiksi csv-muunnoksen kautta. Copy-aktiviteetilla kopioidaan dataa tietovarastosta toiseen, joten se on keskeinen aktiviteetti Data Factoryn toiminnallisuudessa. Tyypillisesti kopioinnin yhteydessä määritellään myös sekä lähde- että kohdedatan rakenne. (Microsoft 2023b.) Näin voidaan tehdä tiedostomuotomuunnoksia, tai esimerkiksi ladata dataa tiedostoista SQL-tietokannan tauluihin. Data Factory tukee varsin laajaa valikoimaa tiedostomuotoja, joten muunnokset onnistuvat usein näin yksinkertaisesti. Copy-aktiviteetissa määritellään lähde (source) ja kohde (sink). Kuviossa 3 on esimerkki Copy-aktiviteetista, jossa lähteenä on tietovarastossa oleva tiedosto.



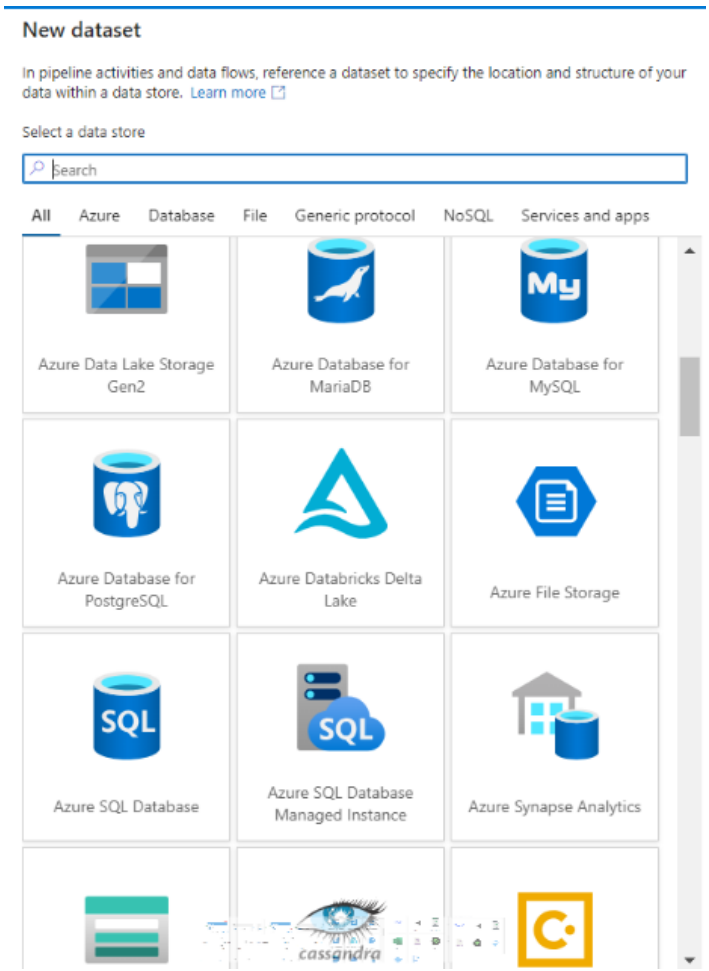
Kuvio 3. Copy-aktiviteetti, lähde

Kuviossa 4 näkyy esimerkki Copy-aktiviteetin kohteesta, joka on tässä Azure SQL-tietokannan taulu. Kuten kuviot 3 ja 4 osoittavat, tarjolla olevat määrittelyt riippuvat valitun tietojoukon (dataset) ominaisuuksista. Niiden ominaisuudet vaihtelevat tietojoukon mukaan.



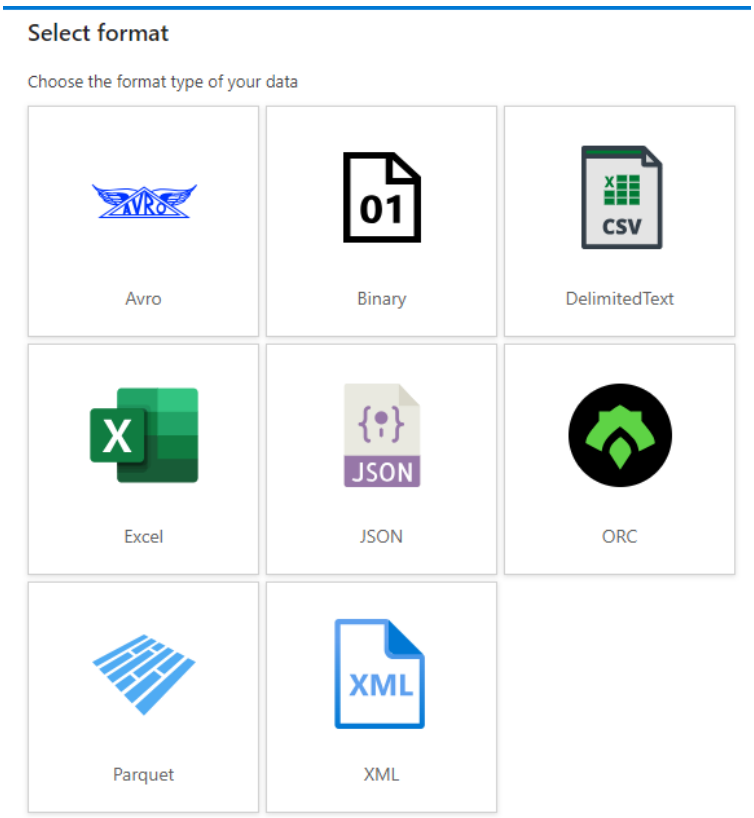
Kuvio 4. Copy-aktiviteetti, kohde

Tietojoukkoa määriteltäessä valittavana on laaja joukko erilaisia tietovarastotyyppejä (kuvio 5). Tässä opinnäytetyössä tietovarastona toimii Azuren datajärvi (data lake), joten valinnan tulee kohdistua siihen. Datajärvi on tietovarasto, johon voi tallentaa sekä rakenteellista että jäsentelemätöntä dataa. Näin datan voi tallentaa tietovarastoon raakamuodossa, kuten tiedostoina tai binääriobjekteina. (Microsoft 2023f.)



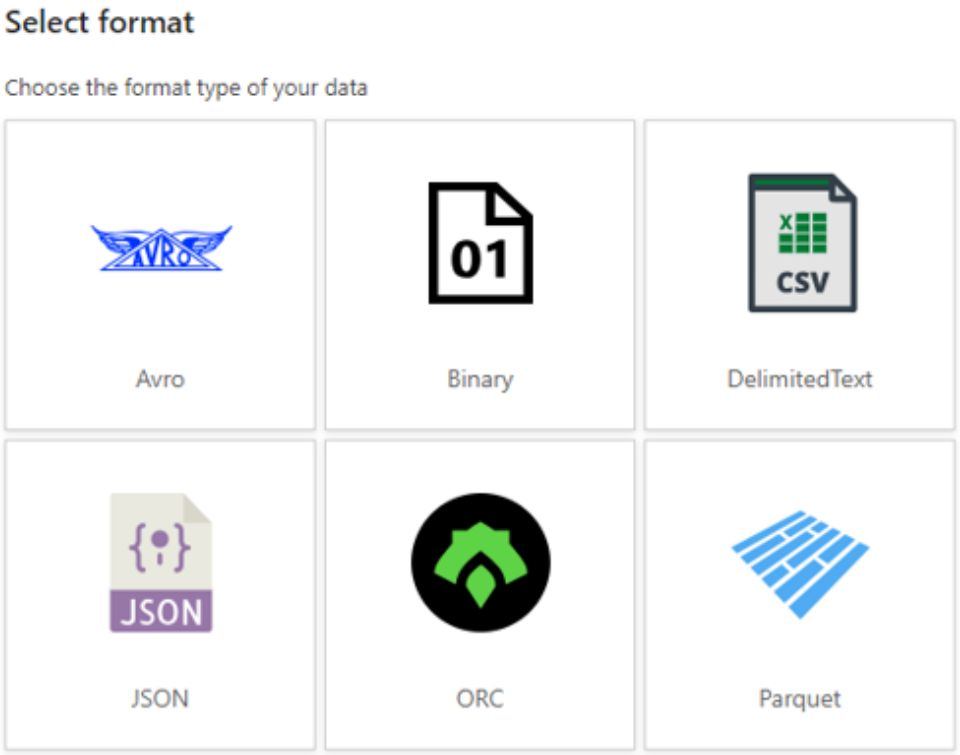
Kuvio 5. Tietojoukon määrittelyvaihtoehtoja

Kun lähdetään määrittelemään Copy-aktiviteetin lähdettä ja valitaan sen tyypiksi kuviossa 5 näkyvä Azure Data Lake Storage Gen2, saadaan näkyviin kyseiselle tietovarastolle datalähteenä mahdolliset tiedostomuodot. Kuten kuvio 6 osoittaa, valittavaksi saadaan joukko yleisesti datan käsittelyssä käytettyjä tiedostomuotoja; myös tässä tapauksessa tarvittava XML. Xml-tiedoston käyttö lähteenä siis onnistuu.



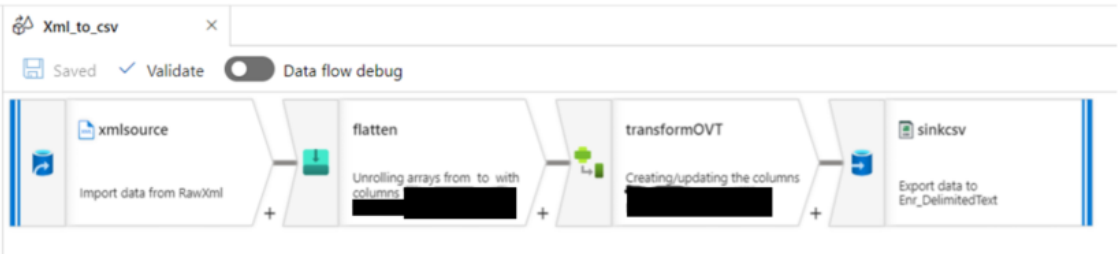
Kuvio 6. Azure Data Laken lähteelle sallimat tiedostomuodot

Kun siirrytään tutkimaan Azure Data Laken kohteelle mahdollisia tiedostomuotoja, tässä tapauksessa tarvittava Excel-muoto ei näy listalla (kuvio 7), joten muunnoksen suora määrittely Copy-aktiviteetilla ei ole mahdollinen. Tietojoukkojen määrittelymahdollisuuksia tutkimalla saadaan lisäksi selville, että Excel on lähteenä mahdollinen (kuvio 6), mutta kun tietojoukkoa tekee kohteelle, se tarjoaa vähemmän vaihtoehtoja (kuvio 7). Myös Microsoftin dokumentaatio vahvistaa tässä todetun: Data Factory tukee Excel-tiedostoja lähteenä (source), mutta ei kohteena (sink) (Microsoft 2023d).



Kuvio 7. Azure Data Laken kohteelle sallimat tiedostomuodot

Vaikka haluttu muunnos periaatteessa onnistuisi, lisähaastetta siihen tuovat lähteenä olevien xml-tiedostojen sisäkkäiset rakenteet. Monimutkaisempia tiedostomuunnoksia on usein mahdollista tehdä Data Factoryn data flow -toiminnallisuuden avulla. Data flow -toiminnon avulla voidaan suunnitella datatransformaatiot visuaalisen käyttöliittymän avulla ilman koodaamista (kuvio 8), ja valmista data flow'ta kutsutaan dataputkessa erillisen Data flow -aktiviteetin kautta. (Microsoft 2023g.) Data flow'n käyttö on tarpeen, jos datassa on esimerkiksi sisäkkäisiä rakenteita, kuten tässä tapauksessa.

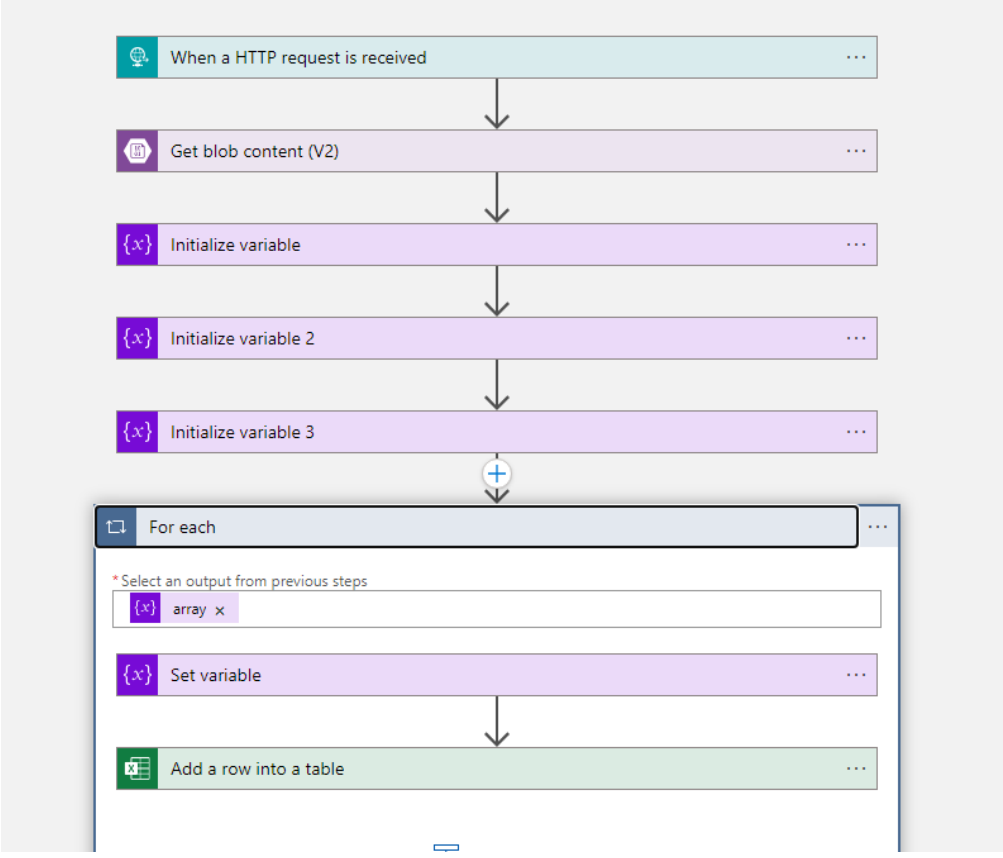


Kuvio 8. Esimerkki data flow:sta

Kuten kuvio 8 osoittaa, monimutkaisten xml-tiedostojen muuntaminen esimerkiksi csv-muotoon onnistuu data flow:n avulla. Muuntaminen Excel-muotoon ei kuitenkaan ole tuettua, lähdetiedostojen muodosta riippumatta.

Lopputulema on, että xml-Excel -tiedostomuunnosta ei voida tehdä Data Factoryn natiivien ominaisuuksien avulla.

Toinen tutkittu vaihtoehto on Azuren palvelu Logic App, joka toimii käytännössä samoin kuin Power Automate. Se on työnkulkujen automatisointiin tarkoitettu low code -ratkaisu, jossa halutut toimenpiteet suoritetaan valmiiden komponenttien avulla. (Microsoft 2023k.) Työkalu on minulle ennestään tuttu, ja sen avulla tiedostomuodon muunnos saatiin toteutettua (kuvio 9). Ratkaisu olisi kuitenkin ollut turhan monimutkainen ja virhealtis. Logic Appia voi kutsua Azure Data Factorysta, mutta yksinkertaista keinoa esimerkiksi virhetilanteiden välittämiseen dataputkelle ei ole. Myöhemmässä vaiheessa kävi lisäksi ilmi, että Excel-tiedostojen muotoiluun liittyy vaatimuksia, joiden toteuttaminen Logic Appilla ei välttämättä olisi onnistunut.



Kuvio 9. Logic App -suunnittelunäkymä tiedostomuotomuunnoksessa

Kolmas vaihtoehto on valmis, kolmannen osapuolen tekemä komponentti/lisäosa. Kyseessä on Invatin tarjoama, Azureen asennettava valmis ohjelmistopaketti, joka tarjoaa API:n csv-tiedostojen muuntamiseksi Excel-

muotoon (Invati Inc. 2021). Xml-muodossa olevien tiedostojen muuntaminen csv-muotoon onnistuu Azure Data Factoryssa helposti, joten ohjelmistopaketilla voisi jatkaa konversiota csv-muodosta Excel-muotoon. Sen tarjoamasta toiminnallisuudesta ei kuitenkaan ole tarjolla kattavaa dokumentaatiota, ja esimerkiksi tiedostojen muotoilumahdollisuuksista ei löytynyt tietoa. Kun lisäksi testiasennuksen kanssa oli ongelmia, tämä vaihtoehto hylättiin.

6.2 Toteutettu vaihtoehto: Azure Functions ja Python

Jäljelle jää neljäs vaihtoehto, joka on muunnoksen toteuttaminen koodaamalla. Asian mielekkyydessä on kaksi puolta: toisaalta omaan koodiin voi helposti jäädä virheitä, mutta se tarjoaa myös paljon joustavuutta mahdollisten muutostarpeiden tullessa. Valmiit työkalut graafisine käyttöliittymineen ovat käteviä, mutta niissä on myös paljon rajoitteita. Ohjelmointitoteutuksessa on pienempi vaara ajautua tilanteeseen, jossa tarvittavia muutoksia ei pystyttäisikään tekemään valitulla toteutustavalla. Lisäksi tässä tarvittava ohjelmointityö on varsin pieni ja yksinkertainen. Kooditoteutukseen pitää valita soveltuva Azure-työkalu ohjelmakoodin suorittamiseen sekä päättää käytettävä ohjelmointikieli.

6.2.1 Ajoympäristön ja ohjelmointikielen valinta

Oman ohjelmakoodin suorittamiseen Azure-ympäristössä on monia vaihtoehtoja, joista tämäntyyppiseen tarkoitukseen soveltuvat esimerkiksi Batch- ja Functions-työkalut. Azure Batch soveltuu parhaiten suuren mittakaavan eräajoihin. Kun käsiteltävänä on paljon datamassaa ja tarvitaan nopeaa suorituskykyä, Batch on sopiva valinta. Batch-työ käynnistää joukon virtuaalikoneita, joissa prosessointi suoritetaan. (Microsoft 2023j.)

Azure Functions on palvelimeton (serverless) ratkaisu, jonka avulla ohjelmakoodia voidaan ajaa ilman infrastruktuurin luomista ja hallintaa. Koodipätkistä muodostetaan funktioita, jotka käynnistyvät niille määriteltyjen laukaisimien (trigger) mukaan. Functions soveltuu pieniin, lyhytkestoisiin tehtäviin. Tyypillisesti yksi funktio suorittaa vain yhden tehtävän, ja funktioita voidaan tarpeen mukaan ketjuttaa. (Serverless360 2023.) Koska tässä tapauksessa kyseessä on pieni ja kevyt tehtävä, moottoriksi valittiin Functions.

Koska toteutus tehdään Azure Functionsin avulla, pitää valita ohjelmointikieli, jota siellä voi käyttää. Tuettuja kieliä ovat C#, Java, Javascript, TypeScript, PowerShell ja Python (Microsoft 2023i). Ohjelmointikieleksi valikoitui luontevasti Python, koska se on itselleni ennestään jonkin verran tuttu, ja soveltuu tähän käyttötarkoitukseen hyvin. Python-kielen syntaksi on kompakti ja yksinkertainen. Lisäksi Python on dataan liittyvässä käsittelyssä yleisessä käytössä, koska se tarjoaa paljon datan käsittelyyn tarkoitettuja kirjastoja, joilla on nopeaa ja tehokasta tehdä tarvittavia muunnoksia. Näitä ovat esimerkiksi NumPy, jonka avulla dataa voi käsitellä moniulotteisina taulukkoina, sekä Pandas, jonka avulla voi tehdä muun muassa tiedostomuotomuunnoksia. (Ogunleke 2022.)

6.2.2 Toteutus

Python-funktiot on tässä tehty Visual Studio Codessa, koska sieltä ne on helppo julkaista Azureen. Python-kirjastojen tutkimisen ja kokeilemisen jälkeen tiedostomuunnos päädyttiin tekemään kahdessa vaiheessa: ensimmäinen funktio muuntaa xml-tiedostot csv-muotoon ja toinen csv-tiedostot Excel-muotoon. Koodi on näin yksinkertaisempi.

Ensin toteutettiin xml-csv-muunnos. Koodi käyttää Azuren tietovarastoja, joten funktioille pitää antaa niihin luku- ja kirjoitusoikeudet. Lisäksi koodin pitää noutaa tietovarastojen yhteysavaimet. Funktion http-kutsu välittää parametreja, joiden arvot koodin pitää lukea. Koodi käy kaikki tietovarastossa olevat xml-tiedostot läpi ja suorittaa sisällön muunnoksen csv-muotoon.

Koska xml-tiedostosta halutaan talteen tietyt kentät, koodi etsii ne xml-tiedoston sisäkkäisistä rakenteista. Tämä tarkoittaa sitä, että xml-tiedoston rakenteen mahdollisesti muuttuessa myös koodia pitää muuttaa. Myös Excel-tiedostojen nimeämisen tulee vastata vaatimuksia ja se on riippuvainen lähdetiedostojen (xml) nimistä ja sisällöstä. Tämä tarkoittaa sitä, että koodissa pitää muodostaa kirjoitettavalle csv-tiedostolle myös nimeämiskäytännön mukainen nimi.

Edellä kuvatut toimenpiteet on varsin helppoa toteuttaa käyttäen hyväksi Pythonin kirjastoja (kuvio 10). Azure.functions tarvitaan, koska koodi ajetaan Functions:issa, ja azure.storage.blob tarvitaan, koska koodi lukee ja kirjoittaa Azuren tietovarastoihin. Xml.etree.ElementTree mahdollistaa xml-tiedoston

lukemisen halutulla tavalla, ja pandasin avulla saadaan muodostettua lopullinen dataframe ja kirjoitettua se csv-tiedostoon. Pandas dataframe on kaksiulotteinen tietorakenne, jossa on sarakkeita: käytännössä se on taulukko (Pandas 2023). Tämä on erittäin tavallinen ja käyttökelpoinen rakenne, kun käsitellään rakenteellista dataa.

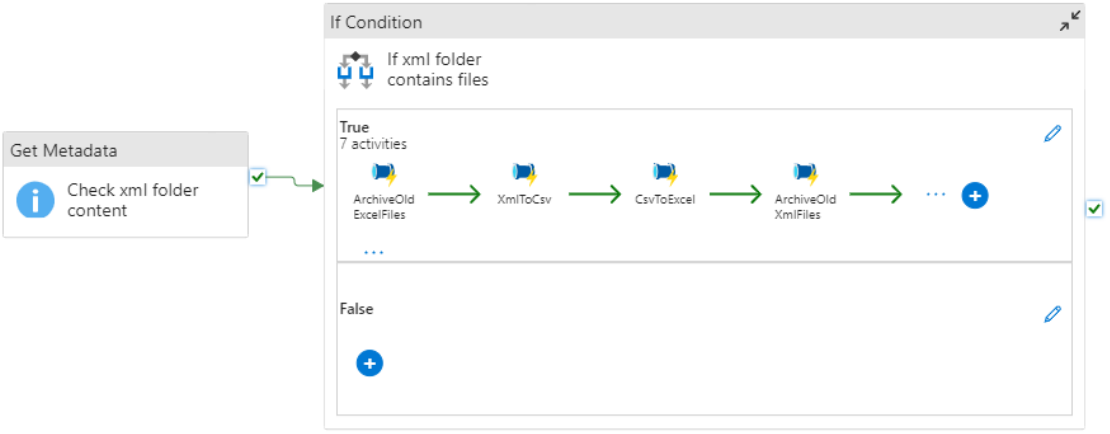
XmlToCsvAllfiles > __init__.py > main

```
1   import logging
2   import io
3   import os
4   import azure.functions as func
5   import pandas as pd
6   import xml.etree.ElementTree as et
7   from azure.storage.blob import BlobClient, BlobServiceClient
8
```

Kuvio 10. Xml-csv -muunnoksen Python-kirjastot

Seuraavaksi toteutettiin csv-Excel-muunnos. Tähän muunnokseen pätevät samat yleiset tarpeet kuin edelliseenkin; funktioiden pitää saada yhteys tietovarastoihin, joten tarvitaan azure.functions- ja azure.storage.blob-kirjastoja. Dataframe luetaan ja muodostetaan lähteenä olevasta csv-tiedostosta pandas-kirjaston avulla, ja siitä muodostetaan Excel käyttäen openpyxl-kirjaston ominaisuuksia. Openpyxl on erittäin tehokas työkalu Excel-tiedostojen kanssa, sillä tiedostojen lukemisen ja kirjoittamisen lisäksi sen avulla voidaan esimerkiksi muokata taulukoita ja muotoilla sisältöä (JavaTpoint 2021).

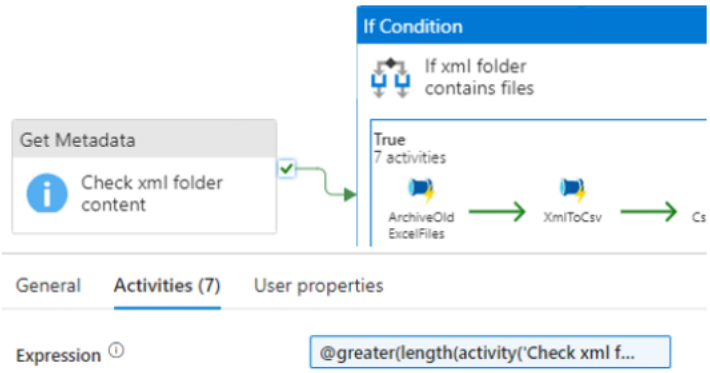
Tiedostomuunnoksen ajo on ajastettu Azure Data Factoryn dataputkeen (kuvio 11), joka kutsuu funktioita. Koska muunnettavia tiedostoja ei tule joka päivä, dataputken suoritukseen liittyy ehtolause: jos tietovaraston määriteltyyn lähdehakemistoon on ilmestynyt tiedostoja, muunnos suoritetaan ja lopputulos viedään määriteltyyn kohdehakemistoon.



Kuvio 11. Data Factoryn dataputki tiedostomuunnokselle

Kuvion 11 dataputken If Condition -aktiviteetilla voidaan hyvin demonstroida, miten low code käytännössä toimii. Edellinen aktiviteetti Get Metadata tarkistaa, onko hakemistossa tiedostoja. If Condition -aktiviteetille on määritelty lauseke, joka ottaa edellisen aktiviteetin tuloksista tietyn arvon ja tarkistaa, onko sen pituus suurempi kuin nolla (kuvio 12). Dataputken suoritus jatkuu, jos palautuva arvo on ”True”.

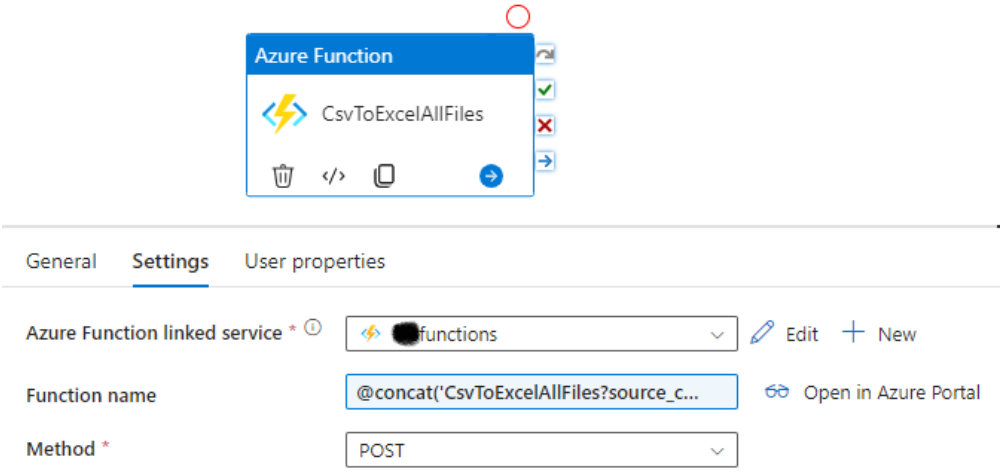
```
@greater(length(activity('Check xml folder content').output.childItems),0)
```



Kuvio 12. If Condition -aktiviteetti

Funktioiden käynnistämiseen voidaan käyttää monentyyppisiä laukaisimia. Käynnistys voi tapahtuma esimerkiksi sanomajonojen, ajastuksen tai tapahtumapohjaisten Azure Event Gridin tai Azure Event Hubin avulla. (Serverless360 2023.) Koska tässä tapauksessa Functionsia kutsutaan Data Factoryn dataputkesta ja kyseessä on pienimuotoinen toteutus, HttpTrigger oli luonteva vaihtoehto. HttpTrigger käynnistää koodin suorituksen http-pyynnön avulla (Serverless360 2023), jolloin väliin ei tarvita mitään lisäkomponentteja. Dataputken Azure Function -aktiviteetti käynnistää funktion suorituksen (kuvio

13). Koska säiliöiden ja hakemistojen nimiä ja sijainteja ei ole haluttu kovakoodata, ne välitetään http-kutsun parametreina funktion nimen yhteydessä.



Kuvio 13. Azure Function -aktiviteetti Data Factory dataputkessa

6.2.3 Tiedostomuunnoksen jatkokehitysmahdollisuuksia

Vielä tässä vaiheessa xml-tiedostot ladataan datajärveen manuaalisesti Excel-konversiota varten ja myös valmiit Excel-tiedostot noudetaan datajärvestä manuaalisesti. Näkisin järkevänä toiminnan automatisoinnin siten, että xml-tiedostoja varten rakennettaisiin oma API-päätepiste, jota Azure Data Factory kutsuisi sovitun ajastuksen mukaisesti. Myös valmiit Excel-tiedostot olisi hyvä saada lopulliseen sijaintiinsa automaattisesti.

Kun xml-excel -muunnoksen luonne on ajan myötä käynyt selvemmäksi, näkisin järkevänä myös muuttaa Azure Functionsin käynnistystavan. Sen sijaan, että funktiot käynnistetään Data Factoryn dataputkessa ajastetusti, Functions voisi käynnistyä suoraan siitä, kun määriteltyyn sijaintiin saapuu uusia xml-tiedostoja. Functions tarjoaa tällaista laukaisinta, mutta toteutus vaatisi ohjelmakoodiin muutoksia.

7 TEHTÄVÄ 2: DATAPUTKI KUUKAUSIRAPORTOINTIIN

7.1 Yleiskuvaus

Azure Data Factoryn dataputki noutaa raportointidatan lähdejärjestelmän tarjoamasta API-rajapinnasta ja siirtää sen Azure-datajärveen. API-rajapinta tarjoaa datan json-muodossa. JSON, JavaScript Object Notation, on tekstipohjainen, kevyt ja ihmisen luettavissa oleva tietomuoto. Ohjelmointikielet tukevat yleisesti json-tietomuotoa ja se on datan siirrossa paljon käytetty muoto. Json-data koostuu avain-arvo -pareista (kuvio 14). (Sriparasa & D'mello 2018, 6. 11.) Yleensä datan muoto ei kuitenkaan ole näin yksinkertainen kuin kuviossa 14 on esitetty, vaan se tyypillisesti sisältää esimerkiksi hierarkkisia tietorakenteita. Json tarjoaa mahdollisuuden käsitellä myös sisäkkäisiä objekteja, jonka ansiosta monimutkaisenkin datan käsittely onnistuu. (Bito 2023.)

```
/* This is an example of student feed in JSON */
{
  "students" :{
    "0" :{
      "studentid" : 101,
      "firstname" : "John",
      "lastname" : "Doe",
      "classes" : [
        "Business Research",
        "Economics",
        "Finance"
      ]
    },
    "1" :{
      "studentid" : 102,
      "firstname" : "Jane",
      "lastname" : "Dane",
      "classes" : [
        "Marketing",
        "Economics",
        "Finance"
      ]
    }
  }
}
```

Kuvio 14. Esimerkki json-tiedostosta (Sriparasa & D'mello 2018, 8)

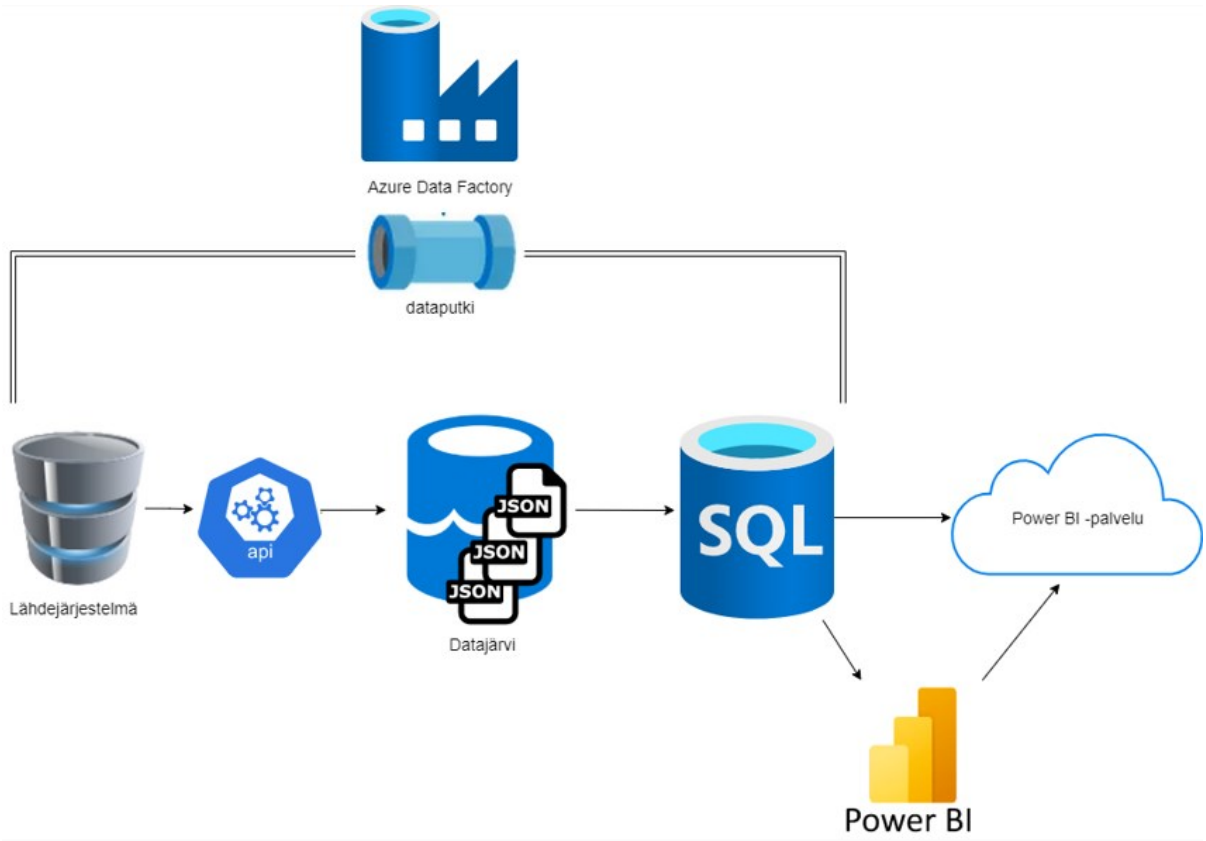
Kun json-tiedostot on noudettu datajärveen, dataputkessa tehdään niille tarvittavat muunnokset, joka voi olla esimerkiksi tiedostojen yhdistämistä. Valmiit muunnokset tallennetaan datajärveen. Tämän jälkeen dataputki siirtää datan

json-tiedostoista Azure SQL-tietokannan tauluihin ja kutsuu tietokannan SQL-proseduureja, jotka suorittavat kannassa tarvittavat ajot ja toimenpiteet.

Käytännössä dataputki koostuu Azure Data Factoryssa kokonaisesta joukosta dataputkirakenteita, joissa ylemmän tason dataputki suorittaa alemman tason dataputken, joka taas suorittaa alemman tason dataputken ja niin edelleen. Rakenteisiin liittyy myös esimerkiksi ehtolauseita ja looppeja. Lisäksi kokonaisuus mahdollistaa dataputken ajamisen uudelleen esimerkiksi virhetilanteen jälkeen siten, ettei datan eheys vaarannu: käytännössä datajärven hakemistoja ja tietokannan tauluja tyhjennetään ennen uuden datan viemistä.

Power BI -palvelussa käytettävät tietojoukot ja raportit määritellään Power BI Desktop -sovelluksella. Desktopilla otetaan yhteys tietokantaan ja noudetaan sieltä tarvittavat tiedot, joista muodostetaan datamalli. Datamalli Power BI:ssä on looginen esitys datan rakenteesta, ominaisuuksista ja suhteista. Tyypillisesti se koostuu yhdestä tai useammasta tietolähteestä, jotka voivat olla mitä tahansa Excel-laskentataulukoista pilvipohjaisiin tietokantoihin. Näiden lähteiden tiedot esitetään taulukkoina, joiden väliset suhteet ovat datamallin kulmakivi. Suhteiden avulla eri tietolähteiden ja taulukkojen tietoja voidaan yhdistää joustavasti. (Hadges 2023.)

Datamallin pohjalta rakennetaan tarvittavat visualisoinnit. Kokonaisuus julkaistaan Power BI -palveluun, josta loppukäyttäjät saavat raportit käyttöönsä sinne määriteltyjen oikeuksien mukaisesti. Viimeisessä vaiheessa Power BI -palveluun ajastetut päivitysajot päivittävät Power BI -tietojoukot uudella, tietokannasta noudetulla datalla, jolloin loppukäyttäjien raportit päivittyvät. Kokonaisuus on kuvattu yksinkertaistettuna kuviossa 15.



Kuvio 15. Raportointikokonaisuuden yleiskuvaus

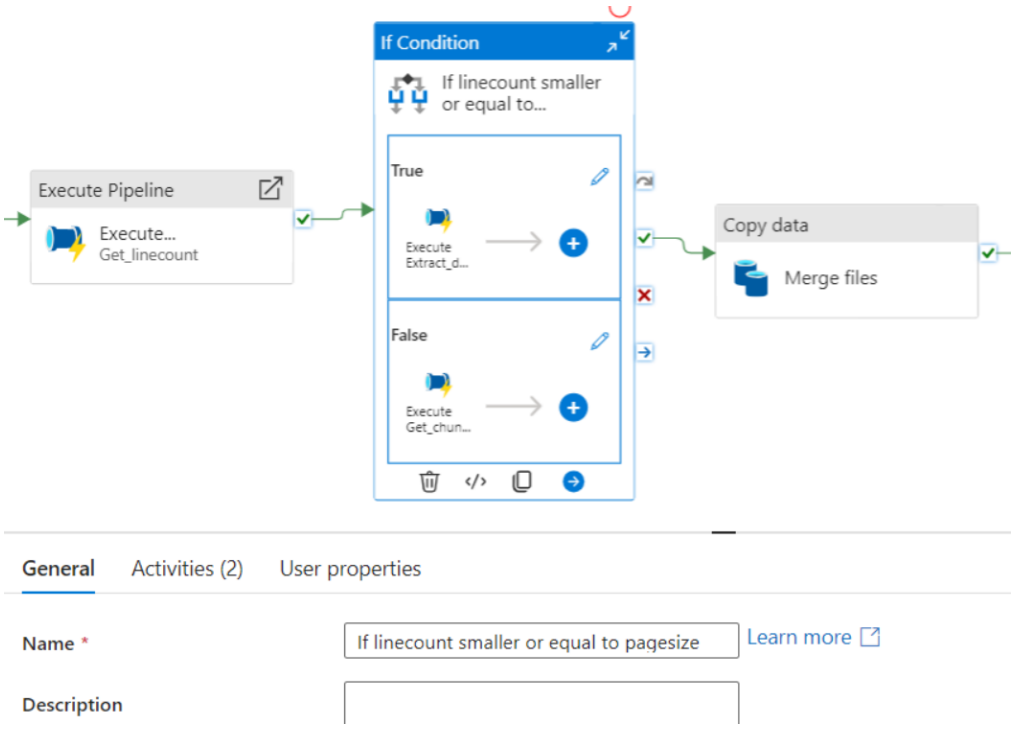
7.2 Raportointidatan tuominen Azureen

7.2.1 Noutaminen API-rajapinnasta

Data siirretään lähdejärjestelmästä API-rajapinnan kautta json-muodossa Azuren datajärveen. Kokonaisuuteen liittyy lukuisia API-päätepisteitä (endpoint), joista osaan annetaan parametreja noutoa varten. API-päätepiste on tietty sijainti API:ssa, joka hyväksyy pyynnöt ja lähettää takaisin vastauksia. Se on järjestelmien ja sovellusten tapa kommunikoida keskenään lähettämällä ja vastaanottamalla tietoa ja ohjeita päätepisteen kautta. (Bryan & Yusuf 2023.)

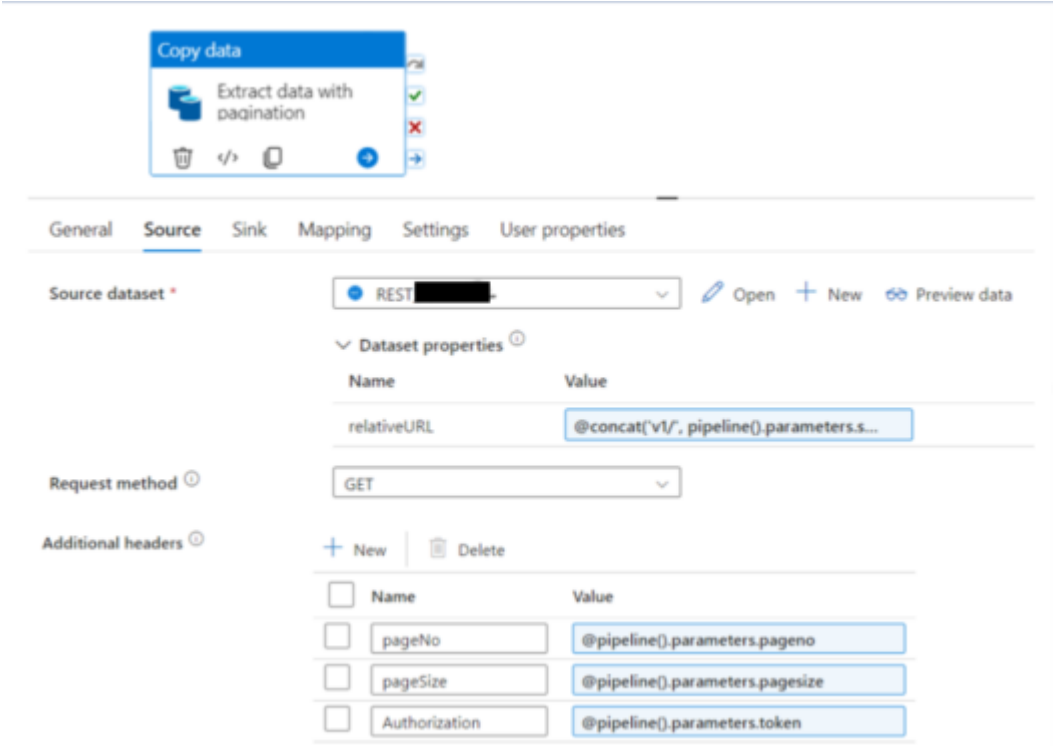
API-päätepisteet, niiden kenttärakenne ja mahdolliset parametrit löytyvät Swaggerista. Swagger on työkalu, jonka avulla API:n rakenne voidaan kuvata automaattisesti (SmartBear Software 2023). Swaggerista näen, mistä osoitteesta tietty päätepiste löytyy, millainen json-rakenne päätepisteellä on ja vaatiiko päätepiste parametreja. Parametreilla voidaan rajata hakua, esimerkiksi noutaa vain tietyn aikavälin data.

Tässä tapauksessa osa API-päätepisteistä noutaa suuren määrän dataa, joten niille on määritelty parametrit pageno ja pagesize sivutusta varten. API-sivutus (API pagination) auttaa noutamaan suuria tietojoukkoja jäsennellyllä ja hallittavalla tavalla, pienempinä osina tai sivuina. Tämä muun muassa parantaa suorituskykyä, auttaa hallitsemaan resurssien käyttöä ja helpottaa virheenkäsittelyä. (Verma 2023.) Dataputkeen pitää rakentaa sivutusta varten oma logiikkansa. Ensin haetaan erillisestä API-päätepisteestä kohteena olevan, sivutusta vaativan päätepisteen kokonaisrivimäärä. Tämä tarvitaan sivutustarpeen määrittelemiseksi ja tiedosta on hyötyä myös virhetilanteissa. Kuvio 16 havainnollistaa, miten dataputken suoritus ohjataan rivimäärän perusteella oikeaan haaraan; jos rivimäärä on suurempi kuin määritelty pagesize, sivun koko, mennään suorittamaan sivutusta.



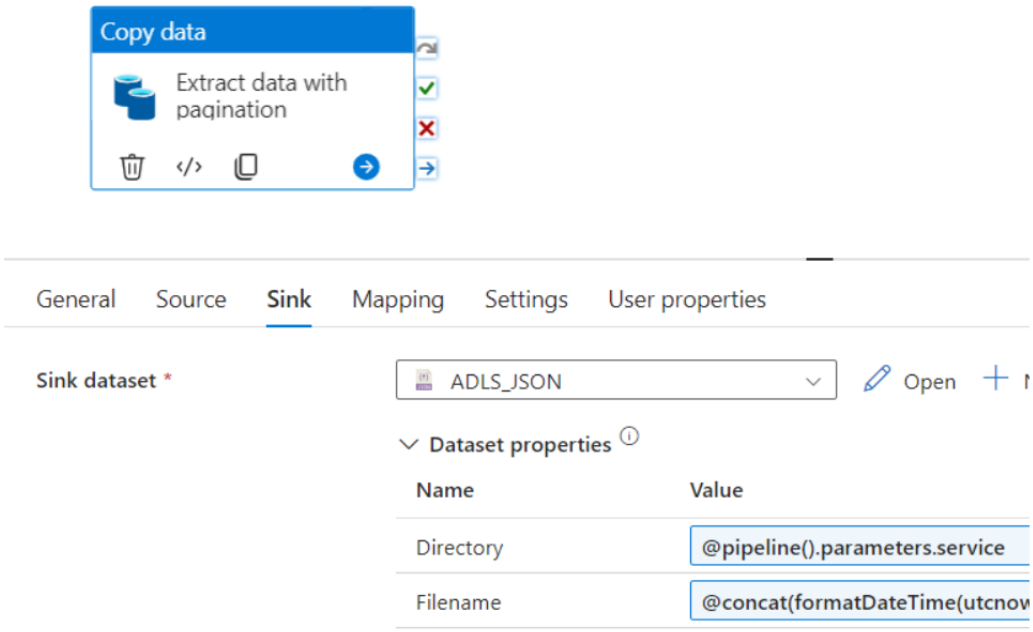
Kuvio 16. Sivutustarpeen määrittely dataputkessa

Itse API-kutsun määrittely tehdään Copy-aktiviteetilla, joka on dataputkessa varsin yksinkertainen (kuvio 17). API:lle on Data Factoryn linked servicessä ja tietojoukossa oma määrittelynsä. Copy-aktiviteetin lähdetietoihin määritellään API-päätepisteen osoite, HTTP-metodi sekä mahdolliset parametrit.



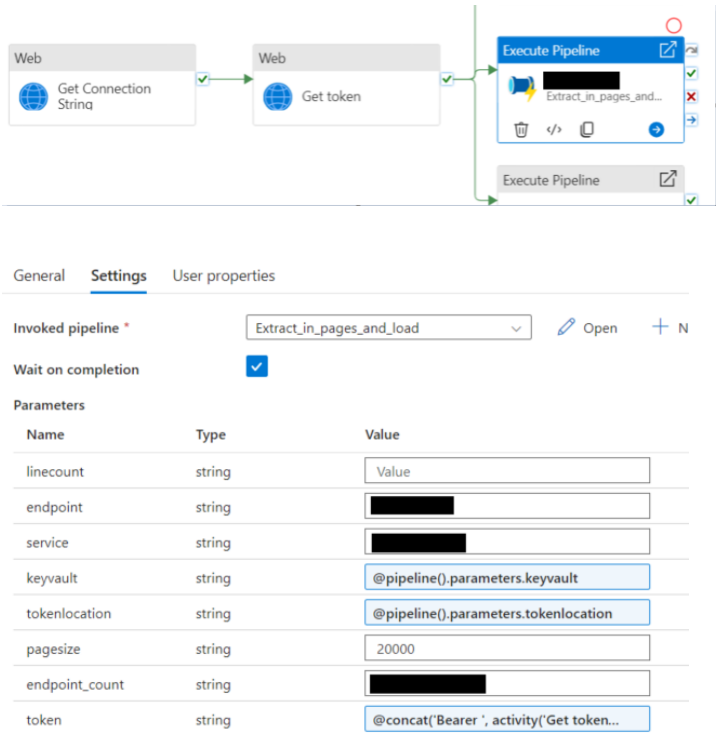
Kuvio 17. Datan nouto parametreilla API-päätepisteestä

Noudettu data viedään datajärveen json-tiedostoina. Json-tyyppiä oleva tietojoukko on linkitetty datajärveen, ja sille määritellään Copy-aktiviteetin kohdetiedoissa kohdehakemisto ja tiedoston nimi (kuvio 18). Kohdehakemisto ja tiedoston nimi määritellään dataputken parametreilla. Azure Data Factoryssä käytetään parametreja ulkoisten arvojen välittämiseen dataputkelle, tietojoukolle, linked servicelle tai data flow'lle. Näin samaa dataputkea voidaan käyttää toistuvasti välittämällä sille joka kerta eri arvoja. (Microsoft 2023e.)



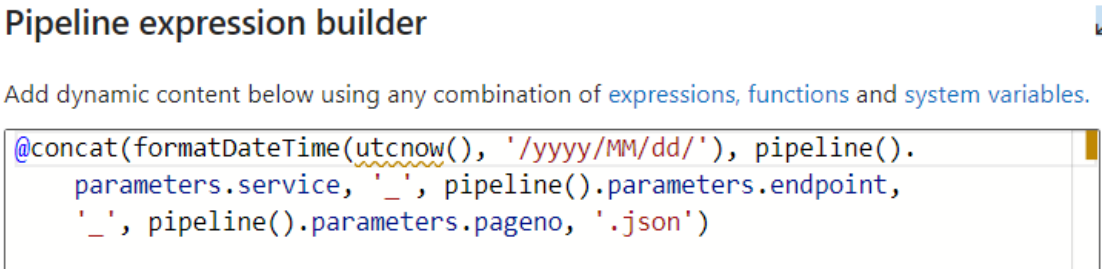
Kuvio 18. Datan vieminen datajärveen json-muodossa

Dataputkessa voidaan kutsua Execute Pipeline -aktiviteetilla alemman tason dataputkea, jolle määritellään halutut parametrit suoritusta varten. Kuvio 19 osoittaa, miten dataputkelle voidaan määrittää parametreja. Kun dataputken kutsussa olevat parametrit vaihdetaan tarpeen mukaan, sama dataputki voi suorittaa tehtäviä eri kohteille.



Kuvio 19. Dataputken parametreja

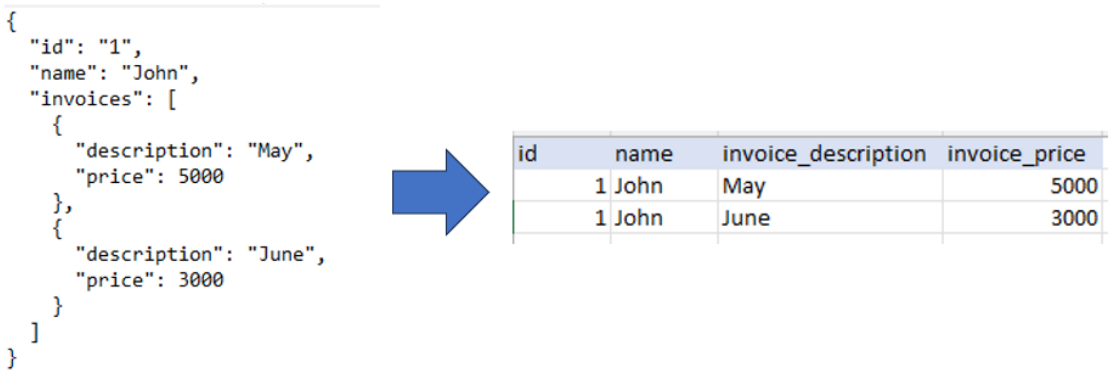
Esimerkiksi kohdetiedoston nimi voidaan määritellä käyttäen hyväksi kuviossa 19 näkyviä parametreja (kuvio 20). Näin voidaan käyttää samaa alemman tason dataputkea eri API-päätepisteiden kanssa, mikä yksinkertaistaa dataputken rakennetta ja myös muutosten tekemistä.



Kuvio 20. Esimerkki tiedostonimen määrittelystä parametrien avulla

7.2.2 Data flow

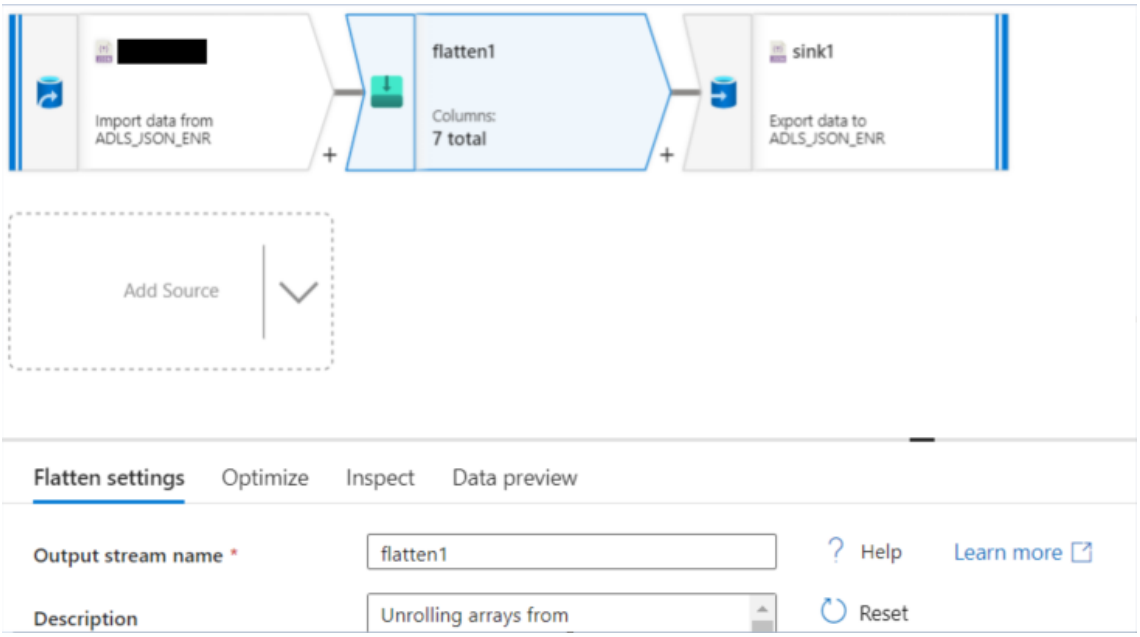
Kun json-tiedostossa on sisäkkäisiä rakenteita, rakenteet halutaan yleensä purkaa tietokannassa omiin kenttiinsä. Tilannetta on havainnollistettu kuviossa 21: json-rakenteesta halutaan purkaa ”invoices”-rakenne tietokannan tauluun siten, että jokainen arvo saadaan omaan kenttäänsä. Toimenpiteestä käytetään englanniksi asiaa hyvin kuvaavaa termiä flatten, joka voidaan suomentaa rakenteen tasoittamiseksi tai litistämiseksi.



Kuvio 21. Json-rakenteiden purkaminen tietokannan tauluun

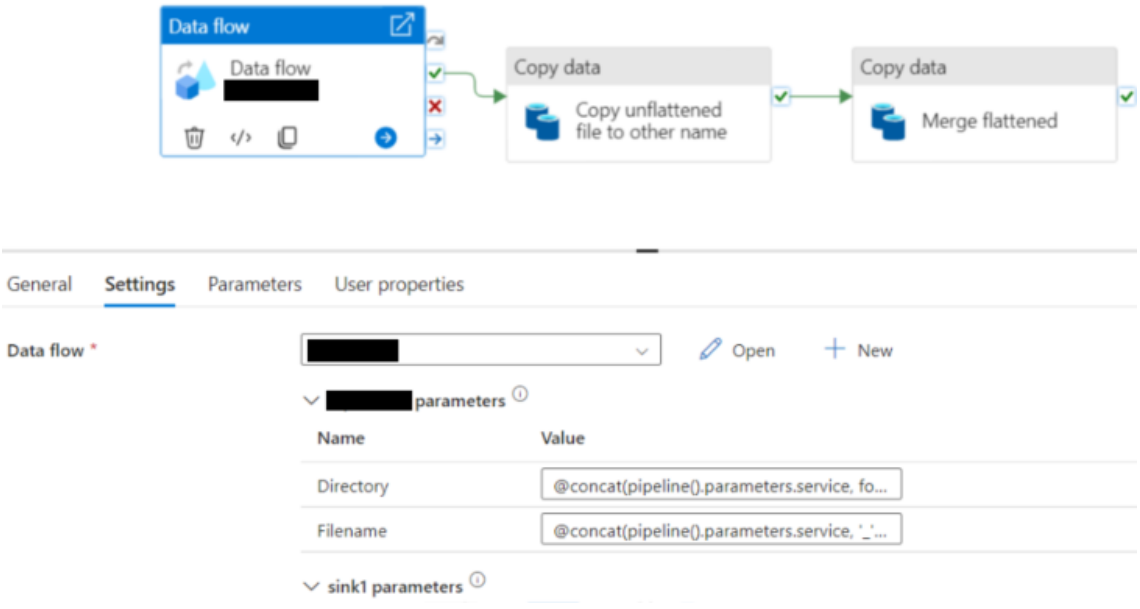
Tällaista muunnosta Copy-aktiviteetti ei kykene tekemään, vaan monimutkaisen json:in (complex json) muuntaminen relaatiokannan tauluun sopivaan muotoon vaatii Data Factoryssa data flow -toiminnallisuuden käyttöä. Tässä työssä tarvittava rakenteen muutos onnistuu hyvin yksinkertaisella data flow’lla (kuvio 22). Data flow tunnistaa tiedoston rakenteen ja osaa ehdottaa sen tasoittamista

automaattisesti, mutta niin haluttaessa on mahdollista myös valita, miten purkaminen suoritetaan.



Kuvio 22. Data flow

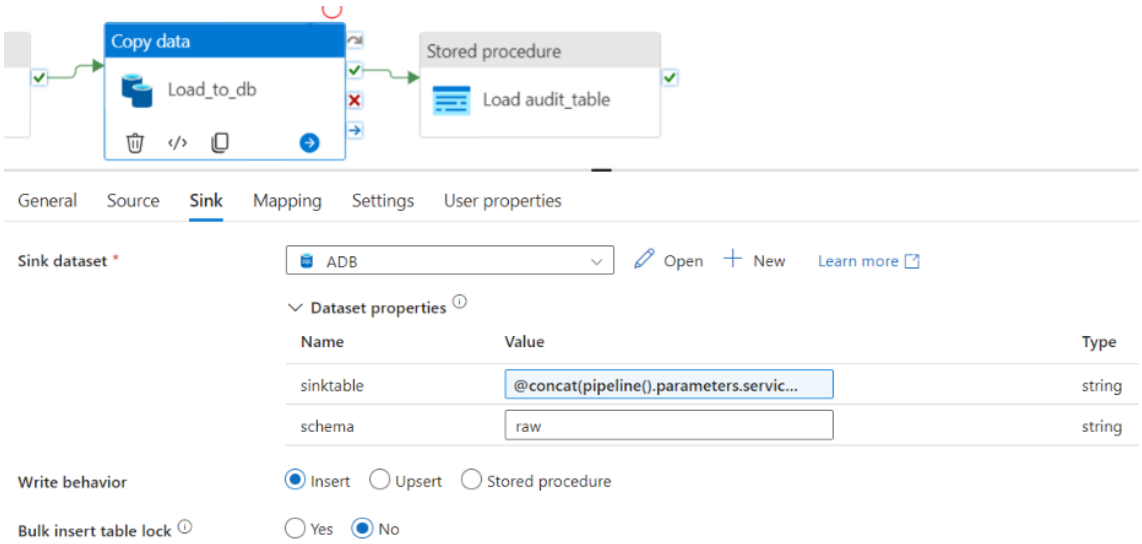
Data flow'n suoritus dataputkessa hoidetaan Data flow -aktiviteetilla (kuvio 23). Samoin kuin kutsuttaessa alemman tason dataputkea, myös data flow:lle määritellään kutsussa halutut parametrit. Kuviossa 23 näkyy, miten hakemisto ja tiedosto määritellään parametrien avulla.



Kuvio 23. Data flow -aktiviteetti

7.3 Tietokannan lataaminen Azure Data Factorysta

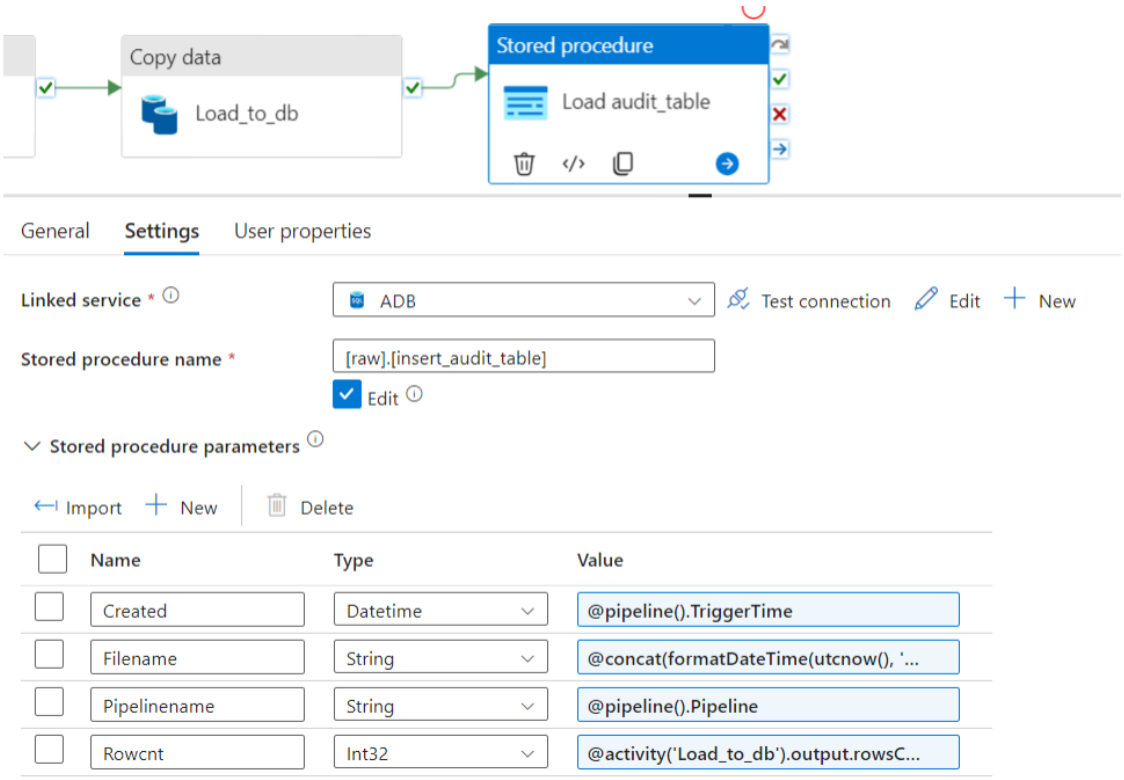
Kun data on noudettu ja tarvittavat muunnokset tehty, se ladataan Data Factoryn dataputkessa SQL-tietokannan tauluihin. Tämäkin tehdään Copy-aktiviteetin avulla. Lähteeksi valitaan datajärvestä json-tiedosto ja kohteeksi Azure SQL-tietokanta (kuvio 24). Tietojoukolle on määritelty parametrit, joiden avulla määritellään, mihin tauluun data viedään.



Kuvio 24. Lataus tietokantaan

Aktiviteetissa olisi mahdollista määritellä Mapping-välilehdellä esimerkiksi mitkä kentät lähdetiedostosta tuodaan ja tehdä tietotyyppimuunnoksia. Tätä ei ole haluttu tässä ratkaisussa käyttää, sillä lähdetiedostojen rakenteet ovat luonnollisesti erilaisia, ja mappingin käyttö vaatisi jokaiselle tiedostolle oman Copy-aktivititeetin rakentamisen. Tässä ratkaisussa tietokantaan on etukäteen luotu taulut, jotka vastaavat jokaisen lähdetiedoston rakennetta. Näin mappingia ei tarvitse käyttää, ja voidaan hyödyntää samaa dataputkea kaikkien tiedostojen lataamiseen. Näin dataputken rakenne pysyy yksinkertaisempana ja mahdollisia muutoksia on helpompi tehdä.

Jotta saadaan talteen historiatieto siitä, mitä ja milloin tietokantaan on ladattu, dataputkeen on rakennettu lisäksi audit-taulun lataaminen (kuvio 25). Lataaminen tehdään Stored procedure -aktiviteetilla, ja myös sille voidaan välittää parametreja.



Kuvio 25. Audit-taulun lataaminen Stored procedure -aktiviteetilla

Stored procedure -aktiviteetti kutsuu tietokannassa olevaa proseduuria (kuvio 26). Kyseessä on hyvin yksinkertainen toimenpide, jossa tietokannan tauluun viedään latauksen ajankohta, ladatun tiedoston ja dataputken nimi sekä ladattujen rivien määrä. Nämä ovat hyödyllisiä tietoja mahdollisia virhetilanteita selvitettäessä. Kun kaikki data on ladattu tietokantaan, Data Factoryn dataputkessa ajetaan Stored procedure -aktiviteettien avulla vielä proseduurit, jotka lataavat datan raakatauluista raportoinnissa käytettyihin taulurakenteisiin. Lopuksi ajetaan proseduurit, jotka sisältävät raporttinäkymien muodostamiseen tarvittavat kyselyt.

```
CREATE PROCEDURE raw.insert_audit_table
@Pipelinename varchar(100),
@Filename varchar(100),
@Rowcnt int,
@Created DATETIME
AS
BEGIN
    INSERT INTO raw.audit_table (pipeline,filename,rowcnt,created)
    VALUES (@Pipelinename,@Filename,@Rowcnt,@Created)
END
```

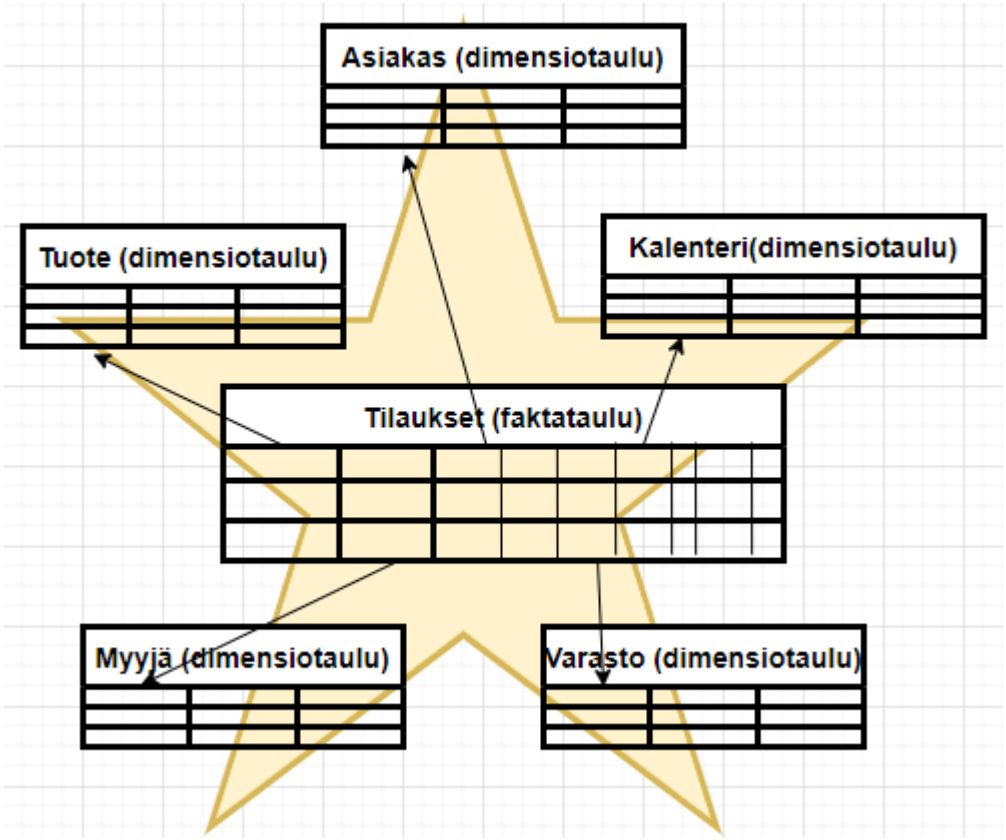
Kuvio 26. Audit-taulun lataava proseduuri tietokannassa

7.4 Tietokannan rakenne

SQL-tietokannan tyypillisiä objekteja ovat esimerkiksi taulut, näkymät, proseduurit, funktiot ja indeksit. Objekteja voi erotella loogisiksi kokonaisuuksi skeemojen avulla. Skeemat ovat objektijoukkoja, joiden avulla tietokannan objekteja voi erotella halutuilla perusteilla, esimerkiksi eri käyttötarkoitusten tai pääsyoikeustarpeiden perusteella. (Gupta 2019.)

Tietokanta sisältää muutaman erillisen palvelukokonaisuuden, joista jokainen on toteutettu tähtimallina. Tähtimalli on varsin tyypillinen tapa järjestää liiketoimintatietoa tai tietovarastoja. Tähtimallissa tietokannan taulurakenne koostuu yhdestä faktataulusta, joka sisältää tapahtuma- tai mittaustietoa, sekä yleensä useista dimensiotauluista, joihin on talletettu tietojen attribuutteja. Faktataulu on kyselyjen ja analyysin varsinainen kohde, ja dimensiotaulut tarjoavat täydentävää tietoa, joka ei yleensä muutu yhtä nopeasti. Faktataulusta saattaa löytyä esimerkiksi vain tilatun tuotteen tuotekoodi, ja tuotteen nimi löytyy dimensiotaulusta tuotekoodia avaimena käyttäen. (TechTarget 2023.)

Tähtimallin nimi tulee ajatuksesta, jossa faktataulun ajatellaan olevan tähden keskiössä, ja dimensiotaulut ovat sen ympärillä muodostaen tähden sakarat (kuvio 27) (TechTarget 2023). Tässä tapauksessa tähtimallia ei ole tarvinnut erikseen suunnitella tietokantaan, vaan se muodostuu suoraan API-rajapintojen datakokonaisuuksista, jotka on tuotu tietokannan tauluiksi. Dataa ladataan tietokantaan kerran kuussa, yhden kuukauden tapahtumat kerrallaan. Näiden tapahtumat sisältävien faktataulujen lisäksi myös dimensiotaulut ladataan aina uudelleen, sillä niissäkin saattaa olla muutoksia.



Kuvio 27. Tähtimalli

Raportointitietokantaan on tehty kolme skeemaa: raw, pres ja archive. Raw-skeema sisältää datajärvistä tuoduista json-tiedostoista Azure Data Factoryn avulla ladatut raakataulut. Taulujen lataustiedot rivimäärineen ja päivämäärineen kerätään erilliseen audit-tauluun, jotta saadaan historiatietoa mahdollisia ongelmanselvitystilanteita varten. Näin saadaan talteen tieto siitä, mitä ja milloin tietokantaan on ladattu.

Pres-skeemasta löytyy ajantasainen tilanne, aktiivisessa käytössä olevat taulut ja näkymät. Kun uusi data on ladattu raw-skeemaan, Azure Data Factory käynnistää tietokannassa proseduurin, joka lataa datan raw-tauluista pres-tauluihin. Pres-skeeman taulut sisältävät raporttiajoissa tarvittavia elementtejä: tauluissa on avaimia ja niiden kolumnien datatyypit on käyty läpi, jotta ne vastaavat raporttien tarpeita. Lisäksi joillekin pres-skeeman tauluille on rakennettu indeksejä, jotta raporttiajo on nopeampi. Indeksointi nopeuttaa taulun sarakkeisiin kohdistuvia kyselyjä luomalla osoittimia datan tallennuspaikkaan tietokannassa (The Data School 2019). Azure SQL-tietokannassa indeksointi on tehty helpoksi, sillä Azure osaa antaa suosituksia tietokannassa tarvittavista

indekseistä sen perusteella, millaisia kyselyjä kannassa on suoritettu. Pres-skeema sisältää vain viimeisimmän kuukauden tiedot, jotta taulujen koko pysyy maltillisena ja kyselyjen suoritus kuukausiraportointia varten on näin nopeampaa.

Data ladataan myös archive-skeemaan, jotta se saadaan kerättyä mahdollista myöhempää tarkastelua ja mahdollisia uusia raportointitarpeita varten. Archive-skeeman tauluihin data kertyy kumulatiivisesti, ja niihin on lisätty myös kenttä, josta ilmenee latauksen ajankohta.

Tietokannassa tehtävät toimenpiteet on rakennettu SQL-proseduureiksi. Toteutus on tehty siten, että proseduurit voidaan ajaa milloin tahansa uudelleen vaarantamatta datan eheyttä: esimerkiksi osaan raporttitauluista data kertyy kumulatiivisesti, joten mahdollisia uudelleenajoja varten SQL-proseduuriin on pitänyt rakentaa tarkistus, joka poistaa ladattavan periodin datan tauluista ennen uudelleenlatausta.

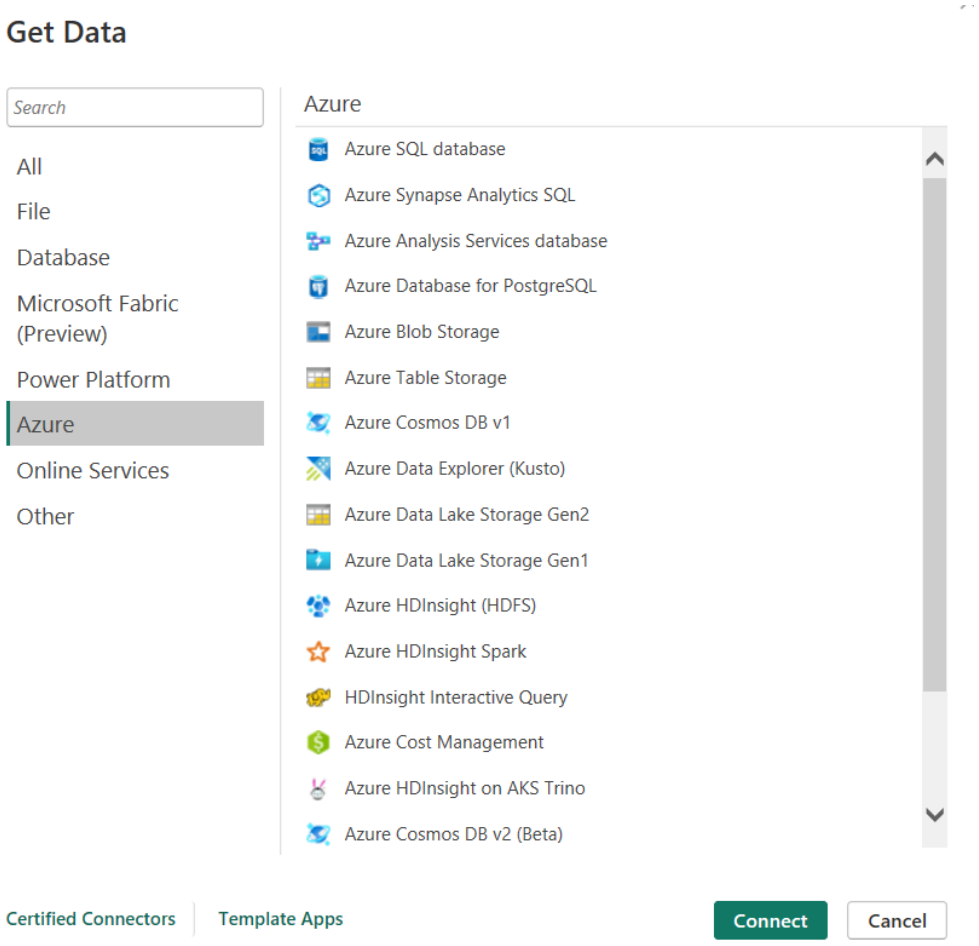
7.5 Power BI -toteutus

Tietokantaa käytetään Power BI:n kautta. Suorituskyvyn kannalta Power BI:n datamalli on hyvä pitää mahdollisimman pienikokoisena sekä suorittaa tarvittavat datamuunnokset jo lähdejärjestelmässä (Philps 2023; Microsoft 2023c). Järjestelmän raportointitarpeet ovat tässä vaiheessa visualisoinneiltaan varsin yksinkertaisia, mutta ne vaativat useiden taulujen tietojen yhdistämisiä tietyillä kriteereillä. Tässä tapauksessa on tehokkaampaa tehdä yhdistelyt tietokannassa SQL-kyselyillä ja rakentaa näin Power BI:tä varten valmiiksi tarvittavat taulut ja näkymät, jotta voidaan välttyä monimutkaiselta datamallilta ja laskennalta Power BI:n puolella. Tietokannassa voidaan suorittaa myös taulujen indeksointia tarvittavilta osin kyselyiden suorituskyvyn tehostamiseksi.

Valmiit taulut ja näkymät on ladattu Power BI -datamalliin. Näin Power BI:ssä on vain tarvittavat tietokentät, eikä dataa tarvitse enää muuntaa. Datan tuominen Power BI:hin on näin nopea ja kevyt toimenpide, eikä Power BI:n puolella tarvitse suorittaa paljoa laskentaa. Raporttien päivittäminen on näin ollen nopeaa. Datamalli on tällä hetkellä hyvin yksinkertainen, mutta sitä voi jatkossa rakentaa ja kehittää joustavasti tarpeen mukaan. Kehitystyössä on vapaus harkita tilanteen

mukaan, onko järkevämpää tehdä lisärakenteita suoraan tietokantaan, vai tuoda taulut Power BI:n datamalliin ja tehdä tarvittava yhdistely siellä.

Power BI tarjoaa erittäin monipuoliset mahdollisuudet esittää tietoa visuaalisesti, esimerkiksi volyymeja, kehitystä ja trendejä, jopa ennusteita. Tässä tapauksessa tarpeet ovat kuitenkin varsin yksinkertaisia, kun halutaan vain saada aiemmin Excelissä tehdyt toimenpiteet Power BI -ympäristöön. Työ aloitetaan ottamalla Power BI Desktopilla yhteys tarvittaviin datalähteisiin. Power BI tukee erittäin laajaa valikoimaa datalähteitä (kuvio 28).



Kuvio 28. Power BI:n datalähteitä

Tässä tapauksessa tarvitaan yhteys Azure SQL-tietokantaan ja Sharepointiin, josta saadaan malliin mukaan muutama Excel-muodossa olevan tukitiedosto. Kun yhteys tietokantaan on auki, valitaan, mitkä taulut tai näkymät halutaan tuoda Power BI -datamalliin. Kun tämä määritys on tehty, data voidaan päivittää

kannasta datamalliin ja visualisointeihin aina haluttaessa. Datamallissa voidaan myös määritellä taulujen välisiä relaatioita.

Myös Power BI:ssä suositellaan käytettäväksi aiemmin kuvattua tähtimallia. Jokainen visualisointi generoi datamalliin kyselyn, joka suodattaa, ryhmittelee ja yhdistää mallissa olevaa dataa. Suorituskykyä siis tukee datamalli, jossa on näitä toimenpiteitä tehokkaasti tukevia tauluja. Tähtimallissa dimensiotaulut tukevat suodatusta ja ryhmittelyä ja faktataulu yhteenvetoja, joten sen rakentamista pidetään parhaiden käytäntöjen mukaisena. (Microsoft 2023h.) Kun tarvittava data on tuotu datamalliin, rakennetaan halutut visualisoinnit. Lopuksi raportti julkaistaan Power BI -palveluun. Power BI -palveluun ajastetaan erikseen tietojoukon päivitys datalähteistä, jotta saatavilla on aina ajantasaista tietoa.

8 POHDINTA

8.1 Prosessin eteneminen

Opinnäytetyön keskeisenä tavoitteena oli automatisoida tiedon tuottamista kahdessa osassa. Ensimmäisessä osassa automatisoitiin asiakkaiden tarvitseman tiedon tuottaminen halutussa muodossa ja toisessa osassa automatisoitiin sisäisten raporttien tuottaminen. Tutkimuskysymys oli, miten dataputki on järkevintä ja tehokkainta toteuttaa määritellyssä käyttötapauksessa. Käyttötapaus on varsin tyypillinen, ja myös toteutus on yleisesti käytetyillä työkaluilla ja alustoilla tehty. Työhön sisältyy prosessin suunnittelu ja teknisten ratkaisujen valinta.

Työn ensimmäisessä osassa toteutettiin tiedostojen muunnos xml-muodosta Excel-muotoon. Tässä oli haasteena sopivimman teknisen toteutusratkaisun löytäminen. Vaihtoehtoja oli useita, mutta toteutukseen päätyi Azure Functionsilla tehty ratkaisu yksinkertaisuutensa ja joustavuutensa vuoksi. Ratkaisu on tehokas, sillä aiemmin koko päivän kestänyt työ hoituu nyt automaattisesti noin viidessä minuutissa. Toteutuksen rakennusprosessi oli iteratiivinen. Ensin etsittiin sopivinta teknistä toteutusvaihtoehtoa tutkimalla ja kokeilemalla, ja sen löydyttyä myös itse toteutus rakentui ja kehittyi iteratiivisesti. Esimerkiksi, ensin Python-koodi teki konversion yhdelle tiedostolle kerrallaan ja ilman Excel-tiedoston muotoilemista, mutta lopullisessa versiossa koodi käy läpi koko hakemiston tiedostot ja Excel-tiedostoille on tehty muotoilua luettavuuden parantamiseksi.

Työn toisessa osassa noudettiin lähdejärjestelmästä raportoinnin tarvitsemat tiedot SQL-kantaan ja vietiin sieltä Power BI -raporteille. Datan liikuttaminen ja muunnokset tehtiin Azure Data Factoryn dataputkessa. Tietokanta ja Power BI:n datamalli rakennettiin tähtimallin mukaisesti. Kokonaisuutena voidaan todeta, että Azure-alusta tarjoaa runsaan valikoiman työkaluja, joista sopivimpien valitseminen kulloiseenkin tarkoitukseen vaatii perehtymistä. Ei voida ajatella, että jokin ratkaisu olisi pohjimmiltaan oikea tai väärä, jos sillä saavutetaan haluttu lopputulos, mutta tehokkuus ja kustannukset saattavat olla erilaisia.

Dataputken rakenne on kehittynyt iteroitumalla työn edetessä useiden viikkojen aikana. Koska Data Factory tukee gitiä, sinne on helppo rakentaa

osakokonaisuuksia eri toimintojen testaamista varten. Lisäksi se mahdollistaa erilaisten toteutusvaihtoehtojen rakentamisen, joista voidaan valita, mitkä viedään lopulliseen toteutukseen. Toteuttaminen on luovaa työtä, sillä samat asiat on mahdollista tehdä monilla eri tavoilla. Myös tietokannan SQL-proseduurit kehittyivät jatkuvasti työn edetessä.

Pyrkimyksenä on ollut tehdä mahdollisimman yksinkertainen dataputki, johon on myös helppo tehdä muutoksia. Periaatteena on välttää samaan tarkoitukseen käytettävien rakenteiden tekeminen useaan kertaan ja hyödyntää parametreja (Microsoft 2023e) esimerkiksi API-päätepisteiden ja tiedostonimien määrittelyssä. Teoriapohja antaa varmuuden ratkaisun järkevyydestä, mutta oman kokemukseni mukaan parasta oppia on nähdä käytössä olevia, muiden tekemiä toteutuksia. Dataputken rakentaminen Azure Data Factoryllä on selkeää, sillä graafinen käyttöliittymä auttaa kokonaisuuden hahmottamista ja eri tarkoituksiin löytyy yleensä suoraviivaisesti aktiviteetti, jolla haluttu asia voidaan toteuttaa.

Vastaus tutkimuskysymykseen oli ensimmäisen tehtävän osalta, että tiedostomuunnos oli järkevintä ja tehokkainta toteuttaa Azure Functionsin avulla. Toisen tehtävän osalta todettiin, että raporttien tuottaminen ja automatisointi oli järkevintä ja tehokkainta toteuttaa käyttäen Azure Data Factoryn natiiveja ominaisuuksia, kuten sopivia aktiviteetteja, parametrejä ja ajastuksia, SQL-tietokannan rakenteita ja proseduureja sekä Power BI:n toiminnallisuutta.

8.2 Tutkimusmenetelmien toteutuminen

Tapaustutkimus mahdollistaa aiheen tarkastelun sen omassa kontekstissa käyttäen useita tietolähteitä, tavoitteena tuoda teoria kosketuksiin empiirisen maailman kanssa (Juuti & Puusa 2020, 199). Pyrkimyksenä on tuottaa tietoa valitusta, selkeästi rajatusta tapauksesta omassa kontekstissaan, eli osana tiettyä ympäristöä (Eriksson & Koistinen 2014, 7). Tässä työssä keskityttiin nimenomaisesti tiettyjen, rajattujen tarpeiden ratkaisemiseen ennaltamäärätyssä ympäristössä. Työ tuotti ratkaisun toimeksiannossa määriteltyihin tavoitteisiin, eikä tarkoituksena ollut kuvata esimerkiksi jonkin käytetyn työkalun kaikkia

mahdollisuuksia ja ominaisuuksia, vaan vain toimeksiannon toteutuksessa relevantit asiat.

Toimintatutkimuksen vaiheet ovat ongelman kartoitus ja suunnittelu, suunnitelman toteuttaminen ja vaikutusten havainnointi sekä tilanteen arviointi ja opitun tunnistaminen (Business Research Methodology 2023). Toimintatutkimuksen luonne mahdollistaa etenemistavan, jossa voidaan tutkia, kokeilla ja arvioida useita kertoja, kunnes lopputulos tyydyttää (Kuula 1999, 218), ja työn käytännön toteutus eteni juuri tämän mukaisesti. Kuvion 1 mukainen toimintatutkimuksen spiraali toteutui työn käytännön toteutuksessa hyvin, sillä prosessi oli iteratiivinen, ja lopputulos jalostui työn edetessä ja tiedon kertyessä.

Tässä työssä on tutkittu ja tehty ratkaisu tietyille käyttötapauksille, mutta tämäntyyppiset tarpeet ovat datamaailmassa varsin tyypillisiä. Tämä työ on pyritty toteuttamaan hyvien periaatteiden mukaisesti, ja siten se on hyödynnettävissä selvitettäessä Azure-alustalle soveltuvia ratkaisuja vastaaviin tarpeisiin. Ratkaisut ja niiden perustelut perustuvat mahdollisimman luotettaviin ja ajantasaisiin lähteisiin. Tästä syystä kirjallisuuslähteiden käyttö on jäänyt teknisiä toteutuksia kuvattaessa vähäiseksi, sillä nämä teknologiat kehittyvät jatkuvasti. Esimerkiksi verkossa oleva Microsoftin dokumentaatio on varmuudella ajantasaista tietoa.

8.3 Onnistumisen arviointi ja jatkomahdollisuudet

Raportointi on ollut jonkin aikaa käytössä. Raporttien tuottamisessa Data Factoryn dataputken ajo kestää tämän hetken datamäärällä noin 20 minuuttia, ja Power BI -raporttien päivittäminen tietokannasta korkeintaan minuutin. Kokonaisuudessaan tarvittava manuaalisen työn määrä on vähäinen toteutuksen jälkeen. Ratkaisu tekee pyydetyt asiat, joten määritellyt tavoitteet on saavutettu.

Erityisesti Power BI -raporttien kehittämiselle on tulevaisuudessa paljon potentiaalia. Sen avulla saataisiin visualisoitua esimerkiksi volyymi- ja historiatietoja, ja kuten on jo hieman toivottukin, tehtyä myös ennusteita tulevasta kehityksestä. Power BI -kehityksen suunnittelu ja toteutus toimisi hyvin jatkotutkimusaiheena, sillä tässä aihetta on käsitelty hyvin yleisellä tasolla.

LÄHTEET

Anttila, P. 2014. Tutkimisen taito ja tiedon hankinta. Metodix – metoditietämystä kaikille. Viitattu 28.5.2023 https://metodix.fi/2014/05/17/anttila-pirkko-tutkimisen-taito-ja-tiedon-hankinta/.

Bigelow, S. 2023. What is Azure DevOps? TechTarget 03/2023. Viitattu 22.9.2023 https://www.techtarget.com/searchwindowsserver/definition/Azure-DevOps-formerly-Visual-Studio-Team-Services.

Bito 2023. Nested Json Example: Json Explained. Viitattu 6.10.2023 https://bito.ai/resources/nested-json-example-json-explained/#4.

Bryan, A. & Yusuf, B. 2023. What is an API endpoint? Contentful 26.1.2023. Viitattu 6.10.2023 https://www.contentful.com/blog/api-endpoint/.

Business Research Methodology 2023. Action Research. Viitattu 28.5.2023 https://research-methodology.net/research-methods/action-research/.

Cote, C., Kamrat Gutzait, M. & Ciaburro, G. 2018. Hands-On Data Warehousing with Azure Data Factory: ETL Techniques to Load and Transform Data from Various Sources, Both on-Premises and on Cloud. Birmingham; Mumbai: Packt Publishing, Limited.

Datacamp 2023. A List of The 17 Best ETL Tools And Why To Choose Them. Viitattu 28.5.2023 https://www.datacamp.com/blog/a-list-of-the-16-best-etl-tools-and-why-to-choose-them.

Dice 2020. Dice Tech Job Report. Viitattu 28.5.2023 http://marketing.dice.com/pdf/2020/Dice_2020_Tech_Job_Report.pdf.

D'Souza, E. 2020. What is Azure Data Factory? Productive Edge 27.7.2020. Viitattu 22.9.2023 https://www.productiveedge.com/blog/azure-data-factory-capabilities.

Dutrée, A. 2021. Data pipelines: What, why and which ones. Towards Data Science 8.9.2021. Viitattu 28.5.2023 https://towardsdatascience.com/data-pipelines-what-why-and-which-ones-1f674ba49946.

Eriksson, P. & Koistinen, K. 2014. Monenlainen tapaustutkimus. Tutkimuksia ja selvityksiä 11/2014. Helsinki: Kuluttajatutkimuskeskus. Viitattu 28.5.2023 http://hdl.handle.net/10138/153032.

Gupta, R. 2019. A Walkthrough of SQL Schema. SQLShack 9.10.2019. Viitattu 22.9.2023 https://www.sqlshack.com/a-walkthrough-of-sql-schema/.

Hadges, H. 2023. Data Modeling Fundamentals in Power BI. phData 13.6.2023. Viitattu 27.10.2023 https://www.phdata.io/blog/data-modeling-fundamentals-in-power-bi/.

Helsingin kaupunki 2023. Helsingin datastrategia. Luku 1.1 Käsitteistö. Viitattu 28.5.2023 https://digi.hel.fi/esittely/helsinki-datastrategia/helsinki-datastrategia-tiivistelma/11-k%C3%A4sitteist%C3%B6/.

Hovi, A. 2018. Data-alan termien selitykset ja kuvaukset. Ari Hovi 6.6.2018. Viitattu 28.5.2023 https://www.arihovi.com/3274-2/.

IBM 2023a. What is an API? Viitattu 28.5.2023 https://www.ibm.com/topics/api.

– 2023b. What is ELT? Viitattu 28.5.2023 https://www.ibm.com/topics/elt.

Invati Inc. 2021. Excel Writer for Azure Data Factory. Viitattu 29.9.2023 https://invati.ai/software-development-accelerators/excel-writer-for-azure-data-factory/.

JavaTpoint 2021. Python Openpyxl. Viitattu 29.9.2023 https://www.javatpoint.com/python-openpyxl.

Juuti, P. & Puusa, A. 2020. Laadullisen tutkimuksen näkökulmat ja menetelmät. Helsinki: Gaudeamus.

Kemppainen, J. 2015. ETL-prosessin suunnittelu. Opinnäytetyö, Haaga-Helia ammattikorkeakoulu. Viitattu 8.11.2023 https://urn.fi/URN:NBN:fi:amk-2015120218845.

Kuula, A. 1999. Toimintatutkimus. Kenttätyötä ja muutospyrkimyksiä. Tampere: Vastapaino.

Lutkevich, B. 2021. What is a data engineer? TechTarget 03/2021. Viitattu 28.5.2023 https://www.techtarget.com/searchdatamanagement/definition/data-engineer.

Microsoft 2023a. Azure Data Factory documentation. Viitattu 28.5.2023 https://learn.microsoft.com/en-us/azure/data-factory/.

– 2023b. Copy activity in Azure Data Factory and Azure Synapse Analytics Viitattu 29.9.2023 https://learn.microsoft.com/en-us/azure/data-factory/copy-activity-overview.

– 2023c. Data reduction techniques for Import modeling. Viitattu 22.9.2023 https://learn.microsoft.com/en-us/power-bi/guidance/import-modeling-data-reduction

– 2023d. Excel file format in Azure Data Factory and Azure Synapse Analytics. Viitattu 29.9.2023 https://learn.microsoft.com/en-us/azure/data-factory/format-excel.

– 2023e. How to use parameters, expressions and functions in Azure Data Factory. Viitattu 20.10.2023 https://learn.microsoft.com/en-us/azure/data-factory/how-to-expression-language-functions.

– 2023f. Introduction to Azure Data Lake Storage Gen2. Viitattu 6.10.2023 https://learn.microsoft.com/en-us/azure/storage/blobs/data-lake-storage-introduction.

– 2023g. Mapping data flows in Azure Data Factory. Viitattu 29.9.2023 https://learn.microsoft.com/en-us/azure/data-factory/concepts-data-flow-overview.

– 2023h. Star schema relevance to Power BI models. Viitattu 13.10.2023 https://learn.microsoft.com/en-us/power-bi/guidance/star-schema.

– 2023i. Supported languages in Azure Functions. Viitattu 29.9.2023 https://learn.microsoft.com/en-us/azure/azure-functions/supported-languages.

– 2023j. What is Azure Batch? Viitattu 29.9.2023 https://learn.microsoft.com/en-us/azure/batch/batch-technical-overview.

– 2023k. What is Azure Logic Apps? Viitattu 29.9.2023 https://learn.microsoft.com/en-us/azure/logic-apps/logic-apps-overview.

Mustonen, J. 2020. On-premises-datan käsittely pilviraportoinnissa: case CGI Suomi Oy. Opinnäytetyö, LAB-ammattikorkeakoulu. Viitattu 8.11.2023 https://urn.fi/URN:NBN:fi:amk-202005057370.

Nelson, M. 2022. Beyond The Buzzword: What Does Data-Driven Decision-Making Really Mean? Forbes 23.9.2022. Viitattu 28.5.2023 https://www.forbes.com/sites/tableau/2022/09/23/beyond-the-buzzword-what-does-data-driven-decision-making-really-mean/.

Ogunleke, S. 2022. Why Is Python Popular for Data Science? MakeUseOf.com 14.1.2022. Viitattu 29.9.2023 https://www.makeuseof.com/why-is-python-popular-for-data-science/.

Pandas 2023. Intro to data structures. Pandas documentation. Viitattu 29.9.2023 https://pandas.pydata.org/docs/user_guide/dsintro.html.

Pekkarinen, M. 2022. ETL-prosessin toteuttaminen Azure Data Factoryllä. Opinnäytetyö, Seinäjoen ammattikorkeakoulu. Viitattu 8.11.2023 https://urn.fi/URN:NBN:fi:amk-2022120827368.

Philps, G. 2021. Data Loading And Transformation Best Practices. Enterprise DNA 14.4.2021. Viitattu 22.9.2023 https://blog.enterprisedna.co/data-loading-and-transformation-best-practices/.

Phuong, P. 2020. A case study in developing an automated ETL solution: concept and implementation. Opinnäytetyö, Turun ammattikorkeakoulu. Viitattu 8.11.2023 https://urn.fi/URN:NBN:fi:amk-2020052614188.

Pubudu, N. 2021. Understanding git and version control systems. Medium 14.5.2021. Viitattu 22.9.2023 https://medium.com/nerd-for-tech/understanding-git-and-version-control-systems-e76da5ec8b34.

RedHat 2022. What is a CI/CD pipeline? Viitattu 22.9.2023 https://www.redhat.com/en/topics/devops/what-cicd- pipeline.

Reeve, A. 2013. Managing Data in Motion: Data Integration Best Practice Techniques and Technologies. Amsterdam: Elsevier.

Saarela-Kinnunen, M. & Eskola, J. 2010. Tapaus ja tutkimus = Tapaustutkimus? Teoksessa J. Aaltola & R. Valli (toim.) Ikkunoita tutkimusmetodeihin 1. Metodin valinta ja aineistonkeruu: virikkeitä aloittelevalle tutkijalle. 3., uudistettu ja täydennetty painos. Jyväskylä: PS-kustannus, 189–199.

Serverless360 2023. Microsoft Azure Functions. Viitattu 29.9.2023 https://www.serverless360.com/azure-functions.

SmartBear Software 2023. What Is Swagger? Viitattu 6.10.2023 https://swagger.io/docs/specification/2-0/what-is-swagger/.

Solarwinds 2023. What is a SQL Database? Viitattu 6.10.2023 https://www.solarwinds.com/resources/it-glossary/sql-database.

Sriparasa, S. & D'mello, B. 2018. JavaScript and JSON Essentials : Build Light Weight, Scalable, and Faster Web Applications with the Power of JSON, 2nd Edition. Birmingham, UK: Packt Publishing, Limited.

TechTarget 2023. Star schema. Viitattu 29.9.2023 https://www.techtarget.com/searchdatamanagement/definition/star-schema.

The Data School 2019. Indexing. Viitattu 13.10.2023 https://dataschool.com/sql-optimization/how-indexing-works/.

Tietoevry 2020. Datasanasto. Viitattu 28.5.2023 https://www.tietoevry.com/fi/uutishuone/kaikki-uutiset-ja-tiedotteet/blogi/2020/ota-datamaailman-termit-haltuun-datasanaston-avulla/.

Towards Data Science 2022. Is Data The New Oil of the 21st Century or Just an Overrated Asset? Viitattu 28.5.2023 https://towardsdatascience.com/is-data-the-new-oil-of-the-21st-century-or-just-an-overrated-asset-1dbb05b8ccdf.

Tutorials Point 2023. Power BI - Introduction. Viitattu 22.9.2023 https://www.tutorialspoint.com/power_bi/power_bi_introduction.htm.

Verma P. 2023. Unlocking the Power of API Pagination: Best Practices and Strategies. DEV Community 6.6.2023. Viitattu 6.10.2023 https://dev.to/pragativerma18/unlocking-the-power-of-api-pagination-best-practices-and-strategies-4b49.

Vilkka, H. 2021. Näin onnistut opinnäytetyössä. Ratkaisut tutkimuksen umpikujiin. Jyväskylä: PS-kustannus.