VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Justus Sydänmaa

KESKITETYN TIETOVARASTON PROTOTYYPPI POSTGRESQL JA AIRBYTE-TEKNOLOGIOILLA

Liiketalous
2024

VAASAN AMMATTIKORKEAKOULU
Tietojenkäsittely

TIIVISTELMÄ

Tekijä	Justus Sydänmaa
Opinnäytetyön nimi	Keskitetyn tietovaraston prototyyppi PostgreSQL ja Airbyte-teknologioilla
Vuosi	2024
Kieli	suomi
Sivumäärä	51
Ohjaaja	Antti Mäkitalo

Tämän opinnäytetyön tavoitteena oli toteuttaa toimeksianto Trineria Oy:lle. Sen tarkoituksena oli luoda keskitetyn tietovaraston prototyyppi, hyödyntäen PostgreSQL- ja Airbyte-teknologioita. Päätavoitteena oli kehittää minimivaatimukset täyttävä tuote (MVP).

Teoreettisena viitekehyksenä on tietojärjestelmien ja tietokantojen hallinta, erityisesti keskittyen tietovarastojen rakentamiseen ja tietojen integraatioprosesseihin.

Opinnäytetyön käytännön osuudessa perehdytään integraatioalustan toimintoihin, tietokantayhteyksien muodostamiseen sekä TypeScript-koodin soveltamiseen, keskittyen yksittäisen tietolähteen käsittelyyn.

Opinnäytetyön tuloksena saatiin luotua toimiva prototyyppi toimeksiantajalle, joka hallitsee Jira-palvelusta tiedon haun. Prototyypin kehittäminen on osoittanut, kuinka tietovarastot voivat tehostaa tiedon käsittelyä ja saatavuutta, mikä on erityisen tärkeää nykyaikaisessa liiketoimintaympäristössä, jossa tieto on keskeisessä roolissa päätöksenteossa.

Avainsanat airbyte, tietovarastot, TypeScript, PostgreSQL

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietojenkäsittely

ABSTRACT

Author	Justus Sydänmaa
Title	Centralized Data Warehouse Prototype with PostgreSQL and Airbyte technologies
Year	2024
Language	Finnish
Pages	51
Name of Supervisor	Antti Mäkitalo

The objective of this thesis was to create a commission for Trineria Oy. Another objective was to develop a prototype of a centralized data warehouse, utilizing PostgreSQL and Airbyte technologies. The primary goal was to develop a product that meets the minimum viable product (MVP) requirements.

The theoretical framework is based on the management of information systems and databases, with a particular focus on the construction of data warehouses and data integration processes.

In the practical section of the thesis, the work delves on into the functions of the integration platform, the formation of database connections, and the application of TypeScript code, focusing on the processing of a single data source.

As a result of the thesis, a functional prototype was created for the client, which manages data retrieval from the Jira service. The development of the prototype demonstrated how data warehouses can enhance data processing and availability, which is especially important in the modern business environment where information plays a central role in decision-making.

Keywords airbyte, data warehouses, TypeScript, PostgreSQL

SISÄLLYS

SANASTO

SQL	Tietokantakyselykieli, jota käytetään tietojen hallintaan ja tietokantojen käsittelyyn.
PostgreSQL	Avoimen lähdekoodin relaatiotietokannanhallintajärjestelmä (RDBMS).
Airbyte	Datan integraatioalusta, joka helpottaa tietojen keruuta ja yhdistämistä eri lähteistä.
Keskitetty tietovarasto	Yhdestä tai useammasta eri lähteestä koostuva tiedon keskusvarasto.
Docker	Konttienhallintasovellus, joka mahdollistaa ohjelmistojen konttien eristetyn ja tehokkaan käytön ja jakelun eri ympäristöissä.
CI/CD	Nykyaikainen ohjelmistokehityskäytäntö, joka korostaa jatkuvaa integraatiota ja toimitusta, nopeuttaen ohjelmistojen kehitystä ja julkaisua.
DevOps	Toimintamalli, joka yhdistää ohjelmistokehityksen ja IT-toiminnan, tavoitteena nopeuttaa ja parantaa palvelujen tuotantoa.
Business Intelligence	Raakadatan muuntamista hyödylliseksi tiedoksi liiketoiminta-analyysien tekemistä varten.
Haaraus (fork)	Lähdekoodin kopioimista ja uuden, itsenäisen sovelluksen luomista sen pohjalta.

API-avain	Tunniste, joka antaa pääsyn tiettyyn ohjelmistorajapintaan.
TypeScript	Microsoftin kehittämä ja ylläpitämä ohjelmointikieli.
ETL	Extract-Transform-Load. Termi tiedon poiminnalle, muokkaukselle ja lataukselle.
ELT	Extract-Load-Transform. Termi tiedon poiminnalle, lataukselle ja muokkaukselle.
Client	Käyttöliittymä, joka pyytää pääsyä palvelimen tarjoamaan palveluun.

1 JOHDANTO

Tänä päivänä valtavat tietomäärät virtaavat monista eri suunnista, ja yrityksille on elintärkeää seurata tätä dataa, sillä se mahdollistaa tehokkaan tiedolla johtamisen. Tieto on äärimmäisen arvokasta, ja se hajautuu useisiin eri tietolähteisiin. Siksi on järkevää, että yritys pystyy keskittämään tarvitsemansa tiedot yhteen paikkaan, mikä helpottaa niiden hallintaa ja tulkintaa.

Tämän opinnäytetyön tavoitteena on kehittää prototyyppi keskitetystä tietovarastosta ohjelmistotalolle. Tietovarasto perustuu PostgreSQL ja Airbyte-teknologioihin ja sen tarkoituksena on kerätä, tallentaa ja hallita eri järjestelmistä saatavaa tietoa yhdessä keskitetyssä tietokannassa. Keskeisenä tavoitteena on riskienhallinta ja tarve siirtää varmuuskopioinnin vastuu yrityksen omalle järjestelmälle kolmannen osapuolen järjestelmän sijaan. Tietokannan suunnittelun ja kehittämisen aikana otetaan huomioon myös jatkokehitysmahdollisuudet. Lähtötilanteena on se, että kasvavalla yrityksellä on kertynyt dataa useampaan eri paikkaan ja on ajankohtaista siirtää kaikki data keskitetysti yhteen paikkaan, jotta tiedon hallinnointi olisi helpompaa sekä selkeämpää.

Työn edetessä seurataan projektin edistymistä tutkimuskysymysten kautta:

1. Onko Airbyte-palvelu soveltuva pienen yrityksen tietovaraston tarpeisiin?

2. Aiheuttaako tietovarasto tarpeetonta tietojen päällekkäisyyttä, kun dataa on jo saatavilla palveluista?

3. Onko tietovaraston rakentaminen itse järkevää vai olisiko viisaampaa hyödyntää valmista palvelua?

1.1 Toimeksiantaja

Työn toimeksiantaja Trineria Oy on vuonna 2014 perustettu teollisen internetin ratkaisuihin erikoistunut ohjelmistotalo, joka työllistää yli 15 IT-alan asiantuntijaa.

Trinerialla on yhteensä kolme toimipistettä Suomessa. Nämä toimipisteet sijaitsevat Vaasassa, Joensuussa ja Helsingissä, jossa sijaitsee myös heidän päätoimipisteensä. Aihe valikoitui Trineria Oy:n aloitteesta. (Trineria, 2022)

2 TEKNINEN TOTEUTUS

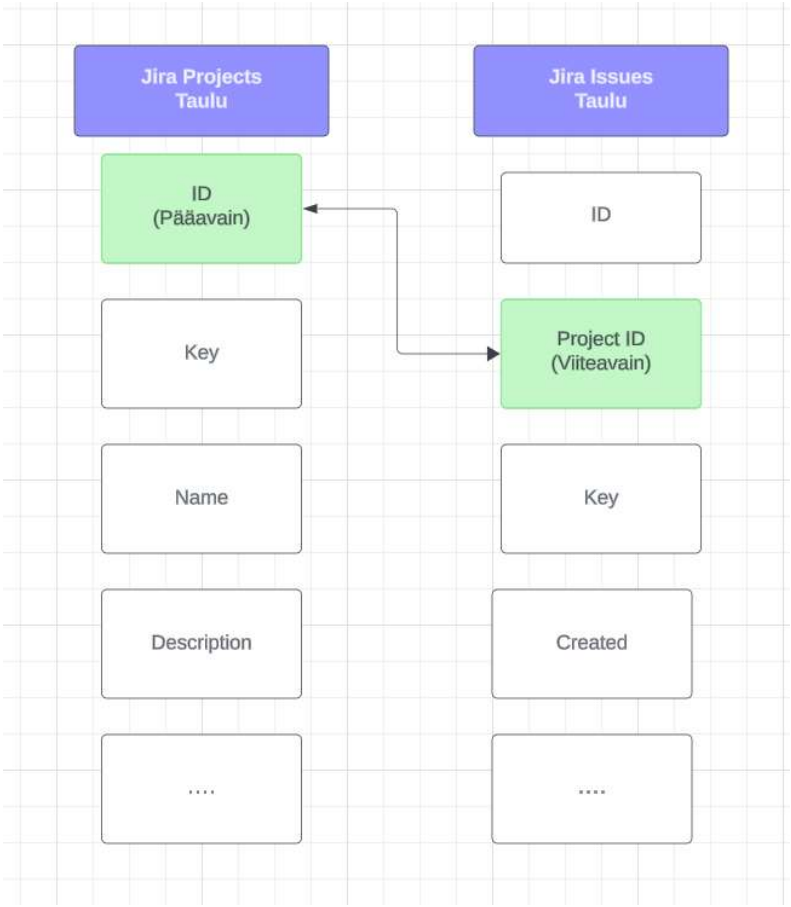
Jotta voidaan syventyä tarkemmin tietovaraston kehitysprosessiin, on hyödyllistä aloittaa tilanteen alustuksella, joka kertoo tarvittavista teknologioista. Tässä kappaleessa vertailen eri teknologioita, esittelen useita vaihtoehtoja ja perustelen, miksi toinen vaihtoehto oli toista parempi tai miksi päädyin tietyntyyppisiin teknologioihin.

2.1 Relaatiotietokanta

Relaatiotietokanta on tietokantamalli, jonka IBM kehitti vuonna 1970. Se mahdollistaa taulujen liittämisen toisiinsa käyttäen yhteistä attribuuttia. Taulut sisältävät tietokannan tiedot, kun taas niiden väliset suhteet määrittävät, kuinka nämä tiedot liittyvät toisiinsa ja vuorovaikuttavat keskenään. Tämä on olennaista tietokannan suunnittelussa ja tiedon käsittelyssä. (Google Cloud, 2024)

Taulujen sarakkeet määrittelevät tietotyypin, ja jokainen tietue sisältää kyseisen tietotyypin arvon. Jokaisella relaatiotietokannan taululla on pääavain (primary key), joka toimii taulun yksilöllisenä tunnisteena. Taulujen välille luodaan suhteita käyttämällä viiteavainta (foreign key), jonka avulla viitataan toisen taulun pääavaimiin. (Google Cloud, 2024)

Esimerkiksi kuvassa 1 on kaksi taulua, "Jira Projects" ja "Jira Issues". "Jira Projects" -taulu sisältää Jira-palvelun projektit, kun taas "Jira Issues" -taulu sisältää projekteihin liittyvät ongelmat eli projektin sisällä olevat yksittäiset tehtävät. Näiden taulujen välille on määritelty yhteys käyttämällä pääavainta ja viiteavainta. "Jira Projects" -taulussa ID toimii uniikkina pääavaimena, kun taas "Jira Issues" -taulussa "Project ID" on määritelty viiteavaimena, joka viittaa "Jira Projects" -taulun pääavaimiin. On tärkeää huomata, että näiden avainten nimet eivät välttämättä ole samat, mutta niiden väliset suhteet on silti määritelty.



Kuva 1. Jira Projects ja Jira Issues taulujen välinen suhde on määritelty avaimia käyttämällä.

Relaatiotietokanta mahdollistaa monimutkaisten suhteiden luomisen eri taulujen välille. Tauluilla voi olla erilaisia suhteita:

- Yhden suhde yhteen
- Yhden suhde moneen
- Monen suhde yhteen
- Monen suhde moneen

Näiden suhteiden avulla kuvataan, millaisessa vuorovaikutuksessa taulut ovat keskenään. Esimerkiksi "yhden suhde moneen" tarkoittaa suhdetta, joka voi esiintyä useaan kertaan. Tämän kaltainen suhde on "Jira Projects"-taulun ja "Jira

Issues"-taulun välillä. Suhteessa yksittäisellä projektilla voi olla useampi ongelma, mutta ongelma voi kuulua vain yhteen projektiin. ”Monen suhde moneen” on looginen liitos, mutta se ei toimi ilman erillistä aputaulua, joka luo kaksi yhden suhde moneen-liitosta. (Peterson, 2023)

2.2 SQL

SQL (Structured Query Language) on ohjelmointikieli, jota käytetään relaatiotietokantojen hallintaan. SQL:n perusominaisuuksia ovat tiedon tallentaminen, hakeminen, päivittäminen sekä poistaminen tietokannasta. Näitä toimintoja käytetään tietokantojen hallinnassa, tiedon käsittelyssä ja järjestelmän rakenteen määrittelyssä. SQL on laajalti käytetty ja se on standardi relaatiotietokantajärjestelmissä. SQL:n suosioon vaikuttaa sen kyky integroitua hyvin eri ohjelmointikieliin. Esimerkiksi kyselyjä voidaan luoda suoraan Java-ohjelmointikieleen, ja TypeScriptiin on saatavilla valmiita kirjastoja, jotka mahdollistavat suorat kyselyt. Kielen oppiminen on lähtökohtaisesti helppoa, sillä se käyttää englanninkielisiä avainsanoja lauseissaan. (Amazon, 2023a)

Ennen kyselyn suorittamista käydään läpi tietyt prosessit. Ensiksi tarkistetaan, että kysely on SQL-sääntöjen mukainen eli se sisältää oikeanlaisen syntaksin. Esimerkiksi SQL-käsky vaatii aina lauseen loppuun puolipisteen. Jos puolipiste puuttuu, kääntäjä palauttaa virheilmoituksen eikä toteuta kyselyä. Tämän lisäksi kyselyä tehdessä tarkistetaan kyselyn tekijän oikeudet. Tietyt lauseet voivat vaatia erityisoikeudet. SQL-kyselyt voidaan jaotella seuraaviin kategorioihin:

- Data definition language (DDL) käskyjä käytetään tietokannan rakenteen määrittelyyn. Esimerkiksi CREATE-komennolla voidaan luoda uusi taulu.
- Data query language (DQL) käskyillä haetaan tietoja tietokannasta. Näitä ovat esimerkiksi SELECT, FROM ja WHERE. Esimerkiksi tällä lauseella haetaan kaikki arvot jira_projects-taulusta: SELECT * FROM jira_projects;

- Data manipulation language (DML) käskyt kirjoittavat uutta tietoa tai muokkaavat olemassa olevia tietueita. DML-käskyjä ovat esimerkiksi INSERT, UPDATE ja DELETE.
- Data control language (DCL) käskyt luovat relaatiotietokantaan oikeuksia, jotka sallivat tehdä eri asioita tietokannassa. Esimerkiksi GRANT-komento antaa käyttöoikeuden valittuihin tauluihin.
- Transaction control language (TCL) käskyjä käytetään tapahtumienhallintaan, kuten ROLLBACK ja COMMIT. ROLLBACK-komento peruuttaa muutokset virheellisen tapahtuman jälkeen, säilyttäen tietokannan eheyden. Esimerkiksi useiden SQL-kyselyiden suorittaminen, jossa jokin kysely epäonnistuu, voi aiheuttaa tarpeen käyttää ROLLBACK-käskyä.

Näiden komentojen avulla voidaan tehokkaasti hallita ja manipuloida tietokannan tietoja eri tavoin. (Amazon, 2023a)

Trinerialla on relaatiotietokantaan soveltuvaa dataa ja suurin osa tiedoista on vahvasti keskinäisessä riippuvuudessa. Liiketoiminnan näkökulmasta on tärkeää, että näitä tietoja voidaan koostaa yhteen. Yritys on pieni ja datan määrä ei ole suuri, jonka vuoksi ei tarvita suurempaa järjestelmää, joka skaalautuisi paremmin. Tässä kontekstissa SQL on tehokas valinta relaatiotietokantojen hallintaan, sillä se mahdollistaa tiedon tehokkaan koostamisen ja käsittelyn.

2.3 NoSQL

NoSQL-tietokannat ovat tietokantoja, jotka tallentavat tiedot muussa kuin relaatiotaulujen muodossa. Niiden suosio kasvoi 2000-luvulla, kun tiedontallennuksien kustannukset laskivat ja sovellukset kasvoivat monimutkaisemmiksi. Tämä asetti ohjelmistokehittäjille paineita, ja NoSQL-tietokannat kehitettiin vastaamaan näihin tarpeisiin. (MongoDB, 2023)

NoSQL-tietokannat mahdollistavat nopean reagoinnin muuttuviin tarpeisiin ja skaalautuvat tehokkaammin kuin perinteiset tietokannat. Tämä nopea reagointi on erityisen tärkeää ketterässä ohjelmistokehityksessä, jossa joustavuus ja nopea kehitys ovat keskeisiä.

Tietokantoja voidaan skaalata kahdella tavalla: horisontaalisesti ja vertikaalisesti. Horisontaalisessa skaalauksessa laskentatehoa ja tallennustilaa jaetaan useiden tietokoneiden välille, mikä mahdollistaa järjestelmän kasvun sujuvasti. Vertikaalisessa skaalauksessa yksittäisen laitteen suorituskykyä parannetaan esimerkiksi päivittämällä sen komponentteja tai vaihtamalla tilalle tehokkaampi tietokone. (MongoDB, 2023)

Lisäksi pilvipalvelujen yleistyminen on edellyttänyt tietojen jakamista useille palvelimille eri alueilla. NoSQL-tietokannat ovat osoittautuneet vahvoiksi tässä ympäristössä, koska ne tarjoavat tehokkaan ja joustavan tavan hallita ja jakaa dataa monimutkaisissa pilviympäristöissä. (MongoDB, 2023)

NoSQL-tietokannat mahdollistavat suurien ja monimutkaisten tietojen tallentamisen. Toisin kuin relaatiotietokannoissa, joissa data on organisoitu tiukasti taulukkoihin, NoSQL-tietokannat tarjoavat joustavuutta erilaisten tietorakenteiden käyttöön. (MongoDB, 2023)

Yksi yleinen NoSQL-tietokantatyyppi on dokumenttitietokanta, joka tallentaa tiedot avain-arvo-pareina (Kuva 2). Tämä mahdollistaa monenlaisten tietotyyppien ja rakenteiden käytön, kuten merkkijonot, numerot, päivämäärät ja objektit. Dokumenttitietokannat tukevat eri formaatteja, kuten JSON, BSON ja XML. (MongoDB, 2023)

Kuva 2. JSON-muodossa oleva dokumentti, joka sisältää avain-arvopareja.

NoSQL tarjoaa joustavamman lähestymistavan kyselyiden luomiseen ja sallii löysemmän lauserakenteen verrattuna SQL-kyselyihin. Esimerkiksi, yllä mainitussa dokumentissa voidaan hakea halutun osoitteen -tiedot käyttämällä kyselyä: address.street = "123 Main St". Tämä kysely suodattaisi ja palauttaisi halutun datan. Kaikki käyttäjät, joiden ikä on alle 30 vuotta, voidaan hakea kyselyllä: age: {$lt: 30}. Tässä $lt-operaattori tarkoittaa “less than” eli “pienempi kuin”, ja se hakee dokumentit, joissa “age” on pienempi kuin 30. NoSQL-tietokantoja käytetään laajasti palveluissa, jotka käsittelevät suuria tietomääriä, kuten reaaliaikaisissa web-sovelluksissa, verkkokaupoissa ja Big Data -ympäristöissä. Niiden avulla voidaan hyödyntää täysin pilviympäristöjä, skaalata joustavasti ja toimittaa dataa viiveettömästi. (Oracle, 2023a)

2.4 Relaatiotietokantojen hallintajärjestelmät

Relaatiotietokannan hallintajärjestelmä on ohjelmisto, joka on suunniteltu relaatiotietokantojen hallintaan. Sen avulla käyttäjät voivat luoda, muokata ja hallinnoida relaatiotietokantoja. Tunnetuimpia relaatiotietokannan hallintajärjestelmiä ovat MySQL, PostgreSQL, MariaDB, Microsoft SQL Server ja

Oracle Database. Nämä järjestelmät tarjoavat laajan valikoiman ominaisuuksia ja toiminnallisuuksia tietojen hallintaan ja analysointiin. (Google Cloud, 2024)

2.4.1 MySQL

MySQL on avoimen lähdekoodin relaatiotietokantojen hallintajärjestelmä, joka tarjoaa tehokkaan suorituskyvyn suurillekin tietomäärille, tehden siitä sopivan valinnan yrityksille kaikista kokoluokista. Tämän joustavuuden ja suorituskyvyn ansiosta MySQL on laajalti suosittu monenlaisten sovellusten ja käyttötarkoitusten tietokantaratkaisuna, kyeten käsittelemään laajoja tietomassoja sekä tarjoamaan luotettavan alustan tietojen hallintaan. (Dyer, 2015)

MySQL käyttää GNU General Public License -lisenssiä, mikä mahdollistaa sen vapaan käytön ja muokkauksen, mutta Oracle omistaa MySQL-tavaramerkin. Tämä omistajuus tuo mukanaan tiettyjä rajoituksia, kuten sen, ettei MySQL:n tavaramerkkiä saa käyttää ilman lupaa tai sisällyttää "MySQL" -tekstiä uusiin sovelluksiin. (Dyer, 2015)

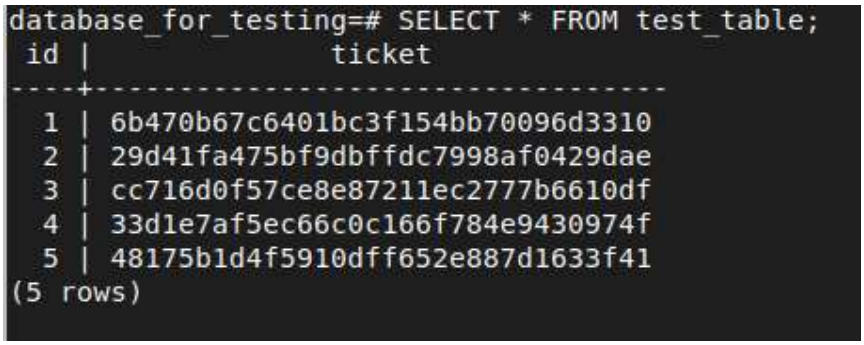
MySQL:n suosion ja avoimen lähdekoodin luonteen ansiosta siitä on kehitetty useita haarauksia, jotka tarjoavat vastaavia ominaisuuksia ilman tavaramerkkirajoituksia. Yksi tunnetuimmista on MariaDB, jonka perusti Michael Widenius, yksi MySQL:n alkuperäisistä kehittäjistä. MariaDB pyrkii tarjoamaan jatkuvia päivityksiä ja parannuksia verrattuna MySQL:ään, ja sekin jakaa GNU General Public License -lisenssin vapaan käytön periaatteet. (Kenler & Razzoli, 2015)

2.4.2 PostgreSQL

PostgreSQL on avoimen lähdekoodin relaatiotietokannan hallintajärjestelmä, joka tunnetaan erityisesti sen vakaudesta ja skaalautuvuudestaan. PostgreSQL:n avoimen lähdekoodin projekti on kasvanut ja kehittynyt kokonaiseksi ekosysteemiksi, jota ylläpitävät yhteisöt eri puolilla maailmaa. Nämä yhteisöt tarjoavat laajennuksia ja työkaluja PostgreSQL:lle. (Ferrari & Pirozzi, 2020)

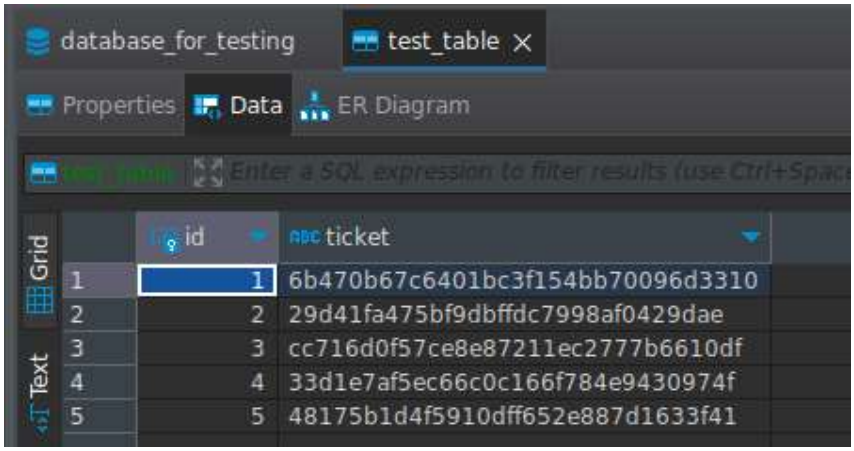
PostgreSQL hyödyntää BSD-lisenssiä, mikä tarkoittaa, että sitä voi voidaan käyttää vapaasti sekä suljetuissa että avoimissa projekteissa. Tämä lisenssi antaa käyttäjille laajan vapauden käyttää PostgreSQL:ää erilaisissa sovelluksissa ja ympäristöissä. (Ferrari & Pirozzi, 2020)

PostgreSQL mahdollistaa tietokannan hallinnan suoraan komentoriviltä (Kuva 3), tarjoten tehokkaan ja tarkasti kontrolloidun tavan tietokantojen käsittelyyn.



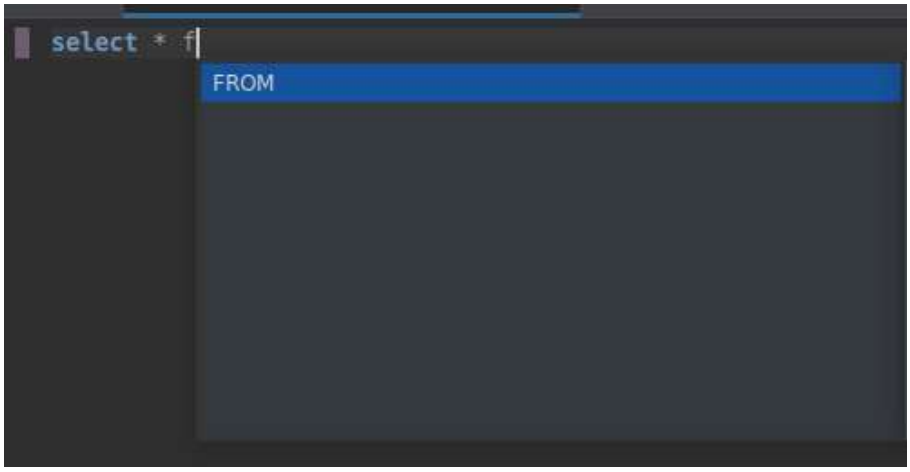
Kuva 3. Relaatiotietokannan hallinta komentoriviltä.

Vaihtoehtoisesti käyttöön löytyy erilaisia sovelluksia kuten DBeaver (Kuva 4), joka tarjoaa käyttäjälle selkeän ja helppokäyttöisen graafisen käyttöliittymän, joka helpottaa tietokantojen hallintaa ja tietojen tarkastelua visuaalisesti. Tällainen monipuolisuus mahdollistaa tietokantojen tehokkaan hallinnan eri käyttötapauksissa ja käyttäjien tarpeisiin perustuen.



Kuva 4. Relaatiotietokannan hallinta DBeaver-sovelluksessa.

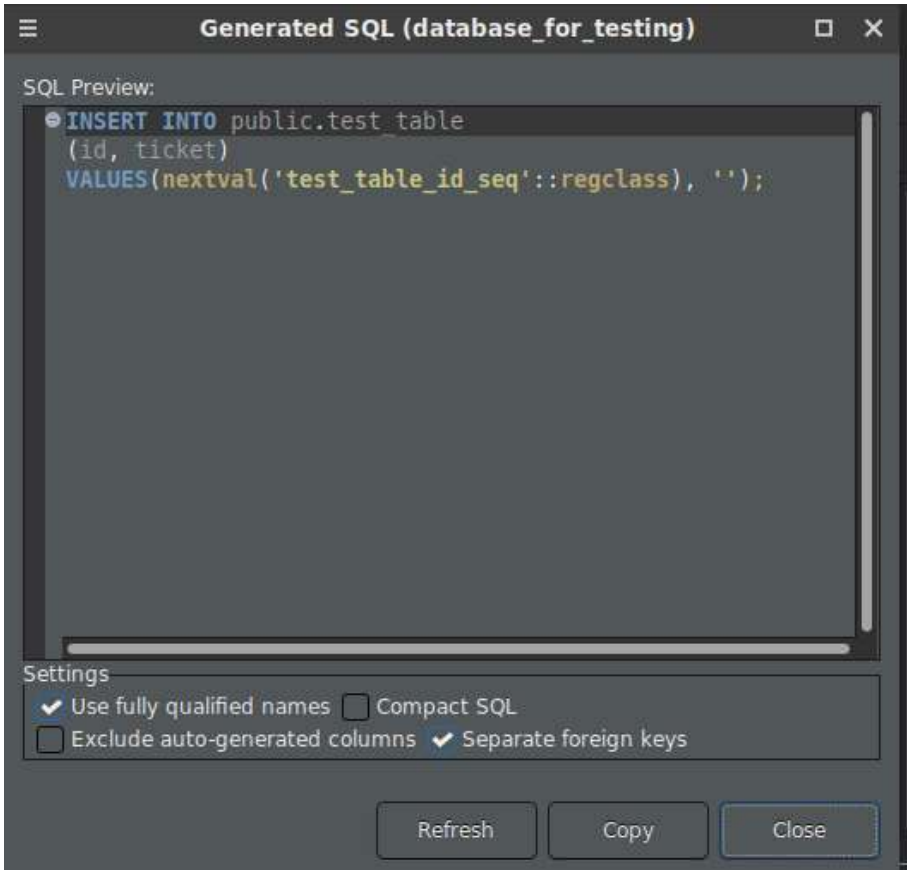
DBeaverissa SQL-käskyt onnistuvat käyttöliittymästä vaivattomasti, sillä sovellus osaa täydentää SQL-käskyjä automaattisilla ehdotuksilla (Kuva 5). Tämä ominaisuus nopeuttaa kirjoittamista ja vähentää virheiden mahdollisuutta, kun käyttäjä saa valmiita vaihtoehtoja SQL-käskyjen täydentämiseksi.



Kuva 5. DBeaver tarjoaa käyttäjälle kyselyn automaattista täydennystä.

Lisäksi sovellus kykenee generoimaan valmiin rungon kyselyille (Kuva 6), mikä erottaa sen perinteisistä komentorivipohjaisista SQL-kyselyistä. Komentorivillä työskennellessä käyttäjän täytyy kirjoittaa kyselyt pääosin itse alusta loppuun ilman automaattista täydennystä tai valmiita runkoja, mikä voi olla aikaa vievää ja

altistaa virheille. DBeaverin tarjoama toiminnallisuus helpottaa huomattavasti SQL-kyselyjen laatimista ja parantaa käyttökokemusta.



Kuva 6. DBeaver-sovelluksessa on myös mahdollisuus generoida automaattisia kyselyrunkoja.

Koska PostgreSQL on laajasti käytetty ja kypsä teknologia, joka on Trinerialla eniten käytössä, sen valitseminen osaksi tietovarastonkehitystä on luontevaa. On tärkeää pitää mielessä, että tulevaisuudessa voidaan siirtyä uudemman sukupolven järjestelmiin, mikäli se todetaan hyödylliseksi ja tarpeelliseksi.

2.5 Business Intelligence

Business Intelligence (BI) -ohjelmisto on järjestelmä, jonka avulla dataa voidaan analysoida, kerätä historiatietoja ja parantaa niiden avustuksella

liiketoimintapäätöksiä. BI-ohjelmiston avulla voidaan havaita tehokkaasti ongelmakohdat, uudet markkinatrendit ja uudet liiketoimintamahdollisuudet. (IBM, 2023a)

Nopeasti muuttuvassa teknologiaympäristössä BI:n käyttö on käytännössä välttämätöntä, sillä tietoa on niin valtavasti ja sen analysointi ilman erillistä ohjelmistoa voi osoittautua haastavaksi tehtäväksi. Tässä kohtaa keskitetty tietovarasto astuu kuvaan tarjoten tehokkaan ratkaisun tietojen keskitettyyn saatavuuteen.

2.6 Keskitetty tietovarasto

Keskitetty tietovarasto on järjestelmä, joka kokoaa useasta eri lähteestä saatavan tiedon yhdeksi, johdonmukaiseksi tietovarastoksi. Tietovaraston avulla pystytään tehokkaasti analysoimaan valtavia määriä dataa, johon tavallinen tietokanta ei pysty. (IBM, 2023b)

Tietovarastot on suunniteltu erityisesti BI-käyttöön, sillä ne mahdollistavat hyödyllisen tiedon kokoamisen ja analytiikan suorittamisen. Tietovarastot on suunniteltu nimenomaan kyselyiden ja analyysien tekemiseen, ja ne sisältävät usein suuria määriä historiallista tietoa. Keskitetyn tietovaraston käyttö mahdollistaa eri lähteistä tulevien suurten tietomäärien yhdistämisen, tarjoten organisaatioille arvokasta tietoa liiketoiminnan kehittämiseen ja päätöksentekoprosesseihin. (Oracle, 2023b)

Tyypillinen keskitetty tietovarasto koostuu yleensä relaatiotietokannasta, tietojen siirtoprosessista ja visuaalisesta käyttöliittymästä, jonka avulla voidaan näyttää dataa käyttäjille. Kuitenkin, koska tämä ratkaisu on prototyyppi, datan visualisointi jää tässä tapauksessa toteutuksen ulkopuolelle. (Oracle, 2023b)

Tietovaraston luomiseen on kaksi päävaihtoehtoa: sen voi rakentaa fyysiselle palvelimelle, jossa palvelimen toiminnasta ja ylläpidosta huolehditaan itse tai vaihtoehtoisesti valita pilvipalvelumuotoisen tietovaraston lisenssimaksulla.

Pilvipalvelussa käyttäjän ei tarvitse huolehtia palvelimen toiminnasta, mikä mahdollistaa keskittymisen olennaiseen toimintaan. (Oracle, 2023b)

Perinteinen tietovarasto, joka on rakennettu fyysisille palvelimille paikan päälle, tarjoaa edelleen monia etuja, kuten paremman hallinnoinnin, turvallisuuden ja pienemmän viiveen. Kuitenkin nämä perinteiset ratkaisut eivät ole yhtä joustavia kuin pilvessä sijaitsevat tietovarastot, ja on vaikea ennustaa tulevaisuutta ja sitä, kuinka tietovarasto tulisi skaalata tulevaisuutta ajatellen. Lisäksi näiden tietovarastojen hallinta voi olla erittäin monimutkaista. (Oracle, 2023b)

Pilvessä sijaitseva tietovarasto tarjoaa käyttäjälle joustavan ja skaalautuvan ratkaisun suuren datavaraston hallintaan. Tallennustilaa on helppo lisätä tarpeen mukaan, mikä tekee siitä erittäin joustavan käyttäjän tarpeisiin. Monet pilvessä sijaitsevat tietovarastot ovat täysin hallittuja ja itseohjautuvia, mikä tekee niiden käytöstä erittäin helppoa myös aloittelijoille. Lisäksi monet pilviperusteiset tietovarastot tarjoavat *pay-as-you-go* -mallin, jossa käyttäjä maksaa etukäteen vain tarvitsemastaan tietovaraston kapasiteetista, mikä eliminoi ylimääräiset kustannukset. Tämä ratkaisu tuo asiakkaille hyviä kustannussäästöjä. (Oracle, 2023b)

Tunnettuja pilvitietovaraston tarjoajia ovat muun muassa Microsoft Azure, Google BigQuery, Amazon Redshift ja SaaS-ratkaisuihin erikoistunut Snowflake. Esimerkiksi Amazon Redshift tarjoaa palvelun lisäksi runsaasti ylimääräisiä ominaisuuksia, joista suurin osa ei ole tarpeellisia Trinerialle.

Trineria pyrkii olemaan alustariippumaton, jonka vuoksi yhtiö hallinnoi itse omaa ratkaisuaan omilla virtuaalipalvelimilla DigitalOcean -palvelussa. Tarvittaessa se voi nopeasti siirtää kaikki palvelut uuden tarjoajan alaisuuteen.

2.7 Tietojen integraatioprosessi

Tietoa kootaan monista eri datalähteistä, ja tätä dataa on tarpeen käsitellä suodattamalla, poistamalla ja muokkaamalla, jotta se soveltuu BI-palveluiden

käyttöön ja tietoanalytiikkaan. Tämän käsittelyprosessin nimi on integraatioprosessi. (Amazon, 2023b)

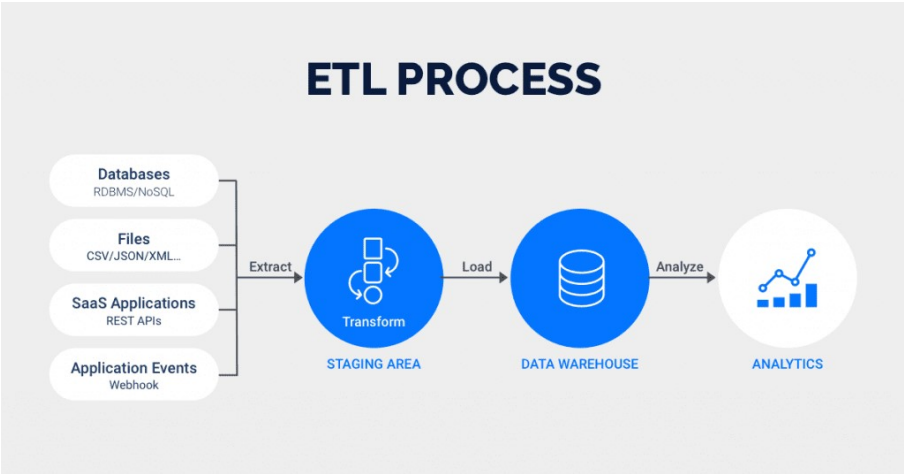
ETL sekä ELT ovat integraatioprosesseja, joissa dataa liikutellaan tietovarastoon. Molemmissa prosesseissa käsittely alkaa hakemalla tiedot lähteistä, mutta tämän jälkeen asiat käsitellään prosessin mukaan eri järjestyksessä. Seuraavissa osioissa käsitellään näiden prosessien eroja tarkemmin.

2.7.1 ETL

ETL tulee sanoista Extract (poiminta), Transform (muokkaus) ja Load (lataus). Prosessissa tiedot kerätään useista lähteistä, muokataan tietovarastolle sopivaksi ja ladataan tietovarastoon lopullista käyttötarkoitusta varten. (Amazon, 2023b)

ETL-prosessin aikana tiedot muokataan tietovaraston tarpeiden mukaisiksi, poistetaan tarpeettomat rivit ja korjataan mahdolliset virheet, kuten virheelliset tunnisteet (ID). Lisäksi prosessi päivittää järjestelmään uusimman tiedon, mikä pitää tietokannan ajan tasalla. Tämä tehostaa datan analysointia, visualisointia ja suuren datamäärän hallintaa, tarjoten arvokasta tukea liiketoiminnan päätöksenteolle. (Amazon, 2023b)

Kuvassa 7 havainnollistetaan ETL-prosessi. Aluksi kerätään dataa ulkoisista lähteistä, sitten se muokataan sopivaan muotoon yrityksen toiminnallisia tarpeita varten yleensä jonkun ulkoisen integraatiotyökalun avulla. Lopuksi se ladataan lopulliseen sijoituspaikkaan, kuten tietovarastoon. Tämän käsittelyn jälkeen tieto ovat saatavilla esimerkiksi BI- sovellusten käyttöön. (Amazon, 2023b)

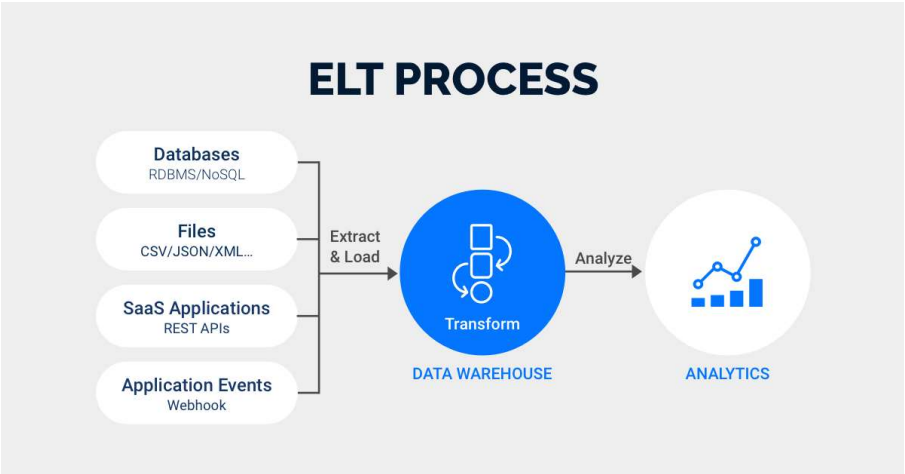


Kuva 7. ETL-putkiston prosessi. (Rivery, 2023)

Menetelmä on suosittu erityisesti vanhojen, legacy-tietovarastojen siirrossa keskitettyyn tietovarastoon, sillä sen avulla voidaan tyhjentää massiiviset, käyttämättömät tietorivit. (Amazon, 2023b)

2.7.2 ELT

Extract, Load, and Transform (ELT) -prosessi on moderni tapa käsitellä ja hallita suuria tietomääriä. Tässä prosessissa (Kuva 8) tiedot ensin poimitaan niiden alkuperäisestä lähteestä, minkä jälkeen ne siirretään suoraan tietovarastoon. Vasta tietovarastoon siirron jälkeen tiedot käsitellään ja muokataan käyttöön sopiviksi. Tämä menetelmä on erityisen tehokas, koska se mahdollistaa tiedon nopean siirron ja käsittelyn keskitetyssä tietokannassa, ilman että tietoa tarvitsee käsitellä siirron aikana. (Amazon, 2023b)



Kuva 8. ELT-putkiston prosessi. (Rivery, 2023)

ELT-prosessin etuna on sen joustavuus ja tehokkuus suurten tietomäärien käsittelyssä. Tiedot voidaan muokata ja käsitellä useita kertoja tarpeen mukaan, mikä on merkittävä etu verrattuna perinteisempään ETL-prosessiin. ETL-prosessissa tiedot käsitellään jo varhaisessa vaiheessa, mikä voi tehdä myöhemmistä muutoksista haastavampia. (Amazon, 2023b)

Trinerialle ELT-prosessin valinta on perusteltua, erityisesti nykyaikaisessa datan käsittelyssä, jossa tietoa voidaan tallentaa lähes rajattomasti. ELT tarjoaa modernin lähestymistavan datan integraatioon, kun taas ETL edustaa perinteisempää lähestymistapaa, joka oli suosittu aikana, jolloin tietovarastot olivat pienempiä ja tietojen tallentaminen kalliimpaa.

2.8 Työkalut

Tässä osiossa keskitytään kolmeen merkittävään työkaluun, jotka ovat olennaisia keskitetyn tietovaraston kannalta: Airbyte, Docker ja Kubernetes. Nämä työkalut yhdessä mahdollistavat tehokkaan ja joustavan tavan käsitellä, siirtää, hallita ja skaalata tietovarastoa keskitetyssä ympäristössä.

2.8.1 Airbyte

Airbyte on avoimen lähdekoodin tietojen integraatioalusta, joka mahdollistaa tietojen replikoinnin eri lähteistä tietovarastoihin, datajärviin ja muihin kohdepalveluihin. (Petrella, 2023)

Nykyään dataa voi virrata lukuisista järjestelmistä, kuten sisäisistä tietokannoista, ulkoisista tietolähteistä, SaaS-tuotteista ja API-liittymistä. Tämän monimuotoisen datan integroitiin Airbyte on kehittänyt Connector Development KIT (CDK)-nimisen ohjelmistokehyksen, joka tarjoaa käyttäjille mahdollisuuden luoda omia liitoksia eri sovellusten välille. (Bitstrapped, 2023)

Airbyte käyttää Docker-kontteja datalähteidensä hallintaan, jolloin jokainen datalähde operoi omassa yksilöllisessä kontissaan. Tämä mahdollistaa datalähteiden helpon seurannan, päivittämisen ja ajoittamisen. Lisäksi konttiteknologian avulla jokainen lähde voi suorittaa itsenäistä datan siirtoa ilman, että se häiritsee muita lähteitä. (Bitstrapped, 2023)

Toisin kuin monet perinteiset integraatiotyökalut, jotka voivat olla kalliita ylläpitää ja vaativat paljon koodia sekä kehittäjien jatkuvaa huomiota, Airbyte tarjoaa kustannustehokkaan ja käyttäjäystävällisen vaihtoehdon. Airbyten avulla käyttäjät voivat pienellä alkupanostuksella luoda vankan perustan datan integraatiolle, jonka jälkeen he voivat keskittyä tiedon siirtoon ja eri järjestelmien väliseen yhdistämiseen. Airbyten joustavuus, erityisesti yhteyksien määrän rajoittamattomuus, mahdollistaa sen, että käyttäjät voivat suunnata huomionsa keskeisiin toimintoihin ilman huolta suurista ylläpitokustannuksista. (Bitstrapped, 2023)

2.8.2 Docker

Docker on avoimen lähdekoodin konttienhallinta-alusta, joka mahdollistaa sovellusten tehokkaan kehittämisen, suorittamisen ja siirtämisen. (Docker, 2023)

Docker eristää palvelut isäntäpalvelimesta ja suorittaa ne Docker-konttien sisällä. Docker-kontit ovat itsenäisiä yksiköitä, jotka on eristetty isäntäpalvelimesta ja sisältävät sovelluksen sekä kaikki siihen liittyvät riippuvuudet, kuten kirjastot ja asetukset. Tämä eristetty rakenne mahdollistaa sen, että sovellukset toimivat yhdenmukaisesti eri ympäristöissä, mikä vähentää yhteensopivuusongelmia ja helpottaa kehitys- ja siirtymisprosessia eri laitteelta toiselle. (Docker, 2023)

2.8.3 Docker kontit

Konttien avulla Docker mahdollistaa nopeamman ja helpomman ohjelmistokehitysprosessin. Ohjelmistokehittäjät voivat luoda, testata ja pakata sovelluksia kontteihin, ja nämä kontit voidaan siirtää ja suorittaa missä tahansa Docker-yhteensopivassa ympäristössä. Tämä tehostaa kehitystyötä, parantaa sovellusten siirtymistä testaus- ja tuotantoympäristöihin sekä edistää DevOps-käytäntöjä ja CI/CD prosesseja. (Docker, 2023)

Dockerin yksinkertainen ja tehokas toimintamalli tekee siitä suositun kehitysympäristön ja helpottaa sovellusten siirtymistä eri vaiheista, kuten kehityksestä testaukseen ja tuotantoon. Dockerin konttiteknologiaa voidaan suorittaa useissa eri ympäristöissä, mukaan lukien paikallisesti omalla tietokoneellasi, virtuaalikoneissa tai pilvipalveluissa. Docker tarjoaa kätevän tavan skaalata tai purkaa sovelluksia ja palveluita tarpeen mukaan, käytännössä reaaliajassa. (Docker, 2023)

2.8.4 Docker image

Docker image on kevyt, itsenäinen paketti, joka pitää sisällään sisällä olevan sovelluksen kaikki tarvittavat asiat, kuten ohjelmistokoodin, asetukset, kirjastot ja riippuvuudet. Image on käytännössä kuvaus siitä, miten sovellus tulisi suorittaa ympäristössä, jossa Docker-kontit ajetaan. Imaget mahdollistavat sovellusten ajon ympäristöstä riippumatta. (GeeksforGeeks, 2023a)

Kun luodaan Docker-image, se yleensä perustuu toiseen Docker-imageen, joka toimii perustana. Tämä alkuperäinen-image voi sisältää esimerkiksi käyttöjärjestelmän, kuten Ubuntun. Sen päälle voidaan lisätä tarvittavat sovellukset, kuten PostgreSQL. Näin vanhasta imagesta luodaan uusi versio, joka sisältää sekä käyttöjärjestelmän että lisätyt sovellukset. Tämä mahdollistaa tehokkaan ja toistettavan tavan pakata ja jakaa sovelluksia eri ympäristöissä sekä hallita niiden versioita ja riippuvuuksia. (GeeksforGeeks, 2023a)

Docker-imagea voidaan kuvitella kuin CD-levynä, joka sisältää tietoa. Tämä CD-levy asetetaan tietokoneeseen, ja tietokone voi asentaa siitä esimerkiksi käyttöjärjestelmän tai muita tietoja.

2.8.5 Kubernetes

Kubernetes on Googlen luoma avoimen lähdekoodin kontinhallinta-alusta, joka hallitsee kontti orkestereita. (Edgeworth, Gooley & Rios, 2018)

Konttien orkestrointi on konttien elinkaaren hallintaprosessi. Suurissa todellisissa pilvipalveluissa on lähes mahdotonta hallita kontteja manuaalisesti, joten konttien orkestrointi järjestelmä on olennainen osa pilvipohjaisen arkkitehtuurin rakentamista. (Indrasiri & Suhothayan, 2021)

Konttien orkestrointijärjestelmän tärkeimpiä ominaisuuksia ovat automaattinen hallinta, joka varaa automaattisesti kontteja käyttöön tarpeen mukaan. Lisäksi järjestelmä tarjoaa korkean saatavuuden, sillä se korjaa epäonnistuneet kontit automaattisesti ja asettaa tilalle uudet kontit.

Skaalautuvuus on toinen tärkeä ominaisuus, jonka avulla konttien määrää voidaan tarvittaessa lisätä tai vähentää sovelluksen skaalautuvuuden säätämiseksi. Järjestelmä myös jakaa resurssit konttien kesken tasaisesti, mikä mahdollistaa tehokkaan resurssienhallinnan. Lisäksi konttien palvelurajapinnat ja kuormituksen tasapainotus mahdollistavat konttien altistamisen ulkoisille järjestelmille ja takaavat kuormanhallinnan konteissa. (Indrasiri & Suhothayan, 2021)

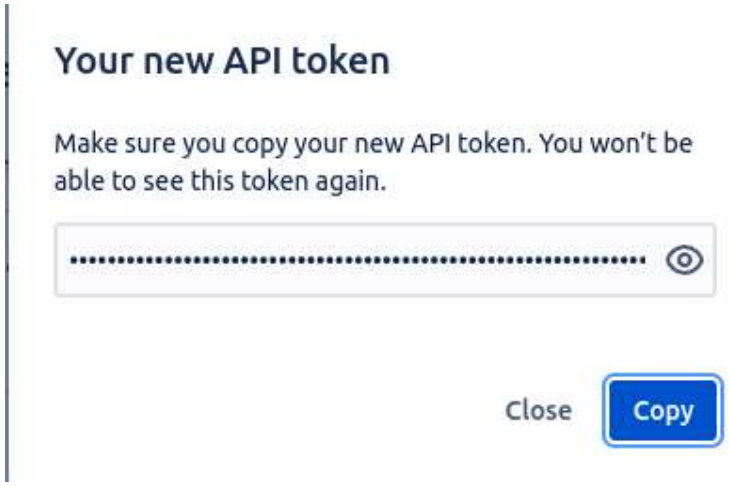
Kubernetes-klusteri koostuu useista pienemmistä palasista, joita kutsutaan solmuiksi tai palvelimiksi. Nämä solmut ovat fyysisiä tai virtuaalisia koneita, jotka toimivat yhdessä muodostaen klusterin. Klusterin solmut jakavat työkuormaa ja vastaavat konttien suorittamisesta. Ne varmistavat, että sovellukset ja palvelut toimivat tasaisesti ja luotettavasti. (Venema, 2020)

Trineria käyttää aktiivisesti Kubernetes-klustereita projekteissaan, ja myös keskitetty tietovarasto on tulevaisuudessa siirtymässä Kubernetes-ympäristöön, mutta se ei kuulu tämän projektin laajuuteen.

3 NOUDETTAVAT TIEDOT

Tietovaraston rakentaminen vaatii selkeää ymmärrystä siitä, mitkä palvelut toimivat tietolähteinä ja millaista tietoa niistä aiotaan kerätä. Tiedon hakutapa vaihtelee riippuen käytettävästä palvelusta.

Tietyissä palveluissa tiedon hankkiminen on yksinkertaista API-avaimen avulla. Kuvassa (Kuva 9) näkyy esimerkki Jira-palvelun API-avaimen hankkimisesta. Tämän avaimen avulla saadaan pääsy Jira-palvelussa oleviin tietoihin ja niiden siirto onnistuu vaivattomasti Airbyte-palveluun.



Kuva 9. Uuden API-avaimen hakeminen Jira-palvelusta.

3.1 CRM

CRM (Customer Relationship Management) on keskitetty alusta, joka auttaa yrityksiä hallitsemaan asiakassuhteitaan. Sen avulla yritykset voivat seurata potentiaalisia asiakkaita ja luoda tietokannan, joka tallentaa asiakkaiden toimintaa. Tämä tarjoaa yritykselle selkeän käsityksen siitä, missä vaiheessa ostoprosessia kukin asiakas on. Trinerialla käytetään Hubspotin CRM-alustaa asiakkuudenhallintaan. Asiakkuudenhallinta on kriittinen osa useiden projektien elinkaarta, ja sen yhdistäminen projekteihin liittyvään dataan, jota Trineria hallinnoi Jirassa, on välttämätöntä. (Hubspot, 2023)

3.2 Jira

Jira on Atlassianin kehittämä ketterään projektinhallintaan, erityisesti ohjelmistokehityksen tarpeisiin suunniteltu työkalu. Se tarjoaa kattavan alustan ohjelmistokehittäjille ja tiimeille, edistäen tehokkaampaa kehitysprosessia. Jiran ydinominaisuuksiin kuuluvat tehtävien seuranta ja monipuolinen projektinhallinta. (Atlassian, 2023a)

Jira integroi Scrum- ja Kanban-menetelmät tarjoamalla dynaamisia ketteriä tauluja, jotka tekevät projektinhallinnasta ja työnkulun visualisoinnista selkeämpää. Näiden taulujen ansiosta tiimit voivat tehokkaasti jakaa tehtäviä, seurata projektien edistymistä ja asettaa tehtäville prioriteetteja. (Atlassian, 2023a)

3.3 Tempo

Tempo on Atlassianin sertifioiman kumppanin kehittämä Jiraan integroitava lisäosa, tarjoaa tehokkaan ratkaisun projektikustannusten hallintaan ja resurssien suunnitteluun automatisoidun ajanseurannan kautta. Työntekijät voivat kirjata työtuntejaan ja tehtäviään Tempoon, mikä antaa reaaliaikaista tietoa työnkulusta ja edesauttaa projektikustannusten kontrollointia.

Lisäksi Tempo tarjoaa käyttäjäystävällisen käyttöliittymän ja monipuoliset raportointiominaisuudet, jotka mahdollistavat selkeän yhteenvedon työntekijöiden ja projektien ajankäytöstä. Tämä on arvokasta päätöksenteon ja resurssien tehokkaan käytön kannalta. (Atlassian, 2023b)

3.4 Visma Fivaldi

Visma Fivaldi on Visman kehittämä taloushallinto-ohjelmisto, joka on kehitetty erityisesti pienten ja keskisuurten yritysten tarpeisiin. Se tarjoaa kattavan valikoiman toimintoja, jotka yksinkertaistavat ja automatisoivat taloushallinnon

prosesseja, mahdollistaen yritysten keskittymisen ydinliiketoimintaansa. (Visma, 2023)

Tämän sähköisen taloudenhallintajärjestelmän avulla yritykset voivat hallinnoida kirjanpitoa, laskutusta, maksuliikennettä ja palkanlaskentaa yhdestä keskitetystä paikasta. (Visma, 2023)

Taloushallinto-ohjelmiston käyttöönotto ei ainoastaan nopeuta kirjanpidon rutiineja, vaan myös vapauttaa arvokasta aikaa muille liiketoiminnan osa-alueille, kuten strategiseen suunnitteluun ja talouden kehittämiseen. Ajantasaiset ja tarkat taloustiedot ovat elintärkeitä luotettavien ennusteiden laatimiselle, jotka ovat perusta informoiduille päätöksille ja edistävät yrityksen pitkäaikaista kehitystä. (Visma, 2023)

4 KÄYTÄNNÖN TOTEUTUS

Tässä luvussa esitellään perusteellisesti projektin perustiedot, vaatimukset ja aikataulu. Aloitus tapahtuu toimeksiantajan esittelyllä, jonka jälkeen syvennytään yhdessä asetettuihin tavoitteisiin.

4.1 Toimeksianto ja tavoitteet

Toimeksiantona on kehittää kevyt tietovarasto, selvittää sen vaatimukset, luoda prototyyppi ja arvioida sen hyödyllisyyttä jatkokehityksen näkökulmasta.

Projektin motiivina on Trinerian kohtaama haaste: arvokkaan tiedon hajautuminen eri järjestelmiin, mikä vaikeuttaa liiketoiminnan ennustettavuutta. Tiedon yhdistäminen on olennainen tavoite, mutta nykytilanteessa se on monimutkaista, sillä tiedot ovat jakautuneet useisiin järjestelmiin, joista osa ei tue tiedonsiirtoa rajapintojen kautta.

Tietovaraston avulla pyritään luomaan yhtenäinen kokonaisnäkymä, joka tukisi tiedolla johtamista yrityksessä. Esimerkiksi Jirasta ja Temposta saadaan projektien aikatietoja, kun taas laskutustiedot ovat Fivaldissa ja asiakastiedot CRM-järjestelmässä.

Tällä hetkellä ei ole olemassa automatisoitua menetelmää näiden tietojen yhdistämiseen ja esimerkiksi projektien katteiden laskemiseen. Tämän prosessin automatisointi ja hallinnan yksinkertaistaminen olisi merkittävä askel yrityksen liiketoiminnan kehittämisessä.

4.2 Lähtötilanne

Trinerialla on olemassa kaikki järjestelmät, joissa data on saatavilla, ja tietyille yleisimmin käytetyille ratkaisuille on omat työkalut niiden yhdistämiseksi.

Esimerkiksi tuntikirjausten laskemiseen ja liukumien hakemiseen on olemassa omat ratkaisunsa. Tietovarastoratkaisuja on kartoitettu ja testattu, ja yrityksellä

on perusymmärrys tietovarastoista ja eri ratkaisuvaihtoehdoista. Kuitenkin systemaattinen selvitystyö puuttuu eikä tarkkaa tietoa siitä, miten asiat kannattaisi toteuttaa, ole vielä olemassa.

Vaikka yritys onkin vielä pieni ja nykyisellä ratkaisulla selvitään, se haluaa toimia ennakoivasti ja kehittää ratkaisun, joka palvelee myös tulevaisuudessa.

4.3 Toiminnalliset vaatimukset

Kehitettävän järjestelmän tulee olla monipuolinen ja turvallinen, jotta se voi palvella Trinerian tarpeita tehokkaasti. Järjestelmän on kyettävä käsittelemään sekä käyttäjän manuaalisesti käynnistämiä toimintoja että automatisoituja prosesseja, jotka aktivoituvat määriteltyjen ehtojen perusteella. Tämä mahdollistaa monipuolisen ja joustavan datan hankinnan eri lähteistä.

Järjestelmän on tarjottava täydet CRUD-toiminnot (Create, Read, Update, Delete) datan elinkaaren hallinnan tukemiseksi. Tämä mahdollistaa käyttäjille uusien tietueiden luomisen, olemassa olevien tietojen haun ja päivityksen sekä tarpeettomaksi käyneiden tietojen poistamisen.

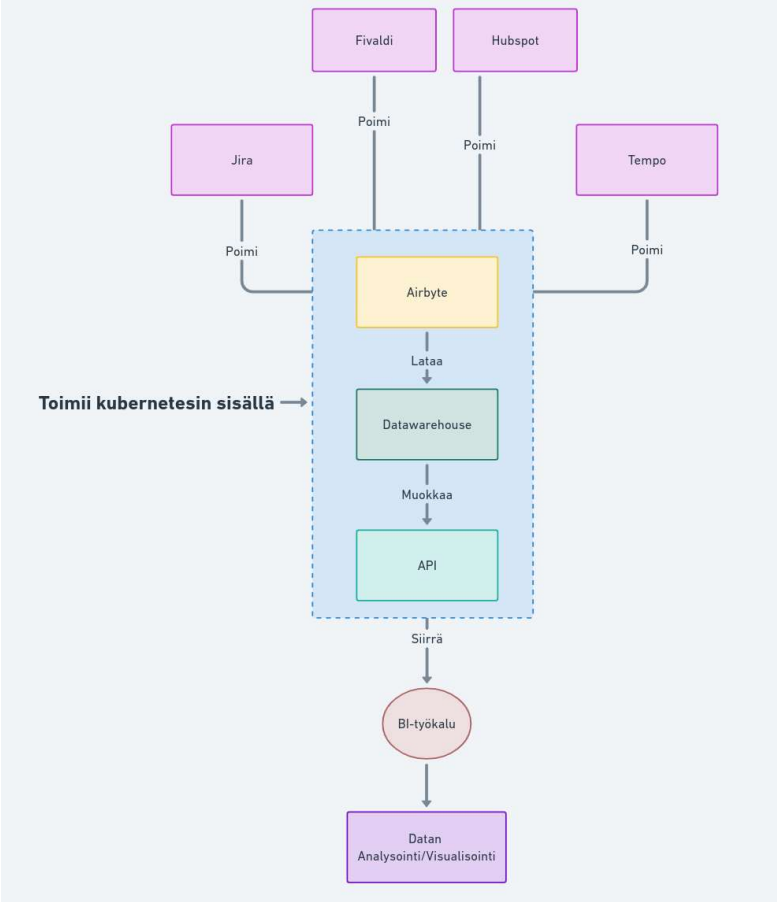
Tiedonsiirto järjestelmässä on toteutettu yksisuuntaisena, mikä yksinkertaistaa integraatiota ja vähentää monimutkaisten tietojen synkronointimekanismien tarvetta. Järjestelmän on myös sovelluttava sujuvasti Kubernetes-klusteriympäristöön, mikä varmistaa sen skaalautuvuuden ja ylläpidettävyyden.

Järjestelmän on noudatettava alan tietoturvastandardeja, mikä sisältää avainten ja konfiguraatioiden suojaamisen ulkopuolisten pääsyltä. Suojauksen toteuttaminen ympäristömuuttujien avulla mahdollistaa herkkien tietojen, kuten API- ja tietokanta-avainten, turvallisen syöttämisen järjestelmään ulkoa. Herkkien tietojen suoran koodaamisen välttäminen lähdekoodiin pienentää tietoturvariskejä ja takaa arkaluonteisten tietojen luotettavan suojelun.

Kaikki Airbyte-integraatiot on testattava automaattisesti varmistaen niiden toimivuus ja luotettavuus. Lisäksi integraatioiden dokumentointi on tärkeää, jotta niiden käyttö ja mahdolliset muutostarpeet ovat selkeitä kaikille osapuolille.

4.4 Arkkitehtuurimalli

Ennen toteutuksen aloittamista olen luonut arkkitehtuurimallin (Kuva 10), joka kuvastaa miten data kulkee eri lähteistä BI-työkaluun. Tämä malli on suunniteltu varmistamaan, että datan kerääminen, käsittely ja analysointi ovat tehokkaita ja sujuvia.



Kuva 10. Arkkitehtuurimalli toteutuksesta.

Datan lähteinä toimivat Jira, Tempo, Fivaldi ja Hubspot. Näistä järjestelmistä data noudetaan käyttäen Airbyteä, joka toimii datan siirtämisen välineenä. Airbyte luo

yhteyden keskitettyyn tietovarastoon, joka vastaa datan tallentamisesta ja tietueiden luomisesta.

Kun data on siirretty tietovarastoon, seuraava vaihe on datan muokkaaminen API:n avulla. API toimii välikätenä, joka muuntaa datan sopivaan muotoon BI-työkalun käyttöä varten. Tämä mahdollistaa datan siirtämisen ja muuntamisen tehokkaasti, varmistaen, että se on oikeassa muodossa analysointia ja visualisointia varten.

Viimeinen vaihe on datan siirtäminen BI-työkaluun, jossa sitä voidaan käyttää analysointiin ja visualisointiin. Tämä mahdollistaa Trinerialle tärkeiden tietojen tehokkaan hyödyntämisen, tarjoten näkemyksiä ja analytiikkaa, jotka auttavat päätöksenteossa ja liiketoiminnan kehittämisessä.

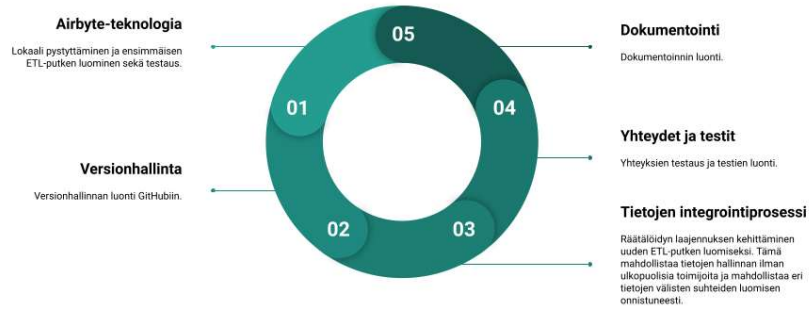
Tiedonsiirto toteutetaan Kubernetes-klusterin sisällä, joka hallitsee kontteja. Tämä konttien hallinta on keskeinen osa arkkitehtuuria, sillä se mahdollistaa eri komponenttien saumattoman yhteistyön ja varmistaa, että datan käsittely ja hallinta tapahtuvat sujuvasti ja luotettavasti.

4.5 Projektin suunnittelu

Projektin suunnittelu alkoi kesäkuussa 2023, jolloin määritettiin lähtötilanne sekä aikataulun. Projektin onnistumisen kannalta keskeistä on kehittää MVP (Minimum Viable Product), eli ohjelmiston ensimmäinen toimiva versio, joka on valmis otettavaksi käyttöön.

MVP sisältää ELT-prosessin, jossa Airbyteä hyödyntämällä tiedot siirretään ensimmäisestä lähteestä keskitettyyn tietovarastoon. Lisäksi luodaan toimiva CI/CD-putki, testit, dokumentaatio sekä osoitus datan oikeasta tallennuksesta oikeassa muodossa.

Alla olevan aikajanan (Kuva 11) ideana on näyttää projektin eri vaiheet siihen pisteeseen asti, kunnes ensimmäinen yhteys on luotu ja dokumentoitu.



Kuva 11. Aikajana projektin eri vaiheista.

Projektin ensimmäinen vaihe liittyy Airbyte-teknologian lokaaliin pystyttämiseen, ensimmäisen ELT-prosessin luomiseen sekä sen testaamiseen. Seuraavaksi luodaan versionhallinta GitHubiin, mikä auttaa seuraamaan ja hallitsemaan projektin kehitystä.

Kolmas vaihe käsittää räätälöidyn lisäosan kehittämisen TypeScriptillä uuden ELT-prosessin tarpeisiin. Tämä lisäosa on suunniteltu erityisesti datan siirtoon tietovarastoon, mikä mahdollistaa datan hallinnan ilman ulkopuolisia toimijoita ja tukee eri datalähteiden välillä luotavien relaatioiden muodostamista.

Neljäntenä vaiheena on yhteyksien testaus ja testien luonti, jotta voidaan varmistaa, että tietojen siirto toimii odotetusti. Viimeisenä vaiheena on dokumentoinnin luonti, mikä on tärkeää projektin ymmärrettävyyden ja ylläpidettävyyden kannalta.

4.6 Testaus

Testaaminen on keskeinen osa ohjelmistokehitysprosessia ja sen avulla varmistetaan, että järjestelmä toimii suunnitellusti, mukaan lukien sen eri toiminnot ja virheenkäsittely. Automaattinen testaus on erityisen tärkeää, sillä se vähentää inhimillisten virheiden riskiä ja takaa, että järjestelmä toimii odotetulla tavalla ilman ihmisen jatkuvaa seurantaa.

Tässä toteutuksessa luodaan yksikkötestejä, jotka keskittyvät erityisesti tietokantayhteyksien onnistuneeseen muodostamiseen ja datan oikeelliseen siirtymiseen tietokantaan. Yksikkötestaus on keskeinen osa laadunvarmistusta, ja se kohdistuu ohjelmiston pienimpiin yksiköihin, kuten funktioihin, metodeihin ja luokkiin. Tämä eristetty testausympäristö mahdollistaa jokaisen komponentin toimivuuden varmistamisen erikseen. (GeeksForGeeks, 2023b)

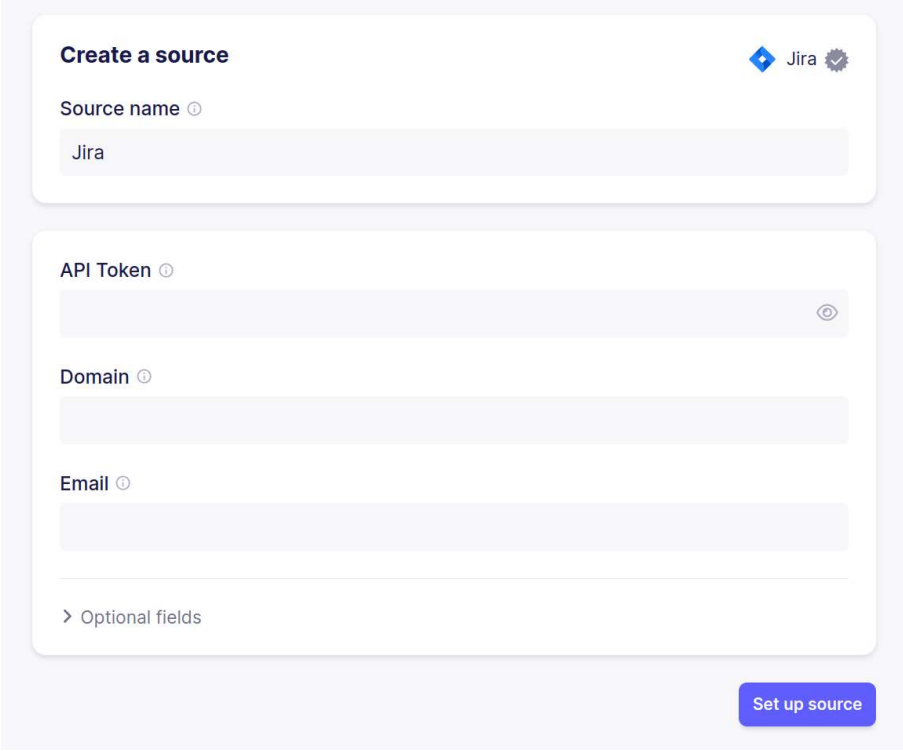
Yksikkötestien toteutuksessa käytetään Jest-testauskirjastoa, joka on laajalti suosittu JavaScript- ja TypeScript-ympäristöissä. Jest tarjoaa tehokkaat työkalut testien kirjoittamiseen ja suorittamiseen, mikä tekee siitä ihanteellisen valinnan räätälöidyn lisäosan testaamiseen. Testauksen avulla varmistetaan, että jokainen koodin osa toimii odotetusti, mikä on välttämätöntä ohjelmiston luotettavuuden ja tehokkuuden kannalta. (Jest, 2023)

5 KEHITYSPROSESSI

Tässä kappaleessa keskitytään keskitetyn tietovaraston kehitysprosessin eri vaiheisiin. Ensin esitellään, miten yhteys luodaan käyttämällä ainoastaan Airbyten käyttöliittymää, ilman tarvetta koodaustaidoille. Tämän jälkeen tarkastellaan, kuinka kehitettyä omaa työkalua voidaan hyödyntää tietokannan hallinnassa, tuoden lisäjoustavuutta ja mukautuvuutta prosessiin.

5.1 Uuden Airbyte-yhteyden luominen

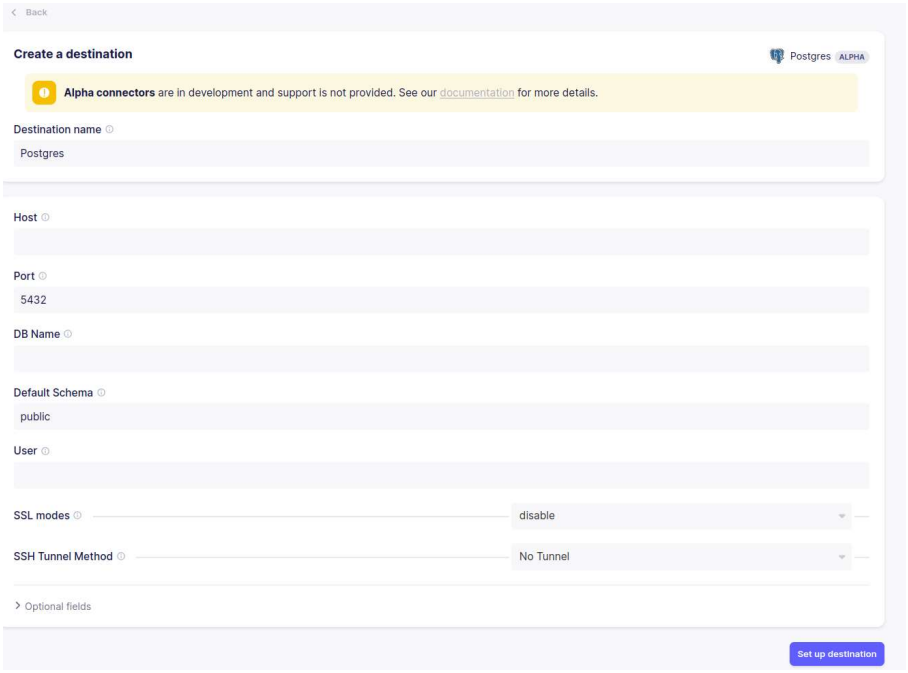
Airbyte tarjoaa modernin käyttöliittymän, jossa uusien yhteyksien luominen onnistuu vaivattomasti (Kuva 12). Uuden yhteyden luominen alkaa datan lähteen (Source) valinnalla ja vaadittavien esitietojen määrittelyllä. Nämä tiedot, kuten API-avain ja käyttäjätunnus, vaihtelevat lähteen mukaan ja ovat välttämättömiä ulkoisen palvelun kanssa yhteyden muodostamiseksi.



Kuva 12. Uuden lähteen luominen Airbyten käyttöliittymässä.

Palvelu suorittaa testin varmistaakseen yhteyden muodostumisen ja ilmoittaa käyttöliittymässä, mikäli yhteys onnistuu. Tämän jälkeen siirrytään luomaan datan määränpää (Destination), joka tässä tapauksessa on PostgreSQL-tietokanta (Kuva 13).

Määränpään määrittely noudattaa samaa kaavaa kuin lähteen luominen, mutta vaatii lisätietojen täyttämistä. Tässä vaiheessa luodaan myös tarvittava tietokanta ja sen käyttäjät, jotta palvelulla on oikeudet siirtää tietoa tietokantaan.



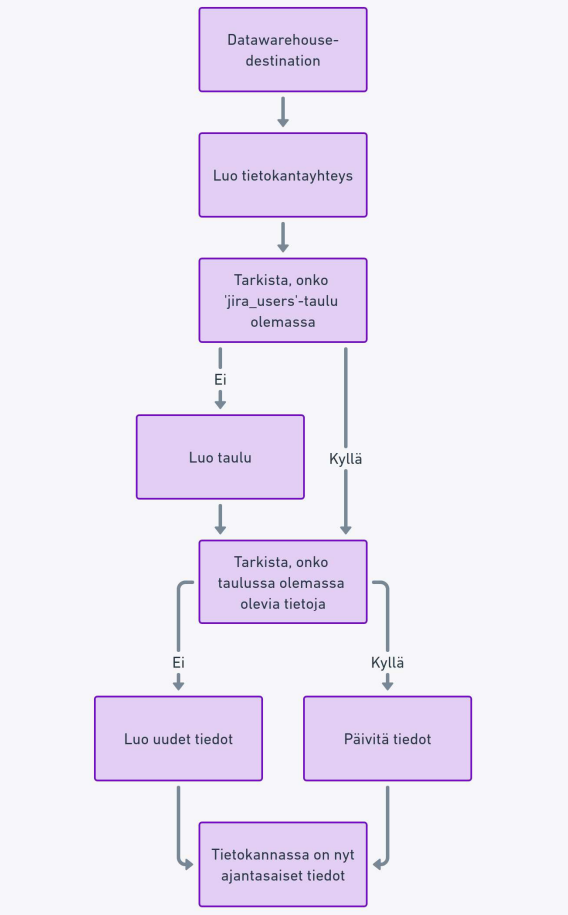
Kuva 13. Uuden määränpään luominen Airbyten käyttöliittymässä.

Näin on luotu onnistunut tiedonsiirtoputki, jonka avulla tietoa voidaan siirtää paikasta A paikkaan B.

5.2 TypeScript-työkalun hyödyntäminen tiedonsiirron tehostamisessa

Tehokkaamman tietojen hallinnan mahdollistamiseksi kehitettiin TypeScriptillä oma räätälöity lisäosa. Tämän lisäosan ansiosta on mahdollista paremmin ohjata ja valita, mitä tietoja siirretään tietovarastoon.

Esimerkkinä käydään läpi, miten käyttäjien tiedot siirtyvät Jirasta keskitettyyn tietokantaan (Kuva 14).



Kuva 14. Esimerkki käyttäjien tietojensiirrosta keskitettyyn tietovarastoon.

5.2.1 Tietokanta yhteyden muodostaminen

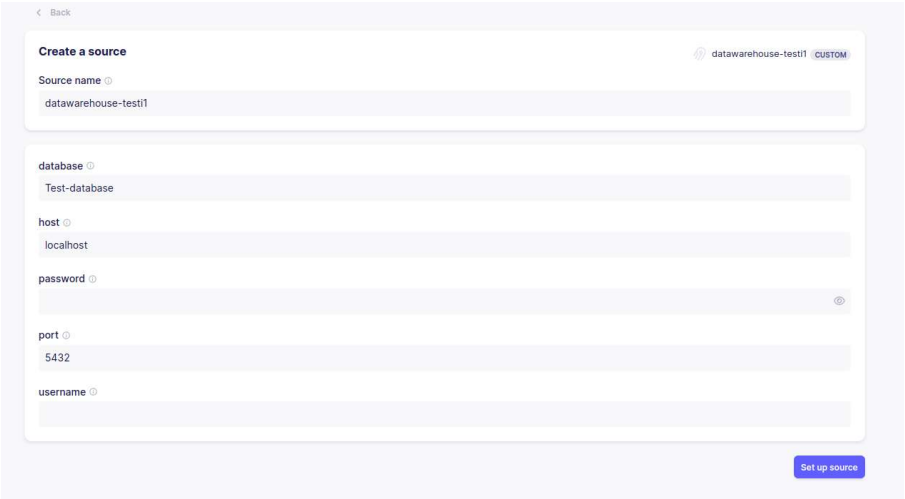
Tietokantayhteyden luominen PostgreSQL-tietokantaan onnistuu tehokkaasti Node.js-ympäristössä käyttämällä node-postgres-kirjastoa. Tämä kirjasto tarjoaa valmiit työkalut PostgreSQL-tietokannan asiakasohjelman (client) käyttöön, kuten Kuva 15 havainnollistaa.

```
async getDatabaseConnection(): Promise<Client> {
    const dbconfig = {
        host: this.host,
        port: this.port,
        database: this.database,
        user: this.user,
        password: this.password,
    };

    const client = new Client(dbconfig);
    try {
        await client.connect();
        console.log("getDatabaseConnection is working");
    } catch (e) {
        throw new Error(`Failed to connect to the database: ${e}`);
    }

    return client;
}
```

Kuva 15. Tietokantayhteyden muodostaminen.

Node-postgres-clientin avulla voidaan luoda ja ylläpitää tietokantayhteyttä Node.js-ympäristössä tehokkaasti ja luotettavasti. Tämän clientin käyttö on keskeistä yhteyden muodostamisessa, mikä toteutetaan getDatabaseConnection-metodilla. Tämä vaihe on välttämätön Jirasta saatavien tietojen siirtämiseksi tietokantaamme, ja sen onnistuminen mahdollistaa Jiran käyttäjätietojen siirron sekä päivityksen keskitettyyn tietokantaamme.

Airbyten käyttöliittymässä käyttäjät syöttävät tarvittavat tiedot – isännän, portin, tietokannan nimen, käyttäjätunnuksen ja salasanan – luodakseen yhteyden tietokantaan (Kuva 16). Tämän yhteyden luominen on kriittinen askel, joka mahdollistaa suoran pääsyn tietokantaan ja tietojen käsittelyn Jira-yhteyden kautta.



Kuva 16. Kustomoidun lähteen luominen Airbyten käyttöliittymässä.

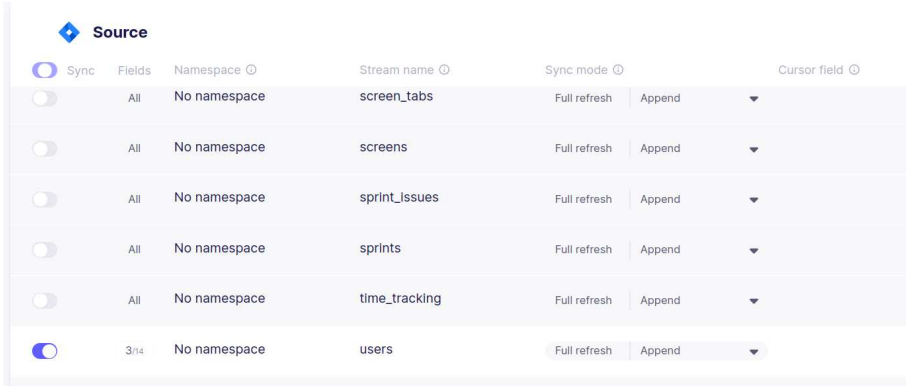
5.2.2 Taulujen luonti

Tietokantayhteyden onnistuneen muodostamisen jälkeen luodaan tarvittavat taulut SQL-kyselyjen avulla. Luodaan taulu jira_users, joka on suunniteltu tallentamaan jira-palvelun käyttäjätietoja (Kuva 17).

```
export async function createJiraUsersTable(client: Client): Promise<void> {
    const createTableQuery = `
    CREATE TABLE IF NOT EXISTS jira_users(
        accountId VARCHAR(255) PRIMARY KEY,
        displayName VARCHAR(255),
        emailAddress VARCHAR(255)
    )
    `;
    await client.query(createTableQuery);
}
```

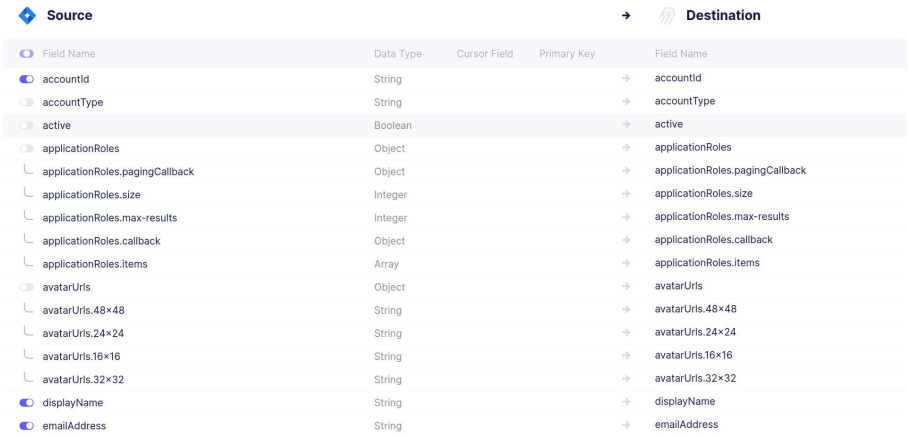
Kuva 17. Luodaan jira_users-taulu.

Tietokantaan tarvittavat tiedot haetaan Jira-yhteyden avulla, joka on konfiguroitu Airbytessä. Airbyten käyttöliittymän kautta on mahdollista valita, mitä tietoja halutaan synkronoida. Tässä prototyypissä (Kuva 18) käyttäjän on luotava yhteys valitsemalla haluamansa tietovirrat käyttöliittymän avulla.



Kuva 18. Käyttäjä valitsee haluamansa tietovirrat käyttöliittymän avulla.

Tämän jälkeen käyttäjän on vielä valittava, mitä kenttiä tietovirta sisältää (Kuva 19), koska ohjelma ei tunnista automaattisesti koodiin kirjoitettuja sääntöjä, kuten tietovirran tiettyjen taulujen valitsemista. Tämä on toteutuksessa ilmenevä puute, joka vaatii myöhempiä muutoksia.



Kuva 19. Käyttäjä valitsee tietovirtojen sisällön käyttöliittymän avulla.

5.2.3 Taulujen datan luonti ja päivittäminen

AddJiraUserRecord-metodia (Kuva 20) käytetään käyttäjätietojen lähettämiseen eteenpäin JiraUserData-tyyppinä (Kuva 21) ja tietokannan päivittämiseen uusilla arvoilla jira_users-taulussa (Kuva 22). SQL-kyselyssä on huomioitu myös tietojen päivitys: jos tietokannassa on jo tietue samalla accountId:llä, kysely päivittää olemassa olevan tietueen displayName ja emailAddress-sarakkeet uusilla arvoilla.

```
private async addJiraUserRecord(unpacked: AirbyteRecord): Promise<void> {
    const user = unpacked.record.data as JiraUserData
    if (!this.userRepository) {
        console.error("UserRepository not initialized.");
        throw new Error("UserRepository not initialized.");
    }
    await this.userRepository.insertUserRecord(user)
}
```

Kuva 20. Tarkistetaan UserRepositoryn olemassaolo.

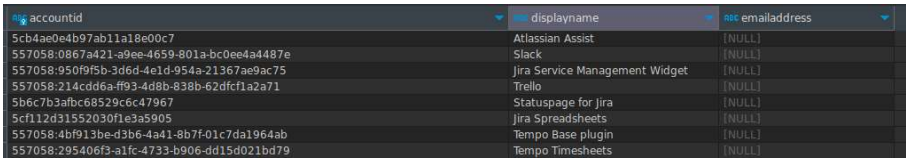
```
export type JiraUserData = {
    "accountId": string,
    "displayName": string
    "emailAddress": string | null,
}
```

Kuva 21. User-tietotyypin määritys.

```
async insertUserRecord(user: JiraUserData): Promise<void> {
    try {
        const newUser = await this.client.query(`
        INSERT INTO jira_users(accountId, displayName, emailAddress)
        VALUES($1, $2, $3)
        ON CONFLICT (accountId) DO UPDATE
        SET displayName = $2,
            emailAddress = $3;
        `, [user.accountId, user.displayName, user.emailAddress])

        if (newUser.rowCount === 0) {
            throw new Error("Failed to insert user record: User record not inserted");
        }
    } catch (error) {
        throw new Error(`Failed to insert user record: ${error}`);
    }
  }
}
```

Kuva 22. Lisätään uusi tietue jira_users-tauluun.

Tietojen siirtojen onnistuessa tiedot tallentuvat tietokantaan (Kuva 23). Tässä esimerkissä siirrettävät tiedot eivät sisältäneet sähköpostiosoitetta, minkä vuoksi emailaddress-rivillä näkyy arvo "NULL".



accountid	displayname	emailaddress
5cb4ae0e4b97ab11a18e00c7	Atlassian Assist	[NULL]
557058:0867a421-a9ee-4659-801a-bc0ee4a4487e	Slack	[NULL]
557058:950f9f5b-3d6d-4e1d-954a-21367ae9ac75	Jira Service Management Widget	[NULL]
557058:214cdd6a-ff93-4d8b-838b-62dfcf1a2a71	Trello	[NULL]
5b6c7b3afbc68529c6c47967	Statuspage for Jira	[NULL]
5cf112d31552030f1e3a5905	Jira Spreadsheets	[NULL]
557058:4bf913be-d3b6-4a41-8b7f-01c7da1964ab	Tempo Base plugin	[NULL]
557058:295406f3-a1fc-4733-b906-dd15d021bd79	Tempo Timesheets	[NULL]

Kuva 23. Tietojensiirto onnistui ja tiedot näkyvät tietokannassa.

6 POHDINTA

Opinnäytetyön tavoitteena oli rakentaa keskitetty tietovarasto Trineria Oy:lle, mikä onnistui hyvin ja odotusten mukaisesti, mahdollistaen toimivan minimivaatimukset täyttävän tuotteen (MVP) luomisen. Lopputulokseen oltiin tyytyväisiä, eikä kehitystyön aikana kohdattu suurempia ongelmia. Näin ollen voidaan todeta opinnäytetyön onnistuneen ja täyttäneen sille asetetut vaatimukset.

Opinnäytetyöni aikana sain arvokasta kokemusta ja kehitin osaamistani monilla alueilla, kuten tietovarastojen rakentamisessa, integraatioprosesseissa, testauksessa sekä liiketoiminnan näkökulmasta. Olen erittäin kiitollinen mahdollisuudesta toteuttaa tämä opinnäytetyö kehitysprojektina Trineria Oy:lle, mikä antoi minulle syvällisen käsityksen siitä, miten projekteja suunnitellaan ja aikataulutetaan, sekä siitä, mitä tekijöitä tulee ottaa huomioon kehitystyötä tehdessä. Tämän opinnäytetyön kautta opin tietovarastojen keskeisen merkityksen yritysmaailmassa, erityisesti haasteista, jotka liittyvät suurten datamäärien hallintaan ja niiden tehokkaaseen hyödyntämiseen monista eri lähteistä.

Tutkimuskysymyksistä ensimmäinen koski Airbyte-palvelun soveltuvuutta pienen yrityksen tietovaraston tarpeisiin. Tutkimus osoitti Airbyten olevan kustannustehokas ja käyttäjäystävällinen tietojen integraatioalusta. Sen joustavuus, erityisesti kyky käsitellä monenlaisia tietolähteitä ilman suuria ylläpitokustannuksia, tekee siitä ihanteellisen ratkaisun pienille yrityksille. Airbyte tarjoaa pienille yrityksille, kuten Trinerialle, mahdollisuuden hyödyntää tietovarastoja ilman tarvetta suuriin IT-infrastruktuurin investointeihin.

Toinen kysymys käsitteli tietovaraston aiheuttamaa tietojen päällekkäisyyttä. Vaikka tietovarastot tuovat mukanaan tietojen päällekkäisyyttä, se ei ole tarpeetonta. Tietovarastot mahdollistavat tietojen säilyttämisen ja monipuolisten versioiden luomisen BI-sovelluksia varten. Tämä on erityisen tärkeää yrityksille,

jotka haluavat varmistaa jatkuvan pääsyn kriittiseen tietoon ja olla riippumattomia alkuperäisen datan lähteestä.

Kolmas kysymys käsitteli tietovaraston rakentamisen järkevyyttä verrattuna valmiiden palveluiden hyödyntämiseen. Trinerian kaltaisille yrityksille, jotka pyrkivät olemaan alustariippumattomia ja joilla on toistaiseksi pienet tarpeet, voi olla järkevää rakentaa ja ylläpitää omaa tietovarastoratkaisuaan. Yritykselle räätälöity ratkaisu mahdollistaa nopean reagoinnin muuttuviin tarpeisiin ja antaa yritykselle aikaa selvittää tietovarastotarpeensa paremmin. Oman tietovarastoratkaisun ylläpidon ja kehityksen mielekkyyttä on vaikea tässä vaiheessa vielä arvioida. On suotavaa kokeilla ja laajentaa tietovarastoratkaisua sekä seurata kehitykseen kuluvaa aikaa ja ongelmakohtia, jotta pitkän aikavälin kustannukset saadaan selville

7 LÄHTEET

Amazon. 2023a. What is SQL? Viitattu 5.6.2023. https://aws.amazon.com/what-is/sql/

Amazon. 2023b. What's the Difference Between ETL and ELT? Viitattu 9.11.2023. https://aws.amazon.com/compare/the-difference-between-etl-and-elt/

Atlassian. 2023a. Welcome to Jira Software. Viitattu 5.7.2023. https://www.atlassian.com/software/jira/guides/getting-started/introduction#what-is-jira-software

Atlassian. 2023b. Timesheets by Tempo - Jira Time Tracking. Viitattu 5.7.2023. https://marketplace.atlassian.com/apps/6572/timesheets-by-tempo-jira-time-tracking?tab=overview&hosting=cloud

Bitstrapped. 2023. Airbyte: The modern ELT data pipeline. Viitattu 16.6.2023. https://www.bitstrapped.com/blog/airbyte-elt-data-pipleine

Datacamp. 2023. What is ETL? Viitattu 31.7.2023. https://www.datacamp.com/blog/a-list-of-the-16-best-etl-tools-and-why-to-choose-them

Docker. 2023. Docker overview. Viitattu 12.10.2023. https://docs.docker.com/get-started/overview/

Dyer, R. 2015. Learning MySQL and MariaDB. O'Reilly Media, Inc. Viitattu 4.7.2023.

Ferrari, L. & Pirozzi E. 2020. Learn PostgreSQL, Packt Publishing. Viitattu 9.6.2023.

GeeksForGeeks. 2023a. What is Docker Images? Viitattu 3.11.2023. https://www.geeksforgeeks.org/what-is-docker-images/

GeeksForGeeks. 2023b. Unit Testing – Software Testing. Viitattu 14.12.2023. https://www.geeksforgeeks.org/unit-testing-software-testing/

Google Cloud. 2023. What is a relational database? Viitattu 7.2.2024. https://cloud.google.com/learn/what-is-a-relational-database

Hubspot. 2023. Free CRM Software with Something for Everyone. Viitattu 5.7.2023. https://www.hubspot.com/products/crm

IBM. 2023a. What is business intelligence? Viitattu 16.6.2023. https://www.ibm.com/topics/business-intelligence

IBM. 2023b. What is a data warehouse? Viitattu 6.6.2023. https://www.ibm.com/topics/data-warehouse

Indrasiri, K. & Suhothayan, S. 2021. Design Patterns for Cloud Native Applications. O'Reilly Media, Inc. Viitattu 9.6.2023.

Jest. 2023. JavaScript Testing Framework. Viitattu 14.12.2023. https://jestjs.io/

Kenler, E. & Razzoli, F. 2015. MariaDB Essentials. Packt Publishing. Viitattu 4.7.2023.

MongoDB. 2023. What is a Document Database? Viitattu 7.12.2023. https://www.mongodb.com/document-databases

Oracle. 2023a. What is NoSQL? Viitattu 16.6.2023. https://www.oracle.com/database/nosql/what-is-nosql/

Oracle. 2023b. What is a Data Warehouse? Viitattu 9.6.2023. https://www.oracle.com/database/what-is-a-data-warehouse/

Peterson, R. 2023. Entity Relationship (ER) Diagram Model with DBMS Example. Viitattu 7.2.2024. https://www.guru99.com/er-diagram-tutorial-dbms.html

Petrella, A. 2023. Fundamentals of Data Observability. O'Reilly Media, Inc. Viitattu 9.6.2023.

PostgreSQL. 2023. What Is PostgreSQL? Viitattu 6.6.2023. https://www.postgresql.org/docs/current/intro-whatis.html

Rivery. 2023. What is the Difference Between ETL and ELT? Viitattu 20.12.2023. https://rivery.io/blog/etl-vs-elt/

Trineria Oy. 2022. Trineria Oy kotisivut. Viitattu 5.6.2023. https://trineria.fi/yritys/

Venema, W. 2020. Building Serverless Applications with Google Cloud Run. O'Reilly Media, Inc. Viitattu 9.6.2023.

Visma. 2023. Visma Fivaldi. Viitattu 19.7.2023. https://www.visma.fi/visma-fivaldi/