# Datan validointi- ja normalisointimenetelmät modernissa tietovarastoinnissa

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Tietojenkäsittelyn koulutus, Hämeenlinnan korkeakoulukeskus
syksy, 2023

Ari Niemi

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittelyn koulutus Tiivistelmä
Tekijä Ari Niemi Vuosi 2023
Työn nimi Datan validointi- ja normalisointimenetelmät modernissa tietovarastoinnissa
Ohjaajat Lasse Seppänen

---

TIIVISTELMÄ

Opinnäytetyön tarkoituksena oli toteuttaa tietovarastointiprojekti, jossa modernia data-arkkitehtuuria hyödyntäen validoitiin ja normalisoitiin ulkoisen tiedontuottajan toimittamaa yrityksiin ja vastuuhenkilöihin liittyvää dataa, jotta datan pohjalta voitiin muodostaa tietovarastoon rajapintojen ja palvelujen käytössä oleva harmonisoitu taulu. Työn toimeksiantaja oli media-alalla toimiva yritys. Opinnäytetyön tekijä oli mukana projektin suunnittelussa ja toteutuksessa.

Opinnäytetyön tietopohja koostuu tietovarastoinnin peruskäsitteistä sekä pilvipohjaisten tietovarastointiratkaisujen piirteistä, datan laatuun ja laadunhallintaan liittyvistä tekijöistä sekä erilaisten datan validointi- ja normalisointimenetelmien esittelystä. Opinnäytetyö on tyypiltään toiminnallinen. Aineiston keräämisen menetelmä oli kehitysprojekti, joka toteutettiin vuoden 2023 alkupuoliskolla validoimalla tiedontuottajan toimittaman datan laatua ja ominaisuuksia, ja toteuttamalla validoinnin tulosten pohjalta normalisointiin liittyviä toimenpiteitä, joilla datan laatua pyrittiin parantamaan.

Johtopäätöksenä voidaan todeta, että tiedontuottajan toimittama data oli varsin kattavaa ja eheää, mutta se ei täyttänyt kaikkia vaatimuksia. Normalisointitoimenpiteiden myötä dataa saatiin puhdistettua attribuutista riippuen muutamasta sadasta tiedosta useisiin tuhansiin, mikä todettiin tärkeäksi toimeksiantajan luotettavuuden ja laatumielikuvan kannalta.

Avainsanat tietovarasto, data, datan laatu, datan validointi, datan normalisointi
Sivut 45 sivua ja liitteitä 1 sivu

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology **Abstract**

Author Ari Niemi Year 2023

Subject Data validation and normalization methods in modern data warehousing

Supervisors Lasse Seppänen

---

ABSTRACT

The purpose of this thesis was to implement a data warehousing project in which company- and person-related raw data provided by an external data provider was validated and normalized by using modern data architecture. The intention of this was to create a harmonized database table inside the data warehouse for the use of interfaces and services. The client of the thesis was a company operating in the media sector. The author of the thesis was involved in planning and implementation of the project.

The knowledge base of the thesis consists of the basic concepts of data warehousing and cloud-based data warehousing solutions, factors related to data quality and quality management, and the methods of data validation and normalization. The thesis is functional. The method of collecting the data was a development project that was implemented in the first half of 2023 by validating the quality and characteristics of data provided by the data provider, and by normalizing the data based on the results of the validation to improve the quality of the data.

As a conclusion, it can be stated that the data provided by the external data provider was quite comprehensive and complete, but it did not meet all the requirements. With the normalization methods, the number of values that could be cleaned varied from a few hundred up to several thousand, depending on the attribute. This was found to be important for maintaining the client's image of reliability and quality among the customer base.

Keywords data warehouse, data, data quality, data validation, data normalization

Pages 45 pages and appendices 1 page

## Sanasto

| | |
|---|---|
| Airflow | Apachen pilvipalveluissa sijaitsevien tehtävien ajastamiseen ja ajamiseen suunniteltu palvelu |
| API | Application Programming Interface, rajapinta |
| Attribute Domain Constraint | Joukko sääntöjä, joilla rajoitetaan datan saamia arvoja |
| Attribuutti | Jotakin luonnehtiva määrite, esim. väri, pituus tai hinta |
| AWS | Amazon Web Services; Amazonin pilvipalveluita tarjoava tytäryhtiö |
| B2B | Business-To-Business; termi, jolla kuvataan yritysten välistä kaupankäyntiä tai liiketoimintaa |
| BI-työkalu | Business Intelligence -työkalu esim. datan analysointiin ja päätöksenteossa käytettävän tiedon tuottamiseen |
| CHAR | Datan tyyppi; yleensä määrämittainen kenttä, jossa säilötään merkkijonomuotoista tietoa |
| CRM | Customer Relationship Management, asiakkuudenhallintajärjestelmä |
| CSV | Comma-separated values (suom. pilkuilla erotellut arvot), tiedostomuoto taulukkomuotoisen tiedon tallentamiseen |
| Data Cleansing | Datan puhdistaminen; prosessi, jolla raakadata valmistellaan esim. tietovarastoon siirrettäväksi |
| Data Mapping | Datan muuntaminen; tapa, jolla datasyöte muutetaan vastaamaan samaa tietoa toisessa muodossa |
| Data Mart | Otoskanta; tietovaraston osa, joka sisältää yleensä vain tiettyyn tarkoitukseen luodun otteen tietokannasta |
| Data Warehouse | Tietovarasto; tietojärjestelmä, johon voidaan kerätä ja tallentaa suuria määriä erityyppistä tietoa useita eri tarpeita varten |
| DataFrame | Python-ohjelmointikielessä tietorakenne, joka järjestää tiedot laskentataulukon tyyppiseksi rivi- ja saraketaulukoksi |
| Datapiste | Tietojoukosta tunnistettavissa oleva elementti, esim. yksi solu tietokantataulussa tai yksi tieto havainnosta tietyllä ajanhetkellä |
| DATE | Datan tyyppi, jolla voidaan säilöä päivämäärän muodossa olevaa tietoa |

| | |
|---|---|
| DECIMAL | Datan tyyppi, jolla voidaan säilöä desimaalilukumuotoista tietoa |
| Double Pipe -operaattori | Operaattori, jota käytetään SQL-kielessä ketjuttamaan tai yhdistämään merkkijonoja |
| ELT | Extract, Load, Transform; tietovarastoinnissa käytettävä prosessi, jossa tiedot tuodaan tietovarastoon raakamuotoisena ennen muuntamista (vrt. ETL) |
| ERP | Enterprise Resource Planning, toiminnanohjausjärjestelmä |
| ETL | Extract, Transform, Load; tietovarastoinnissa käytettävä prosessi, jossa tiedot poimitaan, muunnetaan ja ladataan tietovarastoon (vrt. ELT) |
| Foreign Key | Viiteavain; relaatiotietokannoissa käytetty tieto, jota käytetään viittaamaan toisessa taulussa olevaan tietueeseen |
| FTP | File Transfer Protocol; tapa ladata, lähettää ja siirtää tiedostoja paikasta toiseen internetissä tietokonejärjestelmien välillä |
| Funktio | Python-ohjelmointikielessä koodinpätkä, joka suorittaa tietyn tehtävän silloin, kun sitä kutsutaan muualla ohjelmakoodissa |
| Git | Hajautettu versionhallintajärjestelmä, joka seuraa tiedostoissa tapahtuvia muutoksia, ja jota käytetään yleisesti ohjelmistokehityksessä |
| GDPR | General Data Protection Regulation, EU:n yleinen tietosuoja-asetus |
| INT | Integer; datan tyyppi, joka sisältää kokonaisluvun |
| JSON | JavaScript Object Notation, tiedostoformaatti datan säilömiseen ja siirtämiseen |
| Kysely | SQL-kielessä tapa esittää tietokantaan kohdistuva tietotarvemääritys |
| Metodi | Python-ohjelmointikielessä objektiin tai luokkaan liittyvä funktio, joka voi manipuloida objektin sisältämää dataa tai suorittaa sille toimintoja |
| Muuttuja | Mitattavan kohteen ominaisuus |
| MVP | Miminum Viable Product; versio tuotteesta, jossa on juuri tarpeeksi ominaisuuksia, jotta tuote saadaan julkaistua varhaisten asiakkaiden käyttöön |
| NULL | SQL-ssä käytettävä merkki, joka osoittaa, että tietokannassa ei ole data-arvoa |

| | |
|---|---|
| OLAP | On-Line Analytical Processing, otoskanta |
| Parametri | Python-ohjelmointikielessä muuttujia, jotka määritellään funktiossa |
| Parquet | Avoimen lähdekoodin sarakepohjainen tiedostomuoto, jota käytetään datan tallentamiseen ja siirtämiseen |
| PDF | Portable Document Format, standardisoitu tiedostomuoto sähköisten dokumenttien esittämiseen täsmälleen tarkoitetussa muodossa |
| Polars | Python- ja Rust-ohjelmointikieliin soveltuva kirjasto, joka tarjoaa toimintoja taulukkomuotoisten tietorakenteiden käsittelyyn |
| Primary Key | Pääavain; relaatiotietokannoissa oleva tieto, jonka tarkoituksena on olla uniikki arvo taulussa olevien tietueiden erottamiseksi toisistaan |
| PSA | Persistent Staging Area, tietojen pitempiaikaiseen säilytykseen tarkoitettu latausalueen (staging area) osa |
| Python | Yleiskäyttöinen ohjelmointikieli, joka soveltuu monentyyppisten sovellusten kehittämiseen |
| Raakadata | Jalostamattomassa muodossa oleva data |
| Redshift | AWS:n tietovarastointiratkaisu |
| Relaatiotietokanta | Tietokanta, jossa on useita toisiinsa kytkeytyviä tietokantatauluja |
| Relational Integrity | Datan eheys; termi, jolla kuvataan esim. tietokantataulujen välisiä riippuvuuksia |
| Rivi | Relaatiotietokannassa olevan taulun osa, joka sisältää yhtä entiteettiä kuvaavia tietoja, esim. henkilön etu- ja sukunimi sekä syntymäpäivä; vrt. tietue |
| S3 | Amazon Simple Storage Service; AWS:n tarjoama objektipohjainen tiedon varastointiratkaisu |
| Sarake | Relaatiotietokannan taulussa oleva joukko tietyn tyyppisiä arvoja, yksi jokaiselle taulun riville, vrt. attribuutti |
| Scrum | Ketterä projektinhallintamenetelmä, jota käytetään yleisesti mm. ohjelmistokehityksessä |
| Skeema | Relaatiotietokannan osa, joka määrittää, kuinka tiedot järjestetään, ja joka sisältää esim. taulujen nimet, sarakkeet, datatyypit ja entiteettien väliset suhteet |

| | |
|---|---|
| SQL | Structured Query Language; ohjelmointikieli, joka on suunniteltu relaatiotietokannassa olevien tietojen kyselemiseen, muokkaamiseen ja poimimiseen |
| Staging Area | Latausalue, tietovarastossa oleva tietojen tilapäiseen säilyttämiseen ja muuntamiseen tarkoitettu paikka |
| SSD | Solid-state drive; tietokoneen massamuisti, jossa ei ole liikkuvia osia ja jossa tieto säilyy laitteen ollessa virrattomana |
| String | Datan tyyppi, joka sisältää merkkijonon |
| Taulu | Relaatiotietokannan elementti, joka sisältää tietoja taulukkomuodossa ja joka koostuu sarakkeista ja riveistä |
| Tietue | Yhdistelmätyyppi toisiinsa liittyviä muuttujia, esim. henkilön etu- ja sukunimi sekä syntymäpäivä; vrt. rivi |
| TIMESTAMP | Aikaleima; merkkijono, jolla tunnistetaan, milloin tietty tapahtuma tapahtui ja joka yleensä sisältää päivämäärän ja kellonajan |
| Whitespace | Merkit, jotka kuvaavat tyhjää tilaa, esim. välilyönti, sarkain ja rivinvaihto |
| XML | Extensible Markup Language, datan säilömiseen ja siirtämiseen tarkoitettu tiedostomuoto |

## Sisällys

## Kuvat, ohjelmakoodit ja taulukot



## Liitteet

# 1 Johdanto

Tämän opinnäytetyön aiheena on toimeksiantajayritykselle toteutettu tietovarastointiprojekti, jossa ulkoisen tiedontuottajan raakadataa validoitiin ja normalisoitiin, jotta siitä saatiin tuotettua loppukäyttäjien saatavilla oleva harmonisoitu tietokantataulu. Tutkimuksen teoriaosassa käsiteltiin tietovarastoinnin käsitteitä, datan laadun hallinnan merkitystä liiketoiminnan kannalta sekä erilaisia tapoja ja käytäntöjä, joilla dataa voidaan validoida ja normalisoida.

Tietovarastointiprojektin taustalla oli tarve tuoda tiedontuottajan data saataville toimeksiantajan omaan bisneslogiikkaan niin, että jatkokehitys olisi mahdollisimman helppoa ja ylläpitoon kuluva aika mahdollisimman vähäistä. Luotettavuus, tiedon oikeellisuus ja ajantasaisuus ovat toimeksiantajan kannalta äärimmäisen tärkeitä seikkoja, joten datan laadun validoinnin merkitys korostui.

Projekti toteutettiin vuoden 2023 alkupuoliskon aikana, ja opinnäytetyön tekijä oli itse mukana sen suunnittelussa ja toteutuksessa. Projekti oli opinnäytetyötä aloittaessa jo varsin pitkällä, joten tutkimuksen tulokset toimivat erityisesti katsauksena projektin aikana aiemmin tehtyyn työhön, mutta toisaalta tutkimuksen teko mahdollisti myös tehdyn työn validoimisen teorian näkökulmasta. Tutkimuksessa pyritään myös reflektoimaan omaa työtä ja oppimista.

Tutkimuskysymykset:

- Mitä ominaisuuksia modernissa tietovarastossa on?
- Miksi datan laadun hallinta on tärkeää?
- Millaisia menetelmiä on datan validoimiseen ja normalisointiin?
- Miten datan validointi- ja normalisointimenetelmiä voidaan implementoida käyttöön?
- Minkä verran dataa validoimalla ja normalisoimalla saadaan virheitä poistettua?

# 2 Tietovarastointi

Tässä kappaleessa pohditaan, mitä ominaisuuksia modernin tietovaraston tulisi sisältää. Tarkoitus on kuvata tietovarastoinnin peruskäsitteet ja esitellä tietovarastoinnin perusarkkitehtuurimalli. Lopuksi käsitellään erityisesti AWS:n (Amazon Web Services) pilvipalveluna tarjoamaa tietovarastointiratkaisua, joka on opinnäytetyön käytännön osassa esiteltävän data-arkkitehtuurin osalta käytössä.

## 2.1 Tietovaraston määritelmä

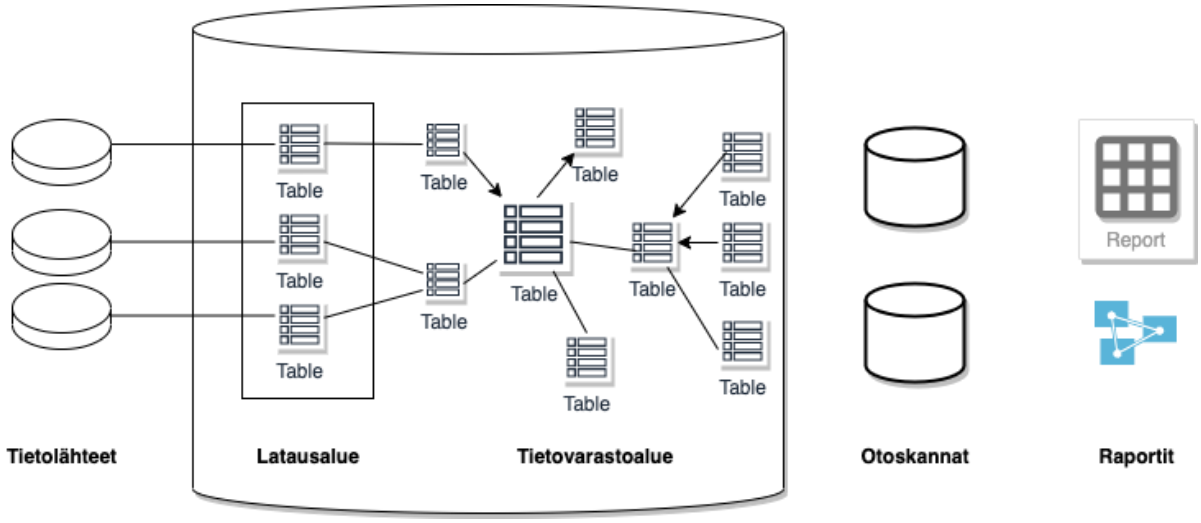
Tietovarasto (Data Warehouse) voidaan nähdä nimensä mukaisesti tiedon keskusvarastona, jossa yhdistetään ja harmonisoidaan eri lähteistä tulevaa dataa, ja yrityskontekstissa sen tarkoituksena on tuottaa dataa esimerkiksi Business Intelligence (BI) -työkaluja, raportointia ja analytiikkaa varten. Tällöin puhutaan datan muuntamisesta tiedoksi, jotta organisaatiossa voidaan tehdä dataan pohjautuvia ja perusteltuja liiketoiminnallisia päätöksiä. (SAP, n.d.)

Tietovaraston tarkoitus on sisältää yhdenmukaistettua ja yhtenäisen historian kattavaa tietoa, ja se on tyypillisesti koostettu käyttötarkoituksensa perusteella yhteen kuuluvista asiakokonaisuuksista, jolloin tietovarastosta tulee looginen ja hallinnoitava tietokokonaisuus, joka tarjoaa luotettavan käsityksen tiedosta. Tiedot voivat olla esimerkiksi yhden liiketoiminta-alueen tai tuoteryhmän kattavia, mutta myös konsernitason tarpeeseen eri tietolähteistä kerättyjä tietoja. Tietovarasto voi sisältää myös osakokonaisuuksia eli niin sanottuja otanta- tai paikallisvarastoja, josta käytetään yleisemmin englanninkielistä termiä data mart. (Törmänen 2017, 3–4.)

Tietovaraston perusarkkitehtuurimalli koostuu viidestä osasta (Kuva 1). Ensin tietoa kerätään eri tietolähteistä, jonka jälkeen sitä muokataan yhteismitalliseen muotoon siivoamalla, jalostamalla ja yhdistelemällä. Tiedot kerätään tietovarastoon, joka koostuu yhdestä tai useammasta tietokannasta, ja josta niitä voidaan jatkojalostaa ja muokata erilaisiin tarpeisiin esimerkiksi käyttäjäryhmän perusteella. Tämän jälkeen tietoa voidaan siirtää esimerkiksi otoskantoihin ja sieltä edelleen analyyseihin, raportteihin ja muihin palveluihin, joista tieto on loppukäyttäjän saatavilla. Näin dataa voidaan hyödyntää sekä suunnittelussa että

operatiivisessa toiminnassa, ja tiedon pohjalta voidaan laatia erilaisia raportteja ja analyyseja, tai tarkastella käyttöliittymästä suoraan. (Törmänen 2017, 6–7, 94.)

Kuva 1 Tietovaraston perusrakentamismalli (Törmänen 2017, 92.)




## 2.2 Datan liikkuminen tietovarastossa

Perinteistä tapaa kuvata datan liikkumista tietolähteistä latausalueelle ja sieltä edelleen tietovarastoon kutsutaan ETL (Extract, Transform, Load) -prosessiksi. Extract-vaihe kuvaa tietojen lataamista tietolähteistä, olivat ne sitten tiedostoja, tietokantoja tai esimerkiksi laskentataulukkoja. Transformation taas kuvaa prosessin osaa, jossa data muunnetaan standardisoituun muotoon, jota tietovarasto tai analyysityökalu osaa lukea. Tämän osan voi ajatella datan puhdistamisena, jotta se saadaan loppukäyttäjän kannalta luettavampaan muotoon. Transformaatio-osassa voidaan asettaa filttereitä, funktioita ja muita kriteerejä, joiden perusteella dataa halutaan muuntaa. Useita datasettejä voidaan myös yhdistää yhteneväiseen muotoon. Load-vaiheella taas viitataan datan tietovarastoon lataamiseen. Lopputuloksena tulisi olla käyttövalmista ja puhdistettua dataa. (Rithika 2023.)

Vaihtoehtoinen tapa ETL-prosessille on kuitenkin ELT (Extract, Load, Transform). Extract-vaihe toimii samaan tapaan kuin ETL-prosessissa, mutta sen sijaan että raakadata muunnettaisiin tässä vaiheessa haluttuun muotoon, ELT-prosessi ajaa datan ensin tietovarastoon säilöttäväksi, jonka jälkeen datalle tehdään muutostoimenpiteet ja

normalisoinnit. ELT-prosessi nähdään ETL-prosessiin verrattuna nopeammaksi sekä lataus- että transformaatiovaiheiden osalta, mutta vaatii tehokkaan tietokannan sekä osaamista Business Intelligence -työkaluista sekä paljon raakadataa. ELT on suunniteltu erityisesti skaalautuvia pilvipalveluja ajatellen, ja raakadata on aina saatavilla lähes reaaliaikaisesti, kun taas ETL-prosessi edellyttää datan päivittämistä tietovarastossa. (Talend n.d.a)

Tietovarastoinnin tietolähteitä voivat olla esimerkiksi organisaation omat operatiiviset järjestelmät, kuten toiminnanohjaukseen (ERP) tai asiakkuudenhallintaan (CRM) liittyvät järjestelmät, tietokannat tai ulkoiset lähteet, kuten kumppanijärjestelmät (SAP, n.d.). Dataa voidaan ladata myös esimerkiksi sähköposteista, verkkosivuilta ja yksittäisistä tiedostoista, joista CSV (comma-separated values) -muotoiset tiedostot ovat yleinen esimerkki (IBM n.d.).

Data voi olla joko strukturoitua tai ei-strukturoitua. Strukturoidulla datalla viitataan dataan, joka on sille määritellyssä kentässä esimerkiksi tiedoston sisällä, ja sitä voi tyypillisesti säilöä relaatiotietokannassa. Strukturoitu data voi sisältää numeroita ja tekstiä, ja sitä hallitaan yleensä SQL-kielellä (Structured Query Language). Esimerkkejä strukturoidusta datasta ovat esimerkiksi nimet, osoitteet, numeerinen data, Microsoft Excel -tiedostot ja tekstitiedostot. Ei-strukturoitu data taas säilötään alkuperäisessä formaatissa ja sille ei ole määritelty mitään datamallia. Ei-strukturoitua dataa on paljon enemmän kuin strukturoitua, ja sitä ovat esimerkiksi teksti, sosiaalisen median aktiviteetit, video- ja audiotiedostot, valvontamateriaali ja lukuisat muut tiedostotyypit. On myös olemassa semi-strukturoitua dataa, joka on strukturoidun datan muoto, joka ei sovi relaatiotietokannan rakenteeseen. (Smallcombe 2023)

Tiedot tuodaan ETL-prosessissa ensin tyypillisesti tiedon latausalueelle, josta käytetään yleisesti englanninkielistä staging area -termiä. Latausalue on sijoitettu raakadataa sisältävien datalähteiden ja tietovarastoalueen tai otoskannan väliin. Latausalueet on usein tarkoitettu tietojen lyhytaikaiseen säilyttämiseen ja käsittelyyn, jolloin niiden sisältö tyhjennetään ennen ETL-prosessin mukaisen latauksen suoritusta, tai sen jälkeen. Kuitenkin joissain tapauksissa arkkitehtuuri voi sisältää latausalueen, jossa säilötään dataa pitempään esimerkiksi säilömisen tai virheenkäsittelyn vuoksi. Latausalueella dataa voidaan monistaa, muuntaa ja testata ennen siirtämistä varsinaiseen tietovarastoon. Latausalueen käytön

hyötyjä ovat myös esimerkiksi mahdollisuus palauttaa korruptoitunutta dataa ja varmuuskopioiden ottaminen. (Varshney 2023)

Tietojen pitempiaikaiseen säilytykseen tarkoitettua latausaluetta kutsutaan persistent staging area -termillä (PSA). Lähdejärjestelmän data ladataan PSA-alueelle ilman transformaatiota raakadatamuodossa ja tietoja ei sieltä tavallisesti poisteta. Tämä mahdollistaa historiatietojen keräämisen, ja raa'assa muodossa oleva data saattaa sisältää myös attribuutteja, joita ei tietovaraston näkökulmasta tarvita. Toisaalta, jos tietovarastoon halutaan myöhemmässä vaiheessa tuoda uutta tietoa, voi se olla PSA-alueelta kaikki historia huomioiden saatavilla. PSA-alue nähdään siis hyödyllisenä mahdollisia muutostarpeita ajatellen, sillä harva tietovarasto säilyy muuttumattomana. (Felix n.d.).

Tietovarastoalue voi koostua useista relaatiotietokannoista, joista jokaisessa data on organisoitu tauluihin ja sarakkeisiin. Jokaiselle sarakkeelle on määriteltävissä tyyppi, kuten numeerisille arvoille integer tai merkkijonoille string. Tauluja organisoidaan skeemoissa, jotka voidaan nähdä kansiorakenteen tyyppisenä elementtinä. Kun data tuodaan tietovarastoon, se säilötään tauluihin, jotka määritellään skeemoissa. Erilaiset kyselytyökalut voivat sitten käyttää skeemaa tauluissa olevan datan kyselemiseen ja analysointiin. (Amazon Web Services, 2023).

Koska monet tietovarastot sisältävät perustasolla suuria määriä yksityiskohtaista dataa, voidaan tietovarastosta luoda otoskantoja, jotka tarjoavat tiettyyn tarpeeseen suunniteltuja näkymiä tietovaraston datasta. Otoskannoista käytetään yleensä OLAP-nimitystä, jonka lyhenne muodostuu sanoista On-Line Analytical Processing. Otoskannat voivat olla relaatiomallisia, moniulotteisia tai hybridimallisia analyysikantoja. Perusanalyysi- ja raportointitarpeet voidaan helpoiten toteuttaa relaatiokannasta SQL-kyselyillä. Relaatiokantaratkaisuissa tieto haetaan relaatio-otoskantaan tietovarastosta. Moniulotteisessa analyysikannassa taas tietokannan rakenne poikkeaa relaatiokannasta, mutta työkalut saattavat tarjota tehokkaammat ja monipuolisemmat mahdollisuudet analyysia varten. Hybridissä kantaratkaisussa on mukana sekä relaatiomallisen että moniulotteisen kannan arkkitehtuuria, mutta se on ratkaisuna hankalampi toteuttaa ja hallita kuin kumpikaan niistä. (Törmänen 2017, 113–114.)

## 2.3 Tietovarastointi pilvipalveluissa

Pilvipohjainen tietovaraston arkkitehtuuri tarkoittaa, että tietovarasto ei sijaitse perinteisesti käyttäjän omilla palvelimilla vaan pilvipalvelussa. Pilvipohjainen tietovarastointi eroaa perinteisestä esimerkiksi kustannusten osalta. Perinteisessä mallissa tietovarastointiin sisältyy usein erilaisia etukäteis- ja ylläpitokustannuksia, joita pilvipalveluissa ei ole. Pilvipalvelut hinnoitellaan käytön ja tarpeen mukaan, joten niiden käyttö voi olla kustannustehokkaampaa. Pilvipalvelujen tarjoama arkkitehtuuri on yleensä myös selvästi nopeampi kuin perinteinen malli, mikä johtuu osaksi siitä, että niissä käytetään usein ETL-prosessin sijaan ELT-prosessia datan lataamiseen ja muokkaamiseen. Lisäksi pilvipalvelujen eduksi nähdään se, että ne tukevat laajemmin eri tavoin strukturoitua dataa ja erilaisia dataformaatteja. Erityisesti suurien datasettien kohdalla pilvipalvelujen etuna on myös helpompi skaalattavuus. (Talend n.d.b)

Amazon Web Services (AWS) tarjoaa tietovarastointiratkaisuna Amazon Redshift -palvelua, jonka se lupaa olevan nopea, täysin hallittava ja massiivistenkin datamäärien käsittelyyn sopiva (Challa, Radovanovich, Yu, Friedmann & Goel 2023, 3.). AWS kertoo palvelun tietovarastointiarkkitehtuurin koostuvan tasoista. Ylin taso on käyttöliittymä, jonka taustalla on esimerkiksi raportointi-, analyysi- ja tiedonlouhintatyökaluja. Keskimmäinen taso sisältää analytiikkamoottorin, jota hyödynnetään tietojen analysoinnissa. Arkkitehtuurin alin taso on tietokantapalvelin, jonne tiedot tallennetaan. Tärkeimpiä ja useimmin käytettyjä tietoja suositellaan tallennettavaksi ns. nopeaan tallennustilaan, joka voi olla esim. SSD-asema (solid-state drive), kun taas tiedot, joita käytetään harvemmin, olisi suositeltava säilöä kustannustehokkaassa objektipohjaisessa varastossa, joka on AWS:n tapauksessa Amazon S3 -palvelu (Amazon Simple Storage Service). AWS myös lupaa, että tietovarastoratkaisuna Amazon Redshift optimoi kyselyjen nopeutta siirtämällä useimmin käytettyjä tietoja nopeaan tallennustilaan. (Amazon Web Services 2003)

# 3 Datan laatu

Tässä luvussa on tarkoituksena kartoittaa, mitä datan laadulla ja laadunhallinnalla tarkoitetaan. Luvussa esitellään myös kahdeksan laatuun liittyvää ulottuvuutta, joiden perusteella datan laatua voidaan tarkastella.

## 3.1 Datan laadun hallinta

Datan laadun hallinta on toimintamalli datan laadusta huolehtimisen avuksi, ja siihen kuuluvat olennaisesti myös datan ohjaaminen, seuranta ja laadun parantamisen prosessit. Datan hyvä laatu on tavoite, johon datan hallinnasta vastaavien henkilöiden tulisi kaikella toiminnallaan pyrkiä. Datan hallinta on tapa varmistaa datan sopivuus käyttötarkoitukseensa sekä datan laadun mittaamiseen, seurantaan ja parantamiseen liittyvien prosessien määrittelyä ja jalkauttamista käytäntöön. (Väre 2019, 191–192.)

Huomioitavaa on myös, että datan laadun hallintaa eivät ole esimerkiksi kertaluontoiset datan siivousoperaatiot. Kun datan laadun hallinnan painopisteet ovat ongelmissa tai pelkässä datassa, painottuvat toimenpiteetkin reaktiiviseen virheiden korjaamiseen, ja datan huonosta laadusta muodostuu kustannuksia ennen kuin ne korjataan. Tavoitteena pitäisi mieluummin olla virheiden syntymisen estäminen, mikä on pitkällä aikajänteellä kustannustehokkaampaa. Dataa tulisikin tarkastella osana liiketoiminnan prosesseja ja asiakaskokemusta, jolloin datan hallintaan saadaan proaktiivisempi ote ja saadaan rakennettua datan hyvä laatu osaksi päivittäistä toimintaa. (Väre 2019, 192–195.)

## 3.2 Datan laadun ulottuvuudet

Laadun yleinen määritelmä on tarkoitukseen sopivuus, joka loppujen lopuksi määräytyy loppukäyttäjän tarpeen mukaan. Datan laadun kannalta on tärkeää, että siitä voidaan luotettavasti yksilöidä henkilö, organisaatio tai muu todellisen elämän asia ja se sisältää tarpeelliset attribuutit. Kaikkien attribuuttien tulee olla sisällöltään ja muodoltaan tarkoitukseen sopivia. Koska tarpeenmukaisuus eli se, mitä varten data ylipäänsä luodaan tai tuodaan järjestelmiin, määritellään aina datan hyödyntäjän tai loppukäyttäjän tarpeen

perusteella, tarvitaan datan laadun määrittelyyn ne organisaation liiketoimintojen edustajat, jotka ymmärtävät, miten loppukäyttäjät, kuten asiakkaat, dataa käyttävät. (Väre 2019, 199.)

Datan laadun merkityksellisimpiä ulottuvuuksia ovat ainakin sisällön kattavuus, oikeellisuus, ainutlaatuisuus, vaatimustenmukaisuus, yhteneväisyys, eheys, järkevyys ja ajanmukaisuus (Väre 2019, 203.). Näitä ulottuvuuksia ja niiden merkitystä tarkastellaan seuraavissa kappaleissa hieman tarkemmin.

Sisällön kattavuus kertoo, onko dataa olemassa. Tällä voidaan tarkoittaa koko tietokantaa, mutta datan laadun arvioinnissa tarkastellaan kattavuutta tarkemmin joko sarakkeen (attribuutin) tai rivin (tietueen) tasolla. Ensin tulisi tarkistaa, onko tietojärjestelmässä paikkaa, jossa datan pitäisi olla, eli onko käyttöliittymässä kenttä tai tietokannassa sarake kyseiselle attribuutille. Jos on, on tarkistettava, onko se tyhjä vai ei. Teknisesti ottaen kenttä ei välttämättä ole tyhjä, vaikka kenttä sisältäisi esimerkiksi nollan tai välilyöntejä, koska tietokannassa tyhjäksi arvoksi määritellään yleensä niin sanottu null-arvo. Datan laadun kannalta tulisi kuitenkin joissain tapauksissa ottaa myös nollat sekä pelkkiä välilyöntejä sisältävät arvot huomioon. Attribuutti- tai sarakekohtainen arviointi tarkoittaa, että sarakekohtaisesti tarkastetaan, moneltako riviltä kyseinen tieto puuttuu. Tietuekohtainen arvio taas mittaa, onko yhdellä rivillä kaikissa arvioinnin kannalta tärkeissä attribuuteissa sisältöä. (Väre 2019, 204–205.)

Jos tietojärjestelmässä on dataa, sen oikeellisuus, eli vastaavuus tosielämään, tulee arvioida. Oikeellisuudella voidaan esimerkiksi tarkoittaa, onko katuosoite oikeasti olemassa ja asuuko henkilö todellisuudessa kyseisessä osoitteessa. Data voi olla oikeellisuuden osalta hyvin puutteellista, koska monissa tapauksissa esimerkiksi osoitetiedon puuttuessa kenttä on täytetty oletusarvolla, kuten ”ei tiedossa” -tekstillä, tai syntymäpäiväksi voidaan ilmoittaa oletusarvoisesti 1.1.1900. Datan oikeellisuus on vaikeasti todennettavissa, ja usein on suositeltavaa käyttää todennuksessa virallisia rekistereitä, vaikka nekin voivat olla tiedoiltaan virheellisiä tai puutteellisia. (Väre 2019, 206–207.)

Ainutlaatuisuus on datan laadun kannalta erittäin tärkeää. Ainutlaatuisesta datasta voidaan puhua, kun se esiintyy yhdessä tietokannassa tai järjestelmässä vain kerran. Esimerkiksi yhtä asiakasta kohden tulee olla vain yksi tietue, ja tuotehierarkiasta tulisi löytyä jokainen

tuotekategoria vain kerran. Useammista toistuvista ilmentymistä käytetään termiä tuplatietue, ja ne voivat aiheuttaa ongelmia esimerkiksi silloin, kun asiakkaan kohdalle merkityt yhteydenotot on kirjattu toiselle saman asiakkaan ilmentymälle. Ainutlaatuisuuden varmistamiseksi tarvitaan säännöt, joilla poikkeustapaukset ja datan puutteet käsitellään. (Väre 2019, 208.)

Vaatimustenmukaisuudella tarkoitetaan, että data täyttää kaikki sille asetetut muodolliset ja sisällölliset vaatimukset. Muodolliset vaatimukset liittyvät datan teknisiin ominaisuuksiin sekä esimerkiksi siirrettävyyteen järjestelmien välillä, kun taas sisällöllisillä vaatimuksilla viitataan datan liiketoiminnalliseen hyödynnettävyyteen. Jos data ei täytä muodollisia vaatimuksia, voi sitä olla vaikea siirtää automatisoituihin prosesseihin. Esimerkiksi sähköpostiosoite, josta puuttuu @-merkki, voi kaataa koko prosessin. Jos taas liiketoiminnalle on tarpeellista, että henkilölle on tiedossa koko virallinen nimi, ei esimerkiksi pelkkä sukunimi tai lempinimi täytä datan sisällöllisiä vaatimuksia. (Väre 2019, 209–210.)

Yhteneväisyydellä viitataan datan sisällön samankaltaisuuteen kaikkialla, missä dataa on, eli onko useisiin paikkoihin siirretty data täysin samaa kuin lähdedata. Yhteneväisyyden puute esimerkiksi siitä syystä, että jossakin vaiheessa data on jäänyt päivittymättä, voi aiheuttaa hankaluuksia esimerkiksi asiakaspalvelussa tai jopa maineenmenetystä, jos asiakasta lähestytään väärillä tai vanhentuneilla tiedoilla. Yhteneväisyyttä voi tarkistaa selvittämällä, missä paikoissa dataa on ja mistä sen sieltä löytää, sekä vertaamalla tietoja toisiinsa. Suora vertailu kuitenkin on haastavaa, koska sama data voi olla tallennettuna eri järjestelmissä eri tavoilla. Data täytyy siis ensin standardoida eli yhdenmukaistaa. Esimerkiksi nimi voi olla yhdessä järjestelmässä vain yksi kenttä, toisessa etunimi ja sukunimi eroteltuna omiin kenttiinsä, tai päivämäärät on voitu tallentaa eri järjestelmiin eri muodossa. (Väre 2019, 2010–2011.)

Datan eheydellä puolestaan pyritään varmistamaan, että järjestelmien väliset viittaukset dataan ovat oikeita, eli tietueen tai rivin tulisi olla yhdistettävissä luotettavasti. Tietokannan sisäisten viittausten tulisi myös olla tietokannan rakenteen mukaisia, mitä kutsutaan viite-eheydeksi. Viitetiedoista puhuttaessa tarkoitetaan avaintietoja tai yksilöiviä tunnisteita, joilla dataa voidaan yhdistää järjestelmien tai tietokantataulujen välillä. Esimerkiksi relaatiotietokannassa jokaisella tietueella on yksilöivä avain, jota sanotaan perusavaimeksi

(englanniksi primary key). Mikäli tähän tietueeseen viitataan toisesta taulusta, kutsutaan kyseistä arvoa toisessa taulussa viiteavaimeksi (foreign key). Avaimena voidaan myös käyttää jotakin ulkoista tunnistetta, kuten henkilön tapauksessa henkilötunnusta tai yritykselle y-tunnusta. Jos avaimia puuttuu tai ne eivät kohdistu oikein järjestelmien sisällä tai välillä, datan yhdistäminen ei onnistu, mikä voi aiheuttaa virheitä tai puutteita. Eri järjestelmien välillä voi olla käytössä niin sanottu ristiviittaustaulu, jossa eri järjestelmien sisäiset tunnisteet yhdistetään toisten järjestelmien tunnisteisiin. (Väre 2019, 212.)

Järkevyys kertoo, onko data malliltaan oletusten mukaista, mikä on lähellä vaatimuksenmukaisuutta. Sitä ei siis ole välttämättä tarpeen huomioida erikseen, mikäli sisältövaatimuksissa on jo huomioitu kyseiset asiat. Esimerkkinä datasta, joka ei ole järkevää, mainitaan tulevaisuuteen merkityt kuolin- tai syntymäpäivät, mutta esimerkiksi jälkimmäisen osalta järkevyyden ratkaisee konteksti, jossa dataa käsitellään. Myös esimerkiksi kuukauden myyntiluku saattaa poiketa järkevästä, jos se eroaa liiaksi normaalista trendistä. Järkevyydestä poikkeava data ei ole automaattisesti virheellistä, mutta syitä poikkeavuudelle voi olla tarpeen tutkia tarkemmin. (Väre 2019, 213.)

Ajanmukaisuudella taas tarkoitetaan, onko data oikea-aikaista. Data ei saisi olla liian vanhaa, ja sen tulisi olla saatavilla oikeaan aikaan. Ajanmukaisuus on usein tärkeää tapahtumatietojen osalta sellaisissa prosesseissa, joissa oikea-aikaisuudella on suuri merkitys. Esimerkiksi terveyteen tai turvallisuuteen liittyvän datan on ehdottomasti syytä olla ajanmukaista. Jos ajanmukaisuus ei kuitenkaan ole täysin kriittistä, voidaan ajanmukaisuuden avulla kuitenkin arvioida datan kokonaisluotettavuutta. Jos dataa ei ole päivitetty yli vuoteen, voidaan sen ajanmukaisuutta tarkastella, tai verrata päivittymistä useampien tietojärjestelmän kesken. Näin voidaan havaita, päivittyykö joku järjestelmä aina muista jäljessä, jolloin voidaan todeta kyseisen järjestelmän sisältävän ongelmia ajanmukaisuuden osalta. (Väre 2019, 2014)

# 4 Datan validointi- ja normalisointimenetelmät

Edellisissä luvuissa on tarkasteltu tietovarastointia ja datan laatuun liittyviä käsitteitä. Tässä luvussa perehdytään tarkemmin datan validoinnin ja normalisoinnin prosessiin. Tavoitteena on tutkia, minkälaisia menetelmiä datan laadun validoimiseksi on käytössä ja miten dataa voi muokata tietovarastointia ajatellen.

## 4.1 Datan validointi

Datan validointi on prosessi, jolla suoritetaan kerätylle tai ulkopuolelta tuodulle datalle tarkistuksia ennen kuin data menee käytettäväksi. Jokaisen datan käsittelyyn liittyvän toimenpiteen, oli se sitten datan kerääminen, analysointi tai strukturointi, tulisi sisältää datan validointia, jotta tulokset olisivat mahdollisimman luotettavia. Datan validoinnista onkin tullut yksi tehokkaan datanhallinnan tukipilareista, koska se mahdollistaa merkityksellisten ja validien datasettien muodostamisen esimerkiksi analytiikkaa ajatellen. (TIBCO n.d.)

Datan laadun validoinnilla pyritään varmistamaan, että datassa olevat arvot näyttäytyvät oikeanlaisina suhteessa toisiinsa ja että ne noudattavat tiettyjä periaatteita. Hyvin toteutettuna validointisäännöt auttavat tunnistamaan ja luokittelemaan suuren osan datassa esiintyvistä ongelmista. Tärkeää on kuitenkin pyrkiä tunnistamaan kaikki mahdolliset validointisäännöt ja varmistaa, että ne ymmärretään oikein, koska puuttuvat ja väärinymmärretyt säännöt voivat puolestaan vaikuttaa datan laadun tulkitsemiseen hyvin paljon. (Maydanchik 2007, 59–60.)

Perustasollaan data koostuu yleisimmin eri attribuuttien yksilöllisistä arvoista, jotka usein kuvastavat tosielämän ihmisiä, asioita, paikkoja tai tapahtumia. Esimerkkeinä mainittakoon, että paino ja pituus voivat olla ihmisen mittoja, ja huonenumero sekä kesto voivat kuvastaa esimerkiksi liiketapaamista. Tosielämän objektit eivät kuitenkaan voi saada arvoikseen mitä tahansa, vaan esimerkiksi ihmisen pituuden tai tapaamisen keston tulisi olla kohtuullinen tosielämään suhteutettuna. Tarkoituksena on siis rajoittaa attribuutin (tai tietokannan näkökulmasta sarakkeen) mahdollisesti saamia yksilöllisiä arvoja. Tällaisia validointisääntöjä

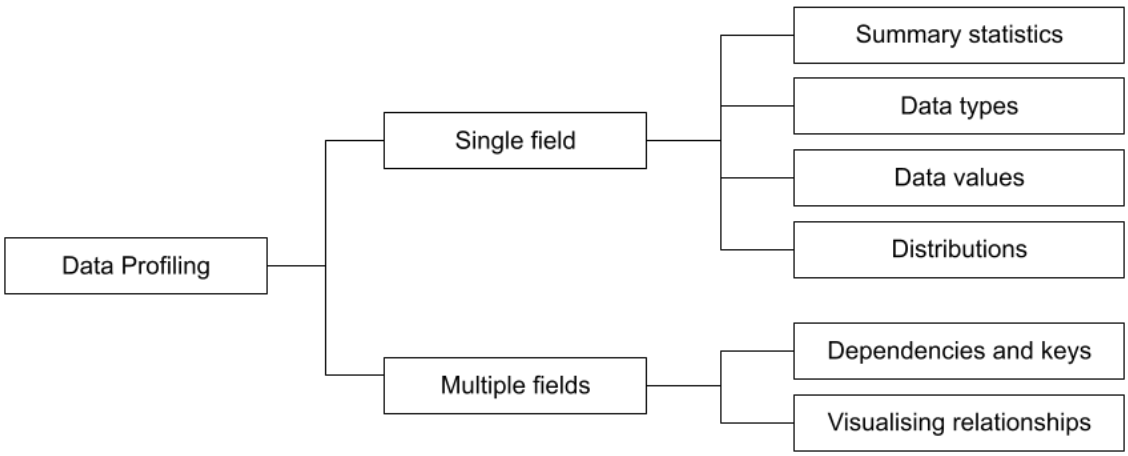
kutsutaan attribuutin muodon rajoittimiksi (englanniksi attribute domain constraints). (Maydanchik 2007, 63–64.)

Toinen tapa validoida dataa hieman monitahoisemmin on luoda datan eheyteen (englanniksi relational integrity) liittyviä sääntöjä. Nämä liittyvät datassa olevien esiintymien keskinäisiin viittauksiin ja ovat esimerkiksi relationaalisten tietokantojen pohjana. Yhdessä attribuutin muodon rajoittamisen kanssa nämä muodostavat datan laadunhallintaan pohjan, mutta on syytä huomioida, että laadunhallintaan liittyviä sääntöjä on muitakin ja ne voivat olla tosimaailman kaltaisesti hyvin monimutkaisia. (Maydanchik 2007, 60.)

## 4.2 Datan profilointi

Datan profiloinniksi kutsutaan yleensä tietovarastointiprojektin alkuvaiheessa tehtävää systemaattista prosessia, jolla pyritään saamaan tietoa lähtödatasta ja sen rakenteesta, ominaisuuksista, tarkoituksesta ja laadusta. Tarkoituksena on selvittää, vastaako data tietovaraston laatuvaatimuksia vai onko sitä tarpeellista puhdistaa ennen tietovarastoon siirtämistä. Tietovarastointiprojektin onnistumisen kannalta mahdolliset korjaukset ovat hyvin tärkeitä, koska lähtödatan mahdolliset virheet voivat aiheuttaa ongelmia myöhemmissä vaiheissa. Profilointiin kuuluu yksittäisten kenttien ja sarakkeiden tutkimista sekä taulujen välisten riippuvuussuhteiden analysointia (Kuva 2). (Kimball 2004)

Kuva 2 Datan profiloinnin tekniikat (Atlan 2022)



Profilointi aloitetaan yleensä yksittäisten kenttien ja sarakkeiden tutkimisesta, ja oletuksena on, että kaikki arvot ovat samaa tyyppiä ja että niiden ominaisuudet ovat samankaltaiset. Yksittäisen sarakkeen tai kentän osalta voi laatia esimerkiksi kokonaistilaston, joka sisältää esimerkiksi sarakkeen sisältämät maksimi- ja minimiarvot sekä keskiarvot (Atlan 2022). Lisäksi olisi syytä tarkastaa, kuinka monella sarakkeen rivillä on tyhjiä tai null-arvoja sekä laskea lukumääriä esimerkiksi taulun sisältämistä riveistä ja yksittäisistä sarakkeen arvoista. (Naumann 2017)

Profiloinnissa halutaan yleensä tutkia myös datan tyyppiä. Yksinkertaisella tasolla voidaan tutkia, onko kenttä teksti- tai numeromuotoinen vai esimerkiksi aikaleima. Yleisimpiä SQL-kielen datatyyppejä ovat esimerkiksi CHAR, INT, DECIMAL ja TIMESTAMP, joista CHAR edustaa tekstimuotoista string-tyyppiä, INT on kokonaisluku, DECIMAL murtoluku ja TIMESTAMP päivämäärää edustava aikaleima. CHAR-tyyppi voi myös sisältää domain-osan, jossa määritellään kuinka monta merkkiä kenttä voi sisältää, esimerkiksi VARCHAR(12). Jos data on esimerkiksi XML- tai JSON-muodossa, voi se olla hienorakenteisempaa. (Naumann 2017)

Datan tyyppiä voi lähestyä myös tilastotieteen näkökulmasta selvittämällä, ovatko kenttien sisältämät arvot tai muuttujat diskreettejä vai jatkuvia. Muuttuja on diskreetti silloin, kun sen mahdolliset arvot ovat lueteltavissa. Esimerkiksi kotikunta ja lukumäärä ovat muuttujina diskreettejä. Jatkuva muuttuja taas saa sitä tarkempia arvoja, mitä tarkemmin ominaisuutta mitataan, ja mahdollisia arvoja ei voi etukäteen luetella. Esimerkiksi aikaa voi mitata mielivaltaisen tarkasti, joten ikä on jatkuva muuttuja. (Tilastokeskus n.d.)

Yksittäisten kenttien profilointia voi tehdä myös tutkimalla datan sisältämiä arvoja tarkemmin. Näin voidaan tunnistaa arvojen sisältämiä ominaispiirteitä ja mahdollisesti toistuvia kaavoja, esimerkiksi tietueen yksilöivät tunnukset, osoitekentät ja kaupungit. Datan arvojen profiloiminen auttaa tunnistamaan, onko data vaatimustenmukaista tai oikeellista. (Atlan 2022). Esimerkki datan oikeellisuuden puutteellisuudesta on esimerkiksi sarake, jonka tulisi sisältää puhelinnumeroita, mutta jonka arvot sisältävät aakkosia (Kimball 2004).

Arvojen jakautumisen tarkastelu on puolestaan hyödyllistä esimerkiksi poikkeavuuksien löytämiseksi. Arvoja voi jaotella kategorian perusteella ja laskea, kuinka paljon esiintymiä

kussakin kategoriassa on. Numeerisen datan osalta voi olla hyödyllistä tehdä esimerkiksi visualisointeja, kuten histogrammeja. Näin voidaan tunnistaa, seuraako arvojen jakauma esimerkiksi jotain tunnettua jakaumaa. (Naumann 2017)

Useamman kentän tai taulun välisten riippuvuuksien profilointi liittyy datan eheyden varmistamiseen. Näin pyritään tunnistamaan esimerkiksi datassa esiintyvät tunnisteet ja yksilöivät arvot sekä arvojen linkittyminen taulujen välillä. Datasta voidaan myös tunnistaa ulottuvuuksia tutkimalla, ovatko yhden kentän arvot muiden kenttien arvojen osajoukkoja. Riippuvuussuhteita voi myös visualisoida esimerkiksi eri tauluissa olevien kenttien välisen korrelaation osalta. (Atlan 2022)

### 4.3 Datan puhdistaminen

Datan puhdistaminen (englanniksi data cleansing) on ETL- tai ELT-prosessissa transformaatiovaiheeseen kuuluva prosessi, jonka tarkoitus on valmistella raakadata esimerkiksi tietovarastoon ja sieltä jatkokäyttöön viemistä varten. Raakadata voi sisältää virheitä, millä voi olla vaikutusta esimerkiksi analyysityökalujen tai koneoppimismallien kannalta. Seurauksena voi olla esimerkiksi vääriä ennusteita tai muita negatiivisia liiketoiminnallisia vaikutuksia, joten päätöksenteon kannalta datan ajantasaisuus ja tarkkuus on tärkeää. Esimerkkejä datan sisältämistä virheistä voivat olla esimerkiksi muotoiluvirheet, kuten väärässä muodossa ilmoitetut päivämäärät tai muut mittayksiköt, sekä poikkeavan pienet tai suuret arvot. Muita yleisiä virheitä ovat esimerkiksi datan korruptoituminen, puuttuva tieto ja kirjoitusvirheet. Näitä virheitä voi korjata esimerkiksi poistamalla tietueiden kaksoiskappaleet sekä jatkokehityksen tai analyysin kannalta epäoleelliset attribuutit sekä selvästi puutteellinen data. Poikkeusarvojen osalta tulee määritellä, saavatko arvot jäädä paikalleen vai onko niitä syytä käsitellä. Myös kirjoitusvirheet ja muut datassa esiintyvät epäjohdonmukaisuudet olisi hyvä korjata, jotta data saadaan noudattamaan haluttua mallia. (Amazon Web Services n.d.)

Transformaatioprosessiin on löydettävissä joitakin tekniikoita datan puhdistamiseksi. Eräs yksinkertaisimmista tavoista on datan filtteröiminen, joka tarkoittaa, että lähtö- tai raakadatasta pudotetaan pois tietyt sarakkeet (attribuutit) tai rivit (tietueet) ennen datan siirtämistä tietovarastoon. (Keboola 2022)

Datan muuntaminen (englanniksi data mapping) puolestaan tarkoittaa, että datasyöte muutetaan vastaamaan samaa tietoa toisessa muodossa. Muuntaminen sisältää esimerkiksi alkukirjainten konvertoinnin niin, että lopputuloksena jokainen tietue sisältää ison alkukirjaimen, merkkijonojen muuntamisen samaan koodausstandardiin, päivämäärä- ja aika-arvojen synkronoinnin sekä esimerkiksi diskreetin merkkijono-tyyppisen muuttujan, kuten sukupuolen, muuntamisen koodimuotoiseksi. Jos siis raakadatassa sukupuoli ilmoitetaan merkkijonona ”mies” tai ”nainen”, voidaan transformaatiossa muuttuja koodata numeeriseksi arvoksi ”0” tai ”1”, mikä auttaa datan luettavuutta esimerkiksi ennustealgoritmien osalta. (Keboola 2022)

Duplikaattiarvojen poistaminen taas tarkoittaa sitä, että datasta poistetaan useammin kuin kerran esiintyvät tietueet, jos tuplatietueet todetaan turhiksi. Transformaatiovaiheessa voidaan luoda myös lähtödatan pohjalta johdettuja muuttujia, jotka lasketaan muiden datapisteiden ja muuttujien perusteella. Data voidaan transformaatiossa myös järjestää tietyn arvon perusteella ennen tietovarastoon tallennusta, jos sen etsintätoiminnallisuutta halutaan parantaa tietyn attribuutin pohjalta. Tyypillistä on myös yhdistää eri tietolähteistä hankittua dataa yhteen tauluun esimerkiksi taulujen sisältämien avainten perusteella. (Keboola 2022)

Datan aggregointi taas on eräänlainen yhteenvedon muoto, jossa luodaan mittareita tai muuttujia datan sisältämien ulottuvuuksien pohjalta. Esimerkkinä ulottuvuudesta mainittakoon maantieteellinen alue, jonka perusteella voidaan laskea ryhmittelemällä kokonaismyynti jokaiselle ulottuvuudelle, olettaen että jokaiselle myyntitapahtumalle on olemassa oma tietue, joka sisältää tiedon maantieteellisestä alueesta. Jos taas transformaatiossa käsitellään strukturoimatonta tai puolistrukturoitua dataa, kuten merkkijonoja tai JSON-/XML-tiedostoja, tiedot voi jakaa esimerkiksi sarakkeisiin ennen tallentamista tietovarastossa olevaan tauluun. Tyypillisesti esimerkiksi osoite saattaa olla yksi merkkijono, josta halutaan erotella katuosoite, postinumero, kaupunki ja maa omiin sarakkeisiinsa. (Keboola 2022)

# 5 Datan validointi- ja puhdistamisprojektin kuvaus

Tässä luvussa esitellään toimeksiantajayritykselle toteutettu projekti, jossa yrityksen data-arkkitehtuurin puitteissa toteutettiin ulkoisen tiedontuottajan toimittamaan tietosisältöön datan laadun validointia ja puhdistustoimenpiteitä ennen datan lataamista tietovarastoon. Luvussa esitellään lähtötilanne ja toimeksiantajan tarpeet, käytössä oleva data-arkkitehtuuri projektin kohteena olevan tiedontuottajan osalta sekä välineet ja ohjelmistot, joita päädyttiin projektin suunnitteluvaiheessa käyttämään.

## 5.1 Toimeksiantajan tarpeet

Toimeksiantajayrityksellä on useita ulkoisia tiedontuottajia ja järjestelmiä, josta raakadataa tulee omaan ympäristöön sisään. Datan ylläpitotehtäviin kuluu vuoden aikana kymmeniä ihmistyöpäiviä, jolloin työn tehokkuus kärsii, kun liiketoiminnallisesti kannattavampien työtehtävien sijaan joudutaan ratkomaan yksittäisiä dataan liittyviä ongelmatilanteita, jotka vaativat usein ongelman laajuuteen nähden paljon työaikaa.

Projekti liittyy uuden tiedontuottajan tarjoamaan dataan ja sen saattamiseen sellaiseen muotoon, että data voidaan varastoida luotettavasti ja viedä tietovarastoa käyttäviin palveluihin loppukäyttäjien saataville. Lähtödata on tässä tapauksessa ulkomaisista yrityksistä saatavaa viranomaisdataa, ja se sisältää myös yritysten vastuuhenkilöitä ja taloudellista dataa, kuten tilinpäätösten tunnuslukuja. Tässä projektissa ei kuitenkaan talouslukuja käsitelty, vaan toteutuksessa keskityttiin yrityksiin, ja vähemmissä määrin myös vastuuhenkilöihin, liittyviin tietoihin.

Tavoitteena oli luoda data-arkkitehtuurin sisälle prosessi, joka sisältää tarvittavat datan validoinnit ja puhdistusmenetelmät uuden tiedontuottajan lähtödataan. Toimeksiantaja oli määritellyt tietosisällön, joka vaaditaan niin kutsutun public-taulun muodostamiseksi tietovaraston sisälle. Kyseisen taulun haluttiin sisältävän jokaiselle yritykselle tietyt perustiedot, ja sen tarkoitus oli toimia loppukäyttäjien saataville menevän datan lähteenä. Loppukäyttäjät puolestaan ovat B2B-asiakkaita, jotka käyttävät dataa esimerkiksi potentiaalisten asiakkaiden prospektointiin, asiakasseurantaan, yritysten analysointiin ja

asiakasrekisterien päivittämiseen. Vaatimuksena on, että datan tulee olla laadukasta, kattavaa ja ajantasaista.

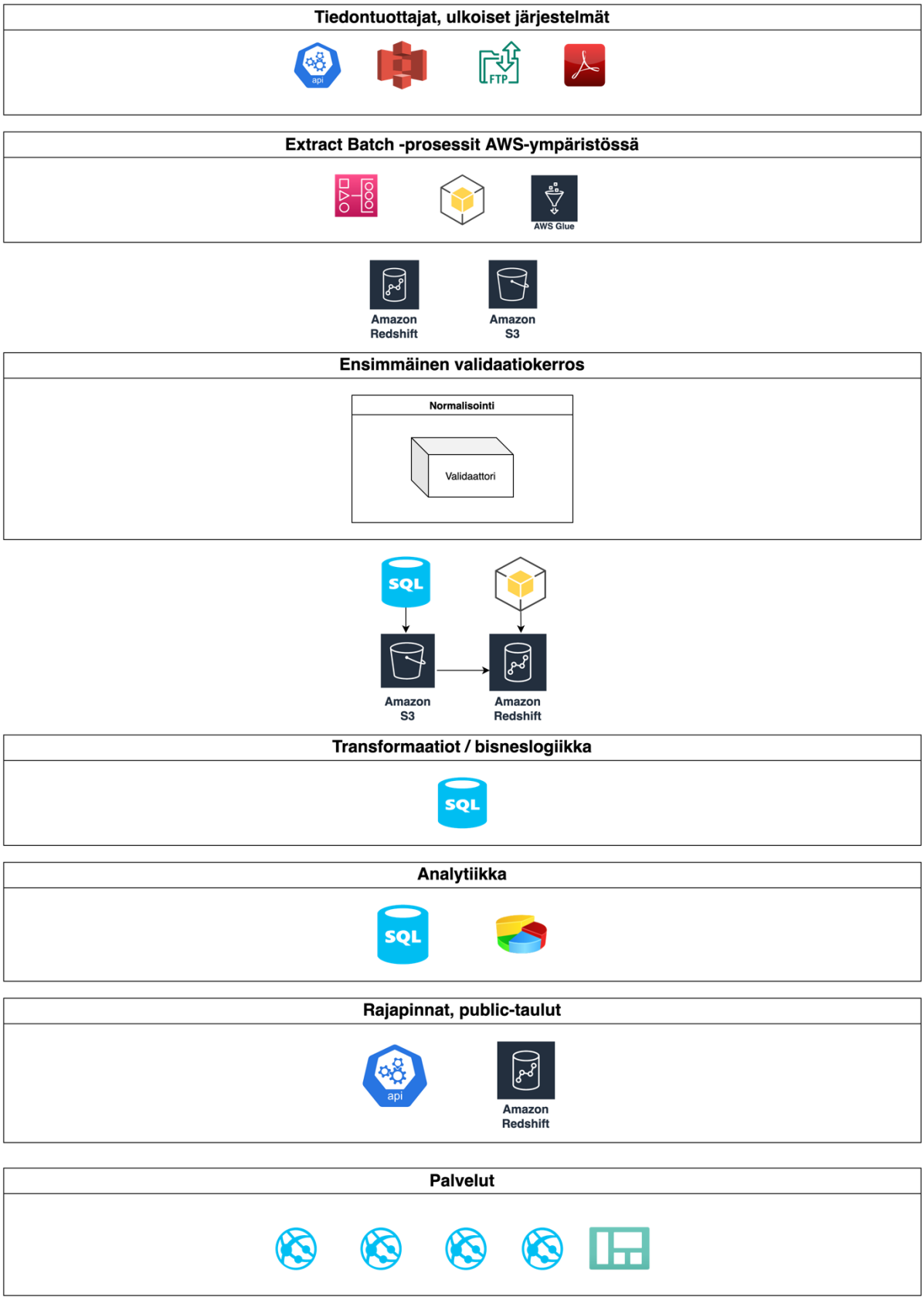
Datan validoinnilla pyrittiin saamaan näkyvyyttä lähtödatan ominaisuuksiin ja laatuun, ja puhdistus- sekä muokkaustoimenpiteillä haluttiin varmistaa, että data on loppukäyttäjien näkökulmasta luotettavaa. Lisäksi ulkoisiin tietolähteisiin kohdistuvalla validoinnilla pyrittiin vähentämään ongelmien selvittelyyn kuluvia työtunteja, jotta työtehtäviä voidaan jatkossa kohdistaa liiketoiminnallisesti kannattavampiin tehtäviin, kuten dataympäristön ja palvelun kehittämiseen.

## 5.2 Data-arkkitehtuuri ja tiedonsiirto

Tässä alaluvussa esitellään projektin käytössä oleva data-arkkitehtuuri (Kuva 3). Ulkoisia tietolähteitä ovat esimerkiksi rajapinnat, FTP-palvelimet, PDF-tiedostot ja Amazon Web Servicesin S3 -palvelu, joka oli myös tämän projektin kohteena olevan tiedontuottajan tapauksessa ratkaisuna. Amazon S3 on pilvipalveluna tarjottava objektipohjainen tallennusratkaisu, jonne tiedontuottaja toimittaa päivittäin dataa CSV-tiedostoina. CSV-tiedostot esikäsitellään ja muunnetaan Pythonin Pandas-kirjastoa käyttäen taulukoksi (DataFrame). Taulukko tallennetaan edelleen S3:een Parquet-tiedostomuotoon, johon päädyttiin, koska se on formaatiltaan sarakepohjainen ja soveltuu siten hyvin datan jatkojalostamiseen ja tietokantaan lataamiseen.

Tämän jälkeen arkkitehtuuri sisältää ensimmäisen validointikerroksen sekä puhdistamisoperaatiot, joita tullaan käsittelemään seuraavassa luvussa. Data ladataan tehtyjen toimenpiteiden jälkeen tietovarastona toimivaan Amazon Redshift -palveluun, joka on relaatiotietokantojen kysely- ja hallintajärjestelmä. Data kulkee tietovarastossa ensin tietojen latausalueelle, josta se jatkaa matkaansa kohti varsinaista liiketoimintalogiikkaa, jossa se on esimerkiksi analytiikkatyökalujen ja rajapintojen sekä niin sanottujen public-taulujen käytössä. Nämä ovat datan lopulliset sijoituspaikat, joista tiedot ovat loppukäyttäjien luettavissa eri palveluiden kautta. Dataa voi tarkastella myös tietokannasta suoraan SQL-kieltä käyttäen.

Kuva 3 Projektin data-arkkitehtuuri



## 5.3 Välineet

Toimeksiantaja tarjosi projektiin myös työvälineet ja ohjelmistot, ja projekti oli toteutettava olemassa olevan data-arkkitehtuurin puitteissa. Työasemana toimi Apple MacBook Pro M1, jossa käyttöjärjestelmänä oli Ventura 13.4.1. Tietovarastoinnissa oli laajasti käytössä Amazonin tarjoamia pilvipalveluita, kuten Redshift ja S3. Erilaisten työnkulkujen hallintaan on käytössä Apachen Airflow-palvelu. Data-arkkitehtuurin osalta pääasialliset ohjelmointi- tai tietokantakielet ovat Python ja SQL, joten niiden käyttö todettiin järkeväksi myös tässä työssä kuvatun projektin osalta. Datan profilointia tehtiin Redshiftissä SQL-kieltä käyttäen, ja datan puhdistamiseen käytettiin Python-ekosysteemiä ja esimerkiksi Polars-kirjastoa.

## 5.4 Menetelmät ja työtavat

Käytännön osan pääasiallinen menetelmä oli kehitysprojekti, joka toteutettiin toimeksiantona. Projektissa oli opinnäytteen kirjoittajan lisäksi mukana muutama muu datakehityksen parissa työskentelevä henkilö. Työtapojen osalta noudatettiin toimeksiantajan työmenetelmiä. Datakehitystiimissä oli käytössä iteratiivinen Scrum-menetelmää mukaileva kehitysprosessi. Työn sisältämä kokonaisuus suunniteltiin ja jaettiin toteutettaviin osiin, joista luotiin Jira-tiketit. Projektin edetessä tikettejä nostettiin työstöön. Koska projekti koski uuden tiedontuottajan tarjoaman datan tuomista osaksi nykyistä palvelukokonaisuutta, oli tavoitteena saada alkuvaiheessa valmiiksi MVP-tasoinen (minimum viable product) datakokonaisuus, eli sisällön osalta ensi vaiheessa loppukäyttäjien kannalta merkityksellisimmät tiedot. Tämän jälkeen tuotteen kehitys jatkuu ketterän kehitysprosessin mukaisesti, ja muutokset toteutetaan Git-versionhallinnan kautta.

Tiedontuottajan toimittaman datan validointiin sovellettiin teoriaosassa esiteltyjä datan laadun ulottuvuuksia ja datan profiloinnin menetelmiä. Validoinnin tavoitteena oli saada näkyvyyttä datan ominaisuuksiin ja löytää mahdolliset virheet. Validoinnin avulla pyrittiin tunnistamaan ne datan attribuutit, jotka haluttiin MVP-tuotekokonaisuuden kannalta tuoda saataville. Lähtödatasta laadittiin attribuuttikohtaisesti tilastoja, jossa pyrittiin löytämään numeerisista attribuuteista minimi- ja maksimiarvot sekä keskiarvo. Datan kattavuutta tutkittiin tarkastelemalla, esiintyykö datassa null-arvoja. Lisäksi tutkittiin attribuuttien tyyppiä, arvojen jakautumista ja useimmin esiintyviä arvoja. Taulujen väliset

riippuvuussuhteet haluttiin myös löytää. Profiloinnin tuloksena syntyneet havainnot ja vaadittavat toimenpiteet dokumentoitiin.

Dataan liittyvät siivous- ja muunto-operaatiot toteutettiin Python- ja SQL-kielillä. Projektin toteutusvaiheessa dokumentoitiin myös datan muuntamiseen ja virheidenkäsittelyyn liittyvät Python-funktiot. Tehtyjen toimenpiteiden vaikutuksia datan laatuun analysoitiin lopuksi.

Opinnäytetyön tekijä oli projektin aikana pääasiallisesti vastuussa datan validoinnista, mutta normalisointitoimenpiteitä jaettiin tiimin kesken toteutettavaksi. Opinnäytteen tekijä oli mukana esimerkiksi kirjoittamassa raakadatan muuntamiseen käytettyjä Python-funktioita sekä normalisoitujen tietojen yhdistämisessä harmonisoituun public-tauluun. Myöhemmin mainittavia koodistojen luomiseen ja henkilökäsittelyyn liittyviä toimenpiteitä ei työssä käsitellä tarkemmin, koska ne olivat itsessään varsin laajoja kokonaisuuksia, ja niiden tarkempaa avaamista ei nähty järkeväksi tämän työn laajuuden puitteissa, ja myös toimeksiantajan toiveena oli, ettei käsittelyjen sisältöä kuvata tutkimuksessa tarkasti.

## 5.5 Eettisyys ja säädökset

Vastuullisuus ja luotettavuus ovat kiinteä osa toimeksiantajan liiketoimintaa, joten projektiin liittyvän datan käsittelyssä otettiin huomioon lait, viranomaisten ohjeistukset ja määräykset. Käyttöön saatuja luottamuksellisia tietoja ei välitetty eteenpäin toimeksiantajan organisaation sisä- tai ulkopuolella.

Tietosuojaan ja tietoturvaan liittyvät seikat otettiin tarkasti huomioon datan käsittelyssä. Organisaatiossa on huomioitu, että tietovarastointi tapahtuu tietoturvallisesti. Pilvipalveluissa vastuu dataan liittyvästä tietoturvasta on palvelun käyttäjällä, joten tietojen salaamisesta on huolehdittu. Lisäksi dataan liittyvien pääsyoikeuksien osalta noudatetaan niin sanottua least privilege -periaatetta, jossa myönnetään tehtävän suorittamiseen vaadittavat pienimmät pääsyoikeudet. Näin huolehditaan, että kukaan ei pääse epähuomiossa tai tahallisesti esimerkiksi tuhoamaan dataa.

Lähtödata on lähtökohtaisesti yrityksiin ja vastuuhenkilöihin liittyvää viranomaisdataa, mikä huomioitiin projektin toteutuksessa siten, että itse datasisältöä ei opinnäytetyössä esitelty. Erityisesti henkilötietoa käsitellessä oli tärkeää huomioida EU:n yleinen tietosuoja-asetus (GDPR), joka asettaa yrityksille ja organisaatioille tarkkoja vaatimuksia henkilötietojen keräämiseen, säilytykseen ja hallinnointiin liittyen. Henkilöiden tai yritysten nimiä tai kytköksiä ei esitelty työssä. Myöskään toimeksiantajan tai tiedontuottajan nimiä ei tuotu työssä esille. Tietosisältöä ja datan sisältämiä attribuutteja kuvattiin yleisellä tasolla ilman oikeita tietosisältöä kuvaavia sarakeotsikoita. Datan sisältämät tarkat rivimäärät oli myös toimeksiantajan toiveesta jätettävä tutkimuksen ulkopuolelle, mikä piti huomioida raportin kirjoitusvaiheessa erityisesti datan validoinnin tuloksia ja normalisoinnin myötä puhdistettujen tietojen määriä kuvattaessa.

# 6 Datan validointi- ja puhdistamisprojektin toteutus

Toimeksiantajan tavoitteena oli muodostaa tiedontuottajan lähtödatan pohjalta yritysten perustiedot sisältävä tietokantataulu, jota kutsutaan jatkossa public-tauluksi. Taulun tarkoitus on toimia tietolähteenä analytiikkatyökaluille, rajapinnoille ja palveluille, joista data on loppukäyttäjien saatavilla. Tässä luvussa kuvataan lähtötilanne ja esitellään public-taulun edellyttämä tietosisältö, esitellään lähtödatalle tehdyt validointitoimenpiteet ja niiden pohjalta tehdyt korjaukset ja muuntotoimenpiteet, ja lopuksi kuvataan public-taulun muodostamiseen tehdyt toimenpiteet.

## 6.1 Tietosisältö

Tietosisällön näkökulmasta public-tauluun haluttiin tuoda saataville ulkomaisten yritysten tärkeimmät perustiedot sen pohjalta, mitä tietoja toimeksiantaja tarjoaa myös suomalaisista yrityksistä. Tietosisältö on kuvattu sen mukaan, missä muodossa data halutaan saataville tuoda.

Yritykselle haluttiin luoda tietokannan sisälle oma yksilöivä tunniste mahdollisen lähtödatassa olevan yritystunnuksen lisäksi. Toinen näistä toimisi myös public-taulun pääavaimena. Pääavain ei saa koskaan olla tyhjä, ja kaikkien arvojen tulee olla uniikkeja.

Suurin osa tietosisällöstä haluttiin tuoda tauluun VARCHAR-tyyppisenä merkkijonona. Yritystunnuksen alkamispäivä ja yrityksen perustamispäivä olivat kuitenkin tietoja, joiden tulisi olla päivämääriä, ja siten DATE-tyyppisiä.

Tiettyjen attribuuttien, kuten yrityksen maan, tilan, yhtiömuodon ja toimialan, osalta haluttiin public-tauluun tuoda näkyviin sekä koodiston mukainen arvo että selite. Koodin tuli sisältää etuliite ja arvoa ilmaiseva numero tai tekstilyhenne, esimerkiksi ”XX_2” tai ”XY_ABC”. Yrityksen tilan tuli olla lopullisessa public-taulussa joko ”Aktiivinen”, ”Lakannut” tai ”Muu”, ja jos tila olisi jotakin muuta kuin aktiivinen, olisi tuotava erillisiin sarakkeisiin päivämäärä, jolloin tila on muuttunut sekä tilan tarkempi selite. Yhtiömuoto ja virallinen toimiala haluttiin tuoda samaan tapaan saataville sekä koodimuotoisina että itse arvoina.

Tieto yrityksen maasta haluttiin generoida tauluun olemassa olevan koodiston mukaisena sen perusteella, minkä maan lähtödatasta tieto tuodaan.

Yrityksen nimi, toimialakuvaus, toimitusjohtajan nimi, kotisivu ja sähköpostiosoite haluttiin tuoda saataville pelkästään arvoina. Puhelinnumeron osalta vaatimuksena oli, että maakoodi ja puhelinnumero-osa erotellaan tarvittaessa omiin sarakkeisiinsa. Sijaintitietojen osalta toivottiin saataville sekä posti- että käyntiosoitteet niin, että osoitteista on eroteltu osoite, postinumero, toimipaikka, kaupunki ja maa omiin sarakkeisiinsa. Lisäksi public-tauluun haluttiin tuoda saataville sijaintikunta ja maakunta.

Public-taulun haluttiin sisältävän jokaiselle attribuutille muutospäivän, joka on tyyppiä TIMESTAMP, ja lähdetiedon tyyppiä VARCHAR. Lähde sisältää tiedon, mistä attribuutin muutos on peräisin. Tässä tapauksessa kuitenkin lähteenä on aina sama tiedontuottaja, joten näitä ei käsitellä projektin toteutuksen osalta tarkemmin.

## 6.2 Datan validointi

Tiedontuottaja toimittaa lähtödataa kolmesta eri maasta. Jokaisen maan osalta ladataan PSA-alueelle useita tiedostoja, jotka koostuvat esimerkiksi yritysten perustiedoista, osoitteista, rekisteröinneistä, taloustiedoista, henkilöistä ja henkilöiden rooleista suhteessa yrityksiin. Tiedostoista muodostetaan PSA-alueelle raakadatamuotoiset taulut, jotka toimivat validoinnin pohjana. Halutun tietosisällön perusteella todettiin, että public-taulun muodostamiseen tarvittavat attribuutit ovat löydettävissä maakohtaisesti kolmesta raakadataa sisältävästä taulusta, joihin viitataan jatkossa nimillä ”yritys”, ”henkilö” ja ”osoite”. Seuraavissa ala-alaluvuissa on kuvattu kyseisten taulujen rakennetta sekä pyritty profiloimaan sisältöä datan laadun ulottuvuuksien näkökulmasta. Dataa tutkittiin pääasiallisesti kokonaisuutena, joka sisälsi kaikkien maiden tiedot, koska näin tulisi olemaan lopullisessa public-taulussakin.

### 6.2.1 Taulujen rakenne ja riippuvuudet

Profilointi aloitettiin tekemällä yleiskatsaus taulujen sisältöön. Taulut vaikuttivat muodoltaan ja tietosisällöltään samankaltaisilta jokaisen maan lähdetietojen osalta. Tauluissa oli saman

verran sarakkeita, ja sarakkeet oli nimetty samoilla nimillä. Yritystasoisten tietojen osalta todettiin datan olevan kokonaisrivimäärältään melko kattavaa, sillä määrät olivat loogisia kotimaiseen dataan verrattuna.

Päätauluna toimii yritystasoisia tietoja sisältänyt taulu, ja se sisälsi suuren osan public-taulun edellyttämistä tiedoista:

- Yritystunnus
- Nimi
- Perustamispäivä
- Tila ja tilan alkamispäivä
- Yritysmuoto sekä lyhenteenä että selitteenä
- Sähköposti ja tieto, onko sähköposti rekisteristä piilotettu
- Puhelinnumero ja tieto, onko se rekisteristä piilotettu
- Kotisivu
- Toimiala sekä koodina että selitteenä
- Toimialaselite
- Tietojen päivityspäivämäärä

Kaikki yritystunnukset olivat uniikkeja, eivätkä ne sisältäneet tyhjiä arvoja. Taulusta löytyi kuitenkin ainoastaan yksi yrityksen perustamiseen viittaava sarake, joten perustamispäivää ja yritystunnuksen alkamispäivää ei saisi toisistaan erillisinä tietoina. Raakadata oli ladattu tietovarastoon VARCHAR-muodossa eli merkkijonoina, joten oli todettavissa, että päivämäärien osalta data ei vielä täyttänyt muodollisia vaatimuksia, koska public-tauluun perustamis- ja yritystunnuksen alkupäivä haluttiin DATE-muodossa.

Osoite-taulu puolestaan sisälsi yrityksiin liittyvää taulua vastaavan yritysavaimen, joten jokaiselle yritystunnukselle oli löydettävissä osoite tai vähintään vastaavuus tietuetasolla. Osoite vaikutti siis olevan yritystasoinen tieto. Osoitetiedoissa oli kolme erilaista kaupunkiin tai kuntaan liittyvää saraketta, joista yksi vaikutti postitoimipaikalta ja yksi sijaintikunnalta, mutta yhden osalta merkitys jäi epäselväksi. Taulusta ei löytynyt erikseen posti- ja käyntiosoitteita. Lisäksi katujen nimet ja numerot olivat eri sarakkeissa. Taulusta löytyi myös osoitteille maatieto sekä laajempaa aluetta tai maakuntaa vastaava tieto.

Henkilöihin viittaava taulu sisälsi henkilölle yksilöivän avaimen sekä yritystunnuksen, henkilön tyypin (yritys tai luonnollinen henkilö), henkilön nimen ja esimerkiksi muutospäivän. Koska public-tauluun haluttiin tuoda toimitusjohtajan nimi, tarvittiin lisäksi henkilöiden roolit sisältävä taulu, joka sisälsi myös henkilön yksilöivän avaimen sekä yritystunnuksen, roolitiedon sekä roolin alku- ja loppupäivämäärän. Henkilöön viittaavat taulut olivat jokaisen maan kohdalla rivimäärällisesti pienempiä kuin yritys- tai osoitetaulut, joten kaikille yrityksille ei ollut saatavissa vastuuhenkilötietoja. Toisaalta rooleja sisältäneet taulut olivat rivimäärältään huomattavasti suurempia kuin henkilö- tai yritystasoiset taulut, joten henkilöllä saattoi olla useampi kuin yksi rooli, ja taulu sisälsi myös jo päättyneitä rooleja.

Lähdetaulujen riippuvuuksia analysoitiin tutkimalla viittausavaimia. Yritys- ja osoitetaulut olivat täysin riippuvaisia toisistaan yritystunnuksen perusteella. Koska yritystunnuksesta haluttiin tehdä public-taulun pääavain, joka ei saa olla tyhjä tai sisältää duplikaatteja, pystyttiin lähtödata toteamaan tämän osalta eheäksi ja ainutlaatuiseksi.

Henkilö-taulu taas kytkeytyi yritystasoiseen tauluun yritysavaimella, ja jokaisella henkilöesiintymällä oli oma henkilöavain. Henkilöavain oli tulkittavissa taulun pääavaimeksi, kun taas yritysavain oli viiteavain. Henkilöavaimiin ei ollut kytkeytynyt useampia yrityksiä, joten lähtödatassa olevat henkilöiden ilmentymät kytkeytyivät aina yhteen yritykseen. Koska todellisuudessa henkilö voi kytkeytyä useampaan kuin yhteen yritykseen vastuuhenkilönä, tulkittiin että lähtödatassa henkilö voi saada eri yrityksissä eri henkilöavaimen.

### 6.2.2 Null-arvojen tutkiminen

Kaikkien tietojen osalta tutkittiin, voivatko attribuutit olla lähtödatassa tyhjiä eli saada arvokseen nullin. Yritystasoisten tietojen osalta yritystunnus oli jo todettu uniikiksi arvoksi, joka ei ole koskaan tyhjä minkään maan osalta, joten sitä ei ollut tarpeen tutkia tarkemmin. Todettiin myös, että muutospäivämäärä sisälsi aina jonkin arvon. Perustamispäivän, yrityksen tilan ja yhtiömuotojen osalta nulleja esiintyi huomattavan vähän, alle 1 % kokonaisrivimäärästä. Yritysnimen sekä toimialakoodin ja -selitteen osalta tyhjien arvojen osuus jäi alle 20 prosenttiin, mutta puhelinnumeron ja toimialakuvauksen kohdalla osuus nousi 70–80 prosenttiin sekä kotisivujen ja sähköpostien osalta jopa yli 90 prosenttiin.

Osoitteiden osalta tyhjien tietojen osuus vaihteli pääasiassa noin 25 prosentista 30 prosenttiin, mutta yhden kuntaan viittaavan attribuutin osalta yli puolet oli tyhjiä. Henkilöiden tai roolien osalta ei tarkistettu null-arvoja erikseen, koska tässä yhteydessä ei oltu kiinnostuneita niinkään kyseisten taulujen kokonaisuudesta, vaan siitä, kuinka monelle yritykselle olisi löydettävissä toimitusjohtaja. Tästä lisää seuraavassa ala-alaluvussa.

Null-arvojen osalta todettiin, että tyhjät yritysnimet vaikuttivat kytkeytyvän yrityksen tilaan, koska näistä vain yhden tila oli merkitty aktiiviseksi ja loput olivat tulkittavissa lakanneiksi. Tämä yksi poikkeustapaus todettiin datan sisältämäksi poikkeusarvoksi. Tila-attribuutin tyhjien arvojen osalta tietueen yritysnimiattribuutti ei kuitenkaan vastavuoroisesti ollut tyhjä. Toimialakuvauksia, kotisivuja, sähköpostiosoitteita ja puhelinnumeroita puuttui huomattavan monelta yritykseltä, mikä oli odotettavissa, koska näin on kotimaisenkin virallisen yritystiedon kohdalla.

Null-arvojen tutkimisella pystyttiin löytämään muutamia poikkeusarvoja sekä tuottamaan yleiskatsaus datan kattavuudesta. Yritystasoisten tietojen osalta kattavuutta pidettiin varsin hyvänä, sillä sarakkeet, joista tietoja puuttui eniten, olivat myös kotimaisen datan osalta niitä, joista tietoa yleisesti ottaen vähemmän saatavilla kuin muista yrityksen perustiedoista. Osoitteiden osalta puuttuvat arvot eivät nekään olleet määrältään liian suuria.

### 6.2.3 Toimitusjohtajat

Henkilöiden osalta oltiin tässä vaiheessa erityisen kiinnostuneita siitä, kuinka monelle yritykselle oli löydettävissä toimitusjohtaja. Datan oikeellisuuden ja ajanmukaisuuden kannalta oli tärkeää, että public-tauluun tulisi uusin ja voimassa oleva tieto, joten henkilödatan osalta tarkistettiin, monelleko yritykselle löytyi toimitusjohtaja. Tässä yhteydessä roolin loppupäivämäärän tuli olla tulevaisuudessa tai tyhjä. Kokonaisuutena rooli löytyi alle 10 prosentille yrityksistä. Datan oikeellisuuden kannalta heräsi kysymys, onko henkilödata tämän osalta puutteellista. Toisaalta todettiin myös, että eri maissa saattaa olla yrityksen vastuurooleihin ja virallisiin rekistereihin liittyen erilaisia käytäntöjä, jolloin suuri ero voisi olla selitettävissä tällä. Asia päätettiin kuitenkin jättää jatkokehityksen yhteydessä mietittäväksi.

### 6.2.4 Arvojen kategorisointi

Yritystasoisten taulujen muuttujista tila, yhtiömuoto ja toimiala ovat selvästi diskreettejä muuttujia, jotka voivat saada ainoastaan tietyn arvon. Näiden attribuuttien osalta tutkittiin, mitä arvoja ne voivat lähtödatassa saada, ja miten arvot jakautuvat kategorioiden välillä.

Yrityksen mahdollisten tilojen määrä vaihteli maiden välillä viidestä kahteenkymmeneen, ja tila saattoi olla myös tyhjä. Suuri osa yrityksistä oli tilan perusteella tulkittavissa joko aktiiviseksi tai lakanneeksi, mutta esimerkiksi fuusioihin ja konkurssiin ajautumisen myötä tila saattoi olla jotain aktiivisen ja lakanneen väliltä. Kategoriat jaettiin kolmeen yläkategoriaan (aktiivinen, lakannut, muu) ja laskettiin arvojen jakautuminen kuhunkin yläkategoriaan. Lakanneiksi tulkittavia oli 50 %, aktiivisia 40 %, muita 1 % ja tyhjiä 9 %. Muiden kuin tyhjien arvojen osalta määrät vertautuivat kotimaiseen yritysdataan varsin hyvin, mutta tyhjien arvojen osalta data todettiin hieman puutteelliseksi.

Yhtiömuotojen osalta jokaisen maan lähtödata sisälsi sekä lyhenteen että alkuperäiskielisen selitteen. Yhtiömuodot olivat osittain yhteneväisiä maiden välillä, mutta niitä kuitenkin oli eri maissa eri määrä. Data kategorisoitiin sen perusteella, onko yritystunnukselle saatavilla lyhenne ja selite, vain toinen niistä tai ei kumpaakaan. Todettiin, että selvästi suurimpaan osaan yritystasoisesta datasta oli saatavilla sekä lyhenne että selite. Joillekin yhtiömuodon selitteelle ei löytynyt tietuetasolla lyhennettä, ja lisäksi muutama lyhenne saattoi viitata useampaan kuin yhteen kokomittaiseen yhtiömuotoon. Viimeksi mainitut olivat kuitenkin tulkittavissa samoiksi yhtiömuodoiksi. Kummankin tiedon puuttuminen oli jo erittäin harvinaista ja tulkittavissa poikkeusarvoiksi. Data todettiin tämän tiedon osalta varsin eheäksi ja johdonmukaiseksi, mutta täydellistä se ei ollut.

Kolmas selvästi diskreetti muuttuja oli yrityksen virallinen toimiala, joka oli yritystasoisissa tauluissa saatavilla sekä numeerisena koodina että selitteenä. Data kategorisoitiin samalla tavalla kuin yhtiömuotoihin liittyvä data tarkistamalla, löytyykö tietueista sekä toimialakoodi että -selite. Jokaisen maan osalta näytti löytyvän joko molemmat tai ei kumpaakaan. Itse yhtiömuotoja oli eri maiden datassa eri verran. Osa koodeista saattoi kytkeytyä useampaan kuin yhteen selitteeseen, mikä vaikutti datan eheyden ja oikeellisuuden kannalta ongelmalliselta.

### 6.2.5 Päivämäärää ja aikaa kuvaavien attribuuttien profilointi

Koska suurin osa public-tauluun vietävästä tietosisällöstä tulisi olemaan VARCHAR-tyyppiä, ei minimi- ja maksimiarvojen laskeminen olisi ollut järkevää useimpien attribuuttien kohdalla. Päivämäärien osalta oli kuitenkin syytä tarkastella näitä tietoja tarkemmin tutkimalla minimi- ja maksimiarvoja sekä yleisimpiä esiintymiä sekä mahdollisia poikkeusarvoja.

Perustamispäivään viittaava sarake oli muodollisesti vaatimustenmukainen, sillä vaikka se lähtödatassa olikin tyyppiä VARCHAR, olivat kaikki tietueet muunnettavissa SQL-kyselyllä päivämääriksi. Erityisesti attribuutin minimiarvot herättivät kysymyksiä tietojen oikeellisuudesta. Kahden maan kohdalla minimiarvo oli ”0001-01-01” ja kolmannen kohdalla ”1277-09-13”. Maksimiarvo taas oli yhden maan kohdalla tulevaisuudessa, kun taas muiden kohdalla joko tutkimista edeltäneenä päivänä tai edeltävänä perjantaina.

Todettiin aiheelliseksi tutkia sarakkeessa esiintyviä yleisimpiä arvoja ja sitä, miten arvot jakautuvat aikajanalla. Näin haluttiin selvittää, esiintyykö tiedoissa poikkeus- tai oletusarvoja. Perustamispäivä-attribuutista etsittiin kymmenen yleisimmin esiintyvää arvoa jokaisen maan osalta. Minimiarvo ”0001-01-01” tuli esiin yhtenä yleisimmistä arvoista, ja arvo ”1901-01-01” esiintyi myös useasti. On mahdollista, että eri maissa on alettu keräämään kyseistä tietoa rekistereihin eri aikoina, mikä aiheuttaa kasautumia tietyille päivämäärille. Rivimäärällisesti esimerkiksi ”0001-01-01”:n esiintymät eivät olleet kovin suuria suhteessa kokonaisrivimäärään, joten tietojen nähtiin olevan edelleen suhteellisen oikeellisia. Kuitenkin ”0001-01-01”-arvon todettiin olevan jonkinlainen rekisteriin tai datasettiin syötetty oletusarvo, kun oikeaa tietoa ei ole ollut saatavilla.

Datan järkevyyden ja oikeellisuuden kannalta oli tarkistettava, esiintyykö tiedoissa poikkeusarvoja. Tätä tutkittiin etsimällä sarakkeesta uniikit arvot ja niiden esiintymien määrä. Listausta tutkiessa havaittiin, että sarakkeissa esiintyi yksittäisiä päivämääriä jo 1200-luvulta alkaen, mutta 1900- ja 2000-luvuille tultaessa yksittäisten päivämäärien esiintymien määrä kasvoi selvästi. Datassa esiintyi myös joitakin tulevaisuuteen sijoittuvia arvoja, joten tiettyjen maiden rekistereissä näytti olevan mahdollista ilmoittaa perustamispäivä ennakkoon.

Muiden ajankohtaa kuvaavien sarakkeiden osalta data vaikutti laadukkaalta. Perustamispäivän sisältämistä poikkeusarvoista huolimatta tiedot vaikuttivat vastaavan tosielämää, joten datan voitiin päivämäärien osalta todeta olevan varsin oikeellista. Tiedot vaikuttivat myös vastaavan oletuksia, joten tältä osin data vaikutti järkevältä.

### 6.2.6 Sallitut ja ei-sallitut arvot

Tiettyjä attribuutteja, kuten yritysnimiä, toimialakuvauksia, kotisivuja, sähköpostiosoitteita, puhelinnumeroita ja osoitetietoja, haluttiin tarkastella erityisesti vaatimustenmukaisuuden ja tietojen oikeellisuuden näkökulmasta.

Yritysnimien osalta oltiin erityisen kiinnostuneita alku- ja loppumerkeistä, sillä loppuasiakkaiden nähtävissä olevan datan haluttiin olevan mahdollisimman siistiä ja sellaista, jota palvelujen hakutoiminnallisuus pystyy käsittelemään. Välilyönteihin alkavia tai päättyviä yritysnimiä ei vaikuttanut juuri löytyvän minkään maan osalta, mutta osittain kyseenalaiselta vaikuttavaa tietoa näytti kuitenkin löytyvän, sillä alku- tai loppumerkkinä esiintyy esimerkiksi lainaus-, ½- ja et-merkkejä sekä kenoviivoja, pilkkuja ja kaksoispisteitä. Näiden esiintymien osuus jäi kuitenkin datan kokonaisrivimäärään verrattuna alle yhteen prosenttiin.

Toimialakuvausten osalta huomattiin, että merkkijonojen pituudet vaihtelivat yhdestä noin kymmeneen tuhanteen merkkiin. Alku- ja loppumerkeissä esiintyi kirjainten lisäksi lukuisia erilaisia erikoismerkkejä, esimerkiksi tapauksessa, jossa yritys oli ilmoittanut harjoittamansa toiminnan väliviivoin: ”- Kirjanpito- ja veroneuvonta - Digitaalinen markkinointi”. Nämä tulkittiin datan ominaisuudeksi eikä virheiksi.

Kotisivujen osalta oltiin kiinnostuneita, täyttivätkö tiedot muodolliset vaatimukset esimerkiksi siitä, missä muodossa tieto on ilmoitettu. Tiedoista kuitenkin löytyi joitakin maakohtaisia eroja. Yhden maan osalta yleisin muoto oli ”domain.fi”, joten merkkijonot eivät muutamia poikkeuksia lukuun ottamatta juurikaan alkaneet ”www”- tai ”http”-merkkijonoilla. Muiden maiden osalta taas selvästi suurin osa kotisivuista alkoi ”www”-merkkijonolla, mutta suoraan domain-tunnuksella alkaviakin oli molempien osalta. ”http”-merkkijonolla alkavia kotisivuja löytyi niistäkin hyvin vähän. Kokonaismäärällisesti ”www”-

yhdistelmä merkkijonon alussa oli kuitenkin pelkkää domainia yleisempi. Muoto ei siis maiden välillä ollut täysin yhtenäinen, mutta data kuitenkin vaikutti täyttävän muodolliset vaatimukset suurilta osin. Esiin nousi silti joitakin yksittäisiä arvoja, joissa kotisivu esiintyi esimerkiksi muodossa ”www-domain.fi”.

Sähköpostien osalta vaatimustenmukaisuutta tarkasteltiin tutkimalla, ovatko sähköposti-attribuutin arvot tulkittavissa sähköpostiosoitteiksi. Tutkiminen aloitettiin tarkistamalla, sisältävätkö osoitteet @-merkin. Merkki puuttui alle prosentilta kaikista niistä, joille oli löydettävissä sähköpostiosoite. Tässä joukossa oli mukana selvästi virheellisiä sähköposteja ja verkko-osoitteita, mutta joissain tapauksissa @-merkki ilmaistiin kaari- tai hakasulkeilla, esimerkiksi ”(at)”. Sähköpostien alku- ja loppumerkkien joukosta löytyi jonkin verran väliviivoja ja muita erikoismerkkejä, vaikka ne olisivatkin @-merkin sisältäneet. Data ei täyttänyt muodollisia vaatimuksia täysin, mutta koska kyse oli yritysten itse rekisteriin ilmoittamista tiedoista, todettiin ettei dataa voisi muokata kovin paljon.

Puhelinnumerot näyttivät lähes poikkeuksetta olevan aina samassa muodossa, eli alussa oli +-merkillä alkava maakoodi, jonka perässä suoraan puhelinnumero-osa ilman välilyöntejä. Pieni osa alkoi suoraan numerolla, ja niistä näyttikin vain puuttuvan maakoodi. Näiden osalta tiedot havaittiin kuitenkin vaatimustenmukaisiksi, sillä attribuutit eivät sisältäneet merkkijonon keskellä erikoismerkkejä tai kirjaimia.

Osoitetietojen osalta todettiin, että datan oikeellisuutta on vaikea validoida ilman erillistä kunkin maan kaikki osoitteet ja postinumerot kattavaa rekisteriä. Huomattiin kuitenkin, että yhden maan kohdalla postinumeroa ilmaiseva merkkijono päättyy aina ”.0”, eli esimerkiksi kuvitteellinen postinumero 1234 olisi ollut muodossa ”1234.0”. Toisen maan osalta postinumerot sisälsivät välilyönnin keskellä merkkijonoa, kun taas kolmannen kohdalla postinumerot sisälsivät pelkkiä numeroita. Data ei ollut maiden välillä täysin yhtenevää, ja etenkään ”.0”-päätteiden kohdalla se ei ollut vaatimustenmukaistakaan. Katunumeroiden osalta tiedot sisälsivät sekä numeroita, että tarkennuksia esimerkiksi taloista ja rapuista. Kadunnimien ja kaupunkien osalta oli havaittavissa joitakin erikoismerkkejä sekä alussa että lopussa, ja sama havainto tehtiin myös sijaintikuntien ja alueiden osalta.

### 6.2.7 Validoinnin johtopäätökset

Tiedontuottajan toimittaman lähtödatan validoinnin jälkeen syntyi käsitys datan laadusta ja ominaisuuksista. Validoinnin pohjalta tehtiin päätökset siitä, mitä toimenpiteitä datan siivoamisen ja normalisoinnin osalta tulisi tehdä. Koska kukin toimenpide saattoi koskea yhtä tai useampaa saraketta, kategorisoitiin toimenpiteet ja tarkistettiin, mitä sarakkeita kukin toimenpide koskee. Toimenpiteet on kuvattu tämän ala-alaluvun lopussa (Taulukko 1).

Lähtökohtana oli, että jokaisen maan lähtödata tulisi yhdistää public-tauluun, josta olisi saatavilla yritystasoiset perustiedot kaikista kolmesta maasta. Tämän mahdollistamiseksi päätettiin luoda normalisoinnin jälkeen tietovaraston latausalueelle yrityksistä, osoitteista, henkilöistä ja rooleista koontitaulut, jotka sisältävät kaiken lähtödatan, ja joihin on jo tehty osa suuri osa puhdistustoimenpiteistä.

Lähtödatan yritystunnuksesta oli tehtävä public-taulun pääavain, koska se oli ainoa saatavilla oleva yksilöivä yritystasoinen tunnus. Koska kotimaisen datan osalta käytössä on myös erillinen yritys-ID, päätettiin myös sille luoda oma sarake. Alkuvaiheessa erillinen yritys-ID olisi kuitenkin sama kuin lähtödatan alkuperäinen yritystunnus, ja päätös erillisen yritys-ID:n muodostamisesta tullaan tekemään jatkokehityksen yhteydessä.

Validoinnin yhteydessä havaittiin, että osa tiedoista, kuten yritysnimet ja osoitteet, sisälsivät alku- ja loppumerkkejä, joita ei haluttu public-taulussa näyttää. Näihin päätettiin luoda siivousfunktiot, jotka on kuvattu tarkemmin myöhemmin. Lisäksi kaikesta lähtödatasta tulevasta tietosisällöstä päätettiin siivota välilyönnit sekä alusta ja lopusta. Osa datasta ei vaikuttanut selvästi edellyttävän välilyöntien siivoamista, mutta se todettiin helpoksi tavaksi varmistaa, että data säilyisi tältä osin laadukkaana, jos välilyöntejä esiintyisi myöhemmin sarakkeissa, joissa niitä ei validoinnin aikana ollut.

Päivämääriä sisältävät tiedot päätettiin normalisoinnin yhteydessä muuntaa DATE-tyyppiin, koska ne olivat lähtödatassa VARCHAR-tyyppiä. Näitä tietoja olivat yrityksen perustamispäivä ja y-tunnuksen alkamispäivä. Lähtödatassa ei ollut tosin saatavilla kuin perustamispäivä, joten se päätettiin tuoda public-tauluun myös y-tunnuksen alkamispäivänä, koska kotimaisessa datassa on saatavilla molemmat. Taulurakenteen yhtenäisyyden vuoksi haluttiin mukaan molemmat, ja yrityksen perustamispäivä sekä y-tunnuksen alkamispäivä

vastasivat tietoina tarpeeksi hyvin toisiaan. Lisäksi perustamispäivän osalta haluttiin muuttaa lähtödatassa esiintynyt oletusarvo ”0001-01-01” tyhjäksi. Lisäksi ei-aktiivisten yritysten osalta haluttiin tuoda näkyville päivämäärä, jolloin tila on viimeksi muuttunut. Tämä oli saatavilla yritystasoisten tietojen viimeisimmästä muutospäivämäärästä, joka oli perustamispäivän tapaan muunnettava VARCHAR-tyyppisestä DATE-tyyppiseksi.

Katuosoitteiden ja puhelinnumeroiden osalta todettiin, että lähtödatan sarakkeita tulee joko yhdistää tai erotella, jotta tiedot saataisiin public-tauluun halutussa muodossa. Katuosoite vaati katunimen ja katunumeron yhdistämistä, ja puhelinnumeroista taas tuli erotella maakoodi ja puhelinnumero-osa.

Toimitusjohtajien osalta taas tuli etsiä yhdistetyistä henkilö- ja roolitauluista uusin ja voimassa oleva tieto. Toimitusjohtajan nimi todettiin riittäväksi tiedoksi yrityksen perustietojen osalta.

Maa, yrityksen tila, yhtiömuoto ja virallinen toimiala edellyttivät erillisten koodistojen luomista tietovaraston sisälle. Maakoodeihin, yrityksen tilaan ja yritysmuotoon liittyvät koodistot päätettiin luoda lähtödatan pohjalta. Toimialoista taas oli saatavilla erillinen virallinen luettelo, jota voitiin hyödyntää toimialakoodiston osalta.

Taulukko 1 Public-taulun normalisointitoimenpiteet

| **Toimenpide** | **Taulut/sarakkeet, joita toimenpide koskee** |
|---|---|
| Taulujen yhdistäminen | Yritykset, osoitteet, henkilöt, roolit |
| Pääavaimen luonti | Yritystunnus |
| Välilyöntien siivous alusta ja lopusta | Kaikki sarakkeet |
| Alku- ja loppumerkkien siivous | Yritysnimi, Osoite: katu, katunumero, kaupunki, toimipaikka |
| Sarakkeiden tietotyyppien muuntaminen | Perustamispäivä, tietojen muutospäivä |
| Oletusarvojen korvaaminen | Perustamispäivä |
| Sarakkeiden yhdistäminen | Katunimi ja -numero |

| | |
|---|---|
| Sarakkeiden erottelu | Puhelinnumero: maakoodi ja puhelinnumero-osa |
| Koodistojen luonti | Maa, yrityksen tila, yhtiömuoto, toimiala |
| Voimassa olevan toimitusjohtajan etsiminen | Toimitusjohtajan nimi |

## 6.3 Datan normalisointi

Tässä kappaleessa on kuvattu datan validoinnin tuloksena syntyneet normalisointi- ja siivoustoimenpiteet sekä se, missä vaiheissa tietovarastointiprosessia kukin operaatio tapahtuu. Siivousoperaatiot on kuvattu siinä järjestyksessä, kuin ne tapahtuvat.

### 6.3.1 Normalisointi- ja siivousoperaatiot

Tietojen puhdistamiseen ja muuntamiseen liittyvät operaatiot toteutettiin tietovaraston ensimmäisessä validointikerroksessa. Datan siivoamiseen liittyvät Python-funktiot luotiin erilliseen Python-tiedostoon, josta niitä oli mahdollista ajaa halutuille tiedontuottajan lähtödatan attribuuteille. Funktioiden todelliset ja tarkoitusta paremmin kuvaavat nimet on korvattu kuvitteellisilla nimillä.

Postinumeroista haluttiin poistaa erityisesti”.0” merkkijonon lopusta, jos sellainen esiintyy (Ohjelmakoodi 1). Funktio lukee value-parametrilla merkkijonon, jonka se split()-metodia käyttäen jakaa kahteen osaan käyttämällä pistettä erottimena. Metodin toinen argumentti ”1” kertoo, että erotus tapahtuu vain kerran. Tulos tallennetaan res-nimiseen lista-tyyppiseen muuttujaan, ja lopuksi funktio palauttaa erotuksen tuloksena syntyneen pistettä edeltävän alkuosan, joka löytyy listan ensimmäisestä elementistä [0]. Todellisuudessa funktio ei siis korvaa pelkästään ”.0”-päätettä tyhjällä, vaan pisteen jälkeinen osa voi olla mitä vain.

Ohjelmakoodi 1 H".0"-päätteen korvaaminen tyhjällä Python-kielellä

```
def funktio_1(value):
    res = value.split('.', 1)
    return res[0]
```

Postinumeroiden osalta haluttiin myös korvata välilyönti tyhjällä. Tässä käytettiin Python-funktiota, joka lukee sisään value-parametrin, josta se replace()-metodia käyttäen korvaa välilyönnit tyhjällä sekä alusta, keskeltä että lopusta (Ohjelmakoodi 2). Tulos tallentuu res-muuttujaan, jonka funktio palauttaa sitä kutsuttaessa.

Ohjelmakoodi 2 Välilyöntien korvaaminen tyhjällä Python-kielellä

```
def funktio_2(value):
    res = value.replace(" ", "")
    return res
```

Perustamispäivä-attribuutin osalta löydettiin isohko joukko arvoja, jotka todettiin dataan lisätyiksi oletusarvoiksi. Tätä varten luotiin funktio, joka muuttaa erikseen määriteltävät oletusarvot tyhjiksi (Ohjelmakoodi 3). Funktio ottaa sisäänsä kaksi parametria: default ja value. Default-parametriin voidaan määritellä merkkijono, jota funktio vertaa value-parametrin sisältämään arvoon. Jos lähtödatan arvo on sama kuin oletusarvo (tässä tapauksessa ”0001-01-01”), palauttaa funktio None-arvon, joka Python-kielessä edustaa tyhjää tai null-arvoa. Jos lähtödatan arvo on jotakin muuta, palauttaa funktio alkuperäisen arvon.

Ohjelmakoodi 3 Oletusarvojen muuttaminen tyhjäksi Python-kielellä

```
def funktio_3(default, value):
    if value == default:
        return None
    else:
        return value
```

Joidenkin attribuuttien osalta havaittiin alku- ja loppumerkkejä, jotka haluttiin poistaa. Tätä varten luotiin funktio, joka value-parametrin sisään luettuaan korvaa strip() -metodia käyttäen arvon alusta ja lopusta kyseiset merkit (Ohjelmakoodi 4). strip() todettiin tässä tapauksessa replace() -metodia sopivammaksi vaihtoehdoksi, koska replace() poistaa merkit kaikkialta, ei vain alusta ja lopusta. Tässä tapauksessa metodille annettiin poistettaviksi merkeiksi pilkut, kaarisulut, välilyönnit, vino- ja kenoviivat, lainaus- sekä puolilainausmerkit, väliviivat, ½-merkit, et-merkit sekä kaksoispisteet. Funktio palauttaa alkuperäisen arvon, jonka alusta ja lopusta on korvattu kyseiset erikoismerkit tyhjällä.

Ohjelmakoodi 4 Ei-sallittujen merkkien poistaminen Python-kielellä

```
def funktio_4(value):
```

```
    res = value.strip(",() \\/\"\'-½&:")
    return res
```

Joissain tapauksissa haluttiin määritellä poistettavat alku- ja loppumerkit funktiota kutsuttaessa, joten luotiin pitkälti edellistä vastaava funktio (Ohjelmakoodi 5), joka ottaa sisään yhden sijasta kaksi parametriä: chars ja value. Chars-parametriin määritellään funktiokutsussa tyhjällä korvattavat merkit, jotka strip() -metodi lukee parametristä. Samoin kuin edellä, funktio korvaa alkuperäisestä arvosta erikseen määritellyt merkit ja palauttaa muunnetun arvon.

Ohjelmakoodi 5 Toinen tapa poistaa ei-sallittuja merkkejä Python-kielellä

```
def funktio_5(chars, value):
    res = value.strip(chars)
    return res
```

Käytännössä jokaisesta lähtödatan attribuutista haluttiin poistaa alussa ja lopussa olevat välilyönnit ja vastaavan kaltaiset merkit, joita kutsutaan englanniksi termillä whitespace. Tätä varten luotiin funktio, joka value-parametrin sisään luettuaan poistaa strip() -metodia käyttäen joko alkuperäisen arvon, josta whitespace-merkit on siivottu, tai tyhjän, jos alkuperäinen arvokin on tyhjä (Ohjelmakoodi 6).

Ohjelmakoodi 6 Välilyöntien poistaminen arvon alusta ja lopusta Python-kielellä

```
def funktio_6(value: str) -> str:
    return value.strip() or None
```

Yrityksen perustamispäivä -attribuutti oli tarpeen muuntaa merkkijonosta päivämäärän tyyppiseksi, jota varten luotiin myös Python-funktio (Ohjelmakoodi 7). Funktio lukee sisään kolme parametria: col_name, format ja df, ja sen tarkoitus on muuntaa Polars-dataframessa (df) oleva string-tyyppinen sarake (col_name) Date-tyyppiseksi sarakkeeksi (format). Funktion ensimmäinen rivi luo logging-moduulia käyttäen informatiivisen viestin, joka ilmoittaa funktion muuntavan col_name-parametrissa ilmaistua saraketta Polars-kirjaston Date-tyyppiin. Itse muuntaminen tapahtuu funktion toisella rivillä, jossa with_columns() -metodia käyttäen res_df-nimiseen muuttujaan luodaan uusi muunnetun sarakkeen sisältävä taulukko. Sarake muunnetaan Date-tyyppiseksi str.strptime()-metodilla, jossa sarakkeen string-tyyppiset arvot parsitaan format-parametrissa määritellyn muodon mukaisiksi. Parsimisen jälkeen cast() -metodi muuntaa luodun sarakkeen Polars-kirjaston Date-tyypiksi.

Jos jokin sarakkeen arvo ei ole muunnettavissa haluttuun Date-tyyppiin, parametri ”strict=True” aiheuttaa virheen ja ohjelman suoritus lakkaa. Funktio palauttaa alkuperäisen taulukon, jossa col_name-parametrissa määritelty sarake on muunnettu Polars-kirjaston Date-tyyppiseksi.

Ohjelmakoodi 7 Merkkijonon muuntaminen Date-tyyppiseksi Python-kielellä

```
def funktio_7(col_name: str, format: str, df: pl.DataFrame) ->
pl.DataFrame:
    logging.info(f"Casting {col_name} to pl.Date")
    res_df = df.with_columns(pl.col(col_name).str.strptime(pl.Date,
format=format).cast(pl.Date, strict=True))
    return res_df
```

Yrityksen tilan muutosajankohdan osalta ei päivämäärä kuitenkaan riittänyt ajankohdaksi, vaan data sisälsi myös kellonajan. Tätä varten luotiin tismalleen samankaltainen funktio (Ohjelmakoodi 8), joka muuntaa string-tyyppisen arvon Polars-kirjaston Datetime-tyyppiin.

Ohjelmakoodi 8 Merkkijonon muuntaminen Datetime-tyyppiseksi Python-kielellä

```
def funktio_8(col_name: str, format: str, df: pl.DataFrame) ->
pl.DataFrame:
    logging.info(f"Casting {col_name} to pl.Datetime")
    res_df =
df.with_columns(pl.col(col_name).str.strptime(pl.Datetime,
format=format).cast(pl.Datetime, strict=True))
    return res_df
```

### 6.3.2 Taulujen yhdistäminen

Yritystasoisten tietojen osalta muodostettiin latausalueelle yrityksistä ja osoitteista taulut, joihin yhdistettiin jokaisesta maasta kaikki data edellisessä kappaleessa kuvattujen puhdistustoimenpiteiden jälkeisessä muodossa. Päivämäärää ja aikaa kuvaavat sarakkeet olivat lopullisessa DATE- ja TIMESTAMP-muodossa. Muut sarakkeet olivat tyyppiä VARCHAR, ja niistä oli esimerkiksi erilaiset erikoismerkit sekä välilyönnit siivottu pois. Yritystasoiseen tauluun luotiin uusi sarake, joka ilmaisi lähtödatan maan. Anonymisoituna esimerkkinä ensimmäisen maan lähtödatan rivit saivat tunnisteekseen ”m1”, ja muiden maiden osalta dataan annettiin vastaavasti tunnisteiksi ”m2” ja ”m3”. Tauluun luotiin myös jatkokehitystä ajatellen sarake erilliselle yritysavaimelle. Osoitteiden osalta toimittiin samalla tavalla kuin yritysten osalta, eli muunnettujen tietojen lisäksi latausalueen tauluun luotiin maata ilmaiseva sarake ja yritysavain-sarake.

Henkilöiden osalta päädyttiin luomaan latausalueelle kotimaista dataa vastaava rakenne, jota ei toimeksiantajan toiveesta haluttu kuvata työssä tarkemmin. Lähtökohta oli kuitenkin sama kuin yritystasoisten tietojen yhdistämisessä, jossa eri maiden tiedot yhdistettiin samoihin tauluihin. Henkilöt ja roolit olivat rakenteen myötä yhdistettävissä yrityksiin, ja public-taulun muodostamiseen tarvittavat tiedot olivat saatavilla. Päivämäärät oli muutettu merkkijonoista aikaleimoiksi.

### 6.3.3 Koodistojen luonti

Yleisesti ottaen tietovarastoon luodaan useita erityyppisiä koodistoja, joista kaikista löytyy suunnilleen samat sarakkeet. Koodistot muodostetaan virallisista lähteistä automaattisesti ja dataan lisätään kaikki lopullisen koodiston tarvitsemat kentät. Näistä koodistoista sitten muodostuu yksi yhtenäinen koodisto, jota käytetään kaikissa lopputuotteissa. Jos virallisessa koodistolähteessä tapahtuu muutoksia, se valuu automaattisesti kaikkiin lopputuotteisiin.

Projektiin liittyvien koodistojen luontia ei haluttu sitäkään kuvata työssä yksityiskohtaisesti. Maakoodien osalta koodisto oli valmiina, ja lähtödata tarvitsi vain yhdistää siihen latausalueen tauluihin luotujen maatunnisteiden avulla. Toimialojen osalta ei käytetty koodiston luonnissa lähtödatasta tulevaa selitettä, joka saattoi olla väärä, vaan erillistä virallista koodistoa, josta saatiin toimialakoodia vastaavat oikeat selitteet. Yhtiömuotojen koodisto taas luotiin lähtödatan maakoodin ja yhtiömuodon lyhenteen perusteella, ja jos lyhenne oli tyhjä, käytettiin koodiston luomisessa lähtödatan selitettä. Yrityksen tilan koodisto muodostettiin erittelemällä lähtödatan tilat maakohtaisesti aktiivisiin, lakanneisiin ja muihin tiloihin, jotta voitiin luoda lopulliseen koodistoon kyseiset kolme tilaa.

### 6.3.4 Public-taulun luonti

Public-taulut luodaan tietovaraston sisälle joka päivä uudestaan, ja pääavain määritellään kyseisen tietokannan skeemassa. Tässä tapauksessa pääavaimeksi määriteltiin lähtödatan yritystunnus, joka toimii jokaisen yritystasoisen tietueen yksilöivänä tietona, ja joka ei saa olla koskaan tyhjä.

Public-taulun tietosisällön luomista varten toteutettiin SQL-skripti, joka etsii tiedot latausalueella olevista tauluista sekä koodistosta. Tiedot oli latausalueella muunnettuna oikeaan tietotyyppiin, joten näiden osalta tarvittiin vain tietojen hakeminen kyselyyn, joka oli perusrakenteeltaan tavanomaista SQL-kieltä (Ohjelmakoodi 9). Jokaiselle attribuutille tai sarakkeelle luotiin lisäksi erilliset sarakkeet, jotka ilmaisevat tiedon lähteen sekä muutospäivämäärän. Nämä lähdetiedot päätettiin tuoda näkyviin ainoastaan, jos itse attribuutti oli saatavilla. Itse SQL-skriptiä ei näytetä työssä, mutta tekniikoiden osalta esitellään ohjelmakoodeina tavat, joilla lopputulokseen päästiin.

Ohjelmakoodi 9 SQL-kyselyn rakenne

```
select
  t1.sarake_1,
  t1.sarake_1_lähde,
  t1.sarake_1_muutospvm,
  …
  tn.sarake_n,
  tn.sarake_n_lähde,
  tn.sarake_n_muutospvm
from taulu_1 t1
join taulu_2 t2 on t2.sarake_1 on t1.sarake_1
…
left join taulu_n tn on tn.sarake_n on t1.sarake_n
where t1.sarake_y = ’XY’
```

Katuosoitteen näyttämiseksi halutussa muodossa tuli jokaisen tietueen tai rivin osalta yhdistää latausalueella olevasta taulusta katunimeä ja katunumeroa ilmaisevat sarakkeet. Redshiftin käyttämällä SQL-murteella sarakkeiden yhdistäminen tapahtui lisäämällä yhdistettävien sarakkeiden väliin niin sanottu double pipe -operaattori (||). Koska sarakkeiden tiedot haluttiin erotella uudessa sarakkeessa väliviivalla, tarvittiin operaattoreita kaksi, ja niiden väliin lisättiin väliviiva puoliheittomerkein ilmaistuna (Ohjelmakoodi 10).

Ohjelmakoodi 10 Esimerkki sarakkeiden yhdistämisestä SQL-kielellä

```
select
  t.katunimi || ' ' || t.katunumero as katuosoite
from taulu t;
```

Puhelinnumeroiden osalta tuli puolestaan erotella latausalueen sarakkeesta maakoodi- ja puhelinnumero-osa, jotta ne voitiin näyttää public-taulussa omissa sarakkeissaan. Erottelu tapahtui lisäämällä SQL-kyselyyn kaksi case-lauseketta (Ohjelmakoodi 11), joista ensimmäinen noutaa puhelinnumeron kolme ensimmäistä merkkiä ja toinen kaiken

neljännestä merkistä eteenpäin. Jos puhelinnumero määriteltiin lähtödatassa piilotettavaksi, saa kumpikin tieto null-arvon eli tyhjän.

Ohjelmakoodi 11 Esimerkki sarakkaiden erottelusta SQL-kielellä

```
select
  case
    when t.piilotettu = 1 then null
    else substring(t.puhelin, 1, 3)
  end as maakoodi,
  case
    when t.piilotettu = 1 then null
    else substring(t.puhelin, 4)
  end as puhelinnumero
from taulu t;
```

Toimitusjohtaja-tieto tuotiin public-tauluun SQL-skriptillä, joka with-lausekkeen avulla etsi jokaiselle yritystunnukselle toimitusjohtajan nimen, jos sellainen oli saatavilla (Ohjelmakoodi 12). With-lausekkeella voitiin luoda toimitusjohtajien osalta tilapäinen datasetti, joka yhdistettiin itse taulun luontiskriptiin yritystunnuksella. With-lausekkeen sisällä uusin ja voimassa oleva rooli löydettiin antamalla henkilöille yritystunnuksen tasolla rivinumero. Rivinumeron voi SQL-kielellä antaa row_number()-funktiota käyttämällä. Funktiolle annetaan ensin partition by -ehdolla sarake, jonka perusteella rivinumerot halutaan antaa, ja rivinumerointi määräytyi order by -ehdolla, jolla edelliseen ehtoon viittaavat tiedot voidaan järjestää esimerkiksi pienimmästä suurimpaan. Order by -ehdon lopussa oleva desc-määritys ilmaisee, että järjestäminen tapahtuu laskevasti, esimerkiksi päivämäärien tapauksessa uusimmasta vanhimpaan. With-lausekkeen tuloksesta voitiin tuoda public-tauluun haluttu tieto eli toimitusjohtajan nimi.

Ohjelmakoodi 12 Rivinumerointi sarakkeen arvon perusteella SQL-kielellä

```
with uusin_tj as (
  select
    sarake_1,
    sarake_2,
    sarake_3,
    row_number() over(
      partition by sarake_1
      order by sarake_2 desc
      ) rn
  from taulu t
)
select
  t1.sarake_1,
  …
  tj.sarake_n
from taulu_1 t1
```

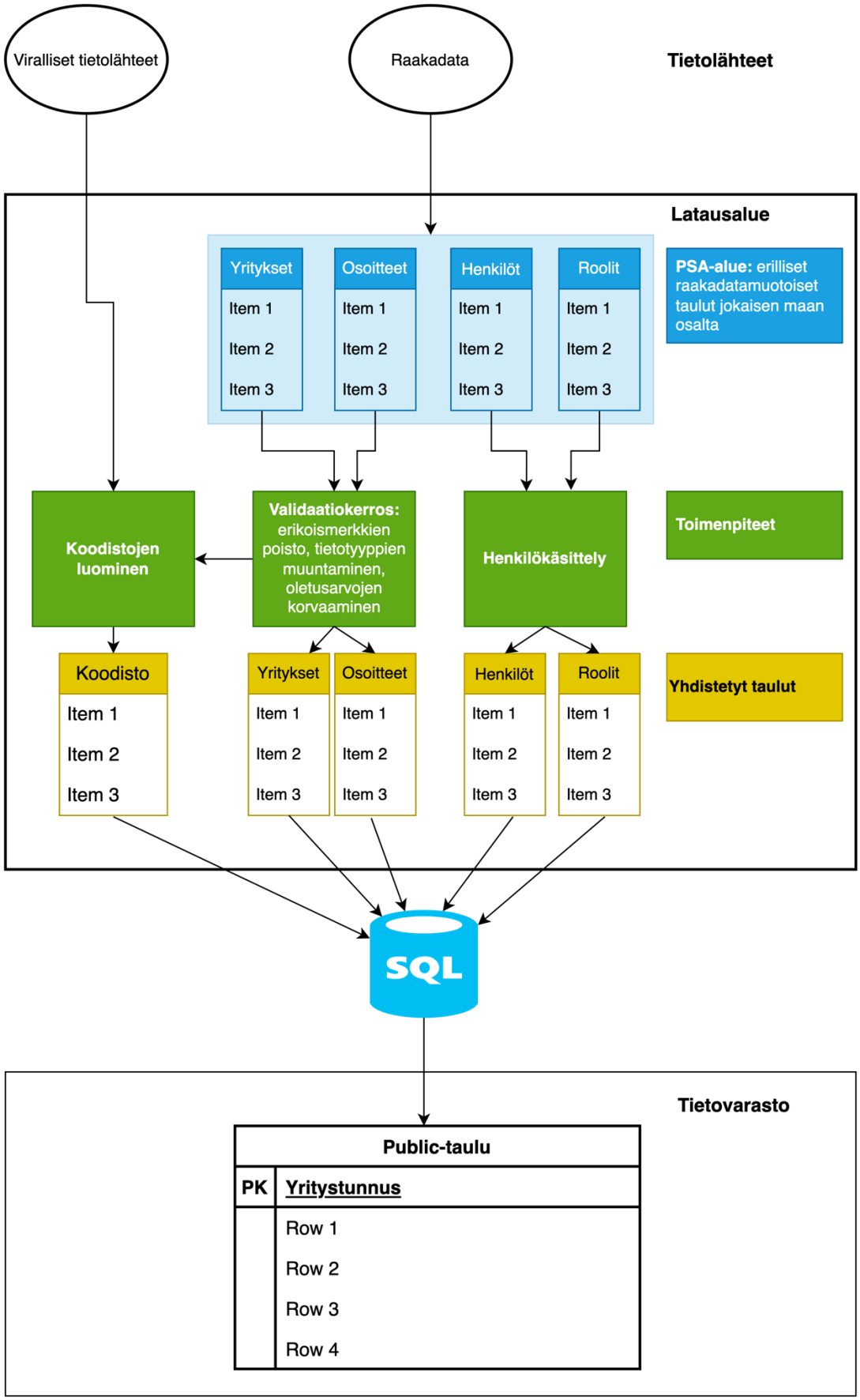
```
left join uusin_tj tj on tj.sarake=1 = t1.sarake_n
```

Koodistot luotiin erillisessä prosessissa, joten public-taulua luodessa tuli liittää koodisto kyselyyn tiettyjen attribuuttien perusteella. Näin tauluun saatiin luotua koodimuotoiset tiedot ilmaisemaan lähtödatan maata, yhtiömuotoa, yrityksen tilaa sekä toimialaa. Koodistosta tuotiin public-tauluun myös alkuperäiskieliset selitteet jatkokäytön helpottamiseksi, koska näiden tietojen osalta ei tarvitsisi enää tehdä yhdistelyä useampaan eri tauluun.

### 6.3.5 Validointi- ja normalisointiprosessin tulokset

Projektin aikana tietovaraston sisälle luodut toimenpiteet dokumentoitiin (Kuva 4). Tässä projektissa tietolähteitä olivat tiedontuottajan toimittama raakadata sekä koodistojen osalta viralliset tietolähteet. Raakadata tuodaan latausalueen sisällä olevalle PSA-alueelle sellaisenaan jokaisen maan osalta erikseen. Ennen datan siirtämistä varsinaiselle latausalueelle yritystasoinen data käsitellään validaatiokerroksessa haluttuun muotoon, ja henkilökäsittely muodostaa henkilö- ja roolidatasta oman henkilömallin. Erilliset koodistot käsitellään ja yhdistetään yhdeksi koodistoksi. Latausalueelle muodostuu tauluja, jotka sisältävät lopullisessa muodossaan olevan datan niin, että eri maiden tiedot on yhdistetty samaan tauluun. Tietovarastoon muodostetaan public-taulu, jossa yhdistellään latausalueelle luotujen taulujen tietoja. Public-taulu on lopulta esimerkiksi rajapintojen ja palveluiden käytettävissä ja tarjoaa ennalta määritellyn tietosisällön jokaiselle yritykselle.

Kuva 4 Validointi- ja normalisointioperaatiot tietovarastointiprosessissa



### 6.3.6 Muutosten määrä

Normalisointien toteutuksen jälkeen oli syytä tarkastella, kuinka paljon dataan oli tullut muutoksia. Koska public-taulussa sarakkeet eivät enää olleet esimerkiksi tietotyyppien muuntamisen ja sarakkeiden yhdistämisen tai erottelun jälkeen suoraan verrattavissa lähtödataan, vertailtiin raakadatan tauluja latausalueelle vietyihin tietoihin. Tässä vaiheessa sarakkeet olivat tyyppimuunnoksia lukuun ottamatta vielä samassa muodossa kuin raakadatassa. Tarkkoja muutosmääriä ei toivottu ilmoitettavan raportoinnin yhteydessä, joten lukuja kuvaillaan muutoin.

Koska yrityskohtaisten tietojen kokonaismäärä liikkuu miljoonissa riveissä, ei muutoksia tullut suhteessa kovin paljon. Yritysten nimiä saatiin normalisointifunktioiden avulla muokattua reilu tuhat. Sähköposteja muuttui muutama sata ja kotisivuja sekä toimialakuvauksia jokunen kymmen. Perustamispäivästä korvattiin tuhansia oletusarvon ”0001-01-01” sisältäneitä tietoja tyhjällä. Muiden tietojen osalta muutoksia ei tullut ollenkaan.

Osoitteiden osalta suurin muutosmäärä tuli postinumeroista, koska lähtödatasta korvattiin ”.0”-päätteet ja välilyönnit. Muuttuneita tietoja oli siis miljoonia. Postiosoitteen kaupungin ja katunumeron osalta muutoksia tuli noin kymmenestä muutamaan kymmeneen, mutta katunimien osalta tuhansia. Tässä vaiheessa huomattiin, että käytössä oli siivousfunktio, jonka ei kuuluisi olla kyseisen sarakkeen osalta käytössä, joten suurin osa muutoksista oli tarpeettomia. Normalisointien korjauksen jälkeen muutosten määrä oli sangen pieni. Muiden sarakkeiden osalta muutoksia ei havaittu.

# 7 Johtopäätökset ja pohdinta

Työn teoriaosan tavoitteena oli tutustua tietovarastoinnin peruskäsitteisiin, datan hallinnan merkitykseen sekä erilaisiin tapoihin validoida ja normalisoida dataa. Tietovarastoinnin peruskäsitteiden opiskelu auttoi ymmärtämään käytännön osassa toteutetun projektin data-arkkitehtuuria paremmin. Havaittiin esimerkiksi, että käytössä oleva datamalli on lähempänä ELT- kuin ETL-prosessia, koska data ladataan tietovaraston sisälle ennen transformaatioiden tekemistä. Datan laadun hallintaan liittyvät kysymykset puolestaan auttoivat ymmärtämään laadukkaan datan merkityksen esimerkiksi liiketoiminnan sekä päätöksenteon kannalta. Datan laadun ulottuvuudet puolestaan olivat tärkeässä roolissa käytännön osan projektissa erityisesti tiedontuottajan dataa validoidessa. Datan normalisointiin tai puhdistamiseen liittyvien käytäntöjen tutkiminen taas antoi käytännön projektiin ajatuksia siitä, miten dataa voisi käsitellä, vaikka toimenpiteet tulikin tehdä soveltaen niitä toimeksiantajan tarjoamaan data-arkkitehtuuriin.

Käytännön osan tavoitteena oli implementoida teoriaosiossa käsiteltyjä validointi- ja normalisointimenetelmiä käyttöön. Tiedontuottajan dataa oli tarpeen analysoida ja profiloida käyttäen teoriaosiossa esiteltyjä menetelmiä, ja siitä syntyneiden havaintojen pohjalta toteuttaa normalisointi- ja puhdistustoimenpiteet datalle. Lopputuotoksena syntyi tiedontuottajan toimittaman datan ja valmiiksi käytössä olleiden virallisista lähteistä ladattavien koodistojen pohjalta niin sanottu public-taulu, joka oli rajapintojen sekä loppukäyttäjille tarjottujen palvelujen saatavilla tietovarastossa.

Projekti onnistui koko tiimin osalta varsin hyvin ja tavoiteltuun MVP-tuotteen julkaisuajankohtaan päästiin. Scrum-prosessia mukaillen vaadittavat työt pilkottiin Jira-tiketeiksi, joita otettiin työstöön sitä mukaa, kun projekti eteni. Raakadata oli ensin tuotava tietovaraston sisälle, jotta sitä voitiin tutkia. Datan ominaisuuksiin sekä normalisointiin ja puhdistamiseen liittyvät ehdotukset dokumentoitiin Jira-tiketteihin ja Confluenceen, jotta päätöksentekoon vaadittava tieto oli kaikkien saatavilla. Näin voitiin suunnitella esimerkiksi koodistojen luomiseen sekä henkilökäsittelyyn liittyvät työt.

Jatkokehitystä ajatellen jäi myös mietittävää. Public-tauluun tuotiin sarake erilliselle yritysavaimelle, mutta sitä ei vielä MVP-tuotteen osalta tauluun generoitu. Myös henkilödatan osalta tulisi selvittää, vastaako toimitusjohtajien määrä todellisuutta.

On kuitenkin huomioitava, että normalisointien jälkeenkin data saattaa olla virheellistä. Versionhallinnan avulla muutosten tekeminen käsittelyyn on kuitenkin suhteellisen helppoa ja nopeaa, joten virheitä huomattaessa ne pystytään korjaamaan kohtalaisen nopeasti. Jatkon kannalta tärkeää on, että yksittäisiä dataan liittyviä virheitä ei tarvitsisi lähteä korjaamaan, vaan että muutoksia voitaisiin tehdä esimerkiksi sarakekohtaisesti. Se, että normalisointiprosessi on ominaisuuksiltaan ja toiminnallisuudeltaan dokumentoitu, vähentänee selvittelyyn kuluvaa työaikaa jatkossa.

Muutosten määrä jäi normalisoinnin jälkeen pääosin maltilliseksi, mikä kertoo siitä, että tiedontuottajan toimittama data oli kokonaisuutena varsin laadukasta. Oli kuitenkin tärkeää havaita vähäisetkin virheet ennen kuin data on loppukäyttäjien saatavilla, koska yksikin virhe saattaa heikentää toimeksiantajan itsestään jättämää mielikuvaa asiakaskunnassa. Datan validointi olikin projektin lopputuloksen kannalta yksi tärkeimmistä vaiheista.

# 8 Yhteenveto

Tutkimuskysymyksiin vastaaminen onnistui kohtalaisen hyvin. Tietovarastoinnin peruskäsitteistön, datan laadun hallinnan ja validointi- sekä normalisointimenetelmien osalta saatiin aikaiseksi tiivis teoriaosio, joka olisi retrospektiivisesti ajateltuna voinut käsitellä aihepiiriä laajemminkin. Tässä työssä kuitenkin haluttiin keskittyä erityisesti datan validointiin ja normalisointiin, joten tietovarastointi oli mukana eritoten teknistä kontekstia antamassa, ja datan laadun hallinnan merkitystä haluttiin käsitellä erityisesti liiketoiminnan näkökulmasta. Teoriaosan kirjoittaminen auttoi ymmärtämään esimerkiksi käytännön osan data-arkkitehtuuria paremmin, ja ylipäätään datan validoinnin merkityksen tiedostaminen oli erittäin hyödyllistä.

Validointi- ja normalisointimenetelmien implementointi projektiin onnistui varsin hyvin. Teoriaosiossa käsitellyt menetelmät mahdollistivat raakadatan varsin monipuolisen tutkimisen ja profiloinnin, mikä oli jälkikäteen ajateltuna myös lopputuloksen kannalta tärkeää. Normalisointia pääsi toteuttamaan itse Python-funktioiden luomisen osalta, sekä analysoimalla koodistojen luontiin ja henkilökäsittelyyn liittyviä prosesseja. Tuloksia saatiin myös mitattua normalisoitujen sarakkeiden osalta.

Datan käsittely SQL-kielellä oli jo entuudestaan varsin tuttua, mutta Python-ohjelmakoodiin tutustuminen oli ajoittain haastavaa. Vaikka ohjelmoinnin perusteet itsessään olivat tiedossa, teki ennestään tuntemattomien kirjastojen ja menetelmien käyttö valmiin koodin lukemisesta hankalaa. Python-kielellä toteutetut normalisointifunktiot toimivat kuitenkin hyvänä keinona tutustua aiheeseen käytännön tasolla.

Käytännön projektin suunnitteluun ja toteutukseen osallistuminen antoi paljon oppia esimerkiksi pilvipalveluna tarjotun tietovaraston rakenteesta ja siitä, minkälainen kokonaisuus raakadatan pohjalta harmonisoidun taulun rakentaminen oikeastaan on. Opinnäytetyötä tehdessä kuitenkin saattoi huomata, että kyseessä oli pintaraapaisu aiheeseen. Tietokantojen hallitsemisen lisäksi olisi data-ammattilaisen hyvä osata ohjelmoida ja tuntea pilvipalvelut kohtalaisen laajasti.

## Lähteet


Amazon Web Services. (2023). *Data Warehouse Concepts.* Viitattu 8.6.2023. https://aws.amazon.com/data-warehouse/

Amazon Web Services. (n.d.). *What is Data Cleansing?* Viitattu 5.7.2023. https://aws.amazon.com/what-is/data-cleansing/

Atlan. (2022). *Data Profiling: Definition, Techniques, Process, and Examples.* Viitattu 27.6.2023. https://atlan.com/data-profiling-101/

Challa, A., Radovanovich, C., Yu, J., Friedmann, L. & Goel, M. (2023). *Data Warehousing on AWS. AWS Whitepaper.* Viitattu 28.6.2023. https://docs.aws.amazon.com/pdfs/whitepapers/latest/data-warehousing-on-aws/data-warehousing-on-aws.pdf#data-warehousing-on-aws

Felix, P. B. (n.d.). *Why You Need A Persistent Staging Area (PSA).* Viitattu 27.7.2023. https://www.leapfrogbi.com/why-you-need-a-persistent-staging-area-psa/

IBM. (n.d.). *What is ETL?* Viitattu 26.6.2023. https://www.ibm.com/topics/etl

Keboola. (2022). *Data Transformation in ETL Process Explained (2023 Guide).* Viitattu 5.7.2023. https://www.keboola.com/blog/etl-transformation

Kimball, R. (2004). *Kimball Design Tip #59: Surprising Value of Data Profiling.* Viitattu 27.6.2023. http://www.kimballgroup.com/wp-content/uploads/2012/05/DT59SurprisingValue.pdf

Maydanchik, A. (n.d.). *Data Quality Rules Tutorial 1 of 4: Attribute Domain Constraints.* Viitattu 14.6.2023. https://www.dataqualitypro.com/blog/data-quality-rules-attribute-domain-constraints-arkady-maydanchik

Maydanchik, A. (2007). *Data Quality Assessment.* Technics Publications. Viitattu 14.6.2023. https://books.google.fi/books?id=g43XBgAAQBAJ&lpg=PA5&ots=-rDryQtj6F&dq=Data%20Quality%20Assessment%20maydanchik&lr&hl=fi&pg=PA60#v=onepage&q=Data%20Quality%20Assessment%20maydanchik&f=false

Morris, K. (2020). *Infrastructure as Code. Dynamic Systems for the Cloud Age.* Viitattu 14.6.2023. https://ebookcentral-proquest-com.ezproxy.hamk.fi/lib/hamk-ebooks/reader.action?docID=6422000&ppg=1

Naumann, F. (2017). *An Introduction to Data Profiling.* Viitattu 27.6.2023. https://hpi.de/fileadmin/user_upload/fachgebiete/naumann/folien/SS17/DP_02_ProfilingIntro.pdf

Rithika, S. (2023). *ETL Process Made Easy: The Ultimate Guide 101.* Viitattu 26.6.2023. https://hevodata.com/learn/etl-process/

SAP. (n.d.). *What is a data warehouse?* Viitattu 8.6.2023. https://www.sap.com/products/technology-platform/datasphere/what-is-a-data-warehouse.html

Smallcombe, A. (2023). *Structured vs Unstructured Data: 5 Key Differences.* Viitattu 26.6.2023. https://www.integrate.io/blog/structured-vs-unstructured-data-key-differences/

Talend. (n.d.a.). *What is ELT?* Viitattu 26.6.2023. https://www.talend.com/resources/what-is-elt/

Talend. (n.d.b.) *Modern Data Warehouse Architecture: Traditional vs Cloud Data Warehouse.* Viitattu 26.6.2023. https://www.talend.com/resources/cloud-data-warehouse-architecture/

TIBCO. (n.d.). *What is Data Validation?* Viitattu 14.6.2023. https://www.tibco.com/reference-center/what-is-data-validation

Tilastokeskus. (n.d.). *Johdatus tilastotieteeseen. 2.1.4 Diskreetti ja jatkuva muuttuja. Muuttujan jakauma.* Viitattu 27.6.2023. https://tilastokoulu.stat.fi/verkkokoulu_v2.xql?course_id=tkoulu_tilaj&lesson_id=2&subject_id=5&page_type=sisalto

Törmänen, A. (2017). *Johdanto tietovarastointiin.* CreateSpace Independent Publishing Platform.

Varshney, H. (2023). *What is a Data Staging Area? | Staging Data Simplified 101.* Viitattu 26.6.2023. https://hevodata.com/learn/data-staging-area

Väre, T. (2019.) *Master data.* Alma Talent.

**Liite 1: Aineistonhallintasuunnitelma**

Kehitysprojektin aikana kerätään teknistä ja dataan liittyvää kvantitatiivista ja laadullista tietoa. Tämä tieto analysoidaan opinnäytetyötä varten. Tieto säilytetään työn toimeksiantajan omistamalla työasemalla OneNote-muistikirjassa tai Excel-taulukoissa. Tiedoista otetaan säännöllisesti varmuuskopioita toiseen työaseman kansioon, mutta pilvipalveluihin niitä ei tallenneta. Ohjausta ajatellen opinnäytetyön työversioita säilytetään ajoittain myös tekijän omalla työasemalla, josta tiedot poistetaan sen jälkeen, kun niitä ei siellä enää tarvita. Tietoja säilytetään työn toimeksiantajan omistamalla työasemalla ainakin vuoden verran opinnäytetyön valmistumisesta.

Kehitysprojektin aikana pidetyistä kokouksista kirjoitetaan muistiinpanoja joko projektiin liittyvälle Confluence-sivulle tai tiettyihin työtehtäviin liittyvissä asioissa suoraan tehtävään liittyvään Jira-tikettiin. Projektin aikana tapahtuvat havainnot ja johtopäätökset kirjataan samaan tapaan joko kyseiseen tehtävään liittyvään tikettiin tai Confluenceen. Valmiin projektin onnistumisesta kerätään tietoa tekijän muistikirjaan ja tuloksena syntyvät tiedot dokumentoidaan Confluenceen, josta ne ovat projektin parissa jatkossa työskentelevien henkilöiden nähtävillä.