Data Vault -menetelmään perustuvan tietovaraston kehityksen automatisaatiomenetelmät

Turun yliopisto
Teknillinen tiedekunta
Tietotekniikan laitos
Data-analytiikka
Pro gradu -tutkielma
Teemu Lehti
Toukokuu 2022

Turun yliopisto
Teknillinen tiedekunta, Tietotekniikan laitos

Teemu Lehti: Data Vault -menetelmään perustuvan tietovaraston kehityksen automatisaatiomenetelmät

Pro gradu -tutkielma, 54 s.
Data-analytiikka
Toukokuu 2022
Ohjaajat: Tuomas Mäkilä, Sami Heino

Data Vault on hyvin standardoitu ja dokumentoitu menetelmä tietovarastointiin. Tietovarastoinnin pääidea on integroida eri lähteistä tulevia datoja keskitettyyn paikkaan, jonka Data Vault toteuttaa tehokkaasti yhdistelemällä aikaisempien menetelmien hyviä ominaisuuksia yhteen. Sen raskas, mutta joustava rakenne tietovaraston mallinnuksessa johtaa automaatiotyökalujen käyttöön, joilla voidaan lähdejärjestelmien metadatan ja Data Vault -mallin määritysten avulla luoda tietovaraston tietokannan objektit automaattisesti.

Tutkielmassa tutustutaan Data Vault -menetelmällä mallinnetun tietovaraston kehityksen automatisaation lähestymistapoihin käymällä läpi erilaisia akateemisia tutkimuksia ja suorittamalla sisäinen haastattelututkimus Aveso Oy:ssä. Tavoitteena on löytää uusia näkökulmia tietovaraston kehityksen automaatioon, analysoida Aveson oman tietovaraston kehitystyökalun, Aveso DW Automationin automaatioratkaisua ja löytää sille kehityskohteita.

Kolme tutkielmassa läpikäytävää lähestymistapaa keskittyvät hieman eri osiin tietovaraston kehityksestä. Puonti et al. lähestymistapa on kevyt ja pääosin datojen latauksiin keskittyvä. Krneta et al. lähestymistapa on hieman pidemmälle kehitetty ja se keskittyy pääosin Data Vault -mallin taulujen luontiin. Krneta et al. esittelevät myös tavan luoda Data Vault -malli automaattisesti lähdejärjestelmien pohjalta. Pankov et al. lähestymistapa keskittyy pääosin yleisen metadatatietokannan kehitykseen, jonka pohjalta olisi mahdollista generoida useamman erilaisen mallinnusmenetelmän tietovarasto-objektit automaattisesti.

Haastattelututkimuksessa tuli esille näkemyksiä tietovaraston kehitykseen ja siihen, millaisia ominaisuuksia voisi tietovaraston kehityksen automatisoivassa työkalussa olla. Haastattelututkimuksen ja akateemisten tutkimusten pohjalta tuli esimerkiksi selville, että Aveso DW Automation -työkalulta puuttuu vielä ominaisuuksia Data Vault -menetelmän ja sen ylläpidon kannalta, sekä sen toimintaa voisi myös tehostaa suoralla yhteydellä lähdejärjestelmiin.

Asiasanat: tietovarasto, Data Vault, automatisaatio, metadatan hallinta

Sisällys

1 Johdanto

Data Vault -menetelmä on koko ajan kasvattamassa suosiotaan maailmalla ja se on Suomessakin jo varsin yleinen uusissa tietovarastointiprojekteissa [1]. Se on joustava ja helposti laajennettava menetelmä monien tietolähteiden integrointiin, sekä sillä on helppo toteuttaa tietojen historiointi. Data Vault -menetelmä on nimenomaan suunniteltu vastaamaan suurten keskitettyjen tietovarastojen tarpeita [2].

Haasteena Data Vault -menetelmällä mallinnetun tietovaraston toteutuksessa on kuitenkin se, että tauluja ja muita tietokannan objekteja joudutaan luomaan reilusti enemmän kuin muilla yleisesti käytetyillä mallinnusmenetelmillä. Data Vault -menetelmällä ei ole käytännöllistä manuaalisesti luoda tietovaraston tauluja, sillä tietovaraston ylläpito tulisi mahdottomaksi. Suhteellisen pienikin muutos tietovarastoon voisi tarkoittaa sitä, että joutuisi kymmeniä, tai tietovaraston koosta riippuen, jopa satojen tietokannan objektien määrityksiä muuttamaan manuaalisesti uudelleen ja uudelleen. Hyvin pienille tietovarastoille Data Vault ei välttämättä ole kovin järkevä menetelmä, sillä Data Vault tekee tietovarastosta selvästi monimutkaisemman. Pienet tietovarastot useasti halutaan pitää yksinkertaisina.

Suuren tietokantaobjektien määrän hallitsemiseen voidaan käyttää erilaisia ulkoisia työkaluohjelmistoja. Data Vault -menetelmä on osittain suunniteltu juuri tällaisten työkalujen kannalta, jotta tietovaraston hallinta olisi mahdollisimman helposti toteutettavissa. Data Vault -mallin osat koostuvat loogisista, helposti käsiteltävistä kokonaisuuksista, joiden kehitys on helppo rinnakkaistaa. Automaatiotyökalut voi-

vat luoda esimerkiksi lähdejärjestelmien metadatan (taulujen sarakkeiden nimien, tietotyyppien ja taulujen välisten yhteyksien) avulla Data Vault -mallilla mallinnetun tietovaraston objektit automaattisesti. Tavoitteena olisi, että manuaaliseksi työksi jäisi vain tämän mallin metadatan, esimerkiksi Data Vault -mallin osien yhteyksien ja avainsarakkeiden määrittäminen, jotta kehittäjät pystyvät suoraan keskittymään liiketoiminnan kannalta tärkeisiin osiin tietovaraston kehityksessä.

Riippuvuus automaatiotyökalusta ei ole suoraan huono asia. Tällainen työkalu voi samalla auttaa myös muihin asioihin tietovaraston kehityksessä. Monia eri osia tietovaraston kehityksestä tehdään nykyään muutenkin täysin manuaalisesti [3]. Työkalusta voisi olla suora apu esimerkiksi tietovaraston ylläpidossa, versioinnissa ja dokumentaation luonnissa. Data vault -mallia luodessa työkalu voisi esimerkiksi avustaa mallin määrittämisessä ja estää huonosti toimivien mallien luonnin. Automaatiotyökalu voi myös varmistaa, että metadataa ei muuteta niin, että tietovarastosta poistuisi dataa. Metadatan versionti auttaisi työn jakamisessa usealle kehittäjälle, sillä kehitystyötä voisi helpommin tehdä useampi samanaikaisesti. Nykyiset ETL-ohjelmistot eivät täysin tue tällaista rinnakkaista kehitystä [4]. Versionnin avulla voidaan myös palata aikaisempaan versioon, esimerkiksi jos huomataan, että metadatan määrittelyissä on jokin virhe.

Aveso DW Automation -työkalu osittain ratkaisee nämä ongelmat, mutta esimerkiksi kaikkia toimintoja metadatan ylläpidon ja Data Vault -mallin taulutyyppien kannalta ei ole vielä implementoitu. Tutkielmassa tarkastellaan automaation mahdollisuuksia, mitä kaikkea on mahdollista ja järkevää tietovaraston kehityksestä automatisoida, sekä kuinka lähellä Aveso DW Automation -työkalu on akateemisen tutkimuksen automaatioratkaisuja. Käydään läpi automaatioesimerkki työkalun nykytoteutuksen kanssa ja etsitään kehityskohteita työkalun jatkokehitystä varten. Tätä varten myös suoritetaan Aveso Oy:ssä haastattelututkimus, josta nähdään Aveson työntekijöiden näkemyksiä tietovaraston kehityksen automatisointiin ja Aveso DW

Automation -työkaluun, jotta tarkemmin syvennytään tutkimuskysymyksiin. Tutkimuskysymyksinä ovat:

- Mitkä ovat Data Vault menetelmän hyödyt ja haitat tietovaraston kehityksessä?
- Millaista tutkimusta on tehty tietovaraston kehityksen automatisoinnista?
  - Kuinka käytännöllisiä malleja tutkimuksessa on esitetty?
  - Mitä osia tietovaraston kehityksestä on automatisoitu?
- Kuinka lähellä Aveso DW Automation -työkalun automaatioratkaisu on tutkimuksessa esitettyjä Data Vault automaatioratkaisuja?
- Onko Aveso DW Automationin automaatioratkaisulla selviä rajoitteita, jotka joillain tavoin heikentävät generoidun tietovaraston toimintaa tai mahdollisuuksia?

Toisessa luvussa perehdytään tietovarastoinnin perusteisiin, kolmannessa tarkemmin Data Vault -menetelmään ja neljännessä luvussa tarkastellaan automaation hyötyjä sekä käydään läpi kolme erilaista lähestymistapaa Data Vault -tietovaraston kehityksen automatisaatioon. Viidennessä luvussa esitetään Aveso DW Automation -työkalun toimintaperiaatteita ja käydään läpi esimerkki Data Vault -tietovaraston kehityksestä sitä käyttäen. Kuudennessa luvussa esitetään suoritetun haastattelututkimuksen suunnitelma ja tulokset. Seitsemännessä luvussa lopulta verrataan Aveso DW Automation -työkalun automatisoinnin toimintaa akateemisessa tutkimuksesta löydettyihin automatisaation lähestymistapoihin ja analysoidaan tarkemmin sen rajoitteita ja kehityskohteita haastattelujen tulosten ja akateemisten lähestymistapojen pohjalta.

2 Tietovarastointi

Tietovarastointi on yksi tärkeimmistä työkaluista, joita yrityksillä on käytettävissä päätöksenteon tueksi. Monet isot organisaatiot ovat jo hyvin pitkään investoineet tietovarastoihin. Ennen tietovarastoita, dataa piti suoraan hakea kriittisistä tuotannossa olevista järjestelmistä [5]. Data Vault -menetelmä ja automatisaatio laajentaa tietovarastojen mahdollisuuksia, joten tietovaraston idea on tämän tutkielman kannalta oleellista ymmärtää.

2.1 Tietovarastoinnin perusteet

Tietovarastolla (engl. data warehouse) yleisesti tarkoitetaan tietokantaa, johon yhdistetään dataa monista eri järjestelmistä analyysia, raportointia tai jotain muuta tarvetta varten [6]. Näitä eri järjestelmiä, esim. muita tietokantoja, ERP-järjestelmiä tai vaikka csv-tiedostoja kutsutaan tietovaraston kannalta *lähdejärjestelmiksi*. Eri lähdejärjestelmistä tulevat datat ovat yleensä hyvin eri muodoissa. Samat käsitteet on esimerkiksi voitu eri järjestelmissä kuvata hyvin erilaisilla nimillä ja teknisesti tallennettu erilaisissa muodoissa. Eri lähdejärjestelmiin myös muodostetaan yhteys eri tavoin ja datoja ladataan eri menetelmillä. Datojen yhdistäminen ei siis ole täysin triviaalia, joten tästä syystä ei ole kannattavaa analysoida lähdejärjestelmien dataa suoraan. Tietovaraston pääideana on olla yksinkertainen näkymä kaikkien lähdejärjestelmien datoille ja tuoda ne helposti saataville keskitettyyn paikkaan, jotta dataa olisi helppo käyttää analysoinnissa ja raportoinnissa.

Yksi tärkeimpiä tietovaraston tehtäviä on myös tiedon historiointi [6]. Lähdejärjestelmät eivät välttämättä pidä yllä historiaa siitä, miten data on muuttunut ajan myötä. Kun tietovarastoon ladataan datan päivityshetkellä myös päivitysajankohta, niin voidaan tämän avulla palata ajassa taaksepäin tarkastelemaan, miltä data on näyttänyt tietyllä tietovaraston päivityshetkellä. Tätä historiadataa voidaan tietenkin myös erikseen analysoida ja raportoida. Tietovaraston dataa ei välttämättä tarvitse poistaa, mutta sitä voidaan tarpeen mukaan merkitä poistetuksi, jos tietyt tietueet ovat lähdejärjestelmästä poistuneet [7]. Näin saadaan poistuneet datat huomioitua ilman, että menetetään datan historiaa.

Tietovaraston hyötynä on myös se, että datan haku tietovaraston kautta ei ruuhkauta kriittisiä lähdejärjestelmiä. Jos analyysiä ja raportointia tehtäisiin suoraan lähdejärjestelmistä, niin se voisi jopa täysin estää jonkin lähdejärjestelmän käytön hetkellisesti, kun suurta datamäärää tarkastellaan samanaikaisesti. Tietovarastoon datat voidaan ajastaa päivittymään sellaisella aikavälillä, jolloin varmasti tiedetään, että lataus on turvallista, esimerkiksi yöllä kerran päivässä. Tietyt taulut voidaan tarpeen mukaan tietovarastoon päivittää useimminkin.

2.2 ETL-prosessi

ETL-prosessi (engl. Extract, Transform, Load) on tietovarastoinnissa käytetty menetelmä esimerkiksi datan lataamiseen lähdejärjestelmän ja tietovaraston välillä. ETL-prosessi ei varsinaisesti ota kantaa latauksen yksityiskohtiin, se voi tarkoittaa käytännössä minkä tahansa kahden järjestelmän välistä integraatiota [4]. Pääidea ETL-prosessissa kuitenkin on, että ensin data ladataan (Extract) järjestelmistä (tietovaraston tapauksessa sen lähdejärjestelmistä), mahdollisesti validoimalla dataa jo jotenkin, ja sitten muuttamalla data lopulliseen haluttuun muotoon (Transform). Tämä data voidaan ladata välivarastoon, josta se sitten lopullisesti ladataan (Load) tavoitejärjestelmään, esim. tietovarastoon.

Suuremmissa tietovarastoissa käytetään nykyään yhä enemmän ETL-prosessin varianttia ELT (Extract, Load, Transform), jossa lähdejärjestelmien dataa ei vielä latauksen alkuvaiheessa muuteta lopulliseen käytettävään muotoon, vaan se ladataan lähdejärjestelmistä lähes sellaisenaan suoraan tietovarastoon. Datan muodon muuttamisen vaihe (Transform) on nyt viimeisenä, joka tehdään vasta tarpeen mukaan suoraan tietovarastossa, esimerkiksi erilaisilla näkymillä tietovaraston dataan. ELT-prosessissa datan lopullinen muokkausvaihe voi myös olla kokonaan oma ETL-prosessinsa, jossa data siirretään välivarastosta lopulliseen varastoon [4].

2.3 Tietovarastoinnin haasteita

Haasteena tietovarastoinnissa on muun muassa suurten datamäärien ja kompleksisuuden hallinta. Esimerkiksi datan analysointia varten voidaan tarvita tietyn aihealueen tauluja monesta eri lähdejärjestelmästä, joiden datoja pitäisi siivota, täydentää ja yhdistellä eri tavoin järjestelmien välillä. Datamäärät voivat olla suuria ja data voi päivittyä usein, joka voi johtaa erilaisiin suorituskykyongelmiin. Lähdejärjestelmistä voi datan lataaminen olla ylipäätään mahdollista vain tiettyinä aikoina, lähdejärjestelmän prosessorin, välimuistin ja esimerkiksi levyn käyttö voi kuormittua, datan indeksointi voi olla vaikeaa, datan rivien yksilöivät avaimet voivat puuttua kokonaan ja data voi olla tallennettu hyvin monimutkaisilla tietorakenteilla [5]. Lähdejärjestelmät voivat myös olla lukinneet jotain dataa väliaikaisesti, vaikka jos itse lähdejärjestelmän sisäinen prosessi käyttää dataa tietyllä hetkellä.

Suuri haaste tietovarastoille on myös datan laadun ja johdonmukaisuuden hallinta [8]. Tietovarastoissa myös helposti sekoitetaan eri tyyppisiä tietokannan muokkauksia, kuten uusien rivien lisäystä, rivien päivitystä ja poistamista samojen prosessien alle, joka yhä enemmän lisää tietovaraston kompleksisuutta ja vaikeuttaa ylläpitoa [5]. Tietovarastoon tulevia datoja muokataan sekaisesti myös senkin takia, että eri työkaluissa on omat ominaisuutensa ja rajoituksensa. Monet tietovarastoin-

tijärjestelmät yrittävät tehdä liian paljon samaan aikaan sen sijaan, että jakaisivat ongelmia pienempiin paloihin ja antaisivat järjestelmän eri osille selvät omat roolinsa tiettyjen ongelmien ratkaisemiseksi [5].

Tietovarastoiden kustannusten alhaisina pitäminen voi olla haastavaa. Mitä enemmän tietovarastoista tehdään vain tiettyyn tarkkaan käyttötarkoitukseen sopivia, niin sitä vaikeammaksi tulee tietovaraston muuttaminen ja jatkokehittäminen liiketoiminnan vaatimusten muuttuessa. Kustannuksiin vaikuttaa kuitenkin moni muukin asia. Itse datan varastointikustannukset kasvavat myös datan määrän kasvaessa. Kun dataa ja datalähteitä on kertynyt tarpeeksi, niin voidaan myös tarvita nopeampi internet yhteys ja enemmän laskentatehoa, jotta kaikki data ehditään käsitellä.

2.4 Tietovaraston kehitysmalleja

Perinteisesti paljon käytetyt tietovarastointimenetelmät ovat 1980 luvulta lähtien kehitetyt Bill Inmonin CIF (Corporate Information Factory) normalisoitu tietovarasto ja Ralph Kimballin kehittämä dimensionaalinen tietovarasto. Inmonin normalisoidussa tietovarastossa data tallennetaan tietokantojen normalisointisääntöjen mukaan, käyttäen kolmatta normaalimuotoa (3NF), jossa data jaetaan tietokantatauluihin niin, että yhdessä taulussa on vain yhteen kategoriaan liittyvää dataa [9]. Normalisoitu malli pitää datasta duplikaatit poissa, jolloin tietty data on aina tallennettuna vain yhdessä paikassa. Kimballin dimensionallisessa mallissa data jaetaan fakta- ja dimensiotauluhin [10]. Faktat sisältävät usein numeerista dataa, esimerkiksi mittausdataa tai transaktiodataa ja dimensiot viitetietoa, jotka antavat lisätietoa faktoille. Dimensionaalisessa mallissa sama data voi helpommin esiintyä monessa eri taulussa.

Suurin ero Inmonin ja Kimballin tietovarastoinnin arkkitehtuureissa on se, että Inmonin tietovarasto rakennetaan ylhäältä alas periaattella, jossa tietovarasto suunnitellaan datan varastoinnin ja integraation näkökulmasta. Kimballin tietovarasto

puolestaan rakennetaan alhaalta ylös, datan hakemisen ja raportoinnin näkökulmasta [6].

Normalisoituun tietovarastoon uuden tiedon lataaminen on helppoa, mutta koska normalisoidussa mallissa on useasti hyvin paljon tauluja, niin merkityksellisen datan saaminen tauluja yhdistämällä voi olla hankalaa. Kimballin dimensionaalinen tähtimalli on tietovaraston käyttäjille paljon helpompi suoraan käyttää ja ymmärtää. Tähtimalli onkin näistä yleisemmin käytetty tietomalli tietovarastoinnissa, sillä sen avulla on helpompi luoda osastokohtainen paikallinen tietovarasto [11]. Normalisoiduilla tietovarastoilla on kuitenkin omat hyötynsä, jonka takia näistä kahdesta mallista ei voi täysin valita parempaa mallia. Normalisoidulla mallilla dataa voi esimerkiksi helpommin käyttää koko tietovaraston laajuudella, yhdistellä eri osa-alueiden datoja, sillä kaikki datat ovat integroitu keskenään. Normalisoitua mallia on myös helpompi muokata liiketoiminnan vaatimusten muuttuessa [6].

3 Data Vault 2.0

Tietovarastojen ongelmia ratkaisemaan Daniel (Dan) Lindstedt julkaisi vuonna 2000 Data Vault tietokantojen mallinnusmenetelmän [12]. Data Vault on ns. hybridimalli, jossa on yritetty yhdistää normalisoitujen mallien ja dimensionaalisen tähtimallien hyviä puolia yhteen.

Dan Lindstedt määrittelee Data Vaultin olevan yksityiskohtiin suuntautunut, historiatiedon jäljittävä ja yksilöllisesti linketetty joukko normalisoituja tauluja, jotka tukevat yhtä tai useampaa toiminnallista liiketoiminta-aluetta. Sen suunnittelu on joustava, skaalautuva, johdonmukainen ja mukautuva yrityksen tarpeisiin. Se on tietomallina suunniteltu nimenomaan vastaamaan nykyajan yritystietovarastojen (EDW) tarpeisiin.

Vuoteen 2013 mennessä Data Vault malli on laajennettu pelkästä mallinnusmenetelmästä Data Vault 2.0 arkkitehtuuriksi ja metodologiaksi, joka sisältää nyt mallinnusmenetelmän lisäksi korkeamman tason käsityksen Data Vault tietovaraston kehityksestä [5]. Data Vault 2.0 metodologian mukainen tietovarasto esimerkiksi tukee myös paremmin suuria ja järjestämättömiä datamääriä (Big data).

3.1 Mallinnuksen perusteet

Data Vault mallinnusmenetelmä osittain perustuu hub-and-spoke malliin, joka on helposti skaalaantuva ja laajennettava verkkomalli [5]. Tämän mallin pohjalta Data Vault -mallin pääosassa on kolme eri entiteettiä eli kokonaisuutta, jotka ovat *hubit*,

linkit ja *satelliitit*. Hubit ovat mallin tärkein osa, joita linkit yhdistävät ja joihin satelliitit antavat näihin liittyvän kuvaavan tiedon. Tietovarastossa hubit, linkit ja satelliitit määritetään esimerkiksi relaatiotietokannan tauluina.

Data Vault -menetelmällä mallinnettu tietovarasto sisältää hyvin paljon tietokannan tauluja ja muita objekteja, sillä hubi-, linkki- ja satelliittitauluja voidaan muodostaa useampi yhtä lähdejärjestelmän taulua kohti. Data Vault -mallin entiteetit ovat kuitenkin tarkkaan standardoituja. Ne ovat tarkoituksella suunniteltu näistä toistuvista rakenteista niin, että samantyyppisten entiteettien kehitys on mahdollista monelta osin rinnakkaistaa [5].

Data Vault 2.0 tuo mukanaan myös uusia tauluja, joita voidaan tietovarastoon sisällyttää. Esimerkiksi Bridge- ja PIT-taulut (Point In Time), joiden käyttö esimerkiksi helpottaa datan lukemista tietovarastosta [13]. Hubit, linkit ja satelliitit ovat kuitenkin Data Vault -menetelmän tärkeimmät kokonaisuudet, joita ilman ei mallia voisi pitää Data Vault -mallina. Data Vault -menetelmän standardin versio 2.0.2 vuodelta 2018 on internetissä ilmaiseksi luettavissa [14].

3.1.1 Hubit

Hubit koostuvat liiketoiminnan käsitteen yksilöivistä avainkentistä [15]. Yksi hubi kuvaa tietovarastossa yhden käsitteen kuten työntekijä, asiakas tai tuote. Hubitaulu voi tietovarastossa kuitenkin sisältää muutakin dataa, kuten latauspäivämäärän tai tietoa lähdejärjestelmästä, josta taulun rivi on tullut. Data Vault 2.0 tuo mallille tavan käyttää kryptografisia hash-arvoja, joiden avulla voidaan muodostaa yksi avainkenttä useasta eri sarakkeesta, joka helpottaa hubien käyttöä ja parantaa suorituskykyä.

3.1.2 Linkit

Linkit kuvaavat liiketoiminnan käsitteiden väliset suhteet yhdistämällä yhden tai useamman hubin pääavaimet keskenään [15]. Linkit teknisesti voivat myös yhdistää linkkejä keskenään, tai vaikka toisen linkin ja hubin, mutta käytännössä tämä ei ole järkevää. Linkkejä toisiinsa yhdistäessä mallin skaalaantuvuus ja suorituskyky kärsivät, sekä mallin ylläpidosta tulee vaikeampaa [5]. Linkeilläkin voi olla muuta yleistä tietoa, kuten latauspäivämäärä ja lähdejärjestelmätietoa. Hash-arvoja käyttäessä linkkitaulu sisältää oman hash-avaimen, joka on kaikkien tämän linkin yhdistämien hubien avainten yhdistelmä, sekä erikseen näiden hubien hash-avaimet.

3.1.3 Satelliitit

Satellittit sisältävät kaiken muun liiketoiminnan käsitettä tai käsitteiden välistä suhdetta kuvaavan tiedon. Satelliitteihin tallentuu lähdejärjestelmistä saatava historioitava tieto [15]. Yksi satelliitti liittyy aina vain yhteen hubiin tai linkkiin, mutta hubeilla ja linkeillä voi olla useampia satelliitteja. Data Vault -mallin kannalta satelliittit tuovat hubeille ja linkeille kontekstin [2].

Hubien ja linkkien tapaan, satelliiteillakin on muuta yleistä tietoa, mutta tärkein satelliitteja kuvaava tieto on niihin liittyvän hubin tai linkin hash-avain, lähdejärjestelmän kuvaavan tiedon sisältävät sarakkeet, sekä mahdollisesti myös näistä sarakkeista yhdistetty hash-ero. Tämän erotuksen avulla voidaan satelliittin rivejä verrata yhden arvon avulla.

Yhden lähdejärjestelmän taulun datat voidaan jakaa usealle satelliitille. Esimerkiksi, jos lähdejärjestelmän taulussa on paljon sarakkeita, jotka sisältävät paljon erityyppistä dataa, niin ne voidaan jakaa kokonaisuuksittain useaan satelliittiin [16]. Kuitenkin, jos näitä sarakkeita esim. raportoinnissa aina kaikkia tarvitaan kerralla, niin ei ole kovin kannattavaa niitä eri satelliittitauluihin ladata. Se vain lisäisi ylimääräisen vaiheen raportointiin, jossa nämä sarakkeet yhdistetään takaisin yh-

teen. Jos vain joidenkin tiettyjen sarakkeiden arvot muuttuvat jatkuvasti, niin myös tällaiset sarakkeet voidaan ottaa omaan satelliittiinsa, jotta muuttumaton data ei turhaan monistuisi satellittitaulussa [16]. Myös eri lähdejärjestelmien datat ladataan yleensä omiin satelliitteihinsa.

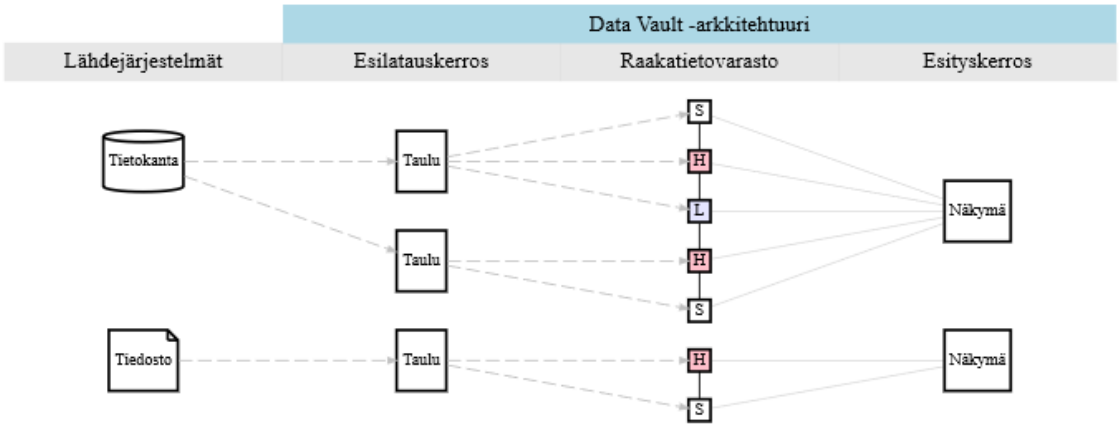
3.2 Arkkitehtuuri

Data Vault 2.0 -arkkitehtuuriin kuuluu kolme osaa. Ensimmäinen osa on datan esilatausalue (engl. Staging layer). Esilatausalueen päätarkoituksena on se, että data saadaan ladattua nopeasti pois lähdejärjestelmistä [15]. Toisena osana on Data Vault -mallinnettu raakatietovarasto, jossa data ladataan hubeihin, linkkeihin ja satelliitteihin. Raakatietovarasto on datan pääasiallinen tallennussijainti tietovarastossa, jossa dataa ei ole suodatettu eikä siihen ole laskettu raportointiin liittyviä liiketoimintasääntöjä. Hash-arvojen avulla on helppoa ja tehokasta varmistaa, että monistettua dataa ei mene tietovarastoon. Vertaamalla satelliitin hash-eroa tietovaraston rivin ja esilatausalueelta ladatun rivin kanssa, näkee suoraan onko rivin jonkin sarakkeen arvo muuttunut.

Kolmantena osana Data Vault -arkkitehtuurissa on esityskerros, jossa data lopullisesti muutetaan raportoinnissa hyödynnettävään muotoon. Varsinaisen Data Vault mallinnetun raakatietovaraston päälle voidaan esimerkiksi luoda *Business Vault* hyödyntäen Data Vault 2.0 määrittämiä PIT- ja Bridge-tauluja sekä datalle voidaan laskea raportoinnissa tarvittavia liiketoimintasääntöjä. Tämän päälle voidaan vielä osastokohtaisesti luoda dimensionaalisia informaatiomartteja, joissa data on suoraan raportoinnissa hyödynnettävässä ja jalostetussa muodossa [13]. Informaatiomartteja varten valitaan tietyt toisiinsa liittyvät hubit, linkit ja satelliitit ja näiden datat yhdistetään selviksi ymmärrettäviksi kokonaisuuksiksi.

Kuvassa 3.1 on esitettynä tämän kolmiosaisen Data Vault 2.0 -arkkitehtuurin idea. Esityskerros on kuvassa kuitenkin esitettynä vain yksinkertaisina näkyminä

raakatietovaraston dataan. Tämän päälle voisi Business Vaultin ja informaatiomartit tehdä raportoinnin ja datan analysoinnin helpottamiseksi.



Kuva 3.1: Data Vault -arkkitehtuuri

3.3 Vertailua muihin kehitysmalleihin

Data Vault -arkkitehtuuri on monelta osin yhteensopiva perinteisen Bill Inmonin tietovaraston kehitysmallin Corporate Information Factory (CIF) kanssa [2]. Bill Inmonin kehitysmalliin kuuluu keskeisenä osana kolmannen normaalimuodon mukaan mallinnettu relaatiotietokanta, johon tietovaraston data varsinaisesti tallennetaan. Data Vault korvaa tämän osan omalla hubeihin, linkkeihin ja satelliitteihin perustuvalla mallinnuksella. Monista muista tietovarastoista poiketen Data Vault mallinnetun tietovaraston viitteellinen eheys on täydellinen koko mallin yli ja sitä jatkuvasti ylläpidetään [2].

4 Tietovaraston kehityksen automatisaatio

Kehityksen automatisaatiota voi lähestyä monesta eri näkökulmasta. Tietovaraston kehitys on monivaiheinen prosessi, jonka täysin automatisoiva ohjelmisto vaatisi hyvin paljon erilaisia toimintoja. Käytännössä tällaisen ohjelmiston kehittäminen olisi siis hyvin hankalaa. Sen sijaan on kehitetty pienempiä työkaluja ja rajapintoja sovelluksiin, joiden avulla voi tiettyjä toimintoja kontrolloida toisten ohjelmien kautta. Automaatio on tärkeää, sillä se esimerkiksi auttaa tietovaraston laadun ylläpitämisessä, mahdollistaa nopeiden muutosten tekemisen liiketoiminnan tarpeiden muuttuessa, vähentää tietovaraston kehityksen, toiminnan ja ylläpidon kustannuksia sekä mahdollistaa nopean kehityksen, jolloin tietovaraston, liiketoiminnan vaatimusten ja datan mahdolliset ongelmat havaitaan nopeammin [17].

Data Vault -mallinnusta on suunniteltu tehtäväksi tällaisen automaatiotyökalun avulla. Data Vault -malli käyttää tarkasti standardoituja, toisistaan erillisiä osia, eli hubeja, linkkejä ja satelliitteja, jotka esimerkiksi helpottavat tietovaraston skeeman määrittämisessä. Näille osille on helpompi määrittää yksinkertaistettu metadatamalli. Linkit mahdollistavat mallin rakenteen muuttamisen liiketoiminnan tarpeiden muuttuessa ilman historiatietojen menettämistä, sillä itse historia on tallennettuna satelliiteissa, joiden ei tarvitse muuttua. Mahdollisuus muuttaa mallia ilman historian menettämistä on Lindstedtin mukaan kriittistä tietovaraston menestymisen ja

pitkän aikavälin kannattavuuden kannalta [2]. Muilla mallinnusmenetelmillä ei ole täysin vastaavia automaatiomahdollisuuksia.

Metadatan hallinta tietovaraston oikean toiminnan ja jatkokehityksen kannalta oleellinen osa tietovarastoa [18]. Tästä huolimatta suurimmalla osalla organisaatioista ei ole keskitettyä järjestelmää metadatan hallintaan ja prosessointiin. Pankov et al. mainitsevat tähän kolme pääsyytä. Ensinnäkään metadatan hallintaan ei ole olemassa vakiintuneita standardeja. Jokaisella työkalulla on oma lähestymistapansa metadatan tallentamiseen ja hallintaan. Metadatan keskitetty hallinta on suhteellisen monimutkainen tehtävä, joten metadatan hallinnan toteuttaminen voisi vaatia lisäresursseja ja investointeja. Organisaatioilla ei myöskään ole selvää ymmärrystä hyödyistä, joita metadatan keskitetty hallinta voisi tarjota.

4.1 Tietovaraston kehitys ilman automaatiota

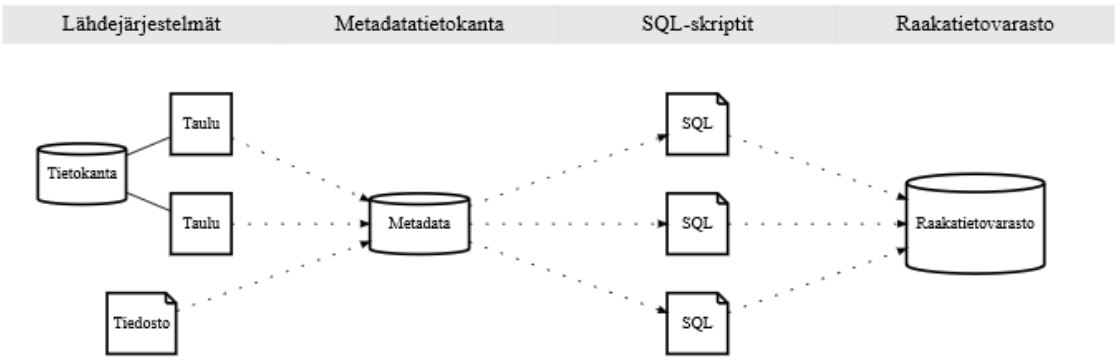
Perinteisesti hyvin suuri osa tietovaraston kehityksestä on tehty manuaalisesti [3]. Phipps ja Davis esittävät viiden vaiheen mallin tietovaraston kehitykseen: Aktiviteetit ennen tietovaraston kehitystä, arkkitehtuurin valinta, tietovaraston skeeman luonti, tietojen lataus tietovarastoon ja tietovaraston ylläpito. Ennen varsinaisen tietovaraston kehitystä huolehditaan esimerkiksi tietovaraston infrastruktuurin pystyttämisestä, kuten kehitysympäristön ja yhteyksien hankkimisesta eri järjestelmien välille.

Data Vault -mallinnetun tietovaraston kehityksessä aloitetaan yleensä esilatausalueen luomisesta, jossa on koostettuna kaikki lähdejärjestelmien taulut, jotka tietovarastoon halutaan ladata. Tämän jälkeen aloitetaan hubien, linkkien ja satelliittien mallin muodostaminen aloittaen hubitaulujen tunnistamisesta ja muodostaen linkkejä näiden hubien välille [12].

4.2 Lähestymistapoja kehityksen automatisointiin

Tietovaraston kehityksen automatisoinnista on jonkin verran tehty akateemista tutkimusta. Tutkimusta on tehty ainakin tietovaraston skeeman automatisoinnin [3] ja yleisen tietovaraston skeeman [18] kannalta. Akateemista tutkimusta on myös tehty hyvin hajanaisesti eri osa alueilla, ja koska tietovaraston kehitykseen ei ole selvää vakiintunutta standardia, niin tutkimuksien yhdistäminen ei ole kovin helppoa. Data Vault on suhteellisen hyvin vakiintunut menetelmä, mutta uutta tutkimusta keskittyen juuri Data Vault -mallinnetun tietovaraston kehityksen automatisointiin ei ole tehty paljoa [4].

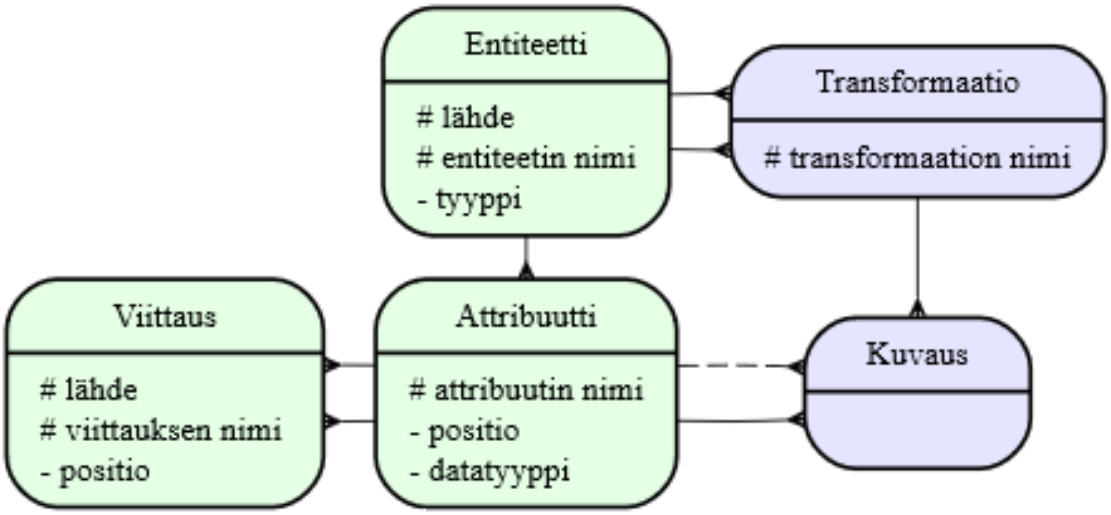
Kuvassa 4.1 on yleiskatsaus automatisaation lähestymistapaan, jonka peruspe-riaatteet ovat hyvin lähellä kaikkia tässä luvussa esiteltyjä lähestymistapoja. Perusideana kaikissa näissä automaatioratkaisuissa on luoda metadatatietokanta, johon erilaisten prosessien avulla ladataan lähdejärjestelmien metadatatiedot ja jonka avulla automaatiotyökalu luo raakatietovaraston tietokantaan tarvittavat objektit esimerkiksi käyttäen SQL-skriptejä. Metadatan tuonti lähdejärjestelmistä metadatamalliin voidaan myös osittain automatisoida [19].



Kuva 4.1: Yleiskatsaus raakatietovaraston objektien generoinnin lähestymistapaan

4.2.1 Puonti et al. lähestymistapa

Mikko Puonti et al. ovat tutkineet mitä Data Vault -menetelmällä mallinnetun tietovaraston ETL-prosessin transformaatioista on mahdollista automatisoida tietovaraston populoinnin vaiheessa. Esittelevät artikkelissaan ”Automating Transformations in Data Vault Data Warehouse Loads” (2017) kevyen tietovaraston metadatamallin, jonka avulla voi luoda tietovaraston SQL skriptit Data Vault mallin raakatietovaraston hubi-, linkki- ja satelliittitaulujen sekä näiden latausproseduurien luontiin. Kuvassa 4.2 Puonti et al. esittelemä metadatamalli.



Kuva 4.2: Puonti et al. esittelemä metadatamalli.

Pääideana metadatamallissa on, että se esittää erikseen kohdemallin rakenteen informaation ja rakenteen välisen tietovirran informaation. Metadatamallin vahvuutena on, että kohdemalli esitetään pienempien tietokantaobjektien mallien välisinä kuvauksina (engl. mapping). Rakenteellinen informaatio on kuvassa vihreällä ja tietovirran informaatio violetilla värillä. #-merkillä on esitetty yksilöivät tai avaimena toimivat attribuutit, mutta mallissa ei kuitenkaan tarkasti oteta kantaa siihen, miten nämä mallin osat yhdistetään toisiinsa ja esimerkiksi tämän mallin vierasavainviittauksia ei ole annettu. Malli on enemmän looginen malli eikä sen osat välttämättä

suoraan viittaa tietokannan tauluihin.

Entiteetti kuvaa tässä tietokannan taulua tai näkymää, mutta tietokannan ei mallin kannalta tarvitse olla relaatiotietokanta. Entiteetti voi siis kuvata jonkin muunkin tyyppistä objektia. Käytännössä entiteetti kuvaa kohdetietovaraston tauluja kuten esilatausalueen, hubien, linkkien ja satelliittien tauluja. Entiteettillä on tyyppi, joka kertoo mikä entiteetti on kyseessä. Entiteetin omistaja (viitaten lähdejärjestelmän tunnisteeseen) ja entiteetin nimi yksilöivät entiteetin. Lähdejärjestelmän tunnisteen merkitystä ei selitetä, mutta oletettavasti se on malliin lisätty, jotta voi kahdesta eri lähteestä tulla samannimiset entiteetit. Attribuutti kuvaa entiteetin saraketason tietoa eli sarakkeiden nimet, tietotyypit ja mahdollisesti myös muutakin sarakkeiden metatietoa, jota tietovarastoon tallentuu. Viittaus kertoo attribuuttitasolla vierasavainviittaukset.

Tietovirran informaation entiteettien välillä kuvaa transformaatio ja attribuuttien välillä kuvaus (engl. mapping). Transformaatio käytännössä liittää siis kaksi entiteettiä yhteen ja kuvaus yhdistää kaksi kahden eri entiteetin attribuuttia yhteen. Tietovirran informaatio siis kertoo minkä entiteettien ja attribuuttien välillä data kulkee ja mihin suuntaan. Kuvaukselle ei attribuutteja ole malliin kuvattu eikä yhteyttä attribuuttiin ja transformaatioon tarkasti selitetty. Mallin voi toteuttaa monella tavalla, eikä ajatuksena tunnukaan olevan, että tämän pohjalta tehty toteutus vastaisi täysin tehtyä PL/SQL koetoteutusta. Toteutusta ei ole edes julkaistu, joten siihen ei voi verrata. Tietovirran informaatio sisältää tiedon jäljitettävyyden (engl. lineage) kannalta tärkeää tietoa ja puonti et al. mainitsevat, että tämän mallin avulla voidaan tietovaraston objektien väliset yhteydet esittää myös graafisesti erilaisilla työkaluilla.

Metadatamallin rakenneosa voidaan täyttää olemassa olevan tietovaraston pohjalta, jossa tietovaraston kaikki taulut (mukaanlukien hubi-, linkki- ja satelliittitaulut) on jo luotuina erillisellä työkalulla. Metadatamallin täyttäminen onnistuu tämän

jälkeen esimerkiksi seuraten vierasavainviittauksia satelliiteista aloittaen esilatausalueen tauluille asti ja keräämällä entiteetti, attribuutti ja viittaustiedot aina uusien objektien tullessa vastaan. Tällä lähestymistavalla on PL/SQL koetoteutus tehty, mutta he mainitsevat myös, että metadatamalli voisi olla myös pääasiallinen tietovaraston objektien määrityssijainti, josta näille tauluille luotaisiin SQL-luontilauseet. Mainitaan myös, että mallin kautta voisi tehdä myös muutoksia tietovarastoon, mutta se ei ole ollut sen alkuperäinen tarkoitus. Pääasiassa automaattisesti generoitujen objektien osalta on keskitytty vain latausproseduurien luontiin, jotka lataavat datan esilatausalueelta Data Vault -mallin tauluihin, sekä tietokannan näkymien luontiin, jotka näyttävät satelliiteissa ja linkeissä olevan uusimman datan ilman historiaa.

Kokonaisuudessaan metadatamalli on kevyt ja voi olla käytännöllinen joissain tapauksissa. Se on helposti käyttöönotettavissa, sillä se ei tarvitse ulkoista työkalua sen käyttämiseen. Käytännössä kuitenkin vaaditaan, että tietovaraston taulut mallinnetaan ulkoisella työkalulla. Malli ei myöskään tarjoa hyviä ominaisuuksia tietovaraston ylläpitoon. He kuitenkin mainitsevat myös, että metadatamallin kautta voisi Data Vault -mallin taulurakenteenkin määrittää ja ylläpitoakin hoitaa, mutta eivät anna näihin yksityiskohtia.

Joulukuussa 2019 Mikko Puonti ja Timo Raitalaakso julkaisivat tälle menetelmälle jatkoa artikkelissa ”Data Vault Mappings to Dimensional Model Using Schema Matching” [20], jossa he keskittyivät transformaatioiden luomiseen myös Data Vault raakatietovaraston ja esityskerroksen välille. Esitetyt menetelmät ovat toistensa kanssa yhteensopivia. He mainitsevat kuitenkin, että transformaatioiden luontia ei menetelmällä täysin pysty automatisoimaan, sillä esityskerroksen määrittelyt vaativat enemmän tapauskohtaista määrittelyä ja liiketoimintasääntöjä. Automaattisesti voi kuitenkin luoda perustan, jonka päältä tietovarastojen asiantuntijat voivat jatkaa monimutkaisempien transformaatioiden tekoa.

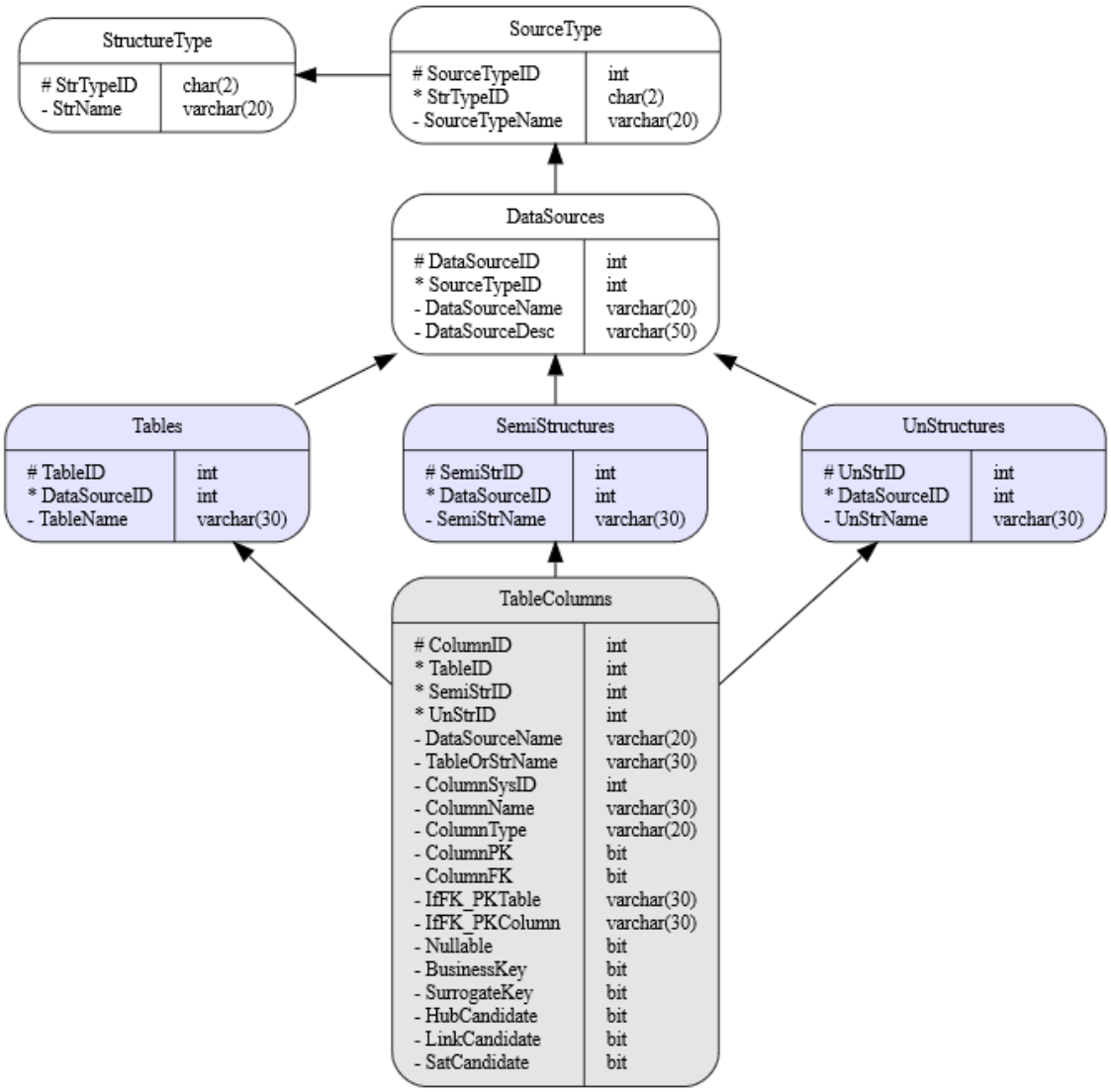
4.2.2 Krneta et al. lähestymistapa

Dragoljub Krneta et al. esittelevät artikkelissaan ”A Direct Approach to Physical Data Vault Design” (2014) ketterän lähestymistavan suuren mittakaavan yritystietovaraston suunnitteluun Data Vault -mallin pohjalta. Se monilta osin on verrattavissa Puonnin et al. esittämään Data Vault raakatietovaraston transformaatioiden automatisointitapaan, mutta esitetty malli on osittain vielä monipuolisempi, jossa automatisoidaan koko Data Vault mallin suunnittelu alusta lähtien. Kuvassa 4.3 Krneta et al. esittelemän metadatatietokannan malli. Pääavaimet on kuvassa merkittynä #-merkillä ja vierasavaimet *-merkillä. Violetilla on merkitty osittain Puonti et al. lähestymistavan entiteettitaulua vastaavat taulut ja harmaalla attribuuttitaulua vastaava saraketaulu.

Esitetty algoritmi hyödyntää tämän metatietomallin lisäksi erillistä mallinnussääntöjen mallia, jonka avulla algoritmi tekee esimerkiksi automaattiset päätökset Data Vault -entiteeteille, eli määrittelee mitkä lähdetaulujen sarakkeet kuuluvat hubeille, linkeille ja satelliiteille. Sääntöjen malli on kuvattuna kuvassa 4.4. Yksittäinen sääntö kuvataan sääntötaulussa SQL-lauseilla. Päätöksentekoon kuuluu myös käyttäjän manuaalisesti käyttöliittymän kautta syöttämät taulujen luonnolliset avaimet, joita ei automaattisesti voi saada lähdejärjestelmistä.

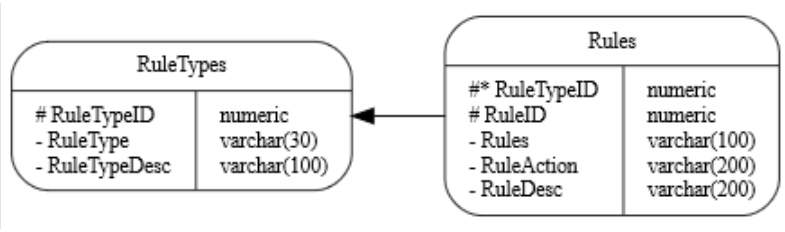
Algoritmia voidaan hyödyntää strukturoiduille, puolistrukturoiduille ja yksinkertaisille strukturoimattomille tietolähteille. Näillä on kehitetyssä fyysisessä mallissa omat roolinsa. Rakenteettoman datan hallintaa voidaan auttaa esimerkiksi muuttamalla se rakenteelliseksi dataksi tekstianalyysiä hyödyntäen, mutta se voidaan tallentaa myös sellaisenaan tietovarastoon käyttämällä geneerisiä tietotyyppejä.

Automatisoinnin neljä päävaihetta ovat metadatamallin käsitteellistäminen, tietolähteiden valinta, Data Vault -mallin tunnistaminen ja Data Vault -mallin määritys. Ensimmäisessä vaiheessa rakennetaan metadatan ja mallinnussääntöjen mallit, generoidaan esilatausalueen taulujen luontilauseet ja määritetään sääntöjen malliin



Kuva 4.3: Krneta et al. esittelemä metadatamalli

halutut mallinnussäännöt sekä metadatamalliin rakenteiden ja lähteiden tyypit. Toisessa vaiheessa metadatamalliin täytetään käyttöliittymän kautta valitut tietolähteet, sekä määritetään tietolähteiden mukaan metadata eri rakennetyypeille ja näiden taulujen sarakkeille metadatamalliin. Kolmannessa vaiheessa käyttäjä valitsee käyttöliittymän kautta sarakkeille luonnolliset avaimet ja määritettyjen sääntöjen avulla käyttöliittymä ehdottaa Data Vault -mallin eli miten eri sarakkeet jakautuvat hubeille, linkeille ja satelliiteille. Neljännessä vaiheessa luodaan automaattisesti näiden määritysten perusteella SQL-skriptit hubi-, linkki- ja satelliittitaulujen luon-



Kuva 4.4: Krneta et al. esittelemä sääntöjen malli

tiin.

Tärkeimmäksi artikkelin esittelemäksi toiminnoksi Krneta et al. mainitsevat tämän automaattisen Data Vault -mallin luonnin suoraan käyttäen lähteen relaatiotietokannan skeemaa. Tämän sanotaan olevan mahdollista Data Vault -mallin ansiosta, sillä se erottaa muuttumattomat oikeiden järjestelmien entiteettien identiteetit eli hubit, linkit ja satelliitit toisistaan. He väittävät myös, että mikä tahansa epäsuora Data Vault -mallin kehitys, joka käyttää konseptuaalista tai loogista (olemassa olevista tietolähteistä johdettua) tietomallia, on vähemmän joustava kuin tällainen suoraan lähdeskeemasta johdettu fyysinen Data Vault -malli, vaikka kehitys olisi osin automaatiolla tuettu.

Mainitaan myös, että esitetty lähestymistapa ei muun muassa käsittele data marttien suunnittelua, datan virtualisointia, tietovaraston skeeman evoluutiota, master datan hallintaa, hybriditietokantoja eikä täysin selvitä ELT/ETL transformaatioiden automatisointia, sillä näiden ongelmien sanottiin olevan osa isompaa meneillä olevaa tutkimusohjelmaa. Vuonna 2016 he julkaisivat artikkelin ”An Approach to Data Mart Design from a Data Vault”, jossa he keskittyivät tarkemmin data marttien luontiin Data Vault -menetelmällä mallinnetun raakatietovaraston pohjalta [21].

Krneta et al. esittelemä lähestymistapa automatisointiin on selvästi raskaampi kuin Puonti et al. esittelemä lähestymistapa. Krneta et al. algoritmi tarvitsee ulkoisen käyttöliittymätyökalun, joka auttaa mallin käytössä. Krneta et al. esittelemä malli kuitenkin tekee enemmän ja hieman eri asioita, kuin Puonti et al. esittele-

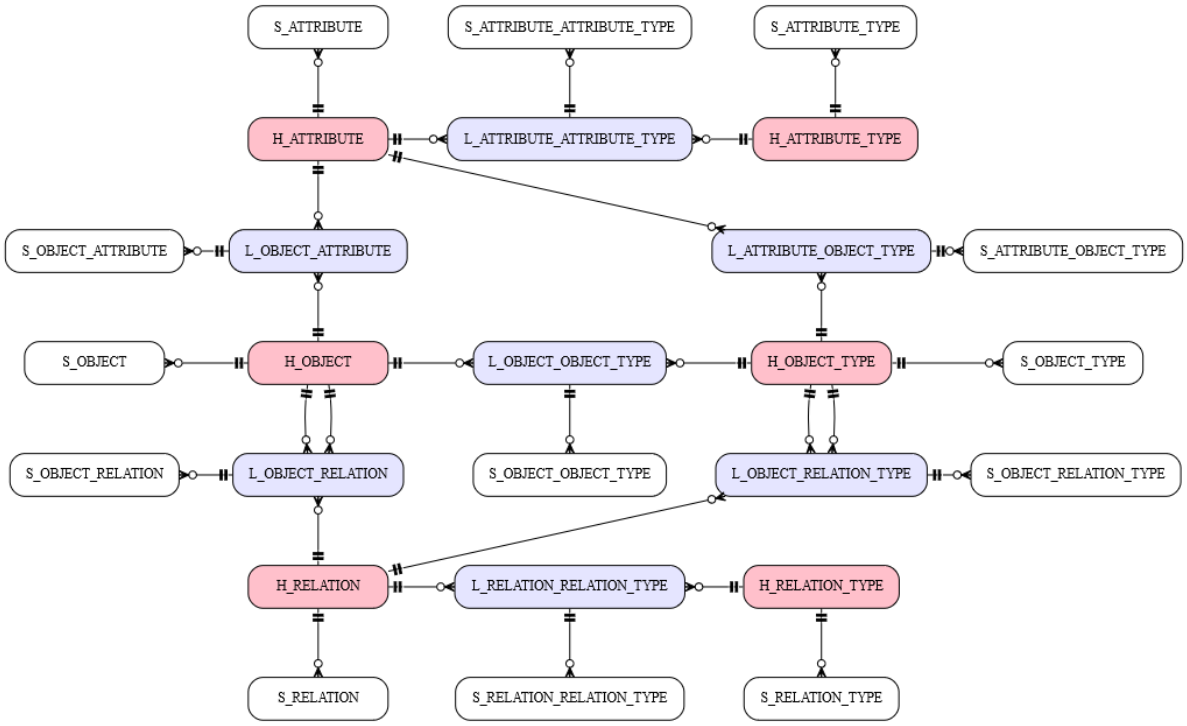
mä malli, joten ne eivät ole täysin toisiinsa verrattavissa. Krneta et al. keskittyvät enemmän Data Vault taulujen generointiin ja Puonti et al. taulujen väliseen ETL-prosessin latauksiin. ETL-prosessin määrittäminen olisi Krneta et al. lähestymistavan avulla myös mahdollista.

4.2.3 Pankov et al. lähestymistapa

Pankov et al. esittelevät artikkelissaan ”Towards a Generic Metadata Warehouse” (2014) yleistetyn Data Vault -mallinnusta käyttävän metatietokannan mallin, jonka pohjalta tietovarastojen standardisointi ja automatisointi on mahdollista kehittää ja implementoida. Tämä ei kuitenkaan Puonnin ja Krnetan malleista poiketen ole suoraan Data Vault -mallinnetun tietovaraston kehitykseen tarkoitettu, vaan on yleinen malli, joka vaatii aivan omaa standardiaan käyttävän työkalun, jotta mallia voi käyttää.

Päätavoitteeksi mainitaan suunnittelu ja toteutus geneeriselle metadatapohjaiselle prototyypille tietovarastojen luomiseen ja hallintaan, sekä ETL-prosessien ja tietomallien generointiin. Metadatamallin halutaan olevan geneerinen, joka sallii minkä tahansa tyyppistä metadataa ja näin antaa näkökulman kokonaisuudessaan kaikille objekteille, liiketoimintasäännöille, prosesseille ja näiden riippuvuuksille organisaation järjestelmissä. Tästä syystä lähestymistapa seuraa Matthew Westin yleisen suunnittelun periaatteita, joiden seuraaminen auttaa laadukkaiden datamallien kehittämisessä [22]. Kuvassa 4.5 lopullinen Pankov et al. esittelemä metatietokannan malli.

Metadatamalli koostuu kolmesta pääosasta. Objektista, attribuutista ja relaatiosta, jotka ovat mallissa hubeina. Objekti on mallin keskeisin osa, joka kuvaa kaikki yksinään esiintyvät käsitteet. Se käytännössä vastaa Puonti et al. mallin entiteettitaulua, mutta on kuvattu vielä korkeammalla abstraktiotasolla, eikä se välttämättä kuvaa Data Vault -menetelmällä mallinnettua entiteettiä. Objektin ominaisuuksia



Kuva 4.5: Pankov et al. esittelemä yleinen metatietomalli

kuvaa attribuutit, eli esimerkiksi ajatellessa tietokannan taulua objektina, niin sen sarakkeet ovat attribuutteja. Relaatiot kuvaavat objektien väliset suhteet, eli tauluilla vastaavasti vierasavainviittaukset. Objektilla, attribuutilla ja relaatiolla on myös omat tyyppinsä, jotka ovat mallissa kuvattu myös hubeina.

Nämä kuusi hubia muodostavat metadatamallin rungon ja Data Vault -menetelmää seuraten näiden väliset yhteydet mallinnetaan linkeillä. Jokaiselle metadatamallin linkille ja hubille yhdistetään myös satelliitti, johon tallennetaan metadatamallin kuvaavat tiedot. Satelliitteihin ja linkkeihin tallentuu historia siitä, miten metadatamalli on muuttunut ajan myötä.

Yleisen metadatamallin sanotaan tuovan paljon hyviä puolia, mutta myös lisäävän kompleksisuutta. Metadatamalliin tallennetusta datasta ei suoraan näe, mitä objekteja mallissa on ja mitkä näiden suhteet toisiinsa ovat. Monimutkaisilla SQL kyselyillä voisi dataa yhdistellä, mutta se olisi nopeasti raskasta ja työlästä. Tähän ehdotetaan metadatan hallintaohjelmaa, joka voi myös tarjota käyttäjälle graafisen

käyttöliittymän (Krneta et al. lähestymistapaa vastaavalla tavalla). Hallintaohjelma piilottaisi kompleksisuuden ja mahdollistaisi keskittymisen suoraan prosessien määrittelyyn eikä mallinnustavan määrittelyyn. Tämän sanotaan pienentävän kustannuksia ja tarvittavaa aikaa ETL-prosessien määrittelyyn. Metadatan hallintaohjelman ehdotetaan esittävän metadatamallin muokkausoperaatiot API-rajapintana. ETL-prosessin generoinnissa puolestaan hallintaohjelma voisi itse käyttää olemassaolevien ETL-työkalujen API-rajapintoja metadatamallin määrittelyjen tuomiseksi tietovarastoon.

Keskittyessä Data Vault -menetelmällä mallinnetun tietovaraston kehitykseen, Pankov et al. esittelemä lähestymistapa on näistä esitetyistä lähestymistavoista selvästi raskain ja epäkäytännöllisin. Menetelmää ei ole Data Vault -menetelmän periaatteilla mallinnettua tietovarastoa varten suoraan suunniteltu. Lähestymistapa ei kuitenkaan ota kantaa tietovaraston objektien generointiin ja koska metadatamalli on hyvin yleisellä tasolla suunniteltu, niin Data Vault -mallinnetun tietovaraston generointi tämän metadatamallin pohjalta voisi olla mahdollista. Pankov et al. eivät tätä kuitenkaan mainitse.

5 Aveso DW Automation -työkalu

Aveso DW Automation -työkalu on kehitetty helpottamaan Data Vault -mallinnetun tietovaraston luontia ja sen toimintaperiaate on monelta osin samankaltainen Krneta et al. esittämään lähestymistapaan verrattuna. Muutkin edellisessä kappaleessa esitetyt lähestymistavat hyödyntävät metatietokantoja, joten kaikilla näillä on joitain samankaltaisuuksia. Aveso DW Automation -työkalun lähestymistapa automatisaatioon on raskaampi kuin Puonti et al. esittelemän lähestymistapa, mutta ei niin raskas kuin Pankov et al. lähestymistapa.

Aveso DW Automation on enemmän kehitystyökalu, kuin suunnitteluun tarkoitettu työkalu. Sillä ei myöskään ole Krneta et al. esittelemän lähestymistavan ominaisuutta automaattisesti luoda Data Vault -malli lähdejärjestelmän skeeman mukaan, vaan Data Vault -malli määritetään kokonaisuudessaan käyttöliittymän kautta. Ideana on yleensä, että Aveso DW Automation -työkaluun tuodaan valmiiksi suunniteltu malli, jonka työkalu generoi tietovarastoon ja jota työkalun avulla voidaan ylläpitää.

Joustavien SQL-templaattien avulla se voi generoida käytännössä lähes mitä tahansa tietokannan objekteja metatietokannan määrittelyjen perusteella. Käytännössä keskitytään hubi-, linkki- ja satelliittitaulujen luontilauseiden, näiden latausproseduurien (ETL-prosessin esilatausalueen ja raakatietovaraston välillä) sekä raakatietovaraston lukua helpottavien näkymien generointiin.

5.1 Toimintaperiaate

Aveso DW Automation -työkalun keskeisimpinä osina ovat metatietokanta, templaattien hallinnan komponentti sekä web-pohjainen käyttöliittymä. Tietovarastoon syötetään käyttöliittymän kautta Data Vault -malli, SQL-templaatit sekä muuta metatietoa. Tämän jälkeen voidaan käyttöliittymästä käynnistää templaattien hallinta, joka generoi kaikille määritetyille entiteeteille SQL-skriptit täyttämällä SQL-templaatteihin määritellyn metadatan. SQL-skriptit voidaan myös automaattisesti ajaa tietovarastoon.

SQL-templaateista voidaan tehdä dynaamisia niin, että jos tietovarastoon on jo joitain tauluja viety, niin uusien skriptien ajaminen tietovarastoon ei ylikirjoita näitä vanhoja olemassa olevia tauluja. Tämä helpottaa ylläpitoa, sillä mallia voi laajentaa huoletta, eikä tarvitse olla kovin tarkkana, että tuhoaisi tietovarastossa olevaa dataa vahingossa. Muitakin ylläpitoa helpottavia ominaisuuksia työkalussa on. Esimerkiksi kaikki määrittelyt voidaan versioida omalla id:llään metatietokantaan ja aikaisempiin määrittelyihin voidaan näin palata, esimerkiksi jos huomataan että uusissa määrittelyissa on jokin virhe. Näin saadaan myös tallennettua historiaa, josta nähdään miten määrittelyt ovat muuttuneet ajan myötä. Tämä on verrattavissa Pankov (et al) esittämään Data Vault -mallinnettuun metatietokantaan, jossa historia tallentuu Data Vault -menetelmän mukaisesti linkki- ja satelliittitauluihin. Erona on kuitenkin se, että Aveso DW Automation -työkalun versiointi tapahtuu manuaalisesti kokonaisuuksittain, jolloin vain tietyt halutut muutokset tulevat versioiduiksi.

Aveso DW Automation ei hallitse generoitujen latausproseduurien suoritusta, se vain generoi ne tietokantaan [23]. Datojen lataus lähdejärjestelmistä esilatausalueelle ja latausproseduurien ajastettun suorituksen hallinta tehdään ulkoisella työkalulla. Aveso DW Automation on oletuksena konfiguroitu käyttämään kaksiosaista esilatausaluetta, joka koostuu extract- ja staging-tauluista. Extract-taulujen raken-

ne ladataan ulkoisella työkalulla lähdejärjestelmien mukaiseksi ja niitä käytetään myös metadatamallin täyttämisessä. Extract-tauluista saadaan esimerkiksi kaikki sarakkeiden tietotyypit lopulliseen tietovarastoon. Staging-taulut ovat täysin SQL-templaateilla määritettyjä tauluja, joihin voidaan laskea hash-arvoja datalle valmiiksi ennen lopullista siirtoa tietovarastoon.

5.1.1 Metadatamalli

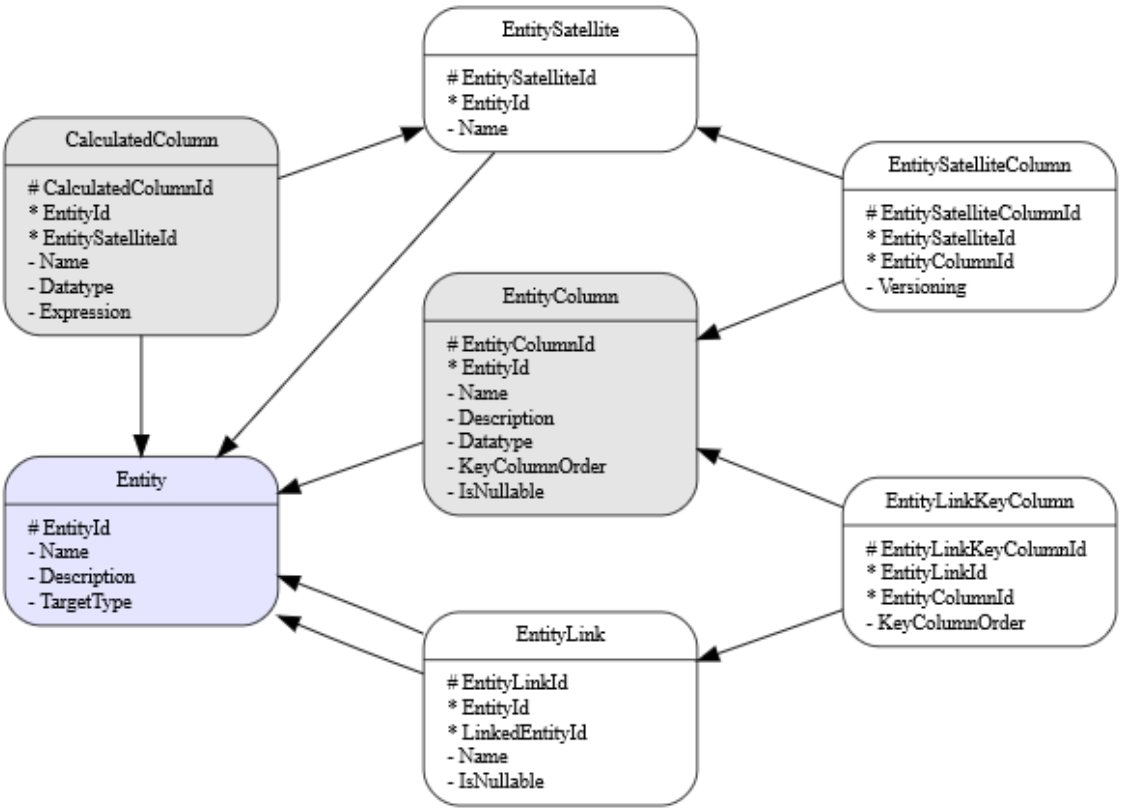
Kuvassa 5.1 on hieman yksinkertaistettu malli Data Vault -mallinnuksen käyttämästä osasta työkalun metatietokannasta. Pääosana mallissa on entiteetti, joka Aveso DW Automationin kannalta kuvaa Data Vault -mallin hubi- tai linkkitaulua. Entiteettitaulu on kuvassa merkitty violetilla värillä, vastaavasti kuin Krneta et al. metadatamallin kuvassa 4.3. Entiteettiin voi kuulua yksi tai useampi satelliitti (tai ei satelliittia ollenkaan) ja satelliitille kuuluu tietyt omat sarakkeet (EntitySatelliteColumn).

Yleisessä saraketaulussa (EntityColumn) on kaikki entiteetille kuuluvat sarakkeet, jotka voidaan tähän hubiin tai linkkiin (entiteetin tyypin perusteella) tai siihen liittyville satelliiteille ladata. Satelliitille voidaan myös lisätä laskettuja sarakkeita, jotka määritellään SQL-lausekkeen avulla (CalculatedColumn). Nämä kaksi saraketaulua on kuvassa 5.1 merkittynä harmaalla värillä.

Linkki yhdistyy hubiin linkitystaulun (EntityLink) avulla ja linkitetyn hubin avainsarakkeet yhdistetään linkin vierasavainsarakkeisiin avainsaraketaulun avulla (EntityLinkKeyColumn).

5.1.2 SQL-templaatit

SQL-templaatit ovat käytännössä tavallisia SQL-skriptejä, mutta niistä on vain osia korvattu erilaisilla tageilla, jotka korvataan generointivaiheessa entiteettikohtaisilla tiedoilla. Esimerkiksi tagi `<entity>` korvaantuu entiteetin nimellä. Tagit käyttävät

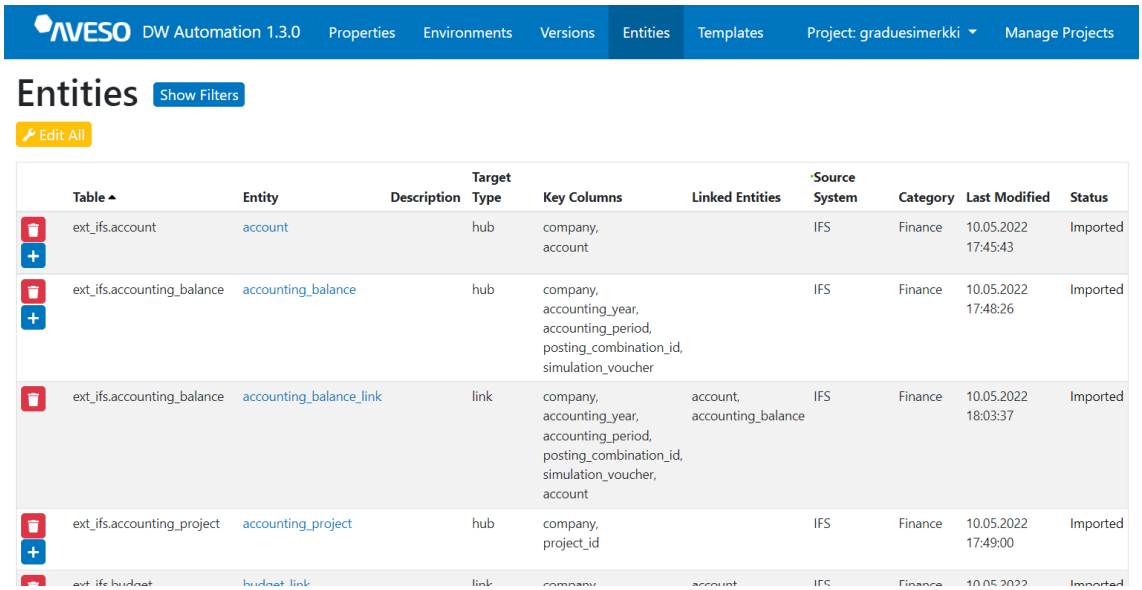


Kuva 5.1: Aveso DW Automation -työkalun käyttämä metatietomalli

XML-merkintäkielestä inspiroitua syntaksia, jossa tageilla voi olla attribuutteja, esimerkiksi muodossa `<tagi attribuutti="arvo"/>`. Kaikilla tageilla ei kuitenkaan ole attribuutteja määritettynä, joten attribuutteja voi käyttää vain tietyillä tageilla. Attribuuteilla on aina jokin tagikohtainen erityismerkitys.

5.1.3 Käyttöliittymä

Käyttöliittymä auttaa metadatamallin täyttämisessä ja huolehtii, että metadatamallia täytetään oikealla tavalla. Käyttöliittymästä hallitaan mallin versioita, voidaan malleille generoida dokumentaatiotiedosto, josta näkee mallin määritykset kokonaisuudessaan. Kun malli on valmis vietäväksi tietokantaan, niin käyttöliittymästä hallitaan julkaisuprosessia eri ympäristöihin. Kuvassa 5.2 esimerkki käyttöliittymän entiteettisivusta.



Kuva 5.2: Aveso DW Automation -työkalun käyttöliittymän entiteettisivu

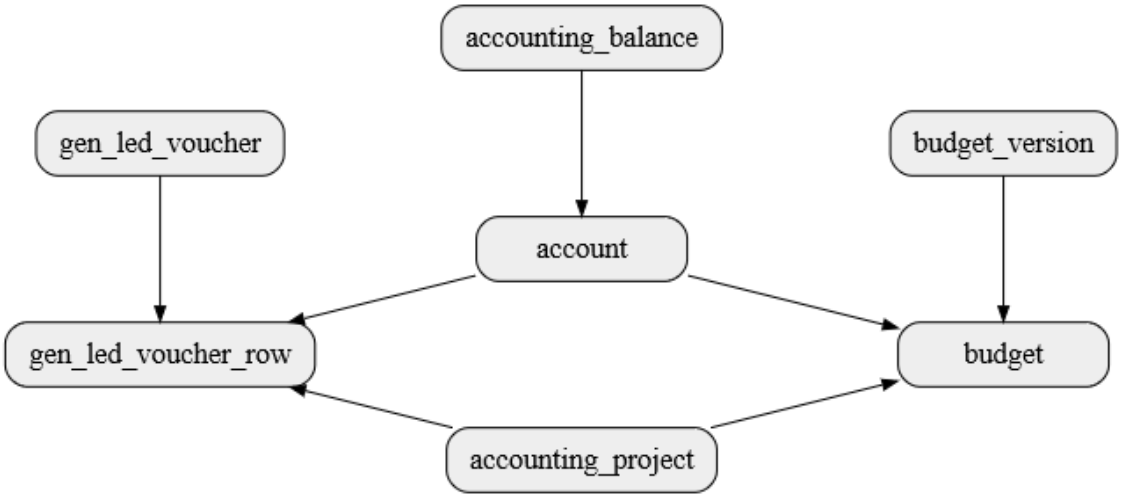
5.2 Mallinnustilanteen esimerkki

Tässä luvussa esitellään millaista mallinnus Aveso DW Automation -työkalulla on ja käydään läpi vaiheita, joita työkalun käytössä täytyy ottaa huomioon. Tarkastellaan myös, miten käyttöliittymän kautta tuodaan metadatatietokantaon Data Vault -mallinnuksessa tarvittava data ja miten lopulliset tietovaraston objektit generoidaan määritetyn mallin perusteella.

5.2.1 Lähdejärjestelmän määritys

Lähdejärjestelmänä käytetään tässä ERP-järjestelmää. ERP-järjestelmät eli toiminnanohjausjärjestelmät ovat yritysten tietojärjestelmiä liiketoimintaprosessien integroituun hallintaan [24]. Esimerkissä käytettävä malli on peräisin IFS:n ERP-järjestelmästä. IFS on maailmanlaajuinen liiketoimintajärjestelmiin keskittyvä yritys. Kuvassa 5.3 on sisäinen, yksinkertaistettu malli IFS ERP:in talousdatoista, joita Aveso DW Automation -työkalulla esimerkissä mallinnetaan. Pääasiassa talousdatana on yleinen pääkirjadata (engl. general ledger) ja budjettidata, joihin yhdistetään tili, projekti

ja muuta oleellista tietoa. Liitettävää dataa on mahdollista kuitenkin IFS:n ERP-ohjelmiston kautta lisätä ja yrityskohtaisesti valita. Esimerkissä on nyt oletettu, että vain oleellisimmat datat tuodaan malliin mukaan, jotta esimerkki pysyy mahdollisimman yksinkertaisena.



Kuva 5.3: Esimerkki IFS:n ERP-järjestelmästä ladattavasta talousdatojen näkymien mallista

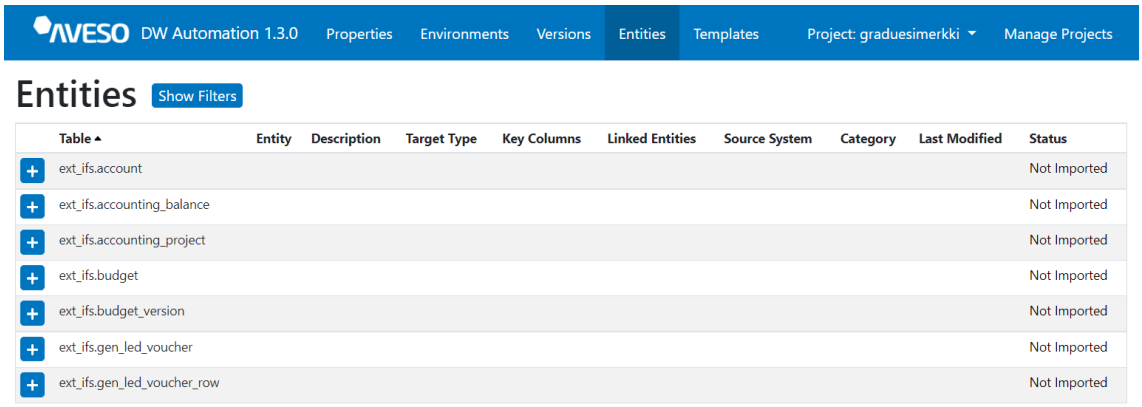
5.2.2 Konfigurointi

Tietovaraston kehityksessä aloitetaan Aveso DW Automation -työkalun kannalta siitä, että luodaan esilatausalueelle extract-taulut, jotka ovat lähdejärjestelmien mukaisessa muodossa oikeilla tietotyypeillä. Extract-taulut voidaan luoda manuaalisesti, mutta jos tauluja on paljon, niin ne voidaan myös generoida automaattisesti ulkoisilla työkaluilla.

Extract-taulujen luonnin jälkeen voidaan aloittaa Data Vault tietomallin ja muun metadatan määrittäminen Aveso DW Automation -työkaluun. Ensin määritellään uusi projekti ja sen alle ympäristö eli käytännössä yhteys siihen SQL-palvelimen tietokantaan, jonne tietovarasto halutaan luoda ja jossa extract-taulut on luotuna. Useampikin ympäristö voidaan luoda, esimerkiksi kun halutaan luoda erikseen

kehitysympäristö ja tuotantoympäristö.

Kun työkalu saa yhteyden palvelimelle, niin voidaan entiteettien eli hubi, linkki ja satelliittirakenteen määrittäminen aloittaa. Entiteettisivulla tulee näkyviin luodut extract-taulut. Kuvassa 5.4 näkyy miltä entiteettisivu tälle esimerkille näyttää nyt näiden alustavien määritysten jälkeen. Kaikki taulut ovat tilassa ”Not Imported”, joka kertoo sen, että millään taululla ei vielä ole mitään määrittelyitä metadatatietokannassa.



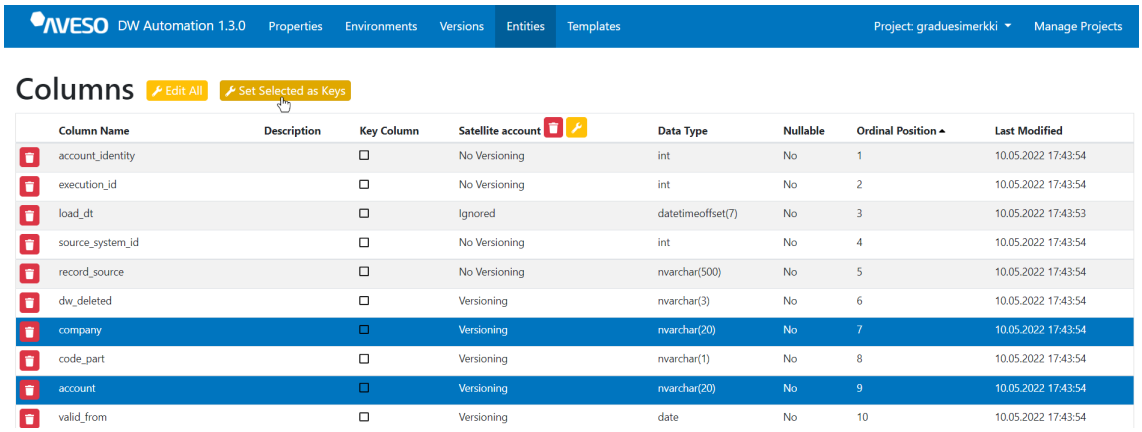
Kuva 5.4: Aveso DW Automation -työkalun käyttöliittymän entiteettisivu extract-taulujen luonnin jälkeen ennen määrittelyjä

Mikäli oletusarvot on määritelty hyvin, niin entiteettien määritys on käytännössä vain sitä, että jokaiselle extract-taululle määritetään näiden taulujen pääavaimet, sekä mihin muotoon niiden datat menevät, eli yhdistyvätkö ne hubiin, linkkiin vai molempiin ja onko näillä hubeilla ja linkeillä liittyviä satelliitteja [23]. Kuvassa 5.5 näkyy account-taulun avainkenttien määritys. Account-taulun avainkentät ovat yritystunnus ja tilinumero. Account-taulusta on kuvassa määritetty hubi ja siihen on liitetty myös satelliitti.

Kun käyttöliittymästä luodaan extract-tauluun liittyvä entiteetti, niin työkalu automaattisesti hakee extract-taulusta kaikkien sarakkeiden metadatat, eli käytännössä sarakkeiden nimet ja tietotyypit EntityColumn-tauluun, sekä luo entiteetil-

le rivin Entity-tauluun. Jos valitaan, että tähän halutaan satelliitti liittää, niin myös EntitySatellite-tauluun luodaan automaattisesti rivi ja EntitySatelliteColumn-tauluun tuodaan viittaukset EntityColumn-tauluun niille sarakkeille, jotka satelliitille halutaan.

Manuaalisesti käyttöliittymästä syötetyt avainkenttien määritykset tallentuvat suoraan EntityColumn-tauluun ja linkeille määritetään erikseen EntityLink-tauluun sen yhdistämät hubit. EntityLinkKeyColumn-tauluun määritetään kuvautumiset, miten näiden hubien avainkentät vastaavat linkin avainkenttiä. Nämä avainkentät saadaan myös automaattisesti, mutta vain jos kaikki vastaavat avainkentät on nimetty täysin samoilla nimeillä.

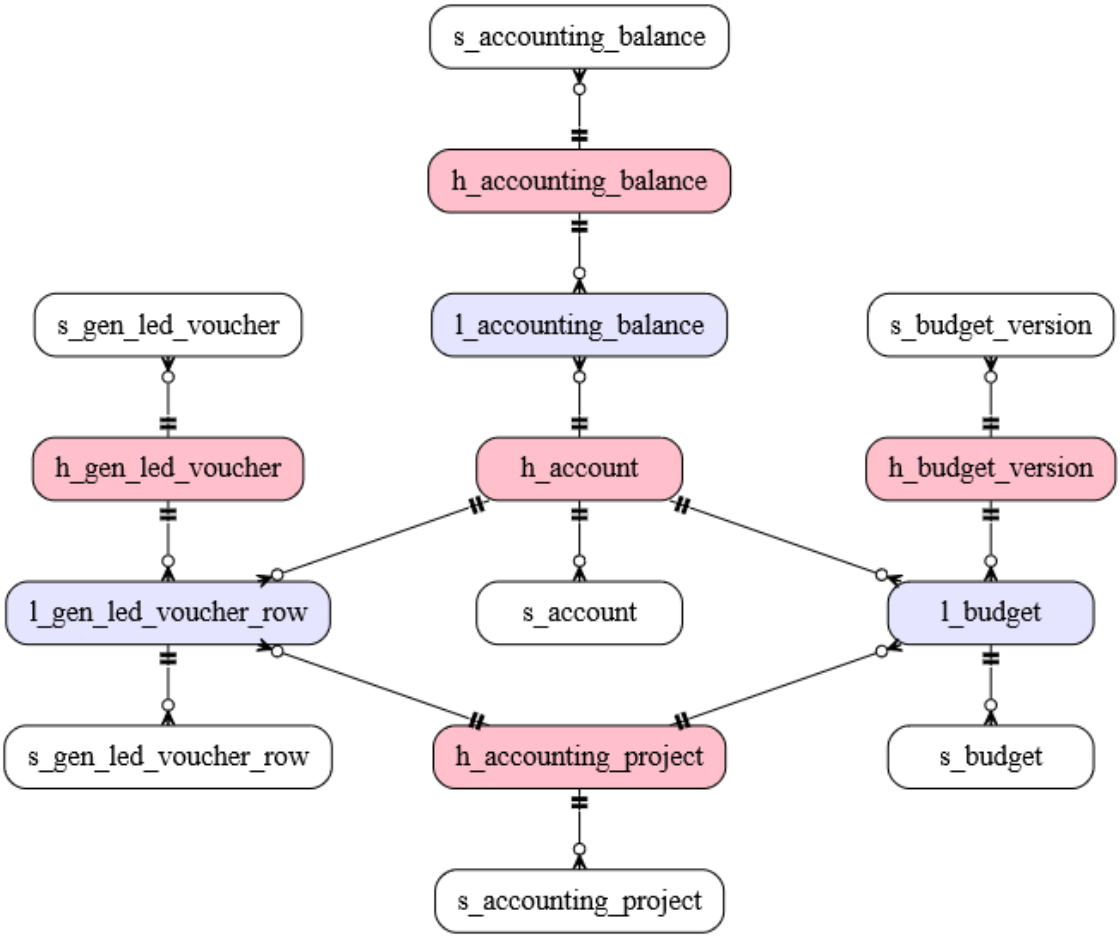


Kuva 5.5: Aveso DW Automation -työkalun käyttöliittymässä account-taulun avainkenttien määritys

5.2.3 Julkaisu

Sen jälkeen kun kaikki halutut entiteetit on määritelty ja malli on valmis julkaistavaksi ympäristöön, niin siitä voidaan tehdä luoda versiohallintaan versio ja tämän jälkeen voidaan tälle versiolle generoida kaikki tietovaraston määrittävät tietokantaobjektit. Kuvassa 5.6 on Data Vault raakatietovaraston malli, jota esimerkissä nyt metadatatietokannan määritykset kuvaavat. Julkaisun jälkeen kaikki nämä hubi-,

linkki- ja satelliittitaulut, sekä proseduurit, jotka lataavat datan extract-tauluista näihin raakatietovaraston tauluihin, ovat nyt tietokannassa. Jokaiselle entiteetille siis SQL-templaattien avulla luodaan SQL-skriptit ja ajetaan nämä tietokantaan.



Kuva 5.6: Esimerkki valmiin tietovaraston Data Vault raakatietovaraston mallista IFS:n ERP-ohjelmiston talousdatoille

6 Haastattelututkimus

6.1 Suunnitelma

Haastattelututkimus päätyi käytännöllisistä syistä sopivammaksi tutkimusmenetelmäksi syventämään käsityksiä tutkimuskysymyksiin. Haastattelun tavoitteena tulee siis olemaan mielipiteiden ja vaikutelmien kerääminen eri tavoista automatisoida tietovaraston kehitystä, Aveso DW Automation -työkalun jatkokehittämistavoista ja toimivuudesta tietovaraston kehityksen automatisoinnissa, sekä Data Vault -mallinnuksesta ja sen roolista tietovaraston kehityksen automatisoinnissa.

Haastattelu pidetään puolistrukturoituna, jossa kysymykset ovat tarkasti laadittuja näiden tavoitteiden pohjalta, mutta kuitenkin sen verran avoimia, jotta haastateltavilta saadaan omat mielipiteet selvästi esiin eikä vain kyllä ja ei vastauksia. Haastattelu äänitetään myöhempää analyysiä varten, sillä haastattelun aikana ei ole tehokasta tehdä muistiinpanoja ja samalla keskittyä haastatteluun. Haastattelun lopullista äänitystä ei kuitenkaan kirjoiteta puhtaaksi, sillä haastattelututkimus on pieni, vain yrityksen sisällä suoritettava. Pienissä tutkimuksissa täysin äänityksen avaaminen tekstiksi ei useasti ole sen vaatiman työn kannalta perusteltua [25]. Pääpointit haastateltavien vastauksista kysymyksiin kuitenkin kirjoitetaan ylös ja vastaukset käydään läpi kategorioittain.

Haastateltavat valitaan sen mukaan kuinka paljon kokemusta heillä on tietovarastoista ja Aveso DW Automation -työkalusta, ottaen haastattelun tavoitteet

huomioon. Taustatietoja on siis hieman käytävä läpi jo haastateltavien valinnassa, mutta myös itse haastattelussa haastateltava saa itse kuvailla omia taustatietojaan. Haastatelvat ovat kaikki Aveson työntekijöitä.

6.1.1 Ryhmähaastattelu

Ryhmähaastattelu sopii hyvin tilanteissa, joissa kerätään kvalitatiivista tietoa [25]. Ryhmähaastattelussa ryhmän jäsenten välinen integraatio ohjaa vahvasti keskustelua, joka monipuolistaa keskustelua ja voi tuoda ryhmän jäseniltä uusia mielipiteitä esille, joita ei yksilöhaastattelussa muuten tulisi. Ryhmähaastattelun tulosten analysointi täytyy kuitenkin tehdä tarkasti, sillä ryhmädynamiikka voi myös helposti vääristää tuloksia. Ryhmiä olisi myös hyvä olla useampi kuin yksi, mutta tässä tutkimuksessa vain yksi ryhmä on mahdollinen haastateltavien määrästä johtuen. Ryhmähaastattelussa kysytään osittain samat kysymykset kuin yksilöhaastattelussa, mutta myös pelkästään ryhmähaastattelussa käytettäviä kysymyksiä laaditaan.

6.1.2 Kysymykset

Kysymykset pääasiassa laaditaan etukäteen ja niille laaditaan myös mahdollisia jatkokysymyksiä, jotta haastattelu etenisi nopeammin tilanteissa, joissa kysymyksiin on vaikea muodostaa vastausta suoraan. Mahdollisesti myös haastattelun aikana voidaan kysymyksiä lisätä vastauksien mukaisesti, mutta pääasiassa samat kysymykset kysytään kaikilta haastatteluun osallistuvilta.

Taustatiedot

Taustatietoa kysytään sillä periaattella, että saadaan kontekstia haastateltavan tuleville vastauksille tutkimuksen pääkysymyksiin.

- Kuinka paljon näet itselläsi olevan kokemusta tietovarastojen kehityksestä?

- Kuinka hyvin mielestäsi ymmärrät Data Vault -menetelmän?
- Minkälaista kehitystä olet Aveso DW Automation -työkalulla tehnyt?

Tietovaraston kehitys

- Miksi tietovarastointi on hyödyllistä?
- Missä vaiheissa tietovaraston kehitystä koet olevan eniten manuaalista työtä?
  - Miksi juuri näissä vaiheissa?
- Mitä näistä vaiheista näet mahdollisesti automatisoitaviksi ja miksi?
- Mitkä ovat Data Vault mallinnuksen ja menetelmän hyödyt ja miksi?
  - Mitä rajoituksia Data Vault menetelmällä on?
- Miten näet Data Vault mallin roolin automatisaation kannalta?
  - Mitä osia tietovaraston kehityksestä Data Vault mallinnuksen automatisoivan työkalun pitäisi automatisoida?

Aveso DW Automation

- Mitkä osa-alueet näet Aveso DW Automation -työkalun tärkeimmiksi kehityskohteiksi?
- Onko työkalussassa selviä rajoituksia, jotka estävät esimerkiksi tiettyjen mallinnustapojen käytön tai poikkeumia Data Vault menetelmän perusperiaatteista. Mitä rajoitteita?
- Näetkö jotain muita heikkouksia/vahvuuksia Aveso DW Automation -työkalussa?

Ryhmähaastattelun kysymykset

- Miksi tietovarastointi on hyödyllistä?
- Missä vaiheissa tietovaraston kehitystä on eniten manuaalista työtä ja miksi?
  - Mitä näistä vaiheista näette mahdollisesti automatisoitaviksi ja miksi?
- Mitkä ovat Data Vault mallinnuksen ja menetelmän hyödyt?
  - Mitä rajoituksia Data Vault menetelmällä on?
- Miten näette Data Vault mallin roolin automatisaation kannalta?
  - Mitä osia tietovaraston kehityksestä Data Vault mallinnuksen automatisoivan työkalun pitäisi automatisoida?
- Mitkä osa-alueet näet Aveso DW Automation -työkalun tärkeimmiksi kehityskohteiksi?
  - Onko Aveso DW Automation -työkalussa selviä rajoituksia, jotka estävät esimerkiksi tiettyjen mallinnustapojen käytön tai poikkeumia Data Vault menetelmän perusperiaatteista.
- Muita ajatuksia tietovarastoinnista, Data Vault menetelmästä, automaatiosta tai jostain muusta näihin liittyvästä?

6.2 Haastattelun tulokset

Haastateltavia henkilöitä oli kolme. Yksilöhaastattelut kestivät kaikki noin puoli tuntia. Yksi haastattelu oli hieman lyhyempi, vain noin 20 minuuttia, mutta kaikki samat kysymykset ehdittiin käydä läpi. Ryhmähaastattelu pidettiin erikseen viikon päästä yksilöhaastatteluista ja siihen osallitui samat kolme henkilöä. Ryhmähaastattelu kesti noin tunnin.

6.2.1 Tietovarastointi

Tietovarastoinnin hyödyt

Tietovarastoinnista kaikki haastateltavat olivat yksimielisiä siinä, että sen päähyötynä on se, että data saadaan tuotua raportointiin ja analysoitavaksi. Ryhmähaastattelukin tämän varmisti. Henkilöiltä, jotka kauemmin olivat keskittyneet tietovarastointiin, tuli esille yksittäisinä hyötyinä esimerkiksi, että tietovarasto kokoaa dataa eri lähteistä, sen avulla voidaan dataa historioida ja ylipäätään saadaan keskitetty paikka datalle, jossa data on yhdenmukaisessa muodossa. Tietovaraston data on siivottua ja oikeellista, jotta datan jakaminen eteenpäin olisi mahdollisimman helppoa. Yksi henkilö mainitsi myös tietovaraston jo itsessään olevan BI-työkalu (engl. Business Intelligence). Esille tulleet tietovarastoinnin hyödyt vastasivat pääosin toisiaan haastateltavien välillä, vaikka pieni näkemysero oli mahdollisesti raportointiin ja tietovarastointiin keskittyvien henkilöiden välillä.

Tietovaraston kehityksen manuaalinen työ ja automatisoitavat vaiheet

Manuaalista työtä tietovarastojen kehityksessä ilmeni pääosin datan latausprosessin kehityksessä, yhteyksien pystytyksessä, tietovaraston tietomallin saamisessa oikean näköiseksi ja ylipäätään latausprosessin alkupään vaiheissa. Manuaalisen työn määrän sanottiin riippuvan esimerkiksi tietovaraston mallinnustavasta. Data Vault -menetelmällä mallinnettu tietovarasto vaatisi vielä enemmän manuaalista työtä, sillä tarvittavia tietokantaobjekteja on paljon. Datan puhdistuksessakin on työtä, mutta se ei ole niin suoraviivaista liiketoimintasääntöjen takia.

Latausprosessin alkupään vaiheiden mainittiin olevan pääosin automatisoitavissa, sillä näissä vaiheissa ei vielä tarvitse ottaa liiketoimintasääntöjä huomioon. Raakatietovarastooon data ladataan käytännössä siinä muodossa kun se on lähdejärjestelmissä. Automatisoitaviksi vaiheiksi mainittiin pääosin esilatausalueen taulujen luonti lähdejärjestelmien mukaisiksi, latausskriptien luonti, jotka lataavat da-

tan lähdejärjestelmistä esilatausalueelle, sekä esilatausalueelta raakatietovarastoon. Raakatietovaraston tietokantaobjektien luonti on myös mahdollista. Vaaditaan vain työkalu, jossa haluttu rakenne määritetään. Ilman automaatiota suuren määrän tietokantaobjektien tekemisen mainittiin myös selvästi olevan virheherkkää.

Ryhmähaastattelussa tuli esille mahdollisia tapoja myös esityskerroksen vaiheiden automatisointiin, mutta mainittiin, että näiden vaiheiden automatisoinnin hyöty on pienempi, sillä aivan kaikkea ei näistä vaiheista voi automatisoida. Jonkin yksinkertaistetun käyttöliittymän kautta voisi liiketoimintasääntöjäkin määritellä, mutta käytännössä näitä sääntöjä tekevät henkilöt osaavat jo valmiiksi käyttää teknisiä työkaluja paremmin. Tällöin esimerkiksi SQL on riittävä kieli tietokantaobjektien määrittelyyn, eikä teknisten henkilöiden tarvitsisi opetella uutta, vaikkakin ehkä yksinkertaisempaa, mutta ei niin monipuolista työkalua. Kysymykseksi jää, saisiko kaikkia monimutkaisia liiketoimintasääntöjä mitenkään määritettyä helposti graafisesti, jotta kuka tahansa voisi sääntöjä helposti muodostaa. Uuden yksinkertaistetun SQL:ää vastaavan kielen tekeminen ei myöskään olisi käytännössä järkevää. Yksinkertaisia liiketoimintasääntöjä, kuten kokonaishinnan laskemista kertomalla hinta kappalemäärällä, voisi kuitenkin graafisestikin määritellä.

6.2.2 Data Vault

Data Vault -menetelmän hyödyt ja haitat

Pääpointtina Data Vault -menetelmästä tuli esille se, että se on hyvin dokumentoitu, standardoitu ja laajasti käytetty tietovarastoinnin menetelmä, joka ratkaisee valmiiksi paljon tietovarastoinnin ongelmia, joten sitä noudattamalla pääsee helpommalla tietovaraston kehityksessä. Suurilla organisaatioilla voi olla myös omia tietovarastoja kehitettynä, joissa on mietitty ongelmia läpi yrityskohtaisesti ja noudatetaan yrityksen omia prosesseja, jolloin Data Vault -tietovarastosta ei suoraan ole hyötyä.

Data Vault -menetelmän muita hyötyjä sanotaan olevan esimerkiksi, että tietovaraston mallia on helppo laajentaa, sillä uusien lähteiden lisäämiseksi ei tarvita muutoksia olemassa olevaan malliin, sekä Data Vault toteuttaa datan talteenkeruun hyvin, joka onkin tietovarastoinnin tärkeimpiä osia. Raakatietovarastoon vain lisätään dataa. Dataa ei koskaan poistu, eikä se muutu, joten historia tallentuu tehokkaasti. Data Vault -menetelmän kerroksellisuus myös jakaa hallintaa eri loogisiin kokonaisuuksiin. Nämä yhdessä tekevät myös tietovaraston ylläpidosta helpompaa. Data Vault -menetelmä standardoituna ja hyvin dokumentoituna tapana mahdollistaa sen, että ulkopuolisetkin henkilöt ymmärtävät tietovaraston toiminnan. Riittää että ymmärtää Data Vault -menetelmän perusperiaatteet. Kehitysprojekteihin on myös helpompi tuoda uusia henkilöitä, sillä ei tarvitse uusia mallinnusmenetelmiä opetella.

Osittain Data Vault -menetelmä kuitenkin siirtää ongelmia myöhempiin vaiheisiin, sillä Data Vaultin sanotaan keskittyvän datan ja historian keräämiseen eikä datan käytettävyyteen. Hyödyntämistä varten datoja täytyy yhdistellä. Se ei siis suoraan ratkaise datan yhdistämisen ongelmia, mutta luo standardoidun kerroksen, jonka avulla näitä ongelmia voi helpommin ratkaista. Data Vault -menetelmän sanotaan myös olevan raskas tietovaraston objektien suuren määrän takia, joka tekee siitä jossain määrin käyttökelvottoman ilman automaatiota.

Data Vault -menetelmän kannalta automatisoitavat vaiheet ja ominaisuudet tietovaraston kehityksessä

Hieman näkemyseroja esiintyi siinä, onko Data Vault -menetelmää suunniteltu sillä periaatteella, että sen pystyisi helposti automatisoimaan. Ryhmähaastattelussa päädyttiin siihen, että todennäköisesti on Data Vault -menetelmän kehityksessä keskitytty ennemmin tietovarastoinnin ongelmien ratkaisuun, mutta on voitu myöhemmin ajatella myös automatisointia ja tehdä mallinnuksen rakenteet helpommin automa-

tisoitaviksi. Mainitaan kuitenkin, että nykyään Data Vault -menetelmän lupauksena on, että malli on toistuva juuri automatisointia varten.

Automatisoitavina osina Data Vault -menetelmästä tuli esille ainakin jonkin tason hallinta latauksille lähdejärjestelmistä, raakatietovaraston objektien generointi ja hieman myös raportointia auttavien osien generointia, jotka lukevat dataa raakatietovarastosta raportointia helpottavassa muodossa. Tähtimallilla mallinnettuihin data martteihin voisi olla tuki ja raportoinnissa tarvittavaa tietomallia voisi myös automaatiotyökalussa määrittää. Muitakin ominaisuuksia, kuten datan jäljitettävyyden hallintaa voisi tukea, jonka avulla saisi tarkkaan sarakekohtaisesti selville, mistä data on tulossa ja minne se on menossa.

Isoissa vastaavissa työkaluissa mainitaan olevan myös koko latausprosessin ajojen kontrollia, ajastusten hallintaa, versiohallintaa ja datan jäljitettävyyden ominaisuuksia, mutta kysymykseksi hieman jää kuinka oleellisia nämä ominaisuudet ovat pienemmässä Data Vault -mallinnukseen keskittyvässä työkalussa. Tämän mainitaan olevan periaatteellinen ero, jolla ei ole selvää vastausta. Asiakkaan kannalta tällaisilla suuremmilla työkaluilla mainitaan mahdollisena ongelmana esim se, että ne voivat olla kalliita (tuhansia euroja kuussa) ja koska latausprosessi hoidetaan täysin tällä työkalulla, niin tällöin ollaan myös lukittuja tähän työkaluun. Kun työkalun käyttö lopetetaan, niin tietovarastoa ei käytännössä voi enää käyttää. Jos työkalu pidetään vain kehityksen apuna erillään tietovarastosta, niin asiakkaan ei tarvitse maksaa lisenssimaksuja eikä työkalun kehittäjien ylläpitää tietokantaa suoraan. Tällöin tietovarastoa voi myös käyttää täysin ilman automaatiotyökalua.

6.2.3 Aveso DW Automation

Vertailua muihin automaatiotyökaluihin

Aveso DW Automation -työkalun mainitaan olevan enemmän kehitetty tietovarastojen kehitysaputyökaluksi, jolla automatisoidaan tietokannan määrittävien SQL-

skriptien generointi, eikä niinkään työkaluksi, jonka avulla hoidetaan tietovaraston suunnittelu ja joka täysin ylläpitää tietovarastoa. Työkalua ei ole yhdistetty tietovaraston latausprosessiin eli ajokontrolliin, joka ajastetusti käynnistää lataukset lähdejärjestelmistä. Monet suurista tällaisista automaatiotyökaluista mainitaan olevan suunniteltu vahvasti juuri tähän, että työkalu itse hoitaa koko tietovaraston. Kysymykseksi jää kuinka kevyeksi tällaisen työkalun haluaa, pitäisikö tietovaraston selvitä ilman automaatiotyökalua. Aveso DW Automation -työkalun voi kokonaan jättää pois eikä se vaikuta tietovaraston suoritukseen.

Monien kaupallisten tuotteiden ongelmana mainitaan myös se, että ne vaativat suuren yrityksen ja yritykselle suuren tietovarastoympäristön, jossa tälläista suurta työkalua kannattaisi alkaa käyttämään, sillä kuukausikustannukset voivat olla merkittävän suuria. Yrityksen pitää haluta maksaa, joten ei kovin pieneen ympäristöön voi suurta työkalua ottaa käyttöön. Aveso DW Automation -työkalun ideana on ollut mahdollistaa työkalun käyttöönotto myös pienemmissä tietovarastoissa/ympäristöissä/yrityksissä, jotka eivät olisi valmiita maksamaan tuhansia euroja pelkästä työkalusta jolla tietovarastoa kehitetään/ylläpidetään. Aveso DW Automation -työkalu on siis suunnattu aivan eri asiakassegmenttiin kuin suuremmat automaatiotyökalut.

Kehityskohteita

Esille tulleita kehityskohteita ovat muun muassa suoran yhteyden saaminen lähdejärjestelmiin, mallinnuksessa sekä esityskerroksen kehityksessä auttavien toimintojen kehittäminen, ja myös toiminnot, jotka auttavat julkaisuiden tekemiseen eri järjestelmiin. Jotkin kehityskohteet ovat olleet jo jonkin aikaa tiedossa ja selvityksessä, mutta myös uusia näkökulmia kehitykseen ilmeni.

Lähdejärjestelmiin yhdistämisen avulla voitaisiin esimerkiksi automaattisesti generoida esilatausalueen taulut. Voisi myös luoda moduuleita, joille voisi lähdejärjestelmistä saadun metadatan avulla luoda esimerkiksi valmiit parametrit lataus-

prosessia hallitsevalle teknologialle. Esimerkiksi Microsoftin Data Factorylle, jota tietovaraston ajokontrollina voi käyttää. Sille voisi tarvittavat parametrit luoda automaattisesti. Aveso DW Automation -työkalu voisi siis auttaa eri teknologiolla tehtävien latausprosessien määrityksessä, vaikka se ei itse latausprosessia hallitsisikaan. Datan inkrementaalinen päivityskin voitaisiin hallita tällaisen yhteyden kanssa paremmin. Sarakkeiden lisäykset voisi myös hallita koko latausprosessi osalta, joten jos lähdejärjestelmään tulee sarakkeeseen muutos, niin koko latausprosessi olisi helppo päivittää automaattisesti. Samoin tärkeänä datan jäljitettävyyden toiminnot, jotka voitaisiin toteuttaa koko latausprosessin osalta.

Aveso DW Automation -työkalusta sanotaan myös, että se voisi olla enemmän mallinnustyökalu, jolle vain määritetään hubit, satelliitit ja linkit ja näille mistä lähteistä datat tulevat. Tällä hetkellä työkalun näkökulma on se, että aloitetaan lähteestä ja lähteelle kerrotaan mihin muotoon datat siitä ajautuvat. Ajatus on nyt myös niin, että varsinainen mallinnus tehdään työkalun ulkopuolella ja malli vain tuodaan työkaluun eikä niin, että itse työkalussa kehitettäisiin. Yksi maininta kehityssuunnasta on se, että työkalulla voisi suoraan hahmotella mallia ja näkisi esikatseluna lähdejärjestelmän taulujen dataa. Visuaalinen mallinnustuki olisi myös kätevä, mutta mainitaan, että se voisi vaatia paljon työtä. Yhteen hubiin satelliitin liittäminen kahdesta eri lähdetaulusta ei vielä tällä hetkellä toimi Aveso DW Automation -työkalulla oikein. Tämäkin voisi korjaantua, jos tämä näkökulma tehtäisiin mallin kannalta eikä lähteiden kannalta.

Työkalusta mainitaan myös, että se voisi myös paremmin tukea esityskerroksen liiketoimintasääntöjä sisältävien näkymien generointia. Toistuvat osat, esimerkiksi linkkien hash viittaukset, voisi automaattisesti tuoda esityskerroksen näkymiin. Tällä hetkellä Data Vault määrittelyt ja esityskerroksen näkymien skriptit ovat eri paikoissa määritettynä, sillä esityskerroksen skriptit ovat erikseen tallennetuina tiedostoina. Tämän takia mainitaan myös, että työkalussa voisi olla jokin kevyt ratkaisu,

jonka avulla voisi käyttöliittymän kautta luoda esityskerroksen objekteja tietokantaan. Tietokantaobjektien kokonaisuus tulisi tällöin julkaisujen näkökulmasta yhden paikan alle (työkalun sisälle). Versiohallintakin olisi tällöin näiden osalta yhtenäinen.

Raportoinnin tarvitseman tietomallin määritys voisi olla työkalun sisällä monipuolisemminkin. Suora tuki voisi olla erilaisille esityskerroksen tarvitsemille taulutyypeille, esimerkiksi PIT- ja Bridge-tauluille, sekä erilaisille linkkitauluille joita Data Vault määrittelee. Myös raakatietovarastossakin käytettäviltä Reference-tauluilta puuttuu myös suora tuki. Yksittäinen maininta on myös sille, että työkalun sisällä voisi olla myös graafisesti mahdollista suunnitella ja mallintaa tämä esityskerroksen tietomalli (data martit).

Julkaisuprosessiin mainitaan myös parannuksena erilaiset muutoksien hallinnan toiminnot, joiden avulla selvemmin näkisi mitä tietovarastossa on muuttunut eri määrittelyjen välillä. Kun julkaisua tehdään, niin voitaisiin olla varmoja mitä muutoksia tietovarastoon menee verrattuna edellisiin määritysten versioihin. Julkaisut voisivat näkyä visuaalisemmin, jotta helpommin näkisi mitä missäkin järjestelmässä on asennettuna.

Aveso DW Automation -työkalun sanotaan olevan olleen enemmän työkalu vain kehityksen apuna ja vähitellen se on muuttunut enemmän ylläpitotyökalun suuntaan uusien ominaisuuksien kautta. Koko ajan on pidetty kuitenkin mielessä minkälaiset muutokset mahdollistavat sen että työkalulla tehty tietovarasto ei tule riippuvaiseksi työkalusta. Tämä on rajoittanut sitä, minkälaisia ominaisuuksia työkaluun on voitu tuoda.

7 Aveso DW Automation -työkalun analyysi

Tässä luvussa käydään läpi mitä Aveso DW Automation -työkalun kannalta merkittäviä asioita haastatteluissa ja akateemisessa tutkimuksessa on tullut esille, sekä analysoidaan työkalun toiminnan rajoituksia ja kehityskohteita. Löytyneet kehityskohteet luokitellaan niiden merkitsevyyden ja tärkeyden mukaan ja näiden pohjalta esitetään korkean ja matalan prioriteetin kehitysehdotuksia.

7.1 Vastaavuus esitettyihin lähestymistapoihin

Aveso DW Automation -työkalussa on monelta osin kehitetty samoja toiminnallisuuksia kuin Puonti et al., Krneta et al. ja Pankov et al. esittämissä automatisoinnin lähestymistavoissa. Metadatatietokannoissa on myös paljon vastaavuuksia.

Puonti et al. lähestymistavan yksinkertainen metadatamalli on monelta osin verrattavissa Aveso DW Automation -työkalun metadatamalliin. Molemmilla on entiteettitaulu ja niillä on samantyyppinen merkitys molemmissa malleissa, sillä molemmissa ne mallintavat tietovaraston taulua. Puonti et al. mallin entiteettitaulu kuitenkin mallintaa myös esilatausalueen taulua ja mitä tahansa Data Vault -mallin hubi-, linkki- tai satelliittitaulua. Aveso DW Automation entiteettitaulu kuvaa suoraan vain hubi- tai linkkitaulua. Satelliitit kuvataan täysin erikseen, mutta ne liitetään suoraan tiettyyn hubiin tai linkkiin, jolloin ei tarvita erillistä kuvaustaulua, joka yh-

distäisi hubin satelliittiin. Puonti et al. lähestymistavassa on myös attribuuttitasolla oma kuvaustaulu, joka kertoo mitkä kaksi attribuuttia kuuluvat yhteen. Aveso DW Automationissa vastaava kuvaustaulu on vain avainsarakkeille, sillä muut sarakkeet ovat Aveso DW Automationissa määritettynä vain kerran ja SQL-templaateilla saadaan attribuuttien yhteydet toisiinsa hoidettua pitämällä sarakkeet aina samassa järjestyksessä.

Krneta et al. lähestymistavalla on myös paljon vastaavuuksia Aveso DW Automationin lähestymistapaan. Sillä ei kuitenkaan ole yhtä tietovaraston taulua kuvaavaa taulua, vaan se on jaettu kolmeksi tauluksi lähteen tyypin mukaan, mutta näiden idea on Krneta et al. lähestymistavassa vastaava. Taulujen suhteet kuitenkin määritetään vasta saraketaulun avulla, johon sarakekohtaisesti määritetään tuleeko sarakkeesta hubi, linkki tai satelliitti. Esilatausalueen tauluja ei kuvata erikseen, samoin kuin Aveso DW Automation -työkalussa. Ehdotettu työkaluohjelma metadatamallin hallitsemiseen on myös monelta osin vastaava Aveso DW Automationin kanssa.

Pankov et al. lähestymistapa on tehty yleisemmällä tasolla, joten metatietomallin määrityksessä on enemmän mahdollisia määritystapoja. Pankov et al. lähestymistavan metatietomalli vaatii tästä syystä enemmän työtä metadatan hallintatyökalun kanssa, joka määrittää miten metatietomallia käytetään. Data Vault -mallin objektien generointiin ja tälle ETL-prosessin määrittämiseen ei Pankov et al. ota suoraan kantaa, mutta mallin yleisyyden vuoksi se voisi sitä kuitenkin tukea. Aveso DW Automationin rakenne on enemmän lukittu juuri Data Vault -mallinnukseen.

7.2 Rajoituksia ja kehityskohteita

Ehkä merkkittävin rajoitus on ollut se, että Aveso DW Automation ei vielä tue satelliittia, jolle tulee sarakkeita useammasta eri lähdetaulusta. Data Vaultin perusperiaatteisiin kuuluu, että esimerkiksi hubille voidaan ottaa useampi satelliitti,

jotka esimerkiksi tulevat eri lähteistä [16]. Aveso DW Automationissa useammasta satelliitista voi tällä hetkellä hyötyä vain kun halutaan satelliittiin jakaa sarakkeet datan muutosnopeuden perusteella. Myöskään esimerkiksi reference-, PIT- ja bridge tauluja Aveso DW Automation ei vielä tue, vaikka ne Data Vault -menetelmään kuuluvatkin [14].

Ylläpitotoimintojen kannalta muutoksenhallinta on erittäin tärkeä toiminto, jota Aveso DW Automation tukee vasta osittain. Kun metadatamalliin tehdään muutoksia, niin on tärkeää myöhemmin tarkasti nähdä mitä nämä muutokset ovat. Tämä toiminto on jo työkalussa toteutettuna, mutta työkalusta puuttuu vielä vertailu tietovaraston objektien tasolla, josta näkisi tarkkaan mitä objekteja tietovarastossa on ja miten uusien metadatamääritysten julkaisu muuttaisi olemassaolevia tietovaraston objekteja.

Rajoituksena on ollut myös se, että Aveso DW Automation -työkalu ei saa suoraa yhteyttä lähdejärjestelmiin. Suoralla yhteydellä lähdejärjestelmiin voitaisiin esimerkiksi luoda Krneta et al. lähestymistavan esittämiä toiminnallisuuksia, joiden avulla voisi helpottaa raakatietovaraston Data Vault -mallin luontia. Joissain tapauksissa Data Vault -mallinnuksen voisi täysin automatisoida tai ainakin luoda mallille pohjan, jota vain hieman tarvitsisi manuaalisesti muokata. Lähdejärjestelmän yhteyden avulla saataisiin esimerkiksi esilatausalueen taulut generoitua automaattisesti. Työkalua on kuitenkin kehitetty kevyenä kehitystä auttavana työkaluna, joten liian suuresti tällaista toimintoa ei työkaluun olla haluttu. Toiminnon voisi kehittää kevyenä niin, että luotu tietovarasto ei tule riippuvaiseksi kehitystyökalusta.

Aveso DW Automationia on kehitetty enemmän kehitystyökaluna kuin mallinnustyökaluna, mutta haastatteluistakin paljastui, että hyvänä kehityssuuntana voisi olla kehittää työkalua enemmän mallinnustyökaluksi. Esimerkiksi toiminnoilla joissa mallia rakennetaan hubien, linkkien ja satelliittien kautta eikä lähdetaulujen kautta. Esityskerroksen Data Vault informaatiomarttien määritys voisi myös olla tar-

kemmin työkalun sisällä, jonka avulla voisi osittain liiketoimintasääntöjä määrittää työkalussa, joka vähentäisi esityskerroksessa manuaalisen työn määrää.

Krneta et al. lähestymistavan tavasta hallita useita eri tyyppisiä lähteitä voisi myös ottaa opiksi. Jotta entiteettejä voitaisiin määrittää Aveso DW Automationiin Data Vault -mallin kannalta, niin entiteeteille tarvitaan tuki, jossa entiteettiä ei ole suoraan yhdistetty esilatausalueen tauluun. Tällä hetkellä entiteetti liittyy aina yhteen esilatausalueen tauluun. Kun entiteetille saataisiin tuki useammalle lähdetaulule, niin tämän avulla voisi myös entiteetille toteuttaa tuen eri lähteistä tuleville satelliiteille.

Aveso DW Automation voisi myös hyötyä Pankov et al. esittämästä geneerisestä ajattelusta, joka ei ota kantaa käytettyyn mallinnusmenetelmään. Aveso DW Automation esimerkiksi käyttää metatietokannassaan paljon Data Vault -menetelmään viittaavia tietoja, joiden käyttöön ei voi Data Vault rakenteen määrittävien SQL-templaattien kanssa kuitenkaan vaikuttaa. Esimerkiksi entiteetillä on käytössä juuri ne attribuutit, jotka tietokantaan on määritetty eikä käyttöliittymän kautta voi uusia arvoja itse määrittää. Yleistämällä voisi erillisiä käyttötapaukseen liittyviä attribuutteja luoda entiteeteille ja yksinkertaistaa SQL-templaattien hallintaa.

Puonti et al. lähestymistavan kevyt metadatamalli ja Pankov et al. lähestymistapa näytti, että Aveso DW Automationin metadatamallin yksinkertaistus voisi myös olla järkevää. Tällainen yksinkertaistus voisi käytännössä olla esimerkiksi saraketaulujen yhdistäminen, sillä tällä hetkellä Aveso DW Automation käyttää erillistä taulua laskettujen sarakkeiden määritykseen. Tämä hieman lisää kompleksisuutta ja vähentää joustavuutta, mutta ei varsinaisesti tuo hyötyä. Samaa voisi harkita kuvaustaulujen kannalta. Aveso DW Automation käyttää kahta eri sarakeliitostaulua. Toisella yhdistetään sarakkeet satelliitteihin ja toisella avainsarakkeet linkkeihin. Puonti et al. lähestymistavan mukaisesti voisi sarakeliitokset hoitaa myös yhdellä yleisellä kuvaustaululla.

7.2.1 Korkeamman prioriteetin kehityskohteita

Korkeamman prioriteetin kehityskohteita ovat:

- Hubille ja linkille tuki satelliiteille, joiden datat tulevat eri lähdetauluista
- Tuki reference-, PIT- ja bridge-tauluille (ja esityskerroksen muille taulutyypeille)
- Metadatamallin muutoksien hallinta tietovaraston objektien tasolla
- Esityskerroksen näkymien osittainen automaattinen generointi
- Suoran yhteyden kehittäminen lähdejärjestelmiin, jonka avulla voisi esilatausalueen taulut ja osittain Data Vault -mallin luoda automaattisesti

Näistä kaksi ensimmäistä ovat erityisen tärkeitä, sillä ne ovat selvimmät rajoitukset Data Vault -menetelmän kannalta. Näille toiminnoille on varmasti tarvetta tulevaisuudessa.

7.2.2 Matalemman prioriteetin kehityskohteita

Matalemman prioriteetin kehityskohteita ovat esimerkiksi:

- Entiteettien määritys suoraan Data Vault -mallin kautta eikä lähtien lähdetauluista
- Datan jäljitettävyyden toiminnot (engl. lineage)
- Esityskerroksen tietomallin määrittäminen työkalun sisällä
- Entiteetteihin liittyvien attribuuttien yleistäminen
- Metadatamallin yksinkertaistaminen esimerkiksi yhdistämällä saraketaulut

Huomioitavaa on kuitenkin, että kehityskohteet eivät täysin ole irrallisia toisistaan, joten niitä on osittain kehitettävä päällekäin. Esimerkiksi metadatamallin yksinkertaistaminen auttaisi myös muiden kehityskohteiden kehittämisessä. Joillain osin matalankin prioriteetin kehityskohteita on siis kehitettävä korkean prioriteetin kehityskohteita kehittäessä.

8 Yhteenveto

Tutkielmassa tutustuttiin Data Vault -menetelmällä mallinnetun tietovaraston kehityksen automatisaatioon käymällä läpi kirjallisuudesta löytyneitä tutkimuksia ja suorittamalla Aveson sisäinen haastattelututkimus. Tavoitteena löytää uusia näkökulmia tietovaraston kehityksen automatisaatioon, analysoida Aveso DW Automation -työkalun automaatioratkaisua ja löytää sille kehityskohteita. Seuraavaksi käydään vielä läpi vastaukset tutkielman tutkimuskysymyksiin.

Mitkä ovat Data Vault menetelmän hyödyt ja haitat tietovaraston kehityksessä? Data Vault yhdistää perinteisten tietovarastointimenetelmien hyviä puolia yhteen ja ratkaisee paljon tietovarastoinnissa esiintyviä ongelmia. Data Vault esimerkiksi hoitaa datan historioinnin tehokkaasti. Standardoituna ja hyvin dokumentoituna menetelmänä sen toiminta on helposti ymmärrettävissä ja selitettävissä ulkopuolisille. Automatisoinnin kannalta on erityisen tärkeää, että Data Vault on hyvin joustava ja skaalaantuva menetelmä. Data Vault menetelmä on kuitenkin raskas taulujen ja muiden tietovaraston objektien suuren määrän vuoksi, jonka takia se ei ole käytännöllinen pienille tietovarastoille ja aina vaatii jonkinlaisen automaatiotyökalun sen kehittämiseen ja ylläpitoon.

Millaista tutkimusta on tehty tietovaraston kehityksen automatisoinnista? Data Vault -menetelmää hyödyntäen on jonkin verran tehty erilaista tutkimusta tietovaraston kehityksen automatisoinnista. Tutkielmassa esiteltiin kolme tällaista lähestymistapaa, jotka kaikki kuitenkin keskittyivät hieman eri osiin tietovaraston ke-

hityksestä. Pankov et al. lähestymistapa keskittyi yleisen metatietovaraston luontiin, jota käyttäen voisi luoda automaatiotyökalun, joka luo tietovaraston haluttuun muotoon, Krneta et al. keskittyi Data Vault -mallin taulujen automaattiseen generointiin käyttäen omaa hieman yksinkertaisempaa metatietovarastoaan ja Puonti et al. esittivät lähestymistavan Data Vault -mallin taulujen välisten transformaatioden automaattiseen generointiin vielä kevyemmän metadatamallin pohjalta.

Kuinka käytännöllisiä malleja tutkimuksessa on esitetty? Näistä kolmesta lähestymistavasta Krneta et al. lähestymistapa on kaikkein käytännöllisin toteuttaa. Puonti et al. kevyt lähestymistapa ei välttämättä suoraan ole käytännöllinen toteutettavaksi, paitsi jos Data Vault -mallin taulut pystytään tehokkaasti tekemään ulkoisella työkalulla tai kehitetään tähän menetelmään tuki myös Data Vault -mallin määritykselle. Pankov et al. lähestymistapa on näistä suoraan kehitettynä epäkäytännöllisin, varsinkin kun keskitytään Data Vault -mallin tietokantaobjektien generointiin.

Kuinka lähellä Aveso DW Automation -työkalun automaatioratkaisu on tutkimuksessa esitettyjä Data Vault automaatioratkaisuja? Krneta et al. lähestymistapa oli näistä lähestymistavoista kaikkein lähimpänä Aveso DW Automationin lähestymistapaa, mutta kaikilla oli kuitenkin esimerkiksi metadatamallin kanssa jotain samankaltaisuuksia. Aveso DW Automationin automaatioratkaisu pääosin siis vastasi tutkimuksessa löydettyjä malleja, mutta osittain sen metadatamalli ei ole niin joustava kuin muut esitetyt metadatamallit. Aveso DW Automationin metadatamallin pieni yleistys ja yksinkertaistus voisi helpottaa esimerkiksi haastatteluissa esille tulleiden rajoitusten hallinnassa.

Onko Aveso DW Automationin automaatioratkaisulla selviä rajoitteita, jotka joillain tavoin heikentävät generoidun tietovaraston toimintaa? Aveso DW Automationin toimintaa rajoittaa se, että se ei vielä tue kaikkia Data Vault -mallin taulutyyppejä, eikä satelliittitauluja tueta vielä täysin. Nykytoteutuksen kanssa satel-

liittien pitää vielä olla samasta lähdetaulusta. Tämä on aika suuri rajoitus Data Vault -menetelmän kannalta. Ylläpitoa helpottaville ominaisuuksille on myös selvä tarve, vaikka niitä onkin osittain jo kehitetty. Esityskerroksen näkymien hallintaa voisi selvästi vielä parantaa ja suora yhteys lähdejärjestelmiin voisi selvästi vähentää manuaalista työtä, jota työkalun kanssa vielä joutuu tekemään.

Tutkielmassa on tullut esille, että Aveso DW Automationin osalta ainakin sen selvimpien rajoitteiden ja Data Vault -mallin automaattisen kehityksen kannalta voisi työkalun jatkokehityksen mahdollisuuksia käydä läpi. Näille ominaisuuksille voisi olla suurin tarve työkalun tulevaisuuden kannalta. Tutkielmassa ei kovin tarkkaan keskitytty tietovaraston esityskerroksen automaattiseen kehitykseen, eikä ylläpitotoimintoihin, joihin voisi vielä selvästi syvemmin perehtyä yleisellä tasolla sekä Aveso DW Automation -työkalun kannalta.

Lähdeluettelo

[1] A. Hovi. ”Tähtimalli vai Data Vault?” (2016), url: https://www.linkedin.com/pulse/t%C3%A4htimalli-vai-data-vault-ari-hovi. [Online; haettu 24.05.2022].

[2] K. Graziano ja D. Linstedt. ”Introduction to Data Vault Modeling”. (2011), url: https://kentgraziano.files.wordpress.com/2012/02/introduction-to-data-vault-modeling.pdf. [Online; haettu 24.05.2022].

[3] C. Phipps ja K. C. Davis, ”Automating data warehouse conceptual schema design and evaluation”, teoksessa *DMDW*, 2002.

[4] M. Puonti, T. Raitalaakso, T. Aho ja T. Mikkonen, ”Automating Transformations in Data Vault Data Warehouse Loads”, *Frontiers in Artificial Intelligence and Applications*, vol. 292, s. 215–230, tammikuu 2017. DOI: 10.3233/978-1-61499-720-7-215.

[5] D. Linstedt ja M. Olschimke, *Building a Scalable Data Warehouse with Data Vault 2.0*. Boston: Morgan Kaufmann, 2016, ISBN: 978-0-12-802510-9.

[6] T. Naeem. ”Data Warehouse Concepts: Kimball vs. Inmon Approach”. (helmikuu 2020), url: https://www.astera.com/type/blog/data-warehouse-concepts/. [Online; haettu 24.05.2022].

[7] P. B. Felix. ”Dealing with Deletes in the Data Warehouse”. (2019), url: https://www.leapfrogbi.com/dealing-with-deletes-in-the-data-warehouse/. [Online; haettu 24.05.2022].

[8] R. Elmasri ja S. Navathe, ”Overview of Data Warehousing and OLAP”, teoksessa *Fundamentals of Database Systems, 7th Edition*. 2016, luku 29.

[9] B. Inmon, *The Corporate Information Factory*. Wiley, tammikuu 2001, ISBN: 978-0471399612.

[10] R. Kimball ja M. Ross, *The Data Warehouse Toolkit, 3rd Edition*. Wiley, 2013.

[11] A. Hovi. ”Kimball vai Inmon?” (Lokakuu 2013), url: https://www.arihovi.com/kimball-vai-inmon/. [Online; haettu 24.05.2022].

[12] D. Lindstedt. ”Data Vault Basics”. (2015), url: https://danlinstedt.com/solutions-2/data-vault-basics/. [Online; haettu 24.05.2022].

[13] K. Graziano. ”The Business Data Vault”. (2015), url: https://www.vertabelo.com/blog/data-vault-series-the-business-data-vault/. [Online; haettu 24.05.2022].

[14] D. Lindstedt. (2018), url: https://danlinstedt.com/wp-content/uploads/2018/06/DVModelingSpecs2-0-1.pdf. [Online; haettu 24.05.2022].

[15] D. Naidoo. ”What is Data Vault? A Complete Guide to using it in your Data Warehouse”. (elokuu 2021), url: https://vp-analytics.com/2021/08/31/what-is-a-data-vault/. [Online; haettu 24.05.2022].

[16] D. Linstedt ja K. Graziano, *Super Charge Your Data Warehouse: Invaluable Data Modeling Rules to Implement Your Data Vault*. Createspace Independent Pub, 2011, ISBN: 9781463778682.

[17] T. Smith. ”Why Automate the Data Warehouse?” (2020), url: https://www.insightsfromanalytics.com/post/why-automate-the-data-warehouse. [Online; haettu 24.05.2022].

[18] I. Pankov, P. Saratchev, F. Filchev ja P. Fatacean, "Towards a Generic Metadata Warehouse", *Materials, Methods & Technologies, Volume 8, 2014 — International Scientific Publications*, 2014.

[19] D. Krneta, V. Jovanovic ja Z. Marjanovic, "A Direct Approach to Physical Data Vault Design", *Computer Science and Information Systems*, vol. 11, s. 569–599, kesäkuu 2014. DOI: 10.2298/CSIS130523034K.

[20] M. Puonti ja T. Raitalaakso, "Data Vault Mappings to Dimensional Model Using Schema Matching", teoksessa *Research and Practical Issues of Enterprise Information Systems*. joulukuu 2019, s. 55–64, ISBN: 978-3-030-37631-4. DOI: 10.1007/978-3-030-37632-1_5.

[21] D. Krneta, V. Jovanovic ja Z. Marjanovic, "An Approach to Data Mart Design from a Data Vault", *INFOTEH-Jahorina BiH*, vol. 15, 2016.

[22] M. West, *Developing High Quality Data Models*. joulukuu 2010, ISBN: 9780123751072.

[23] Aveso, *DW Automation User Guide and Functional Description v1.2.5*, 2021, Sisäinen dokumentti.

[24] T. Anderegg, "MRP/MRPII/ERP/ERM — Confusing Terms and Definitions for a Murkey Alphabet Soup", url: http://wiki.wlug.org.nz/EnterpriseSpeak, [Online; haettu 24.05.2022].

[25] F. Shull, J. Singer ja D. I. K. Sjøberg, toim., *Guide to Advanced Empirical Software Engineering*. Springer London, 2008, ISBN: 978-1-84800-043-8.