Osaamista
ja oivallusta
tulevaisuuden
tekemiseen

Arttu Sahramaa

Pentaho Data Integration (Kettle) -ohjelmiston hyödyntäminen ETL-prosessissa

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

15.5.2020

Tekijä Otsikko Sivumäärä Aika	Arttu Sahramaa Pentaho Data Integration (Kettle) -ohjelmiston hyödyntäminen ETL-prosessissa 37 sivua + 1 liite 15.5.2020
Tutkinto	insinööri (AMK)
Tutkinto-ohjelma	Tieto- ja viestintätekniikka
Ammatillinen pääaine	Ohjelmistotuotanto
Ohjaajat	Lehtori Simo Silander Konsultointijohtaja Jaana Immovaara

Insinöörityön tavoitteena on tutustua ETL-prosessien konsepteihin, tekniikoihin ja ongelmiin sekä tutkia avoimen lähdekoodin Pentaho Data Integration (Kettle) -ohjelmistoa ja miten tätä voi hyödyntää ETL-prosessien muodostamisessa. Tavoitteena on tuottaa LTC-Otso Oy:lle kuvaus Kettle-ohjelmiston toiminnasta sekä antaa valmiudet Kettlen hyödyntämisessä.

ETL-prosessin käsittelyosiossa tutkitaan prosessille tyypillisiä käytänteitä ja ongelmia lähdekirjallisuutta tutkien. Osiossa tutustutaan ETL:n eri osaprosesseihin sekä tyypillisesti käytettäviin teknologioihin ja protokolliin, jotta saadaan kuva ETL-työkalulle tarvittavista ominaisuuksista. Työssä tutustutaan myös Kettlen toiminnallisuuksiin muodostamalla erilaisia käytännön esimerkkejä Kettlen Spoon-nimistä käyttöliittymää käyttäen.

Työn tuloksena saatiin dokumentoitua riittävät tiedot Kettlen toiminnallisuuksista, jotta Kettleä voidaan jatkossa hyödyntää ETL-prosessien muodostamisessa. Erilaisten Kettle-toiminnallisuuksien testaaminen antoi konkreettisen esimerkin Kettlen käytöstä sekä avasi Kettle-prosessien toimintamallia. Tuloksena oli myös Kettlellä toteutettu ETL-prosessi, jonka dokumentaatiota voidaan jatkossa hyödyntää uusien prosessien rakentamisessa LTC-Otso Oy:ssa.

Avainsanat	Dataintegraatio, Kettle, ETL

Author	Arttu Sahramaa
Title	Utilization of Pentaho Kettle Software in ETL Processes
Number of Pages	37 pages + 1 appendix
Date	15 May 2020
Degree	Bachelor of Engineering
Degree Programme	Information and Communications Technology
Professional Major	Software Engineering
Instructors	Simo Silander, Senior lecturer Jaana Immovaara, Director of Consulting

The goal of the Bachelor's thesis is to explore ETL processes as a concept, and to study Pentaho Kettle -software to get an idea how it can be utilized in creation and execution of these processes. The goal is also to give LTC-Otso Oy a description of the capabilities of Pentaho Kettle -software and to provide capabilities to build ETL-processes using it.

To achieve the goal of the thesis, concepts and common problems related to ETL-processes were first studied to get a grasp what is required of the used ETL-tool Kettle. The thesis studies different subprocesses of the ETL process, and commonly used technologies and protocols related to them. The ETL-tool Kettle is then studied by forming different example processes using the tools graphical user interface named Spoon.

The result of the thesis is documentation of the Kettle software. The documentation is sufficient to build ETL-processes using the graphical interface Spoon, and to execute said processes. Different Kettle functionalities were tested to get a concrete idea of the capabilities it provides. Another result of the thesis was documentation and the building of a Kettle process for LTC Otso Oy. The documentation can be utilized as reference point for building new processes in the future.

Keywords	Data integration, Kettle, ETL

Sisällys

Lyhenteet

ETL	Extract/Transform/Load. Tietojen haku (Extract), muokkaus (Transform) ja lataus (Load) prosessi, jonka tavoitteena on viedä dataa jostain järjestelmästä toiseen.
BI	Business Intelligence. Prosessi- ja järjestelmäkokonaisuus, jolla organisaatio valjastaa eri järjestelmiensä datan analysoitavaan muotoon auttamaan yrityksen päätöksentekoa.
IDM	Identity Management. IDM-järjestelmät pyrkivät keskittämään teknisten identiteettitietojen ja käyttöoikeuksien hallinnan.
ERP	Enterprise Resource Planning. ERP-järjestelmät integroivat useita yrityksen liiketoimintaprosesseja yhden järjestelmän alle.

1 Johdanto

Tämän työn tavoitteena on tuottaa tämän työn tilaajalle, LTC-Otso Oy:lle valmiudet Pentaho Data Integration -ETL-työkalun hyödyntämisessä.

ETL on lyhenne sanoista haku (Extract), muokkaus (Transform) ja lataus (Load). ETL-prosessilla tarkoitetaan prosessia, joka hakee tietoa jostain lähdejärjestelmästä (Extract), muokkaa ja validoi tietoa kohdejärjestelmän tarpeisiin (Transform), ja lopulta tuo sen johonkin kohdejärjestelmään (Load). (1.)

ETL-prosessit ovat arkipäivää yritysmaailman järjestelmissä. Yrityksillä on usein käytössään järjestelmiä, joita ei ole suoraan integroitu keskenään, mutta joiden välille vaaditaan kuitenkin datan siirtoa. Yksi esimerkki tällaisesta järjestelmäkokonaisuudesta on BI-, eli Business Intelligence -järjestelmät.

BI-järjestelmät kokoavat dataa tietovarastoihin analysoitavaksi eri lähdejärjestelmistä, kuten asiakastieto-, myynti- tai toiminnanohjausjärjestelmistä. ETL-prosessi toimii BI-järjestelmissä avainasemassa. Prosessi hoitaa tiedon integraation eri lähdejärjestelmistä BI-järjestelmän tietovarastoon. Tieto voi myös esiintyä lähdejärjestelmässä epäkelpoisessa muodossa, jolloin sitä joudutaan muokkaamaan, jotta se saadaan tietovarastoon analysoitavaan muotoon. Tiedon puhdistaminen ja eheyden varmistaminen ovat ETL:n muokkausprosessin osa-alueita. (2.)

Toinen esimerkki järjestelmästä, joka käyttää ETL-prosesseja on IDM-, eli Identity Management -järjestelmät. IDM-järjestelmät pyrkivät keskittämään identiteettitietojen- ja käyttöoikeuksien hallinnan (3). IDM-järjestelmät ovat monesti riippuvaisia usean järjestelmän tiedoista. IDM-järjestelmä saa lähdetietonsa usein jostain muusta järjestelmästä, kuten organisaation HR-järjestelmästä. IDM-järjestelmän tulee pystyä päättelemään HR-järjestelmästä saatujen tietojen perusteella esimerkiksi työntekijöiden työsuhteen tilanteen. Jos työsuhde on päättynyt, myös käyttöoikeudet tulee poistaa. Käyttöoikeuksien poistosta tulee taas välittää tieto eteenpäin useampaan kohdejärjestelmään, joihin voi

lukeutua esimerkiksi moderneja pilvipohjaisia järjestelmiä tai Mainframe-pohjaisia legacy-järjestelmiä. On siis kriittistä että, IDM-järjestelmä pystyy integroitumaan monipuolisiin järjestelmiin mahdollisimman helposti. (4.)

Opinnäytetyön ensimmäisessä osassa tutustutaan lähdekirjallissuutta tutkien ETL-prosessille relevantteihin protokolliin sekä ETL-prosessien rakentamiseen liittyviin käytäntöihin ja haasteisiin.

ETL-prosessin helpottamiseksi on olemassa monia maksullisia- sekä avoimen lähdekoodin työkaluja (5). Opinnäytetyön toisessa osassa tutustutaan avoimen lähdekoodin Pentaho Data Integration (Kettle) -ETL-työkalun ominaisuuksiin. Osion tavoite on antaa tämän työn tilaajalle, LTC-OTSO Oy:lle valmiudet ETL-prosessien rakentamiseen Kettleä käyttäen. Lisäksi Kettleä käyttäen toteutetaan ETL-prosessi, jota kuvaillaan liitteessä 1.

2 ETL

ETL-lyhenteellä tarkoitetaan siis tiedon haku (Extract), muokkaus (Transform) ja lataus (Load) prosessia. ETL-prosessi rakennetaan tyypillisesti lähde- ja kohdejärjestelmien välille, joiden välillä täytyy siirtää dataa säännöllisin väliajoin.

ETL-prosesseihin liittyy laaja kokonaisuus erilaisia tiedonsiirtoon, varastointiin ja muokkaukseen liittyviä teknologioita ja protokollia. Luvussa 2.1 esitellään ETL-prosessin yhteydessä usein käytettyjä tiedonsiirtoteknologioita ja protokollia.

Tiedot haetaan ensin jostain lähdejärjestelmästä. Hakuprosessin haasteisiin lukeutuu mm. käytettävien tiedonsiirtoprotokollien tekniset haasteet, lähdejärjestelmien asettamat rajoitteet ja muut organisaatiopoliittiset haasteet. Tietoja joudutaan usein ennen sen lataamista muokkaamaan tai vähintään validoimaan. Muokkausprosessin tehtävänä on varmistaa, että tiedot tuodaan kohdejärjestelmään oikeassa muodossa. Luvussa 2.2 tutustutaan tarkemmin ETL:n eri prosessien käytäntöihin ja haasteisiin.

2.1 Protokollat ja teknologiat

Järjestelmien ja teknologioiden moninaisuudesta johtuen ETL-prosessi voidaan rakentaa monella tapaa käyttäen monia erilaisia tiedonsiirtomenetelmiä ja protokollia. Tässä luvussa tutustutaan ETL-prosessien yhteydessä useimmin käytettyihin teknologioihin ja protokolliin.

2.1.1 XML

Järjestelmissä esiintyvä data ei esiinny aina välttämättä toisilleen yhteensopivassa muodossa. On siis tarve esittää data universaalissa formaatissa, jotta järjestelmien välinen datan siirto onnistuisi. Extensible Markup Language (XML) on kehitetty, jotta dataa voitaisiin siirtää alustasta riippumatta.

XML on merkintäkieli joka, jonka määrittämien sääntöjen perusteella voidaan tuottaa sekä ihmiselle että koneelle luettavia dokumentteja. XML-formaattia käytetään usein mielivaltaisten datastruktuurien kuvaamiseen (6). Kuvassa 1 esitellään yksinkertaisen XML-dokumentin sisältö, joka kuvaa kuviteltua asiakasdatastruktuuria.

```
1  <?xml version=1.9"?>
2  <Asiakas>
3      <Id>1</Id>
4      <Nimi>Aku Ankka</Nimi>
5  </Asiakas>
```

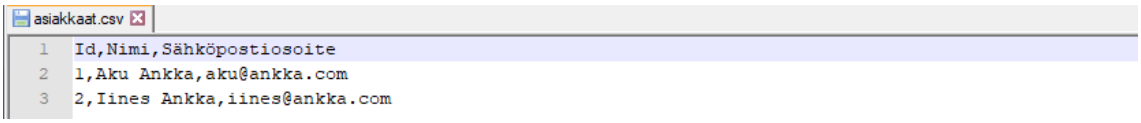
Kuva 1. Esimerkki yksinkertaisesta asiakasdatastruktuurista XML-kielellä kuvattuna.

ETL-prosessissa käytetään usein XML-muotoisia tiedostoja järjestelmien välisessä tiedonsiirrossa. Useimmat ETL-työkalut, Kettle mukaan lukien, sisältävät komponentteja XML-muotoisen datan käsittelyyn.

2.1.2 CSV

XML-formaatin tapaan CSV-, eli Comma Separated Values -formaatti on luotu alustariippumatonta tiedonsiirtoa varten. CSV on yksinkertainen tiedostoformaatti, joka kuvaa taulukkomuotoista dataa. (7.)

Tiedostossa esiintyy kenttiä tai kolumneja, jotka erotetaan toisistaan jotain erotusmerkkiä käyttäen. Tiedostossa rivit tai tietueet erotetaan toisistaan rivinvaihtomerkillä. CSV-tiedostoissa esiintyy valinnainen otsikkorivi, joka kuvaa seuraavien rivien tietueiden struktuuria. Kuvassa 2 on annettu esimerkki CSV-tiedoston sisällöstä.



Kuva 2. Esimerkki asiakastietoja kuvaavasta otsikollisesta CSV-tiedostosta, jossa on käytetty pilkkua erottimena.

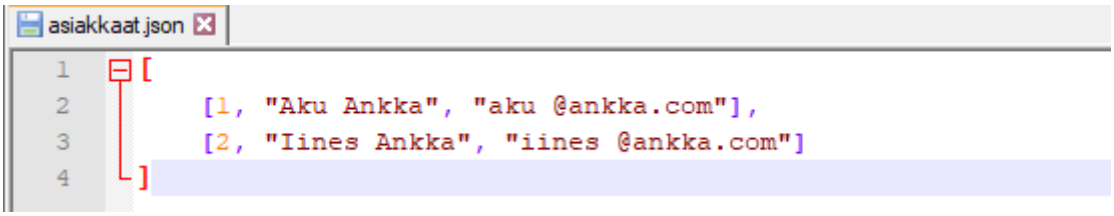
ETL-prosesseissa käsitellään usein CSV-tiedostoja dataa siirrettäessä niiden yksinkertaisuuden ansiosta. CSV-tiedostoilla on helppo kuvata relaatiotietokannassa olevaa taulukkomuotoista dataa, ja tiedostoja on myös helppo käsitellä ohjelmallisesti.

2.1.3 JSON

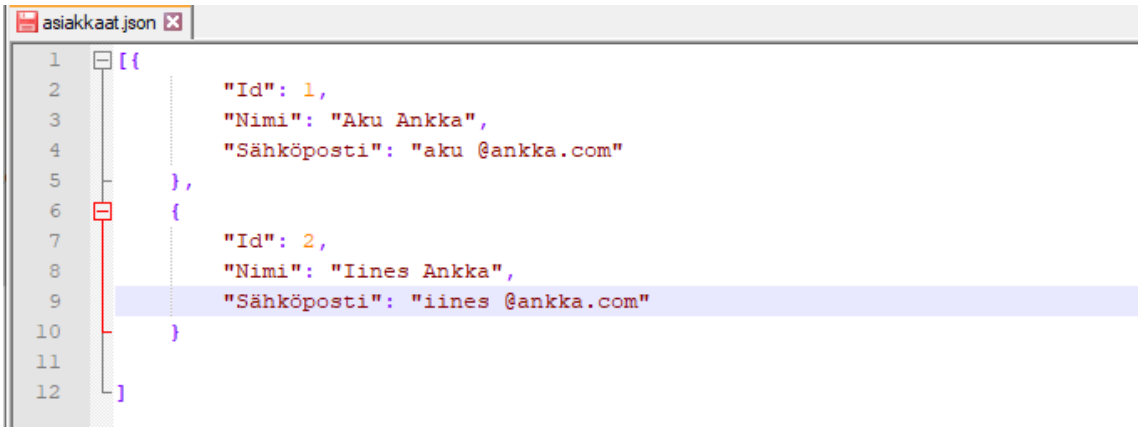
JSON, eli JavaScript Object Notation on JavaScript kieleen perustuva (8) kevyt datansiirtoa varten kehitetty avoin standardi. JSON perustuu kahteen datastruktuuriin:

- joukko avain-arvo pareja, toisin sanoen assosiaatiotaulu (JSON-olio)
- järjestetty arvojen lista tai taulukko (JSON-taulukko).

Kyseiset datastruktuurit esiintyvät lähes kaikissa ohjelmointikielissä jossain muodossa. JSON-formaatti on sen universaalisuutensa ansiosta yleinen formaatti datansiirrossa. Esimerkiksi REST-rajapinnat tarjoavat usein resurssejaan JSON-formaattia käyttäen. Kuvassa 3 on esitelty asiakastietoja kuvaava JSON-taulukko sekä kuvassa 4 JSON-oliota hyödyntävä versio.



Kuva 3. Asiakastiedot esitettynä JSON-taulukkona.



Kuva 4. Asiakastiedot esitettynä JSON-taulukkona, jonka id, nimi- ja sähköpostitiedoissa on hyödynnetty JSON-oliota.

2.1.4 HTTP ja HTTPS

Jotta järjestelmien välinen tiedonsiirto onnistuu, on myös oltava olemassa universaaleja datansiirtoprotokollia. HTTP eli Hyper Text Transfer Protocol on TCP/IP:n päälle rakennettu sovelluskerroksen protokolla, jolla siirretään resursseja, kuten HTML-dokumentteja verkon yli (9). HTTP:stä on tietoturvallisempi versio HTTPS, joka on varustettu SSL- eli Secure Sockets Layer -protokollalla (10). SSL on julkisen avaimen kryptografiaan perustuva tiedonsalausmenetelmä. SSL perustuu varmenteiden käyttöön. SSL-varmenteiden avulla varmistetaan muodostettavan yhteyden luotettavuus sekä salataan HTTP-protokollalla välitettävien viestin sisältö.

HTTP-protokollaa voidaan käyttää myös esimerkiksi XML-tai JSON-tyyppisen datan siirrossa. HTTP-protokollaa käytetään myös tyypillisesti SOAP- ja REST-rajapintojen tiedonsiirtoprotokollana.

2.1.5 FTP ja FTPS

Järjestelmät eivät välttämättä tarjoa aina suoraa rajapintaa datan hakuun tai viemiseen. Data voidaan esimerkiksi joutua siirtämään tiedostoon johonkin välityspalvelimeen, josta data haetaan tai luetaan.

FTP eli File Transfer Protocol on tiedonsiirtoprotokolla, jota käytetään tiedostojen siirtoon verkon välityksellä. (11.) FTP:n on määritetty joukko operaatioita, kuten tiedoston siirto, poisto, lataus ja uudelleennimeäminen, joita FTP-yhteyden ottanut asiakas voi suorittaa palvelimen tiedostoihin. FTP välittää tietoa kahta data- ja komentokanavaa pitkin. FTP-palvelimille voidaan määrittää käyttäjätunnus-salasana-pareja, joiden avulla vastaanotetut yhteydet voidaan autentikoida. FTP-protokollan alaan ei kuitenkaan kuulu minkäänlaista tiedonsalausta, joten välitettävät tunnukset ja data siirretään selväkielisenä.

SFTP-protokolla on FTP-protokollasta erillinen SSH, eli Secure Shell -protokollaa käyttävä tiedonsiirtoprotokolla. SFTP on varustettu FTP:n ominaisuuksien lisäksi laajemmilla tietoturvaominaisuuksilla (12). SFTP-yhteydellä välitettävät tunnukset ja data salataan ennen niiden siirtoa verkon yli.

2.1.6 SOAP

Tiedonsiirron helpottamiseksi useat järjestelmät tarjoavat resurssejaan jonkin ennalta määrätyn rajapinnan kautta. Verkkopalvelut ja rajapinnat määrittävät joukon operaatioita, joilla palvelun alla sijaitsevia resursseja voi hakea tai muokata tiettyä tiedonsiirtoprotokollaa käyttäen.

SOAP, eli Simple Object Access Protocol on XML-pohjainen verkkotiedonsiirrossa käytetty protokolla. SOAP-protokollaa käyttävät verkkopalvelut välittävät ja vastaanottavat tietoa SOAP-protokollan standardin mukaisten XML-tiedostojen avulla. SOAP-protokolla on alustariippumaton, sillä lähes kaikki ohjelmointikielet pystyvät käsittelemään XML-kieltä. (13.)

SOAP-palveluiden toimintoja kuvataan tyypillisesti XML-pohjaisella WSDL-, eli Web Service Description Language -tiedostolla. Tiedosto kuvaa esimerkiksi, mitä parametreja SOAP-palvelu odottaa sille lähetettävissä kyselyissä ja mitä tietoja se palauttaa.

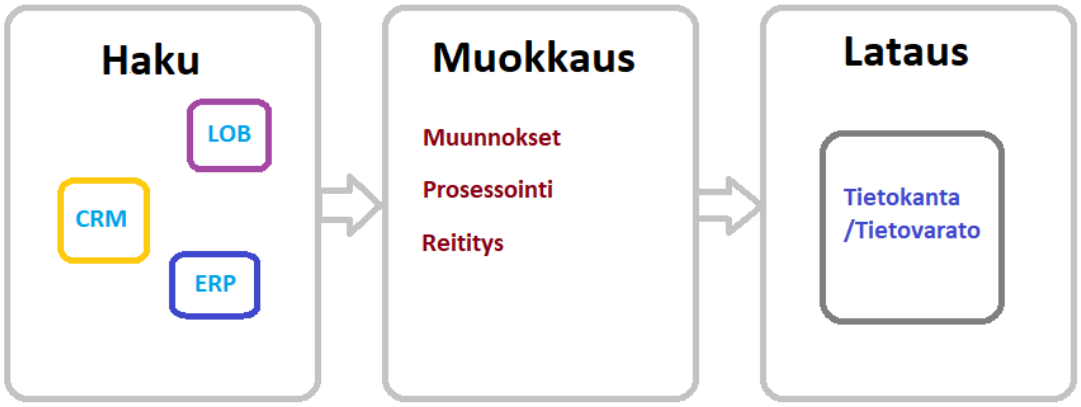
2.1.7 REST

REST eli Representational State Transfer on rajapinta-arkkitehtuuri, jolla välitetään tietoa verkossa HTTP-protokollan avulla. REST on SOAP:n verrattuna uudempi rajapinta-arkkitehtuuri ja on myös nykyään näistä suositumpi valinta uutta rajapintaa rakentaessa (14).

REST-rajapinta määrittää joukon URI- eli Uniform Resource Identifier -osoitteita, joiden kautta päästään kiinni rajapinnan tarjoamiin toimintoihin tai resursseihin (15). REST-rajapinnat tarjoavat resurssejaan tyypillisesti useassa eri muodossa, kuten esimerkiksi JSON-, XML- ja HTML-formaatissa.

2.2 ETL-prosessi

ETL-prosessi kuvataan usein kolmen eri operaation avulla. Nämä operaatiot ovat haku (Extract), muokkaus (Transform) ja lataus (Load). Jokainen operaatio koostuu joukoista omia toimintojaan ja haasteitaan. Kuvassa 5 on hahmoteltu ETL-prosessin kulku. Tieto haetaan ensin jostain lähdejärjestelmästä, kuten asiakastietojärjestelmästä (CRM) tai toiminnanohjausjärjestelmästä (ERP). Hausta saatu tieto viedään muokkausprosessiin, jossa tieto saa lopullisen muotonsa. Muokkausoperaation jälkeen tieto ladataan kohdejärjestelmään.



Kuva 5. ETL-prosessi hahmoteltuna ylätasolla (16).

2.2.1 Haku (Extract) ja Lataus (Load)

ETL-prosessin ensimmäinen askel on tiedon haku kohdejärjestelmästä. Haun tavoite on hakea lähdejärjestelmästä kaikki tarvittu tieto mahdollisimman pienellä resurssien käytöllä jatkoprosessointia varten. Hakuprosessissa on harkittava huolella resurssien käyttöä, jotta se ei vaikuta lähdejärjestelmän toimintakykyyn tämän prosessin aikana. (17.)

Hakuprosessin määrittelyssä on ensimmäisenä harkittava, mitä tietoa haetaan ja mistä. Datan osalta on määritettävä, mikä data on tarpeellista, mitä dataa on saatavilla tai mikä data on muokattavissa tarpeelliseksi muokkausprosessin aikana.

Oman haasteensa datanhakuprosessiin tuo itse datan eristäminen lähdejärjestelmästä. On kartoitettava, onko lähdejärjestelmässä olemassa olevia rajapintoja datan hakua varten ja miten käytettävänä oleva ETL-työkalu soveltuu näiden rajapintojen käyttöön. Yksinkertaisissa tapauksissa data voidaan eristää olemassa olevan rajapinnan kautta, esimerkiksi REST- tai SOAP-rajapintaa käyttäen, tai yhdistämällä suoraan lähdejärjestelmän tietokantaan tietokanta-ajurilla ja suorittamalla SQL-kyselyitä. Lähdejärjestelmän rajoituksista riippuen lähdedata voidaan myös joutua lukemaan suoraan esimerkiksi CSV-tyyppisestä tiedostosta.

Hakuprosessin ajastus määrittää sen, kuinka usein kohdejärjestelmien tietoja virkistetään. On siis myös harkittava, kuinka usein hakuoperaatio suoritetaan, sillä kriittisten

järjestelmien tiedot halutaan virkistää mahdollisimman usein. Virkistystaajuuden kasvaessa kuitenkin järjestelmien kuormitus kasvaa, joten ajastusta on harkittava tarkkaan.

Esimerkkinä toimii IDM-järjestelmä, josta viedään käyttöoikeustietoja johonkin liiketoimintakriittiseen järjestelmään, esimerkiksi asiakastietojärjestelmään. Käyttöoikeuksia poistettaessa tieto halutaan siirtää asiakastietojärjestelmään mahdollisimman nopeasti, jotta väärällä henkilöllä ei ole pääsyä asiakastietoihin. Toisaalta jos asiakastietojärjestelmän käyttöoikeustietoja päivitetään liian usein tietokantaoperaatioilla, voi järjestelmän normaali toimintakyky heikentyä.

Teknisten ongelmien lisäksi haku- ja latausprosessissa haasteena ovat organisaatiopoliittiset- tai sovelluskohtaiset rajoitukset. Esimerkiksi tiettyjen ERP-järjestelmien valmistajat lakkauttavat omien tuotteidensa tuen, jos näiden tietokantoihin otetaan suora yhteys tuotteen ulkopuolelta (18). Lähdeaineisto voi myös sisältää arkaluontoista tietoa, joka voi asettaa datansiirrolle rajoitteita. Arkaluontoisen tiedon siirrossa on valittava tarkasti käytettävät tiedonsiirtoprotokollat. Tiedonsiirrossa tulisi aina käyttää salauksella varattuja tekniikoita. Esimerkiksi HTTP- tai FTP-protokollan sijaan tulisi aina suosia tietoturvallisempia HTTPS- ja SFTP-protokollia. Aineistoa voidaan esimerkiksi säilyttää tiedostossa lähdejärjestelmän palvelimen hakemistossa. Hakemistolle on asetettava tiukat oikeusrajoitteet, jotta mikään ei-haluttu taho ei pääse tähän käsiksi.

Dataa haetaan mahdollisesti useista lähdejärjestelmistä, jotka sisältävät oman käyttäjäkantansa ja oikeutensa. ETL-prosessi tarvitsee siis useita tunnuksia yhdistääkseen jokaiseen järjestelmään. Jotta prosessi voidaan automatisoida, salasanat on säilytettävä jossain kohtaa prosessin konfiguraatiota. Käytettävän ETL-työkalun tulisi pystyä salaamaan (kryptaamaan) salasanat, jotta ne ei esiinny konfiguraatiossa selkokielisinä.

2.2.2 Haku- ja latausstrategiat

Datan haku ja lataus voidaan suorittaa eri strategioilla riippuen siitä, pystytäänkö lähdeaineistosta erottamaan prosessin viimeisen suorituksen jälkeen muuttuneet tiedot.

Jos lähdejärjestelmän tiedoista ei pysty havaitsemaan, mikä data on muuttunut edellisen haun jälkeen, data on haettava kokonaisuudessaan uudelleen jokaisen haun yhteydessä

(Full extraction/Full load). Täysihaku suoritetaan tyypillisesti vain, jos data haetaan ensimmäistä kertaa lähdejärjestelmästä ladattavaksi. Aineiston koosta riippuen tämä voi kuitenkin olla raskas operaatio lähde- ja kohdejärjestelmälle. Operaatiossa joudutaan käyttämään turhaan resursseja muuttumattomien datayksiköiden uusimiseen. Esimerkiksi verkkokaupan tietokannassa voi esiintyä oma taulu tilauksille. Oletetaan, että kohdejärjestelmän kantaan on viety prosessin viimeisessä suorituksessa 20000 tilaustapahtumasta tarvittavat tiedot. Jos seuraavana päivänä tilaustapahtumia esiintyy vain 500 lisää, kaikki 20500 tapahtumaa joudutaan lukemaan kantaan uudelleen.

Lähdejärjestelmästä voidaan tehdä myös osittainen datan haku (Incremental extraction/Incremental load), jos lähdejärjestelmän datasta voidaan havaita päivittynyt data. Datan muutos voidaan havaita esimerkiksi aikaleiman tai jonkin tilatietoattribuutin perusteella. Viimeisestä hausta voidaan myös muodostaa kopio, jotta aineistoja vertailemalla voidaan määrittää, mikä data on muuttunut. Kopion ylläpito ja datan vertailu tuovat toisaalta lisää ei haluttua kuormitusta prosessille. Osittainen lataus toisaalta monimutkaistaa prosessia, sillä datan muutoksista ja muutosjärjestyksestä on pystyttävä pitämään kirjaa. Jos tietoa ei saada ladattua prosessin aikana esimerkiksi virheen takia, on pystyttävä määrittämään, missä järjestyksessä datalle tehdyt muutokset ovat tapahtuneet, jotta kohdejärjestelmään ei viedä ristiriitaista tietoa. (19.)

2.2.3 Muokkaus (Transform)

ETL:n muokkausprosessilla viitataan joukkoon datalle suoritettavia operaatioita, esimerkiksi datan kokoamista, lähde- ja kohdedatan välisten suhteiden muodostamista ja sen puhdistamista, ennen kuin se viedään kohdejärjestelmään. (20.)

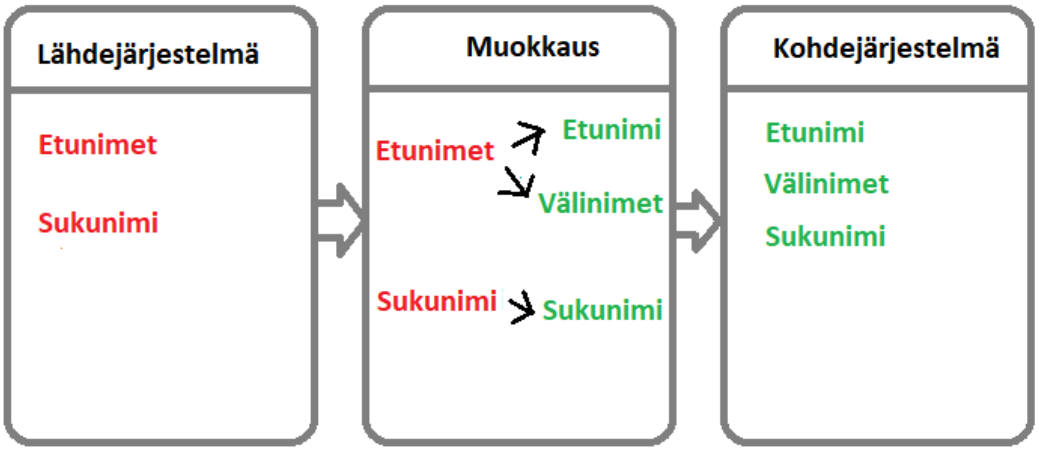
Taulukossa 1 on esitelty tyypillisiä datalle suoritettavia operaatioita muokkausprosessin aikana.

Taulukko 1. Datalle suoritettavia operaatioita muokkausprosessin aikana (21).

Operaatio	**Selite**
Datan vastaavuuksien määritys (Data mapping)	Kohde- ja lähdejärjestelmän datan vastaavuuksien määrittäminen
Datan validointi	Datalle suoritettavia virhetarkistuksia.
Datan puhdistus (Data cleaning)	Virheelliselle datalle suoritettavia korjauksia.
Datan kokoaminen (Data aggregation)	Koostetietojen muodostaminen datan perusteella, esimerkiksi summa.
Datan järjestäminen	Datan organisointi haluttuun järjestykseen.
Johtaminen (Derivation)	Haetun datan perusteella muodostetaan jotain uutta dataa. Esimerkiksi voiton laskeminen kulujen ja tulojen perusteella.
Datan standardisointi	Useasta lähdejärjestelmästä haetun datan muuntaminen standardimuotoon.

Muokkausprosessin osana lähdedata tulee sovittaa kohdejärjestelmään sopivaksi määrittämällä, mitkä lähdedatan kentät sopivat mihinkin kohdejärjestelmän kenttiin. Lähde- ja kohdejärjestelmän kenttien välille tulee muodostaa yhteys, mutta kentät eivät ole aina suoraan sovitettavissa toisiinsa.

Kahdessa järjestelmässä voi esimerkiksi esiintyä molemmissa henkilön nimitieto. Nimitieto lähdejärjestelmässä A esiintyy kahdessa kentässä: etunimet ja sukunimi. Kohdejärjestelmässä esiintyy kuitenkin kolme kenttää: etunimi, välinimet ja sukunimi. Muokkausprosessissa joudutaan tässä tapauksessa kohdistamaan lähdejärjestelmän etunimet kenttä kahteen eri kohdejärjestelmän kenttään etunimi ja välinimet. Pelkän kenttien välisen yhteyden muodostamisen lisäksi joudutaan myös tekemään kentälle muunto-operaatio. Kuvassa 6 on hahmoteltu esimerkin mukainen tilanne.



Kuva 6. Esimerkki datan vastaavuuksien muodostamisesta järjestelmien välillä nimitieto esimerkkiä käyttäen.

Muokkausprosessin aikana etunimet-kenttä tulee jakaa kahtia esimerkiksi välimerkin perusteella, mikä on yksinkertainen operaatio useimmilla ohjelmointikielillä. Etunimet-kentän kohdalla on kuitenkin harkittava erikoistapauksia. Kentässä voi esiintyä esimerkiksi vain yksi nimi, jolloin kentän jakaminen ei onnistu. Tähän on varauduttava käsittelemällä nämä tapaukset erikseen.

Dataa voidaan hakea myös useasta lähdejärjestelmästä. Kuvassa 5 esiteltiin yleiskuvaus ETL-prosessin kulusta. Kuvassa haku-operaatio kohdistuu kolmeen lähdejärjestelmään CRM-, LOB-, ERP-järjestelmään. Järjestelmissä voi esiintyä samankaltaista dataa monenlaisessa eri muodossa. Lähdejärjestelmistä haettava data tulee siis standardisoida, jotta se voidaan viedä yhtenäisenä kohdejärjestelmään käytettäväksi.

2.2.4 Datan validointi

Datan validoinnilla tarkoitetaan muokkausprosessin osaa, jossa lähdejärjestelmästä haetusta datasta havaitaan, sekä mahdollisesti korjataan virheet (22). Esimerkiksi BI-järjestelmille datan tarkkuus on tärkeää, sillä virheellisen tai epätarkan datan tuominen järjestelmän tietovarastoon heikentää myös datan tuottamien analyysien tarkkuutta. Datan validointi on tärkeää myös IDM-järjestelmille. Jos IDM-järjestelmään liittyvä prosessi vie esimerkiksi käyttäjätietoja johonkin kohdejärjestelmään, on tärkeää validoida, että kaikki kohdejärjestelmän vaatimat käyttäjätiedot ovat mukana prosessissa oikeassa

muodossa. Käyttäjän pääsy kohdejärjestelmään saatetaan evätä virheellisten tai puuttuvien tietojen myötä. Taulukossa 2 on kuvattu tyypillisiä datalle suoritettavia validointeja.

Taulukko 2. Muokkausprosessin aikana suoritettavia validointeja (23).

Valdiointi	**Selite**	**Esimerkki**
Arvoalue	Tarkistetaan, kuuluuko haetun kentän arvo sallittujen arvojen alueelle.	Päivämäärän tulee olla tiettyjen rajojen sisällä
Datan tyyppi	Tarkistetaan, onko haetun kentän datan tyyppi oikea.	Kannasta haetun kentän tietotyyppi on oikea.
Ei-sallitut merkit	Tarkistetaan, löytyykö kentästä joitain ei-sallittuja merkkejä.	Puhelinnumero-kentässä ei saa olla kirjaimia jne.
Tyhjä/Null -arvo	Tarkistetaan, että kenttä ei ole tyhjä.	Tyhjää ""-merkkijonoa, null tai 0 arvoa ei sallita pakollisessa kentässä.
Datan muoto	Tarkistetaan, esiintyykö kentän data oikeassa muodossa.	Sähköpostiosoitteessa esiintyy "@XXX.fi"-muotoinen pääte.
Datan pituus	Tarkistetaan, löytyykö datasta ennalta määrätty määrä merkkejä.	Postinumero tulee olla 5 merkkiä pitkä

Validoinnin tarve riippuu myös osittain lähdejärjestelmän ja lähdedatan ominaisuuksista. Luvussa 2.4 annettiin esimerkki yksinkertaisesta nimenmuokkausoperaatiosta. Jos etunimet-kentälle ei ole asetettu rajoituksia lähdejärjestelmässä, validointioperaatio monimutkaistuu. Kenttään voidaan esimerkiksi vahingossa syöttää erikoismerkkejä, joita kohdejärjestelmä ei salli. Lisäksi kohdejärjestelmän etunimi- tai välinimet-kentälle voi olla asetettu kohdejärjestelmässä pituusrajoitus teknisistä syistä. Nyt alkuun yksinkertaiselta vaikuttava nimen validointi ja muokkaus -operaatio vaatii jo kolme tarkistusta: löytyykö etunimet-kentästä kaksi nimeä, kuinka pitkä muodostettava kohdejärjestelmän kenttä on ja löytyykö näistä erikoismerkkejä.

Jos lähdejärjestelmässä datan muodolle on määritelty tiukka noudatettava skeema, validointia tarvitaan vähemmän. Datan skeemalla tarkoitetaan datan luokittelua ja sille asetettuja rajoituksia. Skeemaa noudattaessa datan validointi hoituu täten osin jo silloin, kun data ensimmäistä kertaa syötetään lähdejärjestelmään. (24.) Jos lähdejärjestelmän datalla ei ole skeemaa, tai siihen syötettävälle datalle ei suoriteta tarkistuksia, validoinnin tarve kasvaa.

Kohdejärjestelmän tarpeista riippuen virheellinen data voidaan poistaa käsiteltävästä datasta kokonaan. Vaihtoehtoisesti data voidaan puhdistaa muokkaamalla data oikeaan muotoon muokkausprosessin aikana, mikäli se on mahdollista. Datan puhdistus ETL-prosessin yhteydessä on kiistelty aihe (25), sillä virheellinen data mielletään lähdejärjestelmän tietosisällön ongelmaksi. Datan puhdistus lähdejärjestelmässä voi olla kuitenkin haastavaa ja resurssiraskasta. Lähdejärjestelmään voidaan syöttää jatkuvasti virheellisessä muodossa olevaa dataa, mikäli skeema sen sallii. Tietosisällön laatua tulisi tällöin auditoida jatkuvasti, jotta siirrettävässä datassa ei esiinny virheitä.

2.2.5 Lokitus ja auditointi

ETL-prosessin aikana voi ilmentyä monenlaisia virheitä, jotka voivat liittyä esimerkiksi ennalta tiedossa oleviin lähdedatan eheyden ongelmiin. Jotta virheenselvitys onnistuisi mahdollisimman helposti, prosessin tulisi tuottaa kattavia lokituksia. Lokitus ja lokitiedeostot ovat hyödyllysiä myös prosessin auditoinnissa. Prosessi voi säilöä tietoa esimerkiksi eri osaprosessien suoritusajoista, jotta näistä voidaan havaita resurssien käytön pullonkauloja. Lisäksi prosessi voi säilöä tietoa haettujen ja ladattujen rivien määrästä. Tiedon avulla voidaan valvoa, että prosessin aikana tietoa ei häviä.

Blogissaan Tim Mitchell (26) listaa ETL-prosessin eri osa-alueita, joiden lokitusta kannattaisi harkita:

- Prosessin aloitus- ja pysäytystapahtumat. Prosessikokonaisuuden, sekä osaprosessien aloitus- ja lopetusajankohdat tulisi säilyttää.
- Tila. Osaprosessit voivat onnistua tai epäonnistua itsenäisesti, joten myös osaprosessien suorituksen tila (epäonnistunut tai onnistunut) tulisi lokittaa itsenäisesti.

- Auditointiin liittyvä informaatio. Prosessin ja osaprosessien suoritusajat, haetut ja ladatut rivimäärät. Tarvittaessa yksityiskohtaisempaa tietoa datasta.
- Testaukseen ja virheenkorjaukseen liittyvä informaatio. Hyödyllistä tietoa prosessin kehitysvaiheessa, erityisesti niille prosesseille, joissa suoritetaan paljon muokkausoperaatioita.

Lokien sisällön lisäksi Mitchell kirjoittaa lokitiedostojen säilyttämiseen liittyvistä käytänteistä. ETL-prosessi saattaa koostaan riippuen tuottaa suuren määrän lokitietoa, joka vie paljon levytilaa. Lokeja voidaan säilyttää joko lyhyt tai pitkä aika riippuen niiden tarkoitusperästä. Auditoinneille tarpeelliset lokit halutaan säilyttää tyypillisesti pitkään, kun taas esimerkiksi prosessin ylläpidolle tarkoitettuja lokituksia säilytetään vain se aika, kun ne ovat tarpeellisia. Pitkään säilytettäviä lokeja kannattaa arkistoida säännöllisin väliajoin halvempaan varastotilaan, jotta ne eivät vie turhaan resursseja prosessointia suorittavalta palvelimelta.

3 Pentaho Data Integration (Kettle)

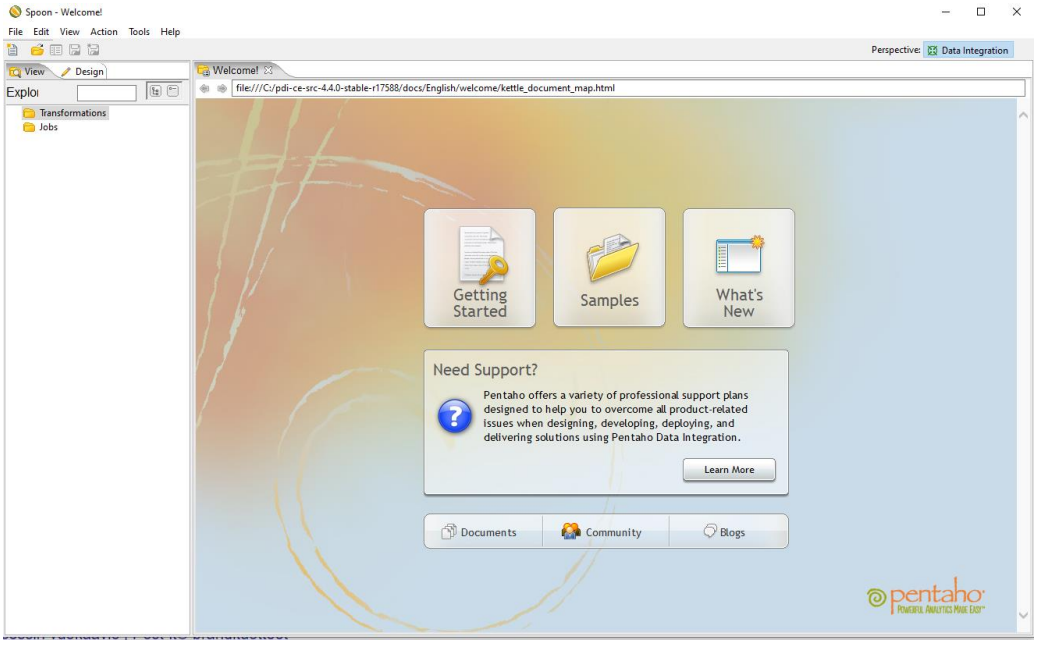
ETL-prosessointia varten on luotu useita ilmaisia ja maksullisia ETL-työkaluja, jotka tähtäävät helpottamaan ETL-prosessin rakentamisessa ja prosessoinnissa.

Pentaho on Hitachi Data Systems -yhtiön omistama BI-ohjelmistokokonaisuus. Tässä luvussa tutustutaan Kettlenä tunnettuun Java-pohjaiseen avoimen lähdekoodin Pentaho Data Integration -ETL-työkalun ominaisuuksiin, joka on osa Pentaho-kokonaisuutta (27). Kettle on varustettu Apache 2.0 -lisenssillä (28).

Kettle sisältää ETL-prosessien suunnittelua ja muotoilua varten graafisen ohjelmointiympäristön, joka tunnetaan nimellä Spoon. Spoonilla suunnitelluista prosesseista muodostetaan XML- tai Ketllelle spesifisiä KTR-tiedostoja (Kettle Transformation File), jotka noudattavat samaa XML-formaattia. Kettle-ohjelmisto koostuu myös Pan- ja Kitchen-nimisistä prosessointimoottoreista, jotka suorittavat näiden tiedostojen pohjalta Spoonilla muodostettuja töitä ja transformaatioita. (29.)

3.1 Käyttöliittymä (Spoon)

Spoon mahdollistaa Kettle-töiden ja -transformaatioiden suunnittelun graafisessa ohjelmointiympäristössä.

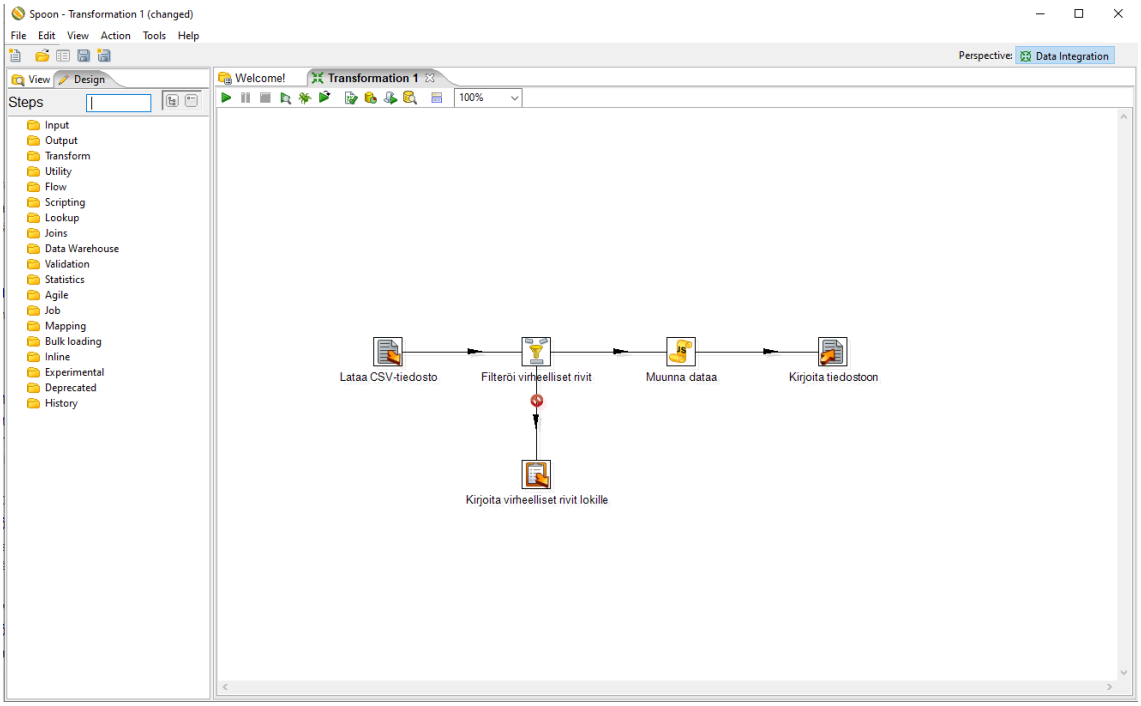


Kuva 7. Spoonin aloitussivu

Spoonin aloitussivu on esitelty kuvassa 7. Spoonilla voidaan muodostaa kahdenlaisia dokumentteja: töitä (Job) ja transformaatioita (Transformation). Transformaatiot kuvaavat tietovirran kulkua, kuten datan hakua lähteestä, sen muokkausta ja lataamista kohteeseen. Työt taas koordinoivat prosessikokonaisuuden kulkua. Työt voivat koostua useasta transformaatiosta tai työstä, joiden suorituksen järjestys määrätään myös työtasolla. Työtasolla voidaan myös suorittaa valmistavia operaatioita ennen transformaation suoritusta, kuten tietokanta- tai verkkopalveluyhteyksien tarkistamista, tai viimeisteleviä operaatioita transformaatioiden suorituksen jälkeen.

Spoonilla suunnitellut transformaatiot ja työt koostuvat useasta yksittäisestä askelesta, joiden välinen kulku kuvataan Spoonissa vuokaaviomaisella esitystavalla. Työ- ja transformaatiosuunnitteluikkunassa löytyy ruudun vasemmasta laidasta Design-välilehti, josta löytyy erilaiset askeleet, joita prosessit voivat käyttää. Askeleet on luokiteltu vali-

kossa loogisiin kokonaisuuksiin kuten esimerkiksi datan syöttö, eli input-askeleisiin. Käytettävät askeleet voidaan raahata hiirellä Design-valikosta suunnitteluruutuun ja niitä klikkaamalla voidaan muodostaa datan virtaa kuvaavia nuolia askelien välille.



Kuva 8. Esimerkki yksinkertaisesta transformaatiosta. Transformaatio kulkee eteenpäin vuokaaviomaiseen tapaan askelien läpi.

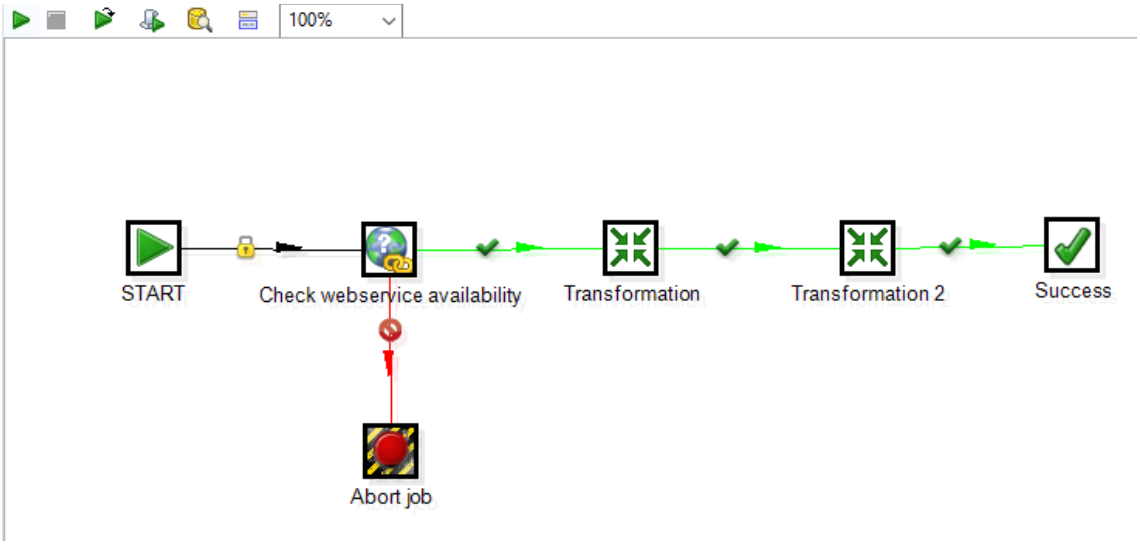
Spoonin kautta voidaan myös suorittaa sillä muodostettuja töitä ja transformaatioita. Käyttöliittymältä voidaan tarkastella suorituksen jälkeen töihin ja transformaatioihin liittyvää statistiikkaa sekä lokitustietoja.

3.2 Kettle-prosessin kulku

Kettlellä voidaan suorittaa yksittäisiä transformaatioita tai töitä, jotka voivat koostua useammasta transformaatiosta tai työstä. Tässä luvussa kuvataan Kettle-prosessin ominaisuuksia esimerkkejä tarkastellen.

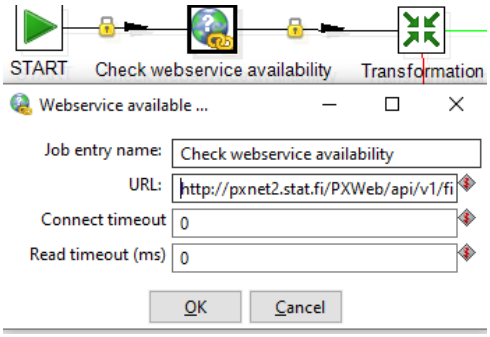
3.2.1 Työ

Työ alkaa Start-askeleella ja onnistuneet työt lopetetaan Success-askeleeseen. Askelien välisillä nuolilla kuvataan askelien välisiä hyppyjä (Hop). Hypyillä voidaan määrittää työssä, mitä tehdä epäonnistuneen ja onnistuneen askeleen jälkeen. Vihreällä nuolella kuvatut hypyt suoritetaan vain, jos edellinen askel on suoriutunut ilman virheitä ja punaisilla vastaavasti ne hypyt, joissa virhe on tapahtunut. Mustalla nuolella ja lukolla kuvatut hypyt suoritetaan aina virheistä riippumatta.



Kuva 9. Esimerkki työstä

Ennen transformaatioiden suoritusta työssä voidaan tehdä alustavia tarkistuksia. Kuvan 9 esimerkissä tarkistetaan, vastaako tietty verkkopalvelu kutsuihin. Työtä ei tarvitse edistää, jos verkkopalvelu ei vastaa, sillä lähdeaineisto ei tässä tapauksessa ole saatavilla.



Kuva 10. Check webservice availability -askeleen konfiguraatioikkuna. Askel on konfiguroitu tarkistamaan URL-osoite http://pxnet2.stat.fi/PXWeb/api/v1/fi/, joka on Tilastokeskuksen ylläpitämä REST-rajapinta.

Kuvassa 10 on esitelty Check webservice availability -askeleen konfiguraatio. Jos verkkopalvelu ei vastaa askeleen lähettämään pyyntöön, siirrytään virhekäsittelyhyppyyn. Virhekäsittelyhyppy johtaa Abort job -askeleeseen, joka keskeyttää työn suorituksen. Tarkistuksen jälkeen aloitetaan Transformation -askeleeseen konfiguroitu transformaatio. Töissä ei itsessään käsitellä dataa, vaan se tapahtuu transformaatioissa. Kettlen dokumentaatio kuvaa töiden roolia Kettle-prosessissa seuraavasti (30):

- Määrittää transformaatioiden virran ja riippuvuudet.
- Valmistelee transformaatioiden suoritusta tarkistamalla ehtoja, kuten tietokannan tai verkkopalvelun yhteyden saatavuus.
- Lähettää tai lokittaa ilmoituksia töiden tai transformaatioiden onnistumisesta tai epäonnistumisesta.

Prosessin alussa työlle allokoidaan oma muisti, joka on yhteinen jokaiselle työn transformaatiolle. Transformaatiot toimivat lähtökohtaisesti isolaatiossa toisistaan. Kettle tarjoaa joukon askelia, joilla voidaan tallentaa ja hakea dataa työn muistista. Taulukossa 3 on esitelty nämä askeleet.

Taulukko 3. Työn muistia käsittelevät askeleet.

Askel	**Selite**
Get rows from result/Copy rows to result	Siirtää muistiin/hakee muistista transformaatiossa käsitellyt rivit.
Get/Set variables	Siirtää/hakee muistista yksittäisiä muuttujia.

Get files from result/Set files in result	Siirtää muistiin/hakee muistista transformaatiossa käsiteltyjä tiedostoja.

3.2.2 Transformaatio

Transformaatio koostuu työn tavalla joukosta askelia. Transformaatiossa käytettävät askeleet on jaettu Spoonissa useaan eri valikkoon, mutta ne voidaan jakaa karkeasti neljään eri kategoriaan:

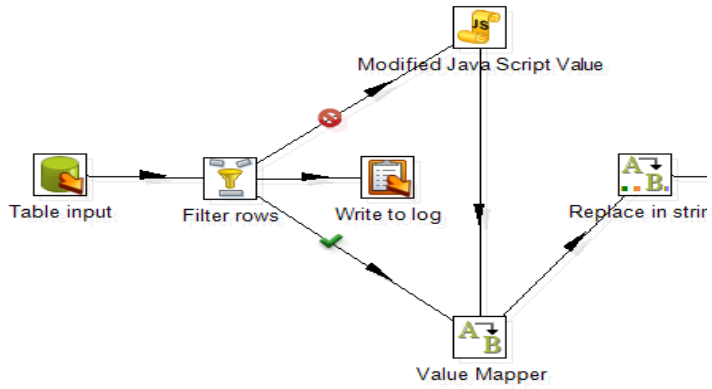
- datan haku (Input)
- datan muokkaus ja validointi
- datan lataus (Output)
- lokitus ja informatiiviset askeleet, sekä transformaation kulun hallinta.

Transformaatioissa datavirtaa käsitellään aina riveinä, jotka haetaan jonkin input-askeleen avulla. Rivien data löytyy kentistä ja kenttien sisältö määritetään kunkin askeleen konfiguraatiossa. Jos datavirta on tyhjä, eli käsiteltäviä rivejä ei ole, askelia ei suoriteta.

Töiden tapaan askelien väliset hypyt esitetään nuolilla. Hyppyjen kulku voidaan määrittää askeleen ominaisuuksien mukaan eri tavoilla:

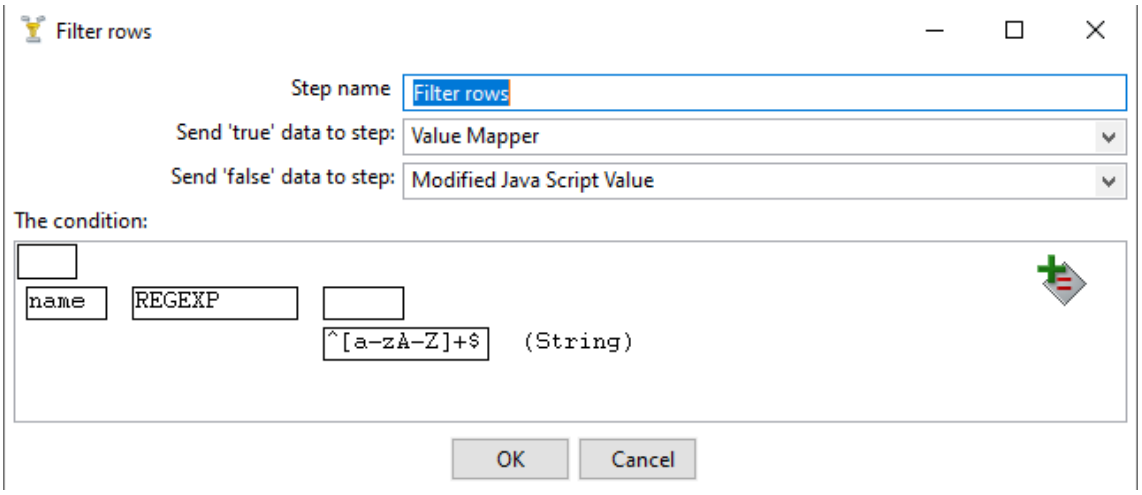
- Main output, eli askeleen pääasiallinen hyppy. Rivit kulkevat kyseistä hyppyä pitkin, jos lähdeaskeleessa ei ole tapahtunut virheitä.
- Virhekäsittelyhyppy. Tätä hyppyä kuljetaan, jos rivin käsittely aiheutti virheen lähdeaskeleessa. Huomattavaa on, että virheen luonteesta riippuen koko transformaatio voi epäonnistua yksittäisen rivin sijasta. Virheen käsittelyä käsitellään tarkemmin seuraavassa alaluvussa.
- Ehdolliset hypyt, jotka päätellään jonkin datavirran kulkua määrittävän askeleen avulla. Esimerkkinä Filter rows -askel, joka suodattaa rivejä kenttien arvojen perusteella.

Kuvan 11 esimerkissä Filter rows -askeleen jälkeen datavirta jakautuu kolmeen eri askeleeseen. Askeleessa voidaan esimerkiksi tarkistaa, löytyykö jonkin rivin kentästä ei sallittuja arvoja, joita ei voida kohdejärjestelmään sellaisenaan viedä. Jos rivin kentät läpäisevät suodatuksen, ne siirtyvät suoraan Value Mapper -askeleeseen. Jos rivi ei läpäise suodatusta, se viedään Modified Java Script -askeleeseen, jossa data voidaan puhdistaa virhearvoista.



Kuva 11. Esimerkki transformaatiosta, jossa Filter rows -askeleesta, lähtee hyppy kolmeen eri askeleeseen.

Tämän askeleen jälkeen rivi siirtyy esimerkissä takaisin samaan datavirtaan onnistuneesti suodatettujen rivien joukkoon. Näiden hyppyjen lisäksi kaikki Filter Rows -askeleessa käsitellyt rivit viedään Write to Log -askeleeseen, riippumatta siitä, suodatettiinko rivi vai ei. Kuvassa 12 on esitelty esimerkki Filter rows -askeleen konfiguraatioista, joissa rivejä suodatetaan sille annetun säännöllisen lausekkeen mukaan.

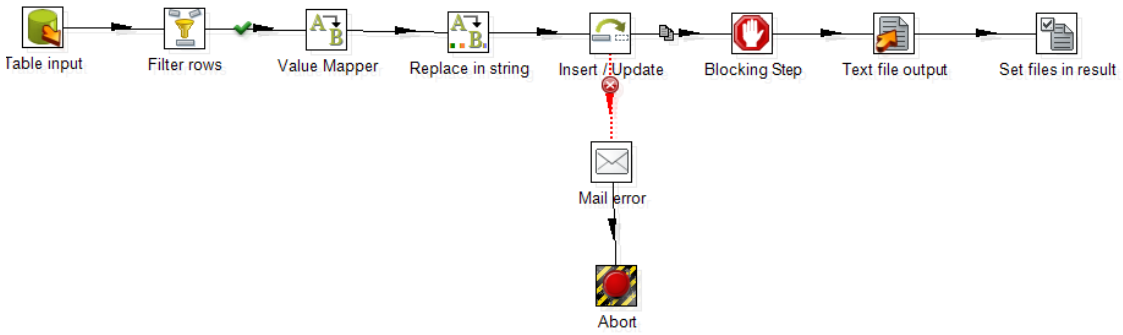


Kuva 12. Esimerkki Filter rows -askeleen konfiguuraatiosta. Askeleessa tarkistetaan regexp- eli säännöllisiä lausekkeita käyttäen, että rivin "name"-kentästä löytyy pelkästään aakkosia.

Vaikka transformaation askeleet vaikuttavat peräkkäisiltä Spoonin vuokaaviomaisessa esitystavassa, se ei takaa, että ne suoritetaan peräkkäin. Kettle käsittelee rivejä transformaation sisällä rinnakkain. Tämä tarkoittaa, että rivit voivat olla käsiteltävinä eri askeleissa samaan aikaan. Transformaatioissa voi kuitenkin esiintyä tilanteita, joissa askelia

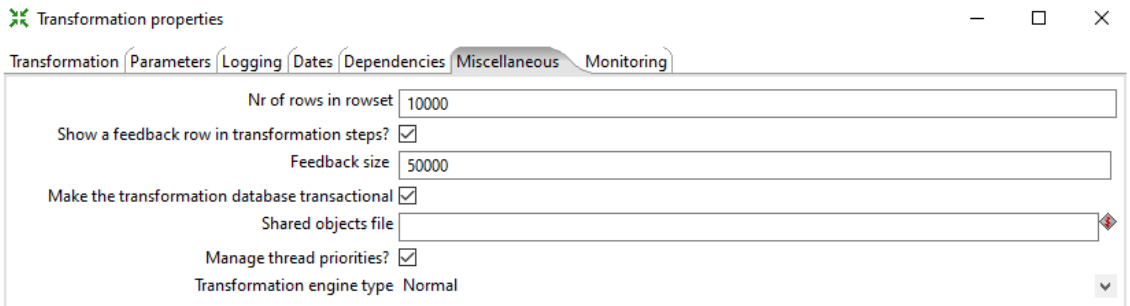
tulee suorittaa peräkkäin. Jos peräkkäisyys tiettyjen askelien välille halutaan taata, jälkimmäisenä suoritettava askel tulisi lähtökohtaisesti sijoittaa eri transformaatioon. Transformaatioissa voidaan käyttää myös Blocking Step -askelta, joka pysäyttää rivien käsittelyn, kunnes kaikki rivit on käsitelty kyseiseen askeleen asti.

Kuvan 13 esimerkissä data haetan jostain tietokannasta Table Input -askeleella, johon on konfiguroitu tietokantayhteyden ottamiseen tarvittavat tiedot, kuten tunnukset, tietokannan tyyppi ja ajurit, sekä käytettävä SQL-lause, jolla tiedot haetaan. Filter rows-, Value Mapper-, ja Repalce in String -askeleet suodattavat ja muokkaavat dataa.



Kuva 13. Esimerkki transformaatiosta, jossa käytetään Blocking Step -askelta

Insert / Update -askel lataa tiedot muokatut tiedot johonkin tietokantaan. Kettlen transformaatio- ja työasetuksissa voidaan määrätä käytettävien operaatioiden transaktionaalisuus. Transaktionaalisuus määrittää, että transformaation aikana päivitetyt rivit palautetaan alkuperäiseen tilaan, mikäli transformaatio kaatuu virhetilanteeseen. Kuvassa 14 nähdään konfiguraatioikkuna, jossa transaktionaalisuus voidaan määrittää.



Kuva 14. Välilehti transformaation asetuksista, jossa voidaan määrittää transaktionaalisuus. Asetuksissa voidaan myös esimerkiksi transformaation lokitukseen ja monitorointiin liittyviä konfigurointeja.

Esimerkin transformaatiossa on määritetty Insert / Update -askeleen virhekäsittelyhyppy Mail-askeleeseen, jonka jälkeen suoritetaan Abort-askel, joka pysäyttää transformaation käsittelyn ja asettaa sen virheeseen. Ne rivit, jotka transformaatio on jo päivittänyt onnistuneesti ennen tämän askeleen suoritusta, palautetaan tilaan, jossa ne olivat ennen käsittelyä, jos transformaatio on asetettu transaktionaaliseksi.

Jos kaikki rivit tuodaan tai päivitetään onnistuneesti kohdetietokantaan, käsitellyistä riveistä viedään koostetietoja tiedostoon Text file output -askeleella. Tiedoston metatiedot viedään työn muistiin Set files in result -askeleella, jotta sitä voidaan käsitellä myöhemmissä transformaatioissa tai työn sisällä. Text file output -askel suoritetaan esimerkin transformaatiossa Blocking Step -askeleen jälkeen. Tämä takaa, että tiedostoa ei muodosteta turhaan, jos transformaation käsittely keskeytyy aiemmissa askeleissa.

3.2.3 Virheiden käsittely

Jotta ETL-prosessi ei kaadu jokaiseen datassa esiintyvään virheeseen, tulee ETL-työkalun pystyä määrittämään erillinen virhekäsittely virheelliselle datalle. Virheellistä dataa ei myöskään saa ladata kohdejärjestelmään, sillä virheellisen datan lataaminen saattaa vaikuttaa tämän järjestelmän toimintakykyyn.

Aiemmissa luvuissa tutustuttiin pinnallisesti Kettle-prosessin virheiden käsittelyyn. Työtasolla voidaan määrittää erillinen työnkulku, jos työhön kuuluvissa transformaatioissa tai töissä esiintyy virheitä. Transformaatioissa virheitä voidaan käsitellä rivitasolla askeleen ominaisuuksien mukaan.

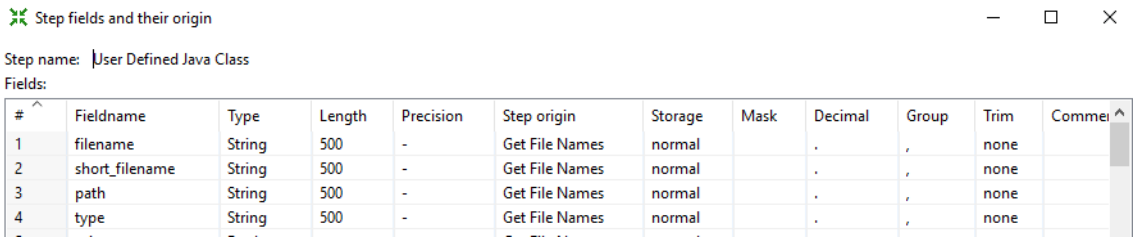
Virheitä voi esiintyä askeleiden tai transformaatioiden konfiguraatiossa, mikä johtaa useimmiten työn tai transformaation kaatumiseen. Kaatumisen voivat aiheuttaa esimerkiksi virheellisesti syötetyt tietokantayhteyden tiedot, puuttuvat tiedostot tai väärin määritellyt datan formaatit askeleen konfiguraatiossa. Erityisen virheherkkiä ovat transformaatioissa käytettävät JavaScript- ja Java-askeleet. Askeleilla voidaan käsitellä rivejä käyttäjän itse kirjoittamalla JavaScript- tai Java-koodilla. Esimerkiksi indeksointi- ja null pointer -virheet aiheuttavat koko transformaation kaatumisen.

Osa askeleista tukee tarkempaa virhekäsittelyjen konfiguraatiota. Näissä askelissa voidaan määrittää seuraavia virheisiin liittyviä parametreja:

- Kentät, joihin viedään tietoja virheestä, kuten virhekoodi ja -kuvaus, virheiden lukumäärä ja virheen aiheuttanut kenttä.
- Virheiden maksimimäärä ennen kuin transformaation suoritus keskeytetään.
- Virheiden maksimimäärä suhteessa onnistuneisiin käsittelyihin.
- Luettujen rivien minimilukumäärä, jonka jälkeen prosentuaalista evaluointia aletaan laskemaan.

3.3 Askeleet

Kettle tarjoaa useita valmiiksi määritettyjä askelia, joilla rivejä voidaan hakea, muokata ja ladata. Suurin osa Kettlen valmiista askelista on helposti konfiguroitavissa. Jokaiseen askeleeseen konfiguroidaan sisään syötettävät ja ulos tulostettavat kentät. Suurin osa askelista pystyy tunnistamaan automaattisesti sille syötettävät kentät edellisen askeleen ulostulon perusteella. Kuvassa 15 on tästä esimerkki. Get File Names -askel hakee tietystä hakemistopolusta tiedostoja, joiden perusteella se muodostaa käsiteltäviä rivejä. Kun askeleesta muodostetaan hyppy User Defined Java Class -askeleeseen, sille syötettävät kentät ovat automaattisesti nähtävissä askeleen kentät-valikossa.



#	Fieldname	Type	Length	Precision	Step origin	Storage	Mask	Decimal	Group	Trim	Comme
1	filename	String	500	-	Get File Names	normal		.	,	none	
2	short_filename	String	500	-	Get File Names	normal		.	,	none	
3	path	String	500	-	Get File Names	normal		.	,	none	
4	type	String	500	-	Get File Names	normal		.	,	none	

Kuva 15. Esimerkki User Defined Java Class -askeleelle syötettävistä kentistä. Lähdeaskeleena toimii Get File Names -askel.

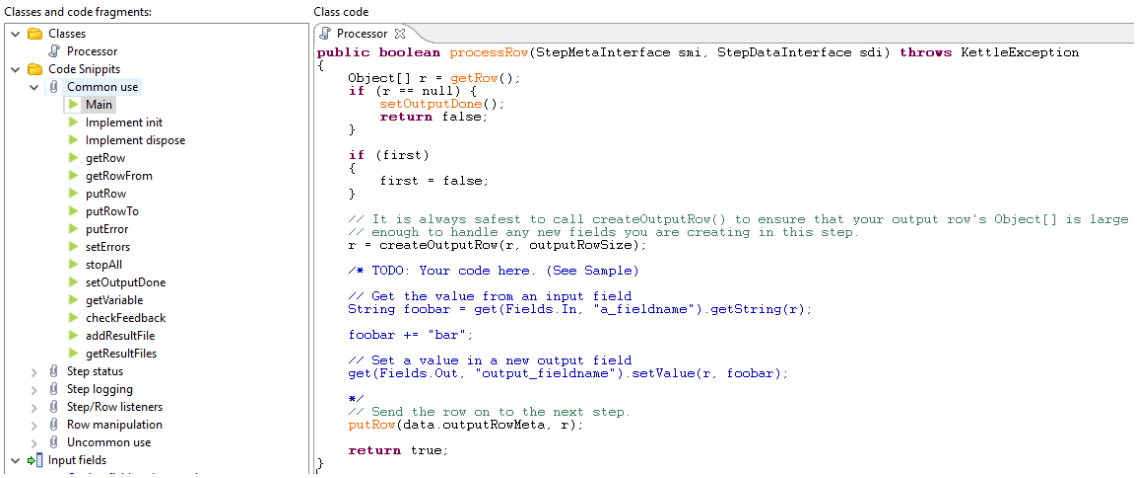
Valmiiden askeleiden lisäksi Kettle tukee kahta enemmän muokattavuutta sallivaa askeletta, jotka ovat nimeltään User Defined Java Class ja Modified JavaScript Value. Nimensä mukaisesti näillä askeleilla voidaan kirjoittaa omaa Java- tai JavaScript-koodia.

Pentahon sivuilta löytyy kattava dokumentaatio erilaisten askelien konfiguraatiosta ja toiminnallisuudesta (31). Seuraavissa alaluvuissa keskitytään vaativampien User Defined Java Class ja Modified Java Script -askelien käyttöön sekä esittelemään, minkälaisia datan haku- ja latausmenetelmiä Kettle tukee.

3.3.1 User Defined Java Class -askel

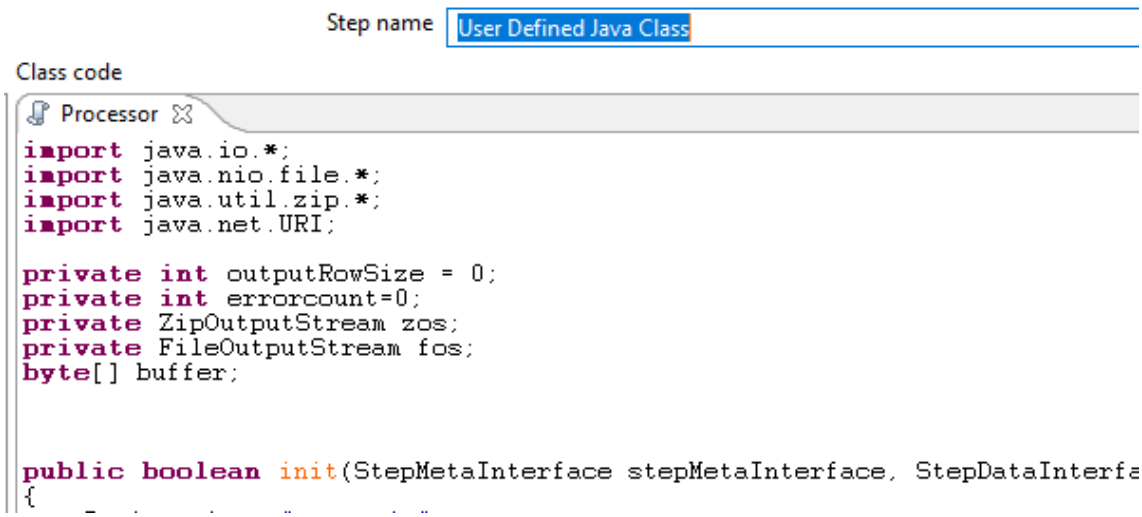
Vaikka Kettlessä on lukuisia valmiita askeleita datan käsittelyä varten, ne eivät aina kata kaikkia käyttötapauksia. Tätä varten Kettleen on rakennettu askelia, joissa ohjelmoija voi itse kirjoittaa koodia, jonka mukaan dataa käsitellään. User Defined Java Class -askeleella voidaan kirjoittaa omaa Java-koodia rivien käsittelyä varten. Kettle käyttää Java-askeleessa sisäisesti avukseen Janino Java-kääntäjää. Spoonin ohjelmointi-ikkunassa on piilotettu Janinon avulla osa luokkamäärityksestä, jotta ohjelmoija voi keskittyä keskeisten metodien implementointiin (32).

Kuvassa 16 on esillä Spooniin rakennettu Java-ohjelmointi-ikkuna. Ikkunan vasemmasta laidasta saa haettua erilaisia mallikoodinpätkiä. Kuvassa on haettu Main-koodi, joka luo processRow-metodille rungon.



Kuva 16. Java-askeleen ohjelmointi-ikkuna Spoonissa.

Koodin alussa voidaan normaalin Java-luokan tapaan alustaa luokkamuuttujia sekä määrittää luokan käyttämiä paketteja, kuten kuvan 17 koodissa on tehty.



Kuva 17. Java-askeleessa määritettyjä luokkamuuttujia tuotavia paketteja.

Koodi initialisoidaan aina transformaation alussa. Luokkaan voidaan kirjoittaa init()-metodi, jolla voidaan alustaa askeleessa käytettäviä muuttujia ja olioita. Init-metodia kutsutaan vain kerran transformaation initialisoinnin yhteydessä. Vastaavasti luokkaan voidaan kirjoittaa dispose()-metodi, jota kutsutaan kerran transformaation käsittelyn jälkeen. Dispose-metodilla voidaan esimerkiksi varmistaa, että luokassa avatut tietovirrat suljetaan.

Kuvan 18 esimerkissä avataan tietovirta test.zip-nimiseen zip-tiedostoon FileOutPutStream- ja ZipOutputStream-olioiden avulla. Oliot alustetaan luokkatasolla määritettyihin fos- ja zos-muuttujiin, kuten kuvassa 17 on esitelty. Näin muuttujia voidaan käyttää myös processRow()-metodin aikana, jossa varsinainen datan käsittely tapahtuu. Dispose-metodissa suljetaan ZipOutPutStream-olion tietovirta, joka delegoi tietovirran sulkemisen myös FileOutPutStream-oliolle.

```
public boolean init(StepMetaInterface stepMetaInterface, StepDataInterface stepDataInterface)
{
    String zip = "test.zip";
    fos = new FileOutputStream(zip);
    zos = new ZipOutputStream(fos);
    buffer = new byte[1024];
    return parent.initImpl(stepMetaInterface, stepDataInterface);
}
public void dispose(StepMetaInterface smi, StepDataInterface sdi) throws IOException
{
    zos.close();
    parent.disposeImpl(smi, sdi);
}
```

Kuva 18. Esimerkki init()- ja dispose()-metodien käytöstä.

Varsinainen rivien käsittely tapahtuu processRow()-metodissa. Kyseistä metodia kutsutaan jokaisen Java-askeleelle syötettävän rivin kohdalla. Käsiteltävä rivi haetaan Object-muotoiseen taulukkoon getRow()-metodilla. Tämä metodi palauttaa null-arvon, mikäli kaikki rivit on käsitelty, joten processRow-metodin alussa tulee tarkistaa, onko rivi olemassa. Tyhjän arvon ilmaantuessa tiedetään, että käsiteltäviä rivejä ei enää ole, joten askel merkataan käsitellyksi setOutputDone()-metodilla. Kettle kutsuu Java-askeleen processRow()-metodia silmukassa niin kauan, kun se palauttaa totuusarvon true, joten käsittelyn loppuessa tulee palauttaa false-arvo.

Jos riviin lisätään metodin aikana uusia kenttiä, ulos syötettävä rivi on muodostettava uudelleen halutun kokoisena createOutPutRow()-metodin avulla. Metodin ensimmäisenä parametrina annetaan rivi, josta data kopioidaan. Toisena parametrina annetaan uuden rivin koko. Jos uudet kentät on määritelty Spoonin fields-ikkunassa, koko voidaan ilmoittaa data.outputRowMeta.size()-metodikutsun avulla. Käsiteltävät kentät voidaan hakea kuvan 19 mukaiesti get()-metodia hyödyntämällä.

```
public boolean processRow(StepMetaInterface smi, StepDataInterface sdi) t
{
    Object[] r = getRow();
    if (r == null) {
        setOutputDone();
        return false;
    }

    if (first)
    {
        first=false;
    }

    Object[] outputRow = createOutputRow(r, data.outputRowMeta.size());


    String filename=get(Fields.In, "filename").getString(r);
    String path = get(Fields.In, "path").getString(r);
    String short_filename = get(Fields.In, "short_filename").getString(r);
```

Kuva 19. processRow-metodin alku

Kuvan 20 koodissa tiedosto kirjoitetaan aiemmin initialisoituun zip-tiedostoon. Lopuksi zip-tiedoston nimi viedään rivissä uuteen zipfilename-kenttään. Rivi kirjoitetaan seuraavaan hyppyyn putRow()-metodin avulla.

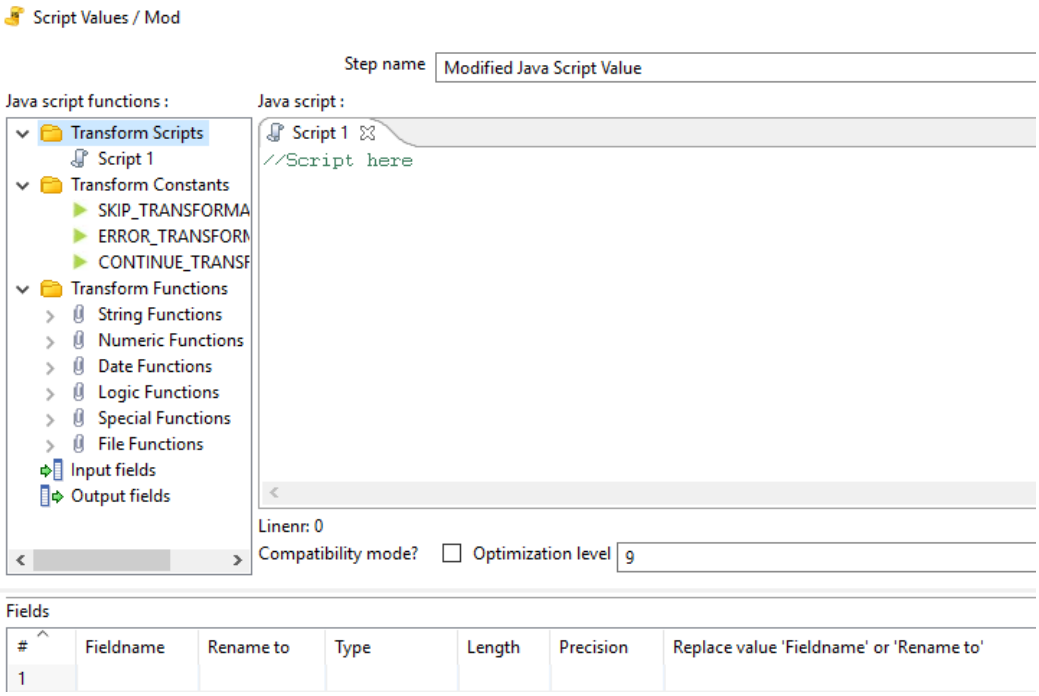
```
try {
    File file = new File(filename);

    FileInputStream fis = new FileInputStream(file);
    zos.putNextEntry(new ZipEntry(file.getName()));
    int length;
    while ((length = fis.read(buffer)) > 0) {
           zos.write(buffer, 0, length);
    }
    zos.closeEntry();
    fis.close();
    } catch (FileNotFoundException ex) {
        errorcount=errorcount+1;
        putError(data.outputRowMeta,r,errorcount,ex.getMessage(),"file not found","1");
    } catch (IOException ex) {
        errorcount=errorcount+1;
        putError(data.outputRowMeta,r,errorcount,ex.getMessage(),"IOException","2");
    }
get(Fields.Out, "zipfilename").setValue(outputRow,"test.zip");
// Send the row on to the next step.
putRow(data.outputRowMeta, outputRow);

return true;
```

Kuva 20. processRow()-metodi jatkuu. Metodissa kirjoitetaan init()-metodissa initialisoituun zip-tiedostoon filename-kentän mukainen tiedosto.

Askeleessa esiintyviä virheitä voidaan käsitellä putError()-metodin avulla. Ne rivit joille kutsutaan putError()-metodia päätyvät Java-askeleen jälkeen virhekäsittelyhyppyyn. Metodille annetaan parametreiksi käsiteltävän rivin metatiedot, virheiden lukumäärä, virheen aiheuttanut rivi sekä virheen kuvas, viesti ja koodi. Ohjelmoijan on itse pidettävä kirjaa virheiden lukumäärästä esimerkiksi luokkamuuttujan avulla, jota kasvatetaan havaittujen virheiden yhteydessä.

3.3.2 Modified JavaScript Value -askel

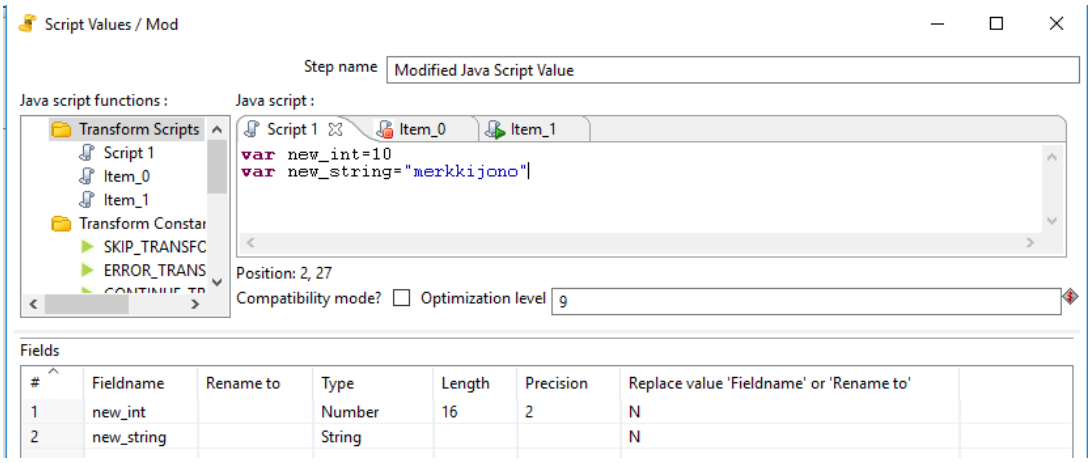
Modified JavaScript Value -askeleen avulla voidaan Java-askeleen tapaan kirjoittaa omaa koodia datan käsittelyä varten JavaScript-koodilla. Kettleen on integroitu Mozillan kehittämä avoimen lähdekoodin Rhino JavaScript-moottori (33), joka kääntää Kettlen JavaScript-askeleella kirjoitetun koodin Javan class-tiedostoiksi (34).



Kuva 21. Modified Java Script Value -askeleen ohjelmointi-ikkuna Spoonissa

JavaScript-askeleessa voidaan kirjoittaa kolmentyyppisiä skriptejä. Nämä ovat transform-, start- ja end-skriptit. Transform-skripti suoritetaan jokaiselle askeleen päätyvälle riville. Transform-skriptiin kannattaa siis kirjoittaa ainoastaan rivin käsittelyyn tarpeellinen koodi, jotta prosessointiaikaa säästyy. Esimerkiksi vakiomuuttujien alustukset kannattaa tehdä start-skriptissä. Start- ja end-skriptit suoritetaan nimiensä mukaisesti ainoastaan transformaation alussa ja lopussa.

Java-askeleen tapaan riviin vietävät uudet kentät tulee määrittää Spoonin Fields-ikkunaan. Spoon osaa tunnistaa askeleessa määritellyt uudet kentät Get Variables -toiminnolla, mikäli ne on määritetty skriptissä var-määreellä. Kuvan 22 esimerkissä on määritetty kaksi uutta kenttää, new_int ja new_string.



Kuva 22. JavaScript-askel, joka luo riviin kaksi uutta kenttää

3.3.3 Input- ja Output-askeleet

Luvussa 2.2. esiteltiin ETL-prosessin kuuluvaa kolmea eri osaprosessia, joista kaksi muodostuu datan hausta ja latauksesta. Tiedonsiirtomenetelmien konfigurointi on siis keskeinen osa ETL-prosessia. Kettle tarjoaa näiden osaprosessien rakentamisen helpottamiseksi useita valmiiksi määriteltyjä askeleita.

Datan hakuaskeleet on lajiteltu Spoonissa pääosin Input-askeleisiin ja latausaskeleet Output-askeleisiin. Taulukossa 4 on esitelty osa Input-askelista sekä niiden toiminnallisuuksia. Dataa voidaan hakea useaa eri protokollaa ja tiedostoformaattia käyttäen. Tuettuja formaatteja ovat esimerkiksi XML, JSON, CSV ja YAML. Dataa voidaan hakea mm. WSDL-tiedostolla määritellystä verkkopalvelusta, JSON-formaattia käyttävästä REST-palvelusta tai tietokannasta SQL-lauseita käyttäen. Taulukossa esiteltyjen askelien lisäksi Kettle tukee mm. datan lukua suoraan Excelistä, Google Analytics -sovelluksesta ja RSS-syötteestä. Lisäksi Kettleen on rakennettu Input-askelia helpottamaan datan hakua tietyistä yritysmaailmassa usein käytetyistä ohjelmistoista, kuten Salesforce-, SAP- ja SAS-ohjelmistoista.

Kettlestä löytyy vastaavasti myös samoja protokollia käyttäviä Output-askelia, joilla dataa voidaan ladata haluttuun kohteeseen.

Taulukko 4. Erilaisia Kettlen Input-askelia

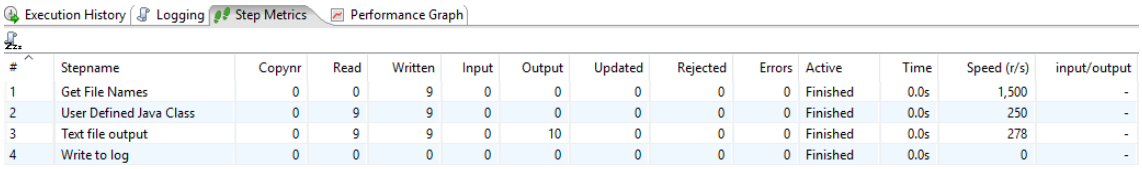
Askel	**Toiminto**
Text file input	Lukee dataa CSV-tiedostosta. Askeleelle määritetään mm. tiedostopolku ja käytettävä erotin.
Json Input	Lukee dataa JSON-muotoisesta lähteestä. Askel voi hakea dataa suoraan tiedostosta tai sille annettavasta URL-osoitteesta HTTP GET -pyynnöllä.
Get Data from XML	Lukee dataa XML-muotoisesta lähteestä. Askel voi hakea dataa suoraan tiedostosta tai sille annettavasta URL-osoitteesta HTTP GET -pyynnöllä.
Table Input	Lukee dataa SQL-tietokannan taulusta. Askeleessa konfiguroidaan tietokantayhteydelle tarvittavat parametrit sekä käytettävä SQL-lause, jolla tiedot haetaan.
LDAP input	Lukee dataa LDAP-lähteestä. Askeleelle määritetään käytettävä LDAP-haku lauseke sekä tarvittavat parametrit LDAP-yhteyden ottamiseen.

3.4 Kettle-prosessien suoritus ja lokitus

ETL-prosessien lokitus on tärkeää, kuten luvussa 2.2.5 esiteltiin. Prosessi halutaan myös useimmiten käynnistää hallitusti joko ajastusta käyttäen tai jonkun muun prosessikokonaisuuden yhteydessä. Kettle ei tue prosessien ajastusta. Ajastusta varten tulee käyttää jotain muuta ulkoista työkalua.

Kettle prosesseja voidaan suorittaa suoraan Kettlen käyttöliittymästä Spoonista. Lisäksi prosesseja voidaan suorittaa käyttämällä Pan- ja Kitchen-komentorivityökaluja. Kettle on myös mahdollista integroida Java-sovellukseen, jolloin prosessi voidaan käynnistää Kettlen tarjoamien Java-rajapintojen kautta.

Kettle tarjoaa kattavia lokitietoja töistä, transformaatioista sekä askelista. Kuvassa 23 on esitelty Spoonin tuottamia lokitietoja transformaation suorituksesta. Transformaation askelista saadaan tietoja mm. kirjoitettujen ja luettujen rivien lukumäärästä, sekä askelien käyttämästä suoritusajasta ja rivien käsittelyn nopeudesta. Lokitusta voidaan määrittää erikseen työ- ja transformaatiotasolla.


Execution History | Logging | Step Metrics | Performance Graph

#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active	Time	Speed (r/s)	input/output
1	Get File Names	0	0	9	0	0	0	0	0	Finished	0.0s	1,500	-
2	User Defined Java Class	0	9	9	0	0	0	0	0	Finished	0.0s	250	-
3	Text file output	0	9	9	0	10	0	0	0	Finished	0.0s	278	-
4	Write to log	0	0	0	0	0	0	0	0	Finished	0.0s	0	-

Kuva 23. Lokitietoja Kettlen käyttöliittymällä transformaation suorituksen jäljiltä

Lokitiedot voidaan ohjata tietokantaan transformaation tai työn asetuksissa. Yksinkertaisempi vaihtoehto on kirjoittaa lokitiedot erilliseen tiedostoon.

3.4.1 Pan ja Kitchen

Pan- ja Kitchen-komentorivityökalujen avulla voidaan suorittaa Kettle-prosesseja Spoonin ulkopuolella (35). Pan erikoistuu transformaatioiden suorittamiseen ja Kitchen vastaavasti töiden. Kettle-asennusten mukana tulee erilliset komentorivityökalut UNIX- ja Windows-ympäristöille.

Pan- ja Kitchen tukevat useita komentorivi-argumentteja. Prosessin suoritukseen ja lokitukseen liittyen tärkeimmät argumentit ovat file, logfile, ja param. File-argumentilla määrätään suoritettava transformaatio tai työ. File-argumentin tulee sisältää polku, johon XML-muotoinen työ tai transformaatio on säilötty. Logfile-argumentilla prosessin lokitiedot voidaan ohjata tiedostoon. Argumentissa annetaan kirjoitettavalle lokitiedostolle polku ja nimi. Param-argumentin avulla suoritettavalle prosessille voidaan syöttää ennalta määriteltyjä parametreja avain-arvo-parin muodossa. Transformaatiot ja työt voivat käyttää näitä parametreja Get Variables -askeleen avulla.

```
Pan.bat /file:C\Kettle\test1.xml /logfile:C:\Kettle\logtest1.log
Pan.sh -file=/opt/kettle/test1.xml -logfile=/opt/kettle/logtest1.log
```

Koodiesimerkissä on havainnollistettu Pan-työkalun käyttöä sekä Windows- että Unix-pohjaisissa ympäristöissä. Pan määrätään suorittamaan test1.xml-tiedoston mukainen transformaatio. Lokitiedot viedään logtest1.log-nimiseen tiedostoon. Lokituksen tasoa voidaan määrätä level-argumentilla. Kuvan 24 lokitiedot on muodostettu oletustasolla määrittämättä level-argumenttia.

```
2020/05/06 18:55:45 - Pan - Start of run.
2020/05/06 18:55:45 - test1 - Dispatching started for transformation [test1]
2020/05/06 18:55:46 - Get File Names.0 - Finished processing (I=0, O=0, R=0, W=9, U=0, E=0)
2020/05/06 18:55:46 - Text file output.0 - Finished processing (I=0, O=10, R=9, W=9, U=0, E=0)
2020/05/06 18:55:46 - User Defined Java Class.0 - Finished processing (I=0, O=0, R=9, W=9, U=0, E=0)
2020/05/06 18:55:46 - Pan - Finished!
2020/05/06 18:55:46 - Pan - Start=2020/05/06 18:55:45.703, Stop=2020/05/06 18:55:46.710
2020/05/06 18:55:46 - Pan - Processing ended after 1 seconds.
2020/05/06 18:55:46 - test1 -
2020/05/06 18:55:46 - test1 - Step Get File Names.0 ended successfully, processed 9 lines. ( 9 lines/s)
2020/05/06 18:55:46 - test1 - Step User Defined Java Class.0 ended successfully, processed 9 lines. ( 9 lines/s)
2020/05/06 18:55:46 - test1 - Step Text file output.0 ended successfully, processed 9 lines. ( 9 lines/s)
2020/05/06 18:55:46 - test1 - Step Write to log.0 ended successfully, processed 0 lines. ( 0 lines/s)
```

Kuva 24. Panin muodostaman lokitiedoston sisältöä

4 Yhteenveto

Insinöörityössä tutustuttiin ETL-prosesseihin ja Pentaho Data Integration (Kettle) -ETL-työkaluun. Työn tavoitteena oli tuottaa LTC-Otso Oy:lle kuvaus Kettlen toiminnallisuudesta, jotta tätä voidaan jatkossa hyödyntää ETL-prosessien muodostamisessa.

Työn ensimmäisessä osassa tutustuttiin ETL-prosessin käytänteisiin lähdekirjallisuutta tutkien. ETL-prosessia käsittelevässä osassa tutkittiin ETL:n eri osaprosesseja, näihin liittyviä ongelmia ja tyypillisesti käytettyjä teknologioita ja protokollia. Työn toisessa osassa tutustuttiin Kettlen ominaisuuksiin erilaisia käytännön esimerkkejä tarkastellen. Työssä kuvaillaan, miten Kettle-prosesseja voidaan muodostaa Kettlen käyttöliittymää Spoonia käyttäen. Lisäksi työssä kuvaillaan Kettle-prosessin suoritukseen liittyviä yksityiskohtia ja valmiita toiminnallisuuksia, joilla dataa voidaan käsitellä.

Työn edetessä selvisi, että Kettlessä on monipuolisesti valmiita ominaisuuksia datan hakua, muokkausta ja latausta varten. Toimintoja ja niiden välistä datavirtaa pystyy konfiguroimaan helposti Kettlen käyttöliittymää Spoonia käyttäen. Insinöörityön tuloksena saatiin riittävät tiedot Kettlen perustoiminnallisuuksista, jotta tällä voidaan muodostaa ja

suorittaa ETL-prosesseja. Työn tuloksena LTC-Otso Oy:ssa saatiin muodostettua liitteessä 1 kuvattu Kettle-prosessi. Rakennetun Kettle-prosessin dokumentaatiota voidaan hyödyntää jatkossa uusia prosesseja muodostaessa.

Lähteet

1 Beal, Vangie. ETL - Extract, Transform, Load. Verkkoaineisto. <https://www.webopedia.com/TERM/E/ETL.html>. Luettu 20.4.2020.

2 Agrawal, Nidhi. 2019. Role of ETL in Business Intelligence. Verkkoaineisto <https://www.mantralabsglobal.com/blog/etl-in-business-intelligence/>. 1.10.2019. Luettu 1.4.2020.

3 Forrest, Stroud. IAM – Identity and Access Management. Verkkoaineisto. <https://www.webopedia.com/TERM/I/iam-identity-and-access-management.html>. Luettu 1.4.2020.

4 Data Integration for Identity and Access Management (IAM). 2018. Verkkoaineisto. CloverDX. <https://www.cloverdx.com/blog/data-integration-for-identity-and-access-management-iam> 15.3.2018. Luettu 2.4.2020

5 Weinberg, Peter. 2020. 20 Great ETL Tools, And The Case For Saying "No" To ETL. Verkkoaineisto. <https://blog.panoply.io/17-great-etl-tools-and-the-case-for-saying-no-to-etl>. 9.1.2020. Luettu 5.4.2020.

6 XML ETL Processing. Verkkoaineisto. ETL-Tools.Info. <https://etl-tools.info/en/examples/xml-etl-processing.htm>. Luettu 5.4.2020.

7 CSV - Comma Separated Values. Verkkoaineisto. Datopian. <https://data-hub.io/docs/data-packages/csv>. Luettu 5.4.2020.

8 Introducing JSON. Verkkoaineisto. Json.org. <https://www.json.org/json-en.html> . Luettu 5.4.2020.

9 HTTP Definition. 2015. Verkkoaineisto. TechTerms. <https://techterms.com/definition/http>. 28.5.2015. Luettu 5.4.2020.

10 Russel, Aaron. 2019. What is SSL. Verkkoaineisto. <https://www.ssl.com/faqs/faq-what-is-ssl>. 2.10.2019. Luettu 5.4.2020.

11 Use FTP to transfer files. 2019. Verkkoainesto. Indiana University. <https://kb.iu.edu/d/aerg>. 18.6.2019. Luettu 6.4.2020.

12 SFTP – SSH Secure File Transfer Protocol. Verkkoaineisto. SSH.com. <https://www.ssh.com/ssh/sftp/>. Luettu 6.4.2020.

13 SOAP Web Services Tutorial: Simple Object Access Protocol Example. Verkkoaineisto. Guru99. <https://www.guru99.com/soap-simple-object-access-protocol.html> Luettu 6.4.2020.

14 Breaker, Doug. 2018. SOAP vs. REST APIs – Which Reigns Supreme? Verkkoaineisto. <https://blog.dreamfactory.com/soap-vs-rest-apis-understand-the-key-differences/>. 18.9.2018. Luettu 12.4.2020.

15 What is REST. Verkkoaineisto. Restfulapi.net. <https://restfulapi.net/> Luettu 12.4.2020.

16 How to Automate the ETL Process For Data From Magento & Google Analytics. 2020. Verkkoaineisto. Countants. <https://medium.com/datadriveninvestor/how-to-automate-the-etl-process-for-data-from-magento-google-analytics-72a6f8f0f6f5>. 1.14.2020. Luettu 7.4.2020.

17 ETL (Extract-Transform-Load). Verkkoaineisto. Dataintegration.info. <https://www.dataintegration.info/etl>. Luettu 1.4.2020.

18 Matt Casters, Roland Bouman & Jos van Dongen. 2010. Pentaho Kettle Solutions: Building Open Source ETL Solutions with Pentaho Data Integration. John Wiley & Sons, Incorporated. 28.9.2010 s. 11.

19 Data Warehouse Infrastructure: Full vs Incremental Loading in ETL. Verkkoaineisto. Panoply. <https://panoply.io/data-warehouse-guide/data-warehouse-etl/> Luettu 10.4.2020.

20 ETL Transform. Verkkoaineisto. Talend. https://www.stitchdata.com/etldatabase/etl-transform/. Luettu 10.4.2020.

21 Alvi, Iqbal. 2018. ETL vs. ELT: Transform First or Transform Later? Verkkoaineisto. <https://datawarehouseinfo.com/etl-vs-elt-transform-first-or-transform-later/>. 30.8.2018. Luettu 10.4.2020.

22 Alley, Garret. 2019. What is Data Validation? Verkkoaineisto. <https://www.alooma.com/blog/what-is-data-validation>. 24.1.2019. Luettu 11.4.2020.

23 Data quality ETL process. Verkkoaineisto. ETL-Tools.Info. <https://etl-tools.info/en/examples/data-quality.htm>. Luettu 10.4.2020.

24 Santosh. 2010. What Is Data Scheme. Verkkoaineisto. <https://www.eukhost.com/blog/webhosting/what-is-data-scheme/>. 27.6.2010. Luettu 12.4.2020.

25 Matt Casters, Roland Bouman & Jos van Dongen. 2010. Pentaho Kettle Solutions: Building Open Source ETL Solutions with Pentaho Data Integration. John Wiley & Sons, Incorporated. 28.9.2010 s. 168.

26 Mitchell, Tim. 2016. ETL Logging. Verkkoaineisto. <https://www.timmitchell.net/post/2016/03/14/etl-logging/>. 6.2.2016. Luettu 15.4.2020.

27 Pentaho Data Integration. Verkkoaineisto. Hitachi Vantara. <https://help.pentaho.com/Documentation/8.2/Products/Data_Integration>. Luettu 15.4.2020.

28 Ohjelmistolisenssi. 2012. Verkkoaineisto. Apache Software Foundation. <https://github.com/pentaho/pentaho-kettle/blob/master/LICENSE.txt>. 16.1.2012. Luettu 20.4.2020.

29 Introduction to Spoon. 2014. Verkkoaineisto. Hitachi Vantara. <https://wiki.pentaho.com/display/EAI/.01+Introduction+to+Spoon#id-.01IntroductiontoSpoon-UserInterfaceOverview>. 30.7.2014.

30 Create Jobs. Verkkoaineisto. Hitachi Vantara. <https://help.pentaho.com/Documentation/5.2/0J0/0C0/030>. Luettu 15.4.2020.

31 Pentaho Data Integration Steps. 2018. Verkkoaineisto. Hitachi Vantara. <https://wiki.pentaho.com/display/EAI/Pentaho+Data+Integration+Steps>. 4.12.2018. Luettu 20.4.2020.

32 User Defined Java Class. Verkkoaineisto. Hitachi Vantara. <https://help.pentaho.com/Documentation/8.2/Products/Data_Integration/Transformation_Step_Reference/User_Defined_Java_Class>. Luettu 30.4.2020.

33 Modified Java Script Value. Verkkoaineisto. Hitachi Vantara. <https://help.pentaho.com/Documentation/8.2/Products/Data_Integration/Transformation_Step_Reference/Modified_Java_Script_Value>. Luettu 30.4.2020.

34 Rhino overview. Verkkoaineisto. Mozilla Corporation. <https://developer.mozilla.org/en-US/docs/Mozilla/Projects/Rhino/Overview>. Luettu 30.4.2020.

35 Use the Pan and Kitchen Command Line Tools to Work with Transformations and Jobs. Verkkoaineisto. Hitachi Vantara. <https://help.pentaho.com/Documentation/7.0/0L0/0Y0/070>. Luettu 1.5.2020.

ETL-prosessin muodostaminen Kettlellä

Poistettu luottamuksellisena.