SQL Server Integration Services

-ETL-prosessien kehittäminen

Timo Kallio

Opinnäytetyö

Ammattikorkeakoulututkinto

Savonia
ammattikorkeakoulu

SAVONIA-AMMATTIKORKEAKOULU **OPINNÄYTETYÖ**
Tiivistelmä

Koulutusala Luonnontieteiden ala	
Koulutusohjelma Tietojenkäsittelyn koulutusohjelma	
Työn tekijä Timo Kallio	
Työn nimi SQL Server Integration Services -ETL-prosessien kehittäminen	
Päiväys 10.4.2012	Sivumäärä/Liitteet 101+2
Ohjaaja Jyrki Linja	
Toimeksiantaja/Yhteistyökumppani -	

Tiivistelmä

Tämän opinnäytetyön tarkoituksena oli tutkia erityyppisten tietolähteiden ja tievarastojen välisten tiedonsiirtojen toteuttamisessa käytettävää ETL-toteutusmallia ja sen eri vaiheita. Työn tarkoituksena oli tutkia myös, että minkä tyyppisiä tietokonesovelluksia ETL-rajapintojen kehittämisessä käytetään. Tähän liittyen työn tarkoituksena oli tutkia lisäksi, että miten SQL Server -tietokantaohjelmistoon liittyvän SQL Server Integration Services -ohjelmiston avulla luodaan ETL-toteutusmallin mukaisia tiedonsiirtorajapintoja.

Tämän opinnäytetyön teoriaosuudessa käytiin läpi ETL-toteutusmalliin liittyvät poiminta-, muunto- ja lataus-vaihe sekä selvitettiin ETL-tiedonsiirtojen toteutuksessa käytettävien tietokonesovellusten käyttötarkoitus, toimintalogiikka ja hyödyt. Työn käytännön osuus muodostui sen sijaan SSIS-ohjelmiston arkkitehtuuriin kuuluvien keskeisten tekijöiden läpikäynnistä. SSIS-ohjelmiston toimintaa tutkittiin lisäksi laajasti käytännön esimerkkien avulla. Tutkimuksen painopiste oli selkeästi SSIS-ympäristön teknisen toiminnan tutkimisessa.

Tutkimuksen avulla voitiin ottaa kantaa siihen, että miten SSIS-ohjelmisto soveltuu erityyppisten ETL-rajapintojen kehittämiseen huomioiden sekä yrityksen että teknisen näkökulman. Tutkimuksen avulla voitiin ottaa kantaa myös SSIS-ohjelmiston käytettävyyteen. Tällä tarkoitetaan esimerkiksi sitä, että kuinka helppoa SSIS-ohjelmiston avulla on luoda ETL-mallin mukaisia tiedonsiirtorajapintoja.

Avainsanat
SQL Server Integration Services, SSIS, ETL, SQL Server, Business Intelligence, BI

SAVONIA UNIVERSITY OF APPLIED SCIENCES **THESIS**
Abstract

Field of Study Natural Sciences			
Degree Programme Degree Programme in Computer Science			
Author Timo Kallio			
Title of Thesis Developing ETL Processes with SQL Server Integration Services			
Date	10.4.2012	Pages/Appendices	101+2
Supervisor Jyrki Linja			
Project/Partner -			

Abstract

The purpose of this thesis was to examine the different phases of ETL model which is used in the implementation of data transfers between different types of data sources and data warehouses. The purpose of this thesis was also to examine the characteristics of software that is used in the implementation of ETL processes. In this context the study concentrated on how to implement ETL data transfers using SQL Server Integration Services, which is a feature in the SQL Server database management system.

The theoretical part of this thesis examines the ETL model's extract, transform and load phases as well as the use, logic and advantages of software which is used in the implementation of ETL data transfers. The practical part of the study discusses the main aspects of SSIS architecture. The functionality of the SSIS platform was also examined extensively with the help of pratical examples. The focus in this study was distinctly on examing the functionality of the SSIS platform.

The results of this study made it possible to take a stand on how the SSIS platform is suited in developing different types of ETL data transfers from the view point of company deploying it and that of the technology. In addition to this, the study made it possible to assess the usability of the SSIS platform such as how easy it is to use the SSIS platform in developing ETL data transfers.

Keywords
SQL Server Integration Services, SSIS, ETL, SQL Server, Business Intelligence, BI

SISÄLTÖ

1 JOHDANTO

Yritysorganisaatioiden operatiivisissa tietojärjestelmissä ja tietovarastoissa olevan tiedon määrä on kasvanut tähän päivään mennessä merkittävästi. Näistä tietomassoista on muodostunut sen myötä tärkeä tekijä, jonka avulla yritys kykenee tekemään liiketoimintansa kasvattamiseen vaikuttavia tärkeitä päätöksiä. Yhä useammassa järjestelmäympäristössä hyödynnetään kuitenkin Microsoft:n kehittämiä teknologioita ja ohjelmistoja. Tästä herää kysymys, että minkä menetelmien ja tekniikoiden avulla näinä päivinä Microsoft-teknologioihin pohjautuvissa järjestelmäympäristöissä hallitaan yritysorganisaatioiden operatiivisten tietojärjestelmien ja tietovarastojen sisältämiä tietomassoja. Aihe on mielenkiintoinen, sillä Microsoft on kehittänyt tähän päivään mennessä useita tehokkaita teknologioita, joiden avulla yritysorganisaatioiden järjestelmäympäristöjen kehitykselle asetettavat vaatimukset voidaan täyttää. Tästä esimerkkinä voidaan mainita muun muassa .NET-ohjelmointiympäristö (eng. *.NET Framework*) ja *MS SQL Server* -tietokantaohjelmisto.

Tässä opinnäytetyössä tarkastellaan eri tyyppisten tietolähteiden ja tietovarastojen välisten tiedonsiirtorajapintojen toteutuksessa käytettävää *Extract - Transform - Load* -toteutusmallia (ts. *ETL*) ja sen eri vaiheita (kappale 2.1). ETL-mallin eri vaiheiden tarkastelun lisäksi tässä työssä tutkitaan, että minkä tyyppisten tietokonesovellusten avulla ETL-mallin mukaisia tiedonsiirtorajapintoja voidaan toteuttaa (kappale 2.2). Tähän liittyen tutkimus keskittyy selvittämään, että miten MS SQL Server -tietokantaohjelmistoon liittyvän *SQL Server Integration Services* -ympäristön (ts. *SSIS*) avulla luodaan ETL-mallin mukaisia tiedonsiirtorajapintoja (luvut 4 ja 5). Ennen SSIS-ympäristön teknisen toiminnan tutkimiseen siirtymistä tutkielmassa esitellään SSIS-ympäristön kehityskulku ja käyttötarkoitus (luku 3). Tämän tutkielman viimeisessä luvussa (luku 6) ennen pohdinta-lukua käydään läpi lisäksi muutamia eri vaihtoehtoja, joiden avulla SSIS-teknologian avulla luotuja ETL-tiedonsiirtorajapintoja on mahdollista testata.

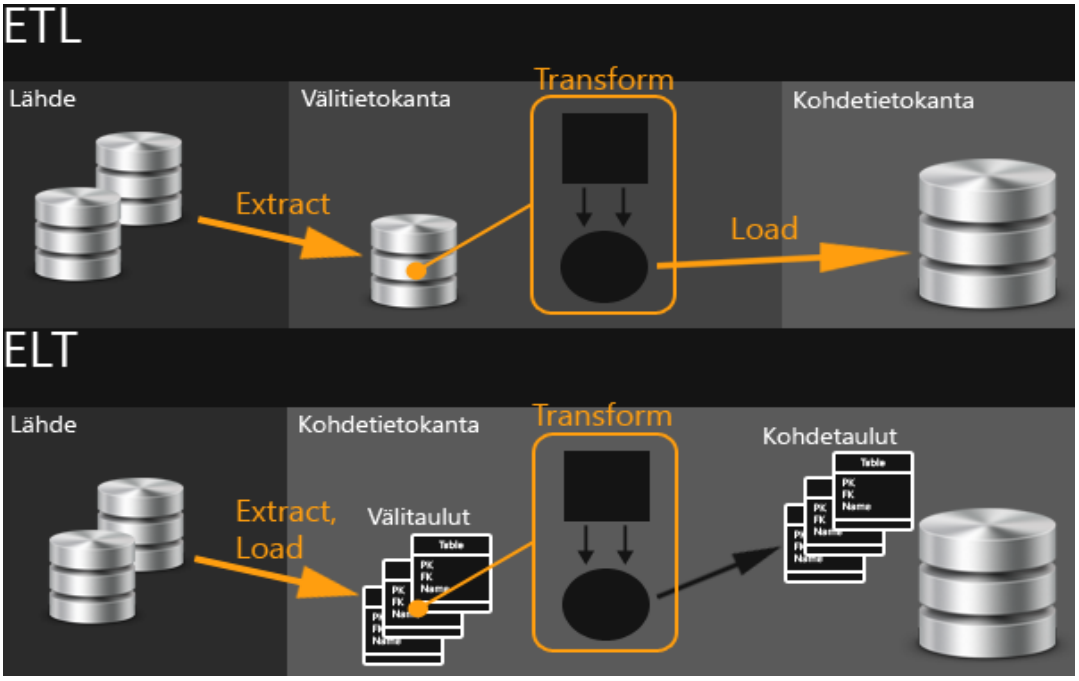
Tämän tutkimuksen päätavoitteena on selvittää, että miten SSIS-ympäristön avulla toteutetaan nykyaikaisia ETL-rajapintoja. Tutkimuksen avulla pyritään ottamaan kantaa myös siihen, että miten SSIS soveltuu ETL-rajapintojen toteutukseen, miten helppoa SSIS-teknologian avulla on luoda ETL-tiedonsiirtorajapintoja ja ylläpitää niitä niiden käyttöönoton jälkeen. Tutkimuksen pyrkimyksenä on myös löytää syitä sille, miksi yrityksen pitäisi valita SSIS ETL-rajapintojen kehitystekniikaksi.

ETL-toteutusmallia ja SSIS-ympäristön toimintaa tutkitaan tässä tutkielmassa eri tyyppisten kirjallisten teosten ja Internet-lähteiden avulla. Näiden lisäksi SSIS-ympäristön toiminnasta tutkittavia seikkoja havainnollistetaan käytännön esimerkkien avulla. SSIS-ympäristön toimintaa tutkitaan pääasiassa Microsoft:n SQL Server Integration Services -tuotedokumentaation avulla, joka on luettavissa Internet-sivustossa http://www.msdn.com.

2 ETL-PROSESSIN TARKASTELU

Extract - Transform - Load on yleisesti tietojärjestelmien välisissä tiedonsiirroissa käytetty toteutusmalli. ETL-mallin mukaisesti toteutettu tiedonsiirto käsittää kolme erillistä vaihetta, joissa tieto poimitaan sen lähteestä (eng. *extract*), muunnetaan kohde järjestelmän edellyttämään muotoon (eng. *transform*) ja lopuksi ladataan kohde järjestelmään (eng. *load*). (Hovi 2009.)

ETL-prosessista puhutaan yleisesti vain tietovarastoinnin yhteydessä. ETL-mallia voidaan kuitenkin hyödyntää myös esimerkiksi kahden operatiivisen tietojärjestelmän välisen tiedonsiirron toteutuksessa. Tässä tapauksessa ETL-malliin liittyvät vaiheet suoritetaan kuitenkin hyvin todennäköisesti eri järjestyksessä, jolloin voidaan puhua ELT-mallista. Tässä mallissa tieto poimitaan kohteena olevan järjestelmän tietokannan välitauluihin (eng. *staging table*), jonka jälkeen tieto muunnetaan oikeaan muotoon välitauluissa ennen sen siirtoa oikeisiin paikkoihin tietokannassa. Normaalisti ETL-prosessiin liittyy siis erillinen välitietokanta (eng. *staging area*), missä tieto muunnetaan oikeaan muotoon ennen sen lataamista kohdejärjestelmän tietokantaan. (Brobst & Pareek 2009, 4.) Kuviossa yksi on kuvattu ETL ja ELT prosessien vaiheita.



KUVIO 1. ETL- ja ELT-mallin vaiheiden kuvaus. On huomioitava, että ELT-mallissa muunto-vaihe kuluttaa tiedonsiirron kohdejärjestelmän tietokannan resursseja.

ETL-prosessi toimii aina itsenäisenä prosessina, jonka vuoksi tiedonsiirrossa käsiteltävien tietojärjestelmien käyttäjät eivät voi vaikuttaa tiedonsiirron kulkuun. Tämä tarkoittaa lisäksi sitä, että tietojärjestelmien käyttäjät eivät voi yleensä käsitellä sitä tietoa tiedonsiirron aikana, mitä ETL-prosessi käsittelee. (Karppinen 2007, 17.) Tiedonsiirrot, jotka on toteutettu ETL-mallin mukaisesti, ovat lisäksi tietyn logiikan mukaan toimivia, minkä vuoksi jokainen tiedonsiirtokerta suoritetaan aina identtisesti tiedonsiirtorajapintaan kehitetyn toiminnallisuuden mukaisesti (Hovi 2009).

ETL-prosesseissa käsitellään usein hyvin suuria tietomääriä, minkä vuoksi tiedonsiirrot varaavat merkittävän osan käsiteltävien tietojärjestelmien suorituskykyresursseista. Tämän vuoksi ETL-prosessin mukaan toteutetut tiedonsiirtorajapinnat toimivat yleensä eräajosovelluksina, jolloin varsinaiset tiedonsiirrot suoritetaan ajastetusti esimerkiksi yöaikaan. Suorituskykyä voidaankin pitää tärkeänä osa-alueena ETL-rajapinnan toteutuksessa, sillä yksittäiseen tietovarastoon kohdistuu usein yhden yön aikana useita erilaisia eräajoina toteutettuja tiedonsiirtoja. Suorituskyvyn ohella toinen kriittinen tekijä, joka tulee ottaa huomioon ETL-tiedonsiirron toteutuksessa, on virheiden käsittely. (Karppinen 2007, 18.) Oikein toteutetun virheiden hallinnan avulla varmistetaan tiedonsiirrossa käsiteltävän tietokannan tietojen eheys. Tämän lisäksi oikein toteutettu virheiden hallinta tehostaa virhetilanteiden selvittämistä ja niiden johdosta tiedonsiirron kohdetietokantaan aiheutuneiden ongelmien korjaamista.

2.1 ETL-prosessin vaiheet

2.1.1 Poiminta (Extract)

ETL-prosessin poimintavaihe on ETL-mallin ensimmäinen vaihe ja se käsittää tiedon siirtämisen tiedonsiirtorajapinnassa yhdestä tai useammasta erityyppisestä tietolähteestä ETL-järjestelmän välitietokantaan (Albrecht & Naumann 2008, 1). Tämän vaiheen toteutus voi olla hyvin yksinkertainen, jolloin ETL-rajapintaan poimittava tieto sijaitsee selkeässä muodossa esimerkiksi yhdessä tai useammassa siirtotiedostossa. ETL-rajapinnan toteutuksen näkökulmasta katsottuna siirtotiedostojen käyttöä tietolähteenä voidaan pitää suositeltavana. Tätä voidaan perustella sillä, että ETL-prosessin tiedon muuntaminen voidaan tehdä osittain jo vaiheessa, missä ETL-rajapinnan tietolähteenä käytettävä siirtotiedosto luodaan (Hovi 2009). Tämän lisäksi, kun poimintavaihe suoritetaan siirtotiedostojen avulla,

kuluttaa vaihe ainoastaan sen palvelimen suorituskykyresursseja, missä itse ETL-prosessi suoritetaan.

Mikäli poimittava tieto sijaitsee siirtotiedostojen sijasta esimerkiksi operatiivisessa tietojärjestelmässä, on sen siirtäminen ETL-rajapintaan usein haastavampaa. Operatiivisen tietojärjestelmän tietokannan rakenne on siis usein monimutkainen ja vaikeasti ymmärrettävä, jolloin ETL-rajapintaan poimittava tieto saattaa sijaita tietokannan monessa eri paikassa. On lisäksi hyvin yleistä, että itse ETL-rajapinnan kehittäjä ei tunne tiedonsiirron lähteenä käytettävän tietojärjestelmän rakennetta, jolloin poimintavaiheen toteutuksessa joudutaan käyttämään apuna lähdejärjestelmän teknisiä asiantuntijoita. (Hovi 2009.) ETL-tiedonsiirron poimintavaiheessa voidaan tässä tapauksessa käyttää apuna myös lähdejärjestelmässä olevaa tietokantanäkymää tai *Web Service* -rajapintaa, joiden avulla tieto välitetään lähdejärjestelmästä ETL-rajapintaan poimittavaksi.

Mikäli ETL-tiedonsiirtorajapinnan tiedon poimintavaihe kohdistetaan suoraan operatiiviseen tietojärjestelmään, tulisi vaiheen toteutuksessa pyrkiä aina siihen, että se kuluttaisi mahdollisimman vähän lähdejärjestelmän resursseja. Tehokkaan poimintavaiheen yhtenä perusmenetelmä on, että ETL-tiedonsiirtorajapintaan poimitaan lähdejärjestelmästä ainoastaan ne tiedot, mitkä ovat muuttuneet tietolähteenä käytettävässä järjestelmässä edellisen tiedonsiirtokerran jälkeen. ETL-prosessin poimintavaiheen muuttuneiden tietojen tunnistamis-menettelyä kutsutaan nimellä *delta tunnistus* (eng. *delta detection*). (Martin s.a. .)

2.1.2 Muunto (Transform)

Tiedon muunto-vaihetta voidaan pitää ETL-prosessin keskeisimpänä vaiheena, sillä tämä vaihe mahdollistaa käytännössä tiedon hyödyntämisen tiedonsiirron kohteena olevassa järjestelmässä. Tieto varastoidaan yleisesti tiedonsiirron lähde- ja kohdejärjestelmässä muodollisesti ja rakenteellisesti siis eri tavoin, jolloin rajapinnassa siirrettävä tieto tulee ladata kohdejärjestelmään sen tietokannan edellyttämän rakenteen mukaisesti.

ETL-prosessin muunto-vaiheelle on ominaista, että tieto muunnetaan kohdejärjestelmän edellyttämään muotoon käyttäen apuna ETL-kehitysympäristön valmiita komponentteja, jotka toteuttavat oman toimintansa mukaisen muunnon käsiteltävään tietoon. Vaihtoehtoisesti tiedon muuntaminen voidaan suorittaa myös itse luodun logiikan mukaisesti. (Henry, Hoon, Hwang, Lee & DeVore 2005, 4.) Tätä

vaihtoehtoa tulisi kuitenkin kyetä välttämään, sillä monimutkainen logiikka vaikeuttaa tiedonsiirtorajapinnan ylläpitoa. Mikäli rajapinnan toteutusta kaikesta huolimatta edellyttää omien logiikoiden hyödyntäminen tiedon muuntamisessa, on tämä logiikka järkevää koota omaksi erilliseksi komponentiksi rajapinnan muun toteutuksen ulkopuolelle esimerkiksi tietokanta-proseduuriin tai ohjelmakoodi-luokkakirjastoon.

ETL-prosessin muunto-vaiheessa käsiteltävään tietoon tyypillisesti kohdistettavat toimenpiteet voidaan luokitella selvästi niiden luonteen mukaisesti kolmeen eri ryhmään, joita ovat tarkastus, muuntaminen ja koonti. Seuraavissa luetteloissa annetaan esimerkkejä näihin ryhmiin kuuluvista toimenpiteistä, jotka Hovin (2009) mukaan lukeutuvat ETL-prosessin muunto-vaiheeseen.

Käsiteltävään tietoon kohdistuvia tarkastuksia ovat esimerkiksi:

- Tietojen duplikaatti-tarkastukset
- Pakollisten tietojen esiintyvyyden tarkastaminen
- Viite-eheys-tarkastusten suorittaminen.

Käsiteltävän tiedon muuntamiseen voidaan luokitella esimerkiksi:

- Syntymäaikojen muuntaminen sosiaaliturvatunnuksen alkuosan edellyttämään muotoon
- Valuutta-esitysten muuntaminen oikeaan muotoon
- Koodi-tietojen muuntaminen oikeaan muotoon, jolloin esimerkiksi sukupuoli tallennetaan lähdejärjestelmässä muodossa *M/F* ja kohdejärjestelmässä *M/N*.

Tiedon koonniksi voidaan luokitella esimerkiksi seuraavat toimenpiteet:

- Raportointia tehostavien luokittelutietojen muodostaminen erillisten selite-tietojen ja niihin liittyvien tunnistekoodien avulla
- Yksittäisen asiakastiedon koostaminen useammasta tietolähteestä tietyn logiikan mukaisesti, jolloin esimerkiksi puhelinnumero-tiedon puuttuessa tietolähteestä A sitä etsitään tietolähteestä B
- Surrogaattien (ts. *perusavain*) muodostaminen uusille tietokannan riveille.

2.1.3 Lataus (Load)

ETL-rajapinnassa käsiteltävän tiedon lataus tiedonsiirron kohdejärjestelmään muodostaa ETL-prosessin viimeisen vaiheen. Tässä vaiheessa kohdejärjestelmän tietokantaan ladataan tietoa, joka on validoitu ja muunnettu kohdejärjestelmän tietomallin mukaiseen muotoon (Albrecht & Naumann 2008, 1). ETL-prosessin lataus-

vaiheen tärkeimpiä huomionkohteita ovat latausprosessin suorituskyky ja sen järjestyksen huomioiminen, jota noudattaen tieto ladataan kohdejärjestelmän tietokannan eri tauluihin (Karppinen 2007, 22).

Lataus-vaihe voidaan suorittaa ETL-prosessissa käytännössä monella tapaa. Lataus-vaiheessa on mahdollista hyödyntää esimerkiksi SQL-kielen INSERT-lauseita, tiedonsiirrossa käytettävän tietokantaohjelmiston omia latausominaisuuksia tai hyödynnettävän ETL-kehitysohjelmiston sisältämiä ominaisuuksia (Hovi 2009). Valittaessa lataus-vaiheessa käytettävää tekniikkaa on huomioitava esimerkiksi, että kuinka suuria tietomääriä vaiheessa käsitellään tai missä muodossa ladattava tieto on. Tiedon muodolla viitataan tässä tapauksessa siihen, että mikäli tieto sisältää esimerkiksi ainoastaan selite- ja lukumäärätietoja, voidaan latauksessa käyttää apuna tehokkainta ja suoraviivaisinta lataustekniikkaa. Mikäli ladattava tieto käsittää sen sijaan runsaasti esimerkiksi viiteavainarvoja, tulee latausvaiheeseen sisällyttää myös logiikkaa, jonka myötä kaikkien lataustekniikoiden käyttäminen ei ole välttämättä mahdollista.

ETL-prosessin lataus-vaiheen toteutuksessa tulisi tavoitella sitä, että vaihe kuluttaisi mahdollisimman pienen osan tiedonsiirron kohdejärjestelmän resursseista. ETL-prosessin latausvaiheen suorituskyvyn kannalta oleellisena voidaan pitää sitä, että latausprosessissa pyritään esimerkiksi välttämään niiden toimenpiteiden suorittamista, joita siirrettävään tietoon tulee kohdistaa ETL-prosessin muuntovaiheessa. Latausvaiheen suorituskykyä voidaan parantaa myös optimoimalla tiedonsiirron kohdejärjestelmän tietokannan asetukset tiedonsiirtoa varten. Optimointia voidaan suorittaa esimerkiksi poistamalla indeksit tiedonsiirron ajaksi niistä kohdejärjestelmän tietokannan tauluista, joita ETL-prosessi käsittelee (Karppinen 2007, 22). Tämän lisäksi latausnopeutta voidaan tehostaa esimerkiksi optimoimalla kohdejärjestelmän tietokantamoottorin *transaction*-lokin toiminta (Dieter 2009). On huomioitava, että lyhentämällä ETL-prosessin lataus-vaiheen kestoa vaikutetaan myös tiedonsiirron ja kohdejärjestelmän välisen mahdollisen transaction-käsittelyn kestoon. Transaction-käsittelyn aikana kohdejärjestelmän tietokannan tiedot ovat siis usein lukittu ainoastaan latausta suorittavan prosessin käyttöön, jolloin muiden prosessien pääsyä tietokannan tietoihin edellyttää transaction-käsittelyä hallitsevan prosessin päättäminen.

ETL-prosessin lataus-vaiheessa ei ole yhdentekevää, että missä järjestyksessä rajapinnassa siirrettävä tieto ladataan kohdejärjestelmän tietokannan tauluihin. Tämä johtuu siitä, että relaatiotietokannoissa kahden taulun välille ei ole mahdollista luoda

eheää liitosta, mikäli tietoa, mihin toisesta taulusta halutaan viitata, ei ole vielä lisätty toiseen tauluun. Tietovarastojärjestelmissä ETL-prosessin lataus-vaihe lähtee liikkeelle siitä, että siirrettävä tieto ladataan ensimmäisenä niin sanottuihin ali-dimensio-tauluihin. Tämän jälkeen tieto ladataan dimensio-tauluihin, joihin tallennetaan myös viiteavain (eng. *foreign key*), jonka avulla on tarkoitus viitata ali-dimensio-taulussa olevaan tietoon. ETL-prosessin latausvaiheessa viimeisenä siirrettävä tieto ladataan faktatauluhin (eng. *fact table*), sillä näistä tauluista on tarkoitus viitata dimensio-tauluissa oleviin tietoihin. (Karppinen 2007, 22.) Operatiivisissa tietojärjestelmissä ETL-prosessin latausvaiheen tietojen latausjärjestys noudattaa samoja perus periaatteita kuin mitä tietovarastojärjestelmään tietoa ladatessa noudatetaan. Tällöin siis esimerkiksi luokittelutiedot ladataan ensimmäisenä kohdejärjestelmän tietokantaa. Tämän jälkeen tietokantaan ladataan tiedot, joihin on tarkoitus tallentaa tarvittaessa myös luokittelutietoihin viittaava viiteavain.

2.2 ETL-välineet

ETL-välineiksi voidaan kutsua sovelluksia, joiden avulla kehitetään ETL-mallin mukaisia tiedonsiirtorajapintoja tai muun kaltaisia tiedonsiirtoja (ts. *integraatioita*). ETL-kehitysympäristöjen perusominaisuuksiin lukeutuu yleisesti valmiita palveluita, joita tarvitaan tyypillisesti ETL-rajapinnan eri vaiheiden kehityksessä. (Hovi 2009.) On siis huomattava, että ETL-kehitysympäristöjen yhtenä päätarkoituksena on vähentää suoranaisen ohjelmoinnin tarvetta tiedonsiirtorajapintojen kehityksessä (Albrecht & Naumann 2008, 1). Ohjelmoinnilla tarkoitetaan tässä yhteydessä SQL-skriptien tai jonkin ohjelmointikielen, kuten esimerkiksi *.NET*-pohjaisen kielen, käyttöä.

ETL-välineiden perus toimintalogiikka muodostuu yleisesti graafisesta käyttöliittymästä, missä käsitellään tietyn mukaisen toiminnallisuuden omaavia tehtäväkomponentteja (Henry & ym. 2005, 2.). Tiedonsiirtorajapinnan toteutus muodostuu siis erilaisista tehtäväkomponenteista, jotka konfiguroidaan toimimaan halutulla tavalla komponenttien sisältämien ominaisuuksien mukaisesti. Kehitettävään tiedonsiirtorajapintaan lisättävien tehtäväkomponenttien suoritusjärjestys määrittää tiedonsiirron etenemisen ETL-kanavaa (eng. *ETL pipeline*) pitkin.

Nykyaikaiset ETL-välineet ovat siirtyneet perinteisten eräajopohjaisten tiedonsiirtojen kehittämisen tukemisesta enemmän kohti reaaliaikaisten integraatioiden toteuttamista. ETL-kehitysympäristöihin on alettu kehittämään yhä enemmän myös

ominaisuuksia, joiden avulla rajapinnassa siirtyvää tietoa pystytään monitoroimaan ja profiloimaan tehokkaammin. (Hovi 2009.) Näiden lisäksi merkittävä tekijä nykyaikaisten ETL-välineiden toiminnassa on metatieto, joka määrittelee esimerkiksi rajapinnassa siirrettävän tiedon rakenteen sen lähteessä ja kohteessa.

Miksi kehittäjien pitäisi sitten käyttää ETL-välinettä tiedonsiirtorajapinnan kehittämisessä? Nopeasti ajateltuna tiedonsiirrot ovat kuitenkin melko yksinkertaisia toteuttaa esimerkiksi tietokantaproseduurien ja erillisen ohjelmointikielen avulla. Hyödyntämällä tiettyä ETL-kehitysympäristöä kehitystyössä yrityksessä toimivat kehittäjät saavat siis esimerkiksi yhtenäisen tavan tiedonsiirtorajapintojen luontiin. Tämän voidaan katsoa tehostavan esimerkiksi aiemmin luotujen ETL-prosessien tai kehityksessä käytettyjen toimintatapojen soveltamista uusiin kehitysprojekteihin. (Albrecht & Naumann 2008, 1.) Yhtenäisen kehitystekniikan käytön lisäksi tiedonsiirtorajapintojen kehittäjät hyötyvät ETL-kehitysvälineiden yleisestä toimintalogiikasta. ETL-kehitysympäristöissä kehitystyö tehdään siis graafisen käyttöliittymän kautta, johon lisättävät tehtäväkomponentit kätkevät toteutuksen taustalle kaiken ohjelmakoodin, jonka mukaan tiedonsiirtorajapinta toimii. Tämä mahdollistaa esimerkiksi tietyn tiedonsiirtorajapinnan tehokkaamman takaisinmallinnuksen, sillä rajapinnan toteutusta on helpompaa tulkita visuaalisen kuvauksen kautta kuin lukemalla suoraan ohjelmakoodia. Myös tämä tehostaa osaltaan aiemmin luotujen ETL-prosessien hyödyntämistä uusissa tiedonsiirtorajapintojen kehitysprojekteissa. (Krudop 2007.)

Toinen kysymys, joka nousee voimakkaasti esille ETL-välineiden käytöstä tiedonsiirtorajapintojen kehityksessä, liittyy yrityksessä hyödynnettävän ETL-kehitysympäristön valintaan. Miksi yksittäinen yritys siis valitsisi valmiin ETL-välineen kehitystyön tueksi, kun tämän kaltaisen ohjelmiston kehittäminen myös itse on suhteellisen vaivatonta? Yhtenä merkittävänä syynä valmiin ETL-välineen käyttöön voidaan pitää sitä, että itse tehdyn ETL-välineen avulla ei pystytä vastaamaan erityyppisiin vaatimuksiin, joita kehitettäville tiedonsiirtorajapinnoille nykyään asetetaan. Mikäli yrityksessä hyödynnetään siis itse kehitettyä ETL-ohjelmistoa, on hyvin todennäköistä, että tämän ohjelmiston rinnalla käytetään hyväksi myös jotain toista ETL-kehitysympäristöä. Toinen merkittävä syy valmiin ETL-kehitysympäristön käytölle löytyy nykyaikaisten tiedonsiirtojen suorituskykyvaatimuksista. Nykyaikaiset ETL-kehitysympäristöt ovat siis optimoituja suurten tiedonsiirtojen hallintaan esimerkiksi moniajon (eng. *multi threading*) ja edistyneen muistin käytön avulla. (Mimno 2001, 1.) Näiden ominaisuuksien hyödyntäminen nousee siis merkittävään rooliin erityisesti tietovarastoihin kohdistettavissa tiedonsiirroissa. Myös

reaaliaikaisesti suoritettavat tiedonsiirrot asettavat huomattavia suorituskykyvaatimuksia kehitettäville tiedonsiirtorajapinnoille. Siirrettäessä tietoa reaaliaikaisesti on siis huomioitava, että tiedonsiirtoprosessin käytettävissä olevat suorituskykyresurssit saattavat olla hyvin pienet johtuen tiedonsiirrossa käsiteltävän tietojärjestelmän päivittäisestä käytöstä samanaikaisesti.

3 SQL SERVER INTEGRATION SERVICES -YMPÄRISTÖN ESITTELY

3.1 SSIS-ympäristön kehityskulku

3.1.1 Data Transformation Services (DTS)

Microsoftin kehittämän SQL Server Integration Services -ympäristön perus toimintalogiikka pohjautuu vahvasti *Data Transformation Services* (ts. *DTS*) -nimisen palvelun toimintaan, jonka Microsoft kehitti SQL Server 7.0 -ohjelmiston yhteyteen (Krueger 2010). DTS oli aikanaan tiedonsiirron työväline, jonka avulla oli tarkoitus ratkaista ongelmia, jotka syntyvät, kun tietoa halutaan siirtää erityyppisten tietolähteiden, kuten esimerkiksi tietokantojen tai tietokantojen ja tiedostojen, välillä. DTS hyödynsi *OLE DB* -standardin mukaista tiedonsiirtorajapintaa yhteyden luonnissa tietolähteiden välille. Tämän lisäksi kytkeytyminen *ODBC*-standardin mukaisiin tietolähteisiin oli mahdollista OLE DB -liittymän välityksellä. (Larsen & Garden 2000.) Tästä voidaan siis ymmärtää, että DTS mahdollisti tiedonsiirron esimerkiksi Excel, Access, SQL Server ja Oracle ohjelmistojen välillä.

DTS-ympäristön avulla luodut tiedonsiirtorajapinnat koostuivat erityyppisistä tehtäväkomponenteista (eng. t*ask*), joita voitiin liittää toisiinsa. Tehtävien välisiin liitoksiin voitiin luoda lisäksi tiettyjä ehtoja (eng. *constraint*), joiden avulla tiedonsiirtoon oli mahdollista luoda halutun mukainen suorituslogiikka. DTS-ympäristön valmiita komponentteja hyödyntämällä rajapinnassa oli mahdollista toteuttaa esimerkiksi tiedon muodon muuntamista (eng. *transformation*), SQL-kyselyiden suorittamista ja erityyppisiin tietolähteisiin kytkeytymistä. Näiden lisäksi DTS-ympäristön perus palveluihin kuuluivat tiedonsiirron seuranta lokien avulla, transaction-käsittely ja globaalien muuttujien hyödyntäminen tiedonsiirron kulussa. (Larsen & Garden 2000.)

DTS oli itsessään tehokas tiedonsiirron työväline, jonka avulla oli mahdollista luoda monimutkaisiakin liittymiä tiedonsiirrossa käsiteltävien järjestelmien välille. Tästä huolimatta tiettyjen ETL-prosesseille tyypillisten peruspalveluiden puuttuminen hankaloitti merkittävästi DTS-ympäristön käyttöä kokoluokaltaan suurissa tiedonsiirroissa. Puutteiden aiheuttamat ongelmat pyrittiin ratkaisemaan kehittämällä rajapinnan yhteyteen räätälöityä toiminnallisuutta suoraan erillisen ohjelmakoodin avulla (Knight, Veerman, Dickinson, Hinson & Herbold 2008, 2). Monimutkaisen ohjelmakoodin kirjoittaminen hankaloittaa luonnollisesti rajapinnan ylläpitoa ja

pahimmassa tapauksessa ylläpidossa voidaan päätyä tilanteeseen, missä ainoastaan tiedonsiirtorajapinnan kehittäjä on kykenevä ylläpitämään luotua rajapintaa. DTS ei siis ollut aikanaan missään nimessä täysin valmis ja kaikkeen kykenevä tiedonsiirron työväline.

3.1.2 SQL Server Integration Services (SSIS)

SQL Server Integration Services -nimeä kantavaa palvelukokonaisuutta voidaan pitää vielä melko nuorena ohjelmistokehityksen teknologiana. SSIS-ympäristön ensimmäinen versio *SQL Server 2005 Integration Services* julkaistiin nimensä mukaisesti SQL Server 2005 -ohjelmiston yhteyteen. SQL Server 2008 -ohjelmiston yhteydessä julkaistiin SSIS-ympäristön toinen versio *SQL Server 2008 Integration Services*, joka on SSIS-ympäristön viimeisin julkaisu. (Knight ym. 2008, 2.)

Vaikka voidaan ymmärtää, että SSIS-ympäristön perus toimintalogiikka pohjautuu DTS-ympäristön toimintaan, niin on silti väärin ajatella, että SSIS pohjautuisi teknillisen toteutuksensa puolesta DTS-ympäristöön (Arshad 2009). SSIS on siis täysin uusi tiedonsiirron työväline, jonka tavoitteena on tarjota tiedonsiirtoon nykyaikaiset ETL-työvälineet. Tarkemmin tarkasteltuna SSIS-ympäristö tarjoaa suoraan rajapintojen kehitystyössä käytettäväksi valmiita ETL-pohjaisia tehtäväkomponentteja, joiden avulla on mahdollista esimerkiksi hallita serverin tiedostojärjestelmässä olevia tiedostoja, poimia tietoa Web Service -rajapintojen kautta, käsitellä XML-muodossa olevaa tietoa, muodostaa tiedosta analyysiraportteja (eng. *data profiling*) tai suorittaa tiedonlouhintaa (eng. d*ata mining*) (Knight ym. 2008, 8). DTS-ympäristössä edellä mainittujen tehtävien suorittaminen vaati lähes poikkeuksetta erillisen ohjelmakoodin kirjoittamista.

Kuten DTS-ympäristössä, myös SSIS-ympäristössä, kehitystyö tehdään graafisen käyttöliittymän kautta. Suurin ja merkittävin eroavaisuus DTS ja SSIS ympäristöjen käyttöliittymien välillä on, että SSIS-ympäristössä kehitystyötä pystytään tekemään kehitysympäristön käyttöliittymän erillisillä pinnoilla (eng. *design surface*). Käytännössä siis SSIS-ympäristön käyttöliittymä jakautuu kolmeen erilliseen pintaan, joissa voidaan hallita rajapinnan suorituksen kulkua (eng. *control flow*), tiedonsiirtoa (eng. *data flow*) ja tapahtumia (eng. *event handlers*). Tämä erittely voidaan huomata osittain myös SSIS-ympäristön teknisen toteutuksen arkkitehtuurissa. SSIS-rajapinnan suorituksen kulkua ja tiedonsiirtoa hallitaan siis toisistaan erillään toimivien ajonaikaisenmoottorin (eng. *run-time engine*) ja tiedonsiirtoa hallitsevan moottorin (eng. *data flow engine*) avulla. Tämä jaottelu mahdollistaa tehokkaammin

esimerkiksi SSIS-ympäristön laajennettavuuden itse luotavien tehtäväkomponenttien avulla. Tämän lisäksi toisistaan erillään toimivien moottoreiden avulla saavutetaan korkeampi suorituskyky tiedonsiirtoon. (Microsoft 2005.)

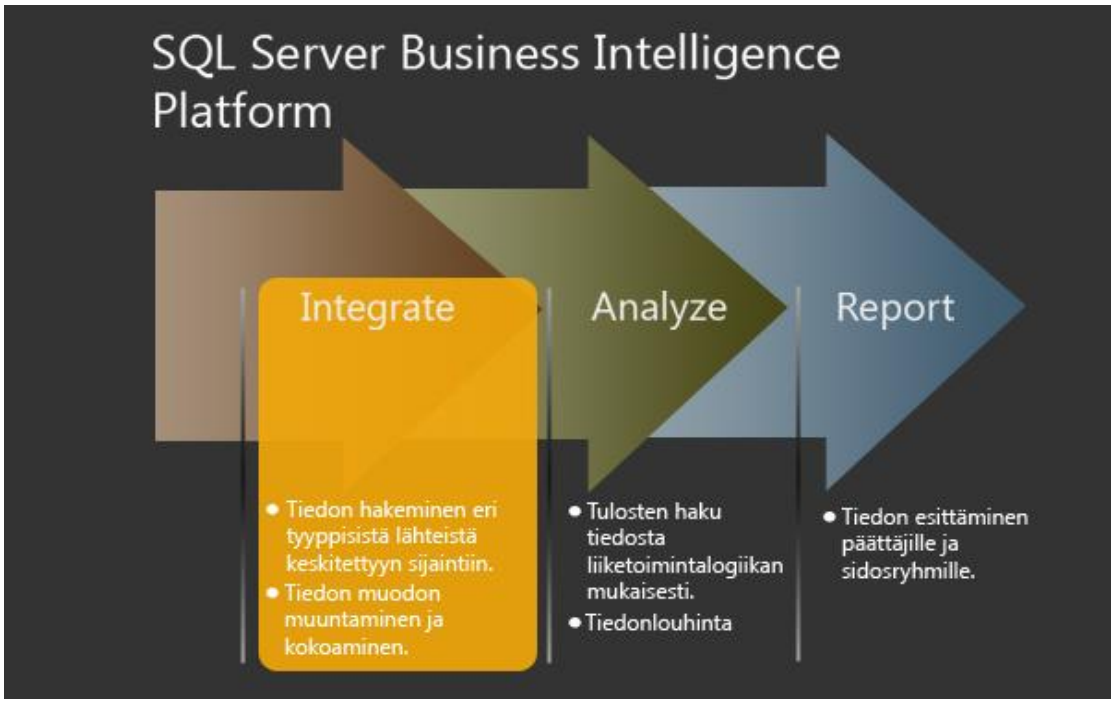
SSIS 2005 - ja myöhemmin SSIS 2008 -ympäristön julkaisun myötä Microsoft on pystynyt korjaamaan aiemmin DTS-ympäristön käytössä todettuja puutteita. SSIS on toteutettu lisäksi täysin omana projektinaan erillään DTS-ympäristön teknisestä toteutuksesta, minkä johdosta DTS-teknologiaa voidaan pitää vanhentuneena. SSIS 2008 -ympäristö tarjoaa kuitenkin vielä tuen esimerkiksi DTS-teknologialla luotujen rajapintojen suorittamiseen SSIS-ympäristön sisällä. Tämän lisäksi SSIS-ympäristö mahdollistaa DTS-rajapintojen muuntamisen muotoon, missä rajapinnan teknistä toteutusta voidaan hallita SSIS-ympäristössä. Tulevaisuudessa SQL Server, ja siten myös SSIS, eivät tule enää tukemaan DTS-teknologian käyttöä. (Knight ym. 2008, 5, 609.)

3.2 SSIS Microsoft BI -arkkitehtuurissa

Business Intelligence (ts. *BI*) voidaan ymmärtää kokoelmana erilaisia keinoja, joiden avulla pyritään hallitsemaan liiketoimintaan liittyvää tietoa. Toisin sanottuna BI:n tavoitteena on muuntaa liiketoiminnan tuottama tieto muotoon, missä sitä voidaan hyödyntää esimerkiksi liiketoiminnan kehittämisessä. Informaatioteknologian näkökulmasta katsottuna BI tarkoittaa joukkoa erilaisia teknologioita, joita hyödyntämällä tietojärjestelmien loppukäyttäjät pystyvät saattamaan tiedon liiketoimintansa sisällä tietovarastoista tai operatiitivista tietojärjestelmistä oikeiden henkilöiden saataville oikeassa muodossa. Yleisimmät informaatioteknologian tarjoamat BI-ratkaisut liittyvät tiedon raportointiin, analysointiin ja integrointiin. (Hovi 2009.) Myös nykyajan web-portaalit, missä tietoa jaetaan loppukäyttäjien kesken, ja perinteiset toimisto-ohjelmistot voidaan yhdistää BI-ratkaisuihin (Johansson & Hotti 2010). Microsoft:n tarjoamien BI-tuotteiden näkökulmasta katsottuna tiedon analysointi ja raportointi suoritetaan *SQL Server Analysis Services* ja *SQL Server Reporting Services* ohjelmistojen avulla. Web-portaalien ilmentymänä on sen sijaan *Sharepoint*-teknologia ja toimisto-ohjelmistojen *Microsoft Office* -tuoteperhe.

SQL Server 2005 -ohjelmiston ilmestymisen myötä Microsoft julkaisi sarjan uusia BI-tuotteita. Näiden tuotteiden avulla Microsoft pyrki tarjoamaan yritysorganisaatioille ohjelmistot, joiden avulla organisaatiot pystyvät hallitsemaan liiketoimintaansa liittyvää tietoa toiminnan jokaisella osa-alueella. SQL Server 2005 Integration

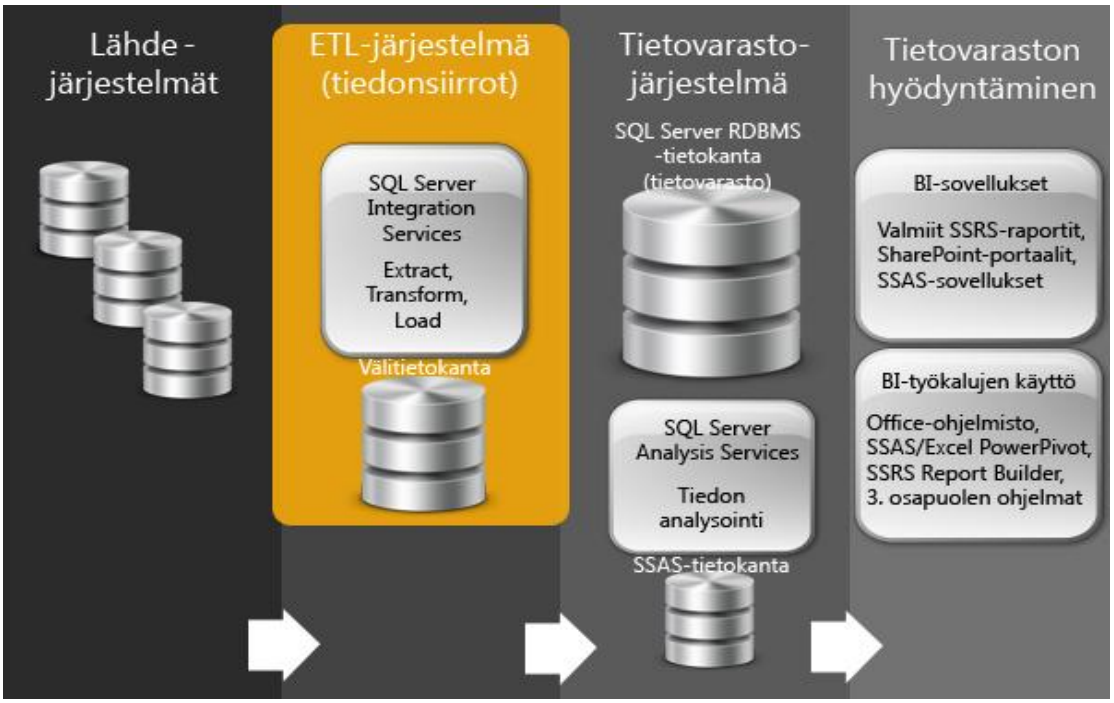
Services -ohjelmisto muodosti yhden kokonaisuuden Microsoft:n uudesta BI-tuoteperheestä. Tämän kokonaisuuden avulla loppukäyttäjille tarjottiin mahdollisuus organisaatiossa olevan tiedon integrointiin. Tarve, mikä tällä pyrittiin siis täyttämään oli, että organisaatiot pystyisivät kokoamaan liiketoimintansa sisällä olevan tiedon yhteen keskitettyyn sijaintiin sen analysointia ja raportointia varten. (Payne 2005, 1.) Tieto, mikä sijaitsee organisaation sisällä ja ulkopuolella, on siis levittäytynyt usein hajalleen hyvin laajalle alueelle. Tehokkaiden analyysien ja raporttien muodostamista edellyttää luonnollisesti tiedon käsittely yhdessä kokonaisuudessa. Kuviossa kaksi havainnollistetaan SQL Server 2005 -ohjelmiston BI-mallin eri vaiheita ja sitä, mihin niissä pyritään.



KUVIO 2. SQL Server 2005 -ohjelmiston BI-mallin vaiheet. Integrointi on ensimmäinen vaihe, kun tietoa jalostetaan SQL Server 2005 -ohjelmiston BI-mallin mukaisesti. (Payne 2005, 2.)

Nykyajan BI-trendi on siirtymässä suuntaan, missä BI:n tulisi koskettaa yhä suurempaa kohderyhmää yritysorganisaation sisällä. BI-ratkaisujen ei tule siis enää vastata ainoastaan liiketoiminnan johdon asettamiin vaatimuksiin, vaan myös esimerkiksi yksittäisellä työntekijällä on tarve päästä käsiksi jalostettuun tietoon. Yksittäistä työntekijää saattaisi kiinnostaa tuloslukujen muodossa esimerkiksi se, miten hän menestyy työssään. (Johansson & Hotti 2010.) Tästä voidaan ymmärtää, että liiketoiminnan sisällä käsiteltävän tiedon määrä voi muodostua nykyään hyvin suureksi kokoluokaltaan laajoissa organisaatioissa, jolloin teknisestä näkökulmasta katsottuna SSIS-ympäristön tulee kyetä toimimaan myös laajojen tietovarastojen

yhteydessä. Kuviossa kolme on esitetty SSIS-ympäristön sijoittuminen Microsoft:n BI-tietovarasto-järjestelmän (eng. *Microsoft Data Warehouse / Business Intelligence system*) arkkitehtuurissa. Mikäli tietovarastojärjestelmän arkkitehtuuria tarkastellaan tarkemmin, voidaan huomata, että sen BI-malli on perusrakenteeltaan sama mitä SQL Server 2005 -ohjelmiston BI-malli. Microsoft:n tietovarastojärjestelmän BI-mallista voidaan huomata siis selkeästi integrointi, analysointi ja raportointi vaiheet mainitussa järjestyksessä.



KUVIO 3. Mukailtu malli Microsoft DW/BI -järjestelmän arkkitehtuurista. Tiedon analysointia ja hyödyntämistä edellyttää, että tieto kootaan lähdejärjestelmistä tietovarastoon. Microsoft DW/BI-arkkitehtuurissa ETL-prosessit suoritetaan SSIS-ohjelmiston avulla. (Mundy, Thornthwaite & Kimball 2011, 83.)

4 SQL SERVER INTEGRATION SERVICES -YMPÄRISTÖN TARKASTELU

4.1 SSIS-arkkitehtuurin keskeiset osat

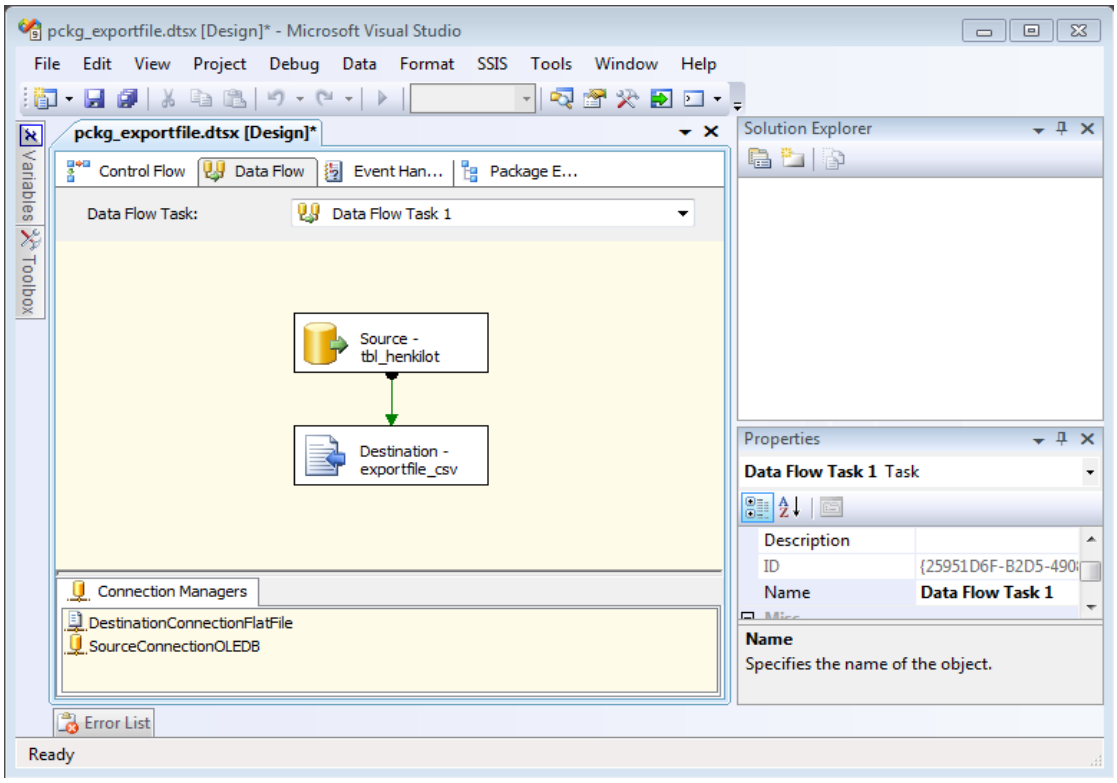
4.1.1 Package

Paketti (eng. *package*) on SSIS-ympäristön keskeisin käsite. SSIS-paketti voidaan ymmärtää itsenäisenä sovelluksena, joka muodostuu toisiinsa erityisillä *ehto-liitoksilla* (eng. *precedence constraint*) liitetyistä tehtäväkomponenteista. Tehtäväkomponenttien toiminnan luonne ja suoritusjärjestys määräävät SSIS-paketin toimintalogiikan perustasolla. Yksittäinen SSIS-paketti muodostuu fyysisesti yhdestä *DTSX*-tiedostopäätteisestä tiedostosta, joka sisältää XML-muodossa olevaa tietoa. (Knight ym. 2008, 7.)

SSIS-paketti, jonka sisältämän toiminnallisuuden mukaisesti esimerkiksi tietojärjestelmien välinen tiedonsiirto suoritetaan, luodaan BIDS-kehitysympäristön (eng. *Business Intelligence Development Studio*) tai *SQL Server Import And Export Wizard* -nimisen sovelluksen avulla. SQL Server Import And Export Wizard -tiedonsiirtotyökalun toiminta pohjautuu SSIS-teknologiaan ja se on saatavilla suoraan kaikissa SQL Server 2008 R2 -ohjelmiston versioissa. (Microsoft s.a.; Knight ym. 2008, 21.) Kuvassa yksi havainnollistetaan SQL Server Import And Export Wizard -sovelluksen avulla luodun SSIS-paketin sisältämää XML-tietoa ja kuvassa kaksi paketin rakennetta BIDS-kehitysympäristössä. SSIS-paketin, jonka sisältöä havainnollistetaan kuvissa yksi ja kaksi, avulla pystytään luomaan *CSV*-tyyppinen siirtotiedosto SQL Server -tietokannassa olevan taulun sisältämästä tiedosta. Vaikka kuvissa yksi ja kaksi havainnollistettu SSIS-paketti on hyvin yksinkertainen, käsittää se silti kokonaisuudessaan 252 riviä XML-tietoa. Kuvasta yksi voidaan lisäksi huomata, että esimerkiksi SSIS-pakettiin määritellyn tiedonsiirron lähdetietokannan ja kohteena olevan CSV-siirtotiedoston *connection string* -merkkijonot on mahdollista löytää XML-tiedon joukosta. Tämä on syytä huomioida, mikäli connection string -merkkijonot käsittävät tietoturvan kannalta arkaluotoista tietoa, kuten esimerkiksi tietokantojen käyttäjätunnuksia tai salasanoja.

```
<DTS:FlatFileColumn>
<DTS:Property DTS:Name="ColumnType">Delimited</DTS:Property>
<DTS:Property DTS:Name="ColumnDelimiter" xml:space="preserve">_x002C_</DTS:Property>
<DTS:Property DTS:Name="ColumnWidth">0</DTS:Property>
<DTS:Property DTS:Name="MaximumWidth">11</DTS:Property>
<DTS:Property DTS:Name="DataType">3</DTS:Property>
<DTS:Property DTS:Name="DataPrecision">0</DTS:Property>
<DTS:Property DTS:Name="DataScale">0</DTS:Property>
<DTS:Property DTS:Name="TextQualified">-1</DTS:Property>
<DTS:Property DTS:Name="ObjectName">h_o_nro</DTS:Property>
<DTS:Property DTS:Name="DTSID">{6146E9A6-E560-4E0C-A103-F555AFF75C84}</DTS:Property>
<DTS:Property DTS:Name="Description"></DTS:Property>
<DTS:Property DTS:Name="CreationName"></DTS:Property></DTS:FlatFileColumn>
<DTS:FlatFileColumn>
<DTS:Property DTS:Name="ColumnType">Delimited</DTS:Property>
<DTS:Property DTS:Name="ColumnDelimiter" xml:space="preserve">_x000D__x000A_</DTS:Property>
<DTS:Property DTS:Name="ColumnWidth">0</DTS:Property>
<DTS:Property DTS:Name="MaximumWidth">11</DTS:Property>
<DTS:Property DTS:Name="DataType">3</DTS:Property>
<DTS:Property DTS:Name="DataPrecision">0</DTS:Property>
<DTS:Property DTS:Name="DataScale">0</DTS:Property>
<DTS:Property DTS:Name="TextQualified">-1</DTS:Property>
<DTS:Property DTS:Name="ObjectName">h_pomonro</DTS:Property>
<DTS:Property DTS:Name="DTSID">{9DC816DC-3D7A-44FB-AC1D-618C5F6A6820}</DTS:Property>
<DTS:Property DTS:Name="Description"></DTS:Property>
<DTS:Property DTS:Name="CreationName"></DTS:Property></DTS:FlatFileColumn>

<DTS:Property DTS:Name="ConnectionString">C:\Temp\exportfile.csv</DTS:Property>

</DTS:ConnectionManager></DTS:ObjectData></DTS:ConnectionManager>

<DTS:ConnectionManager>
<DTS:Property DTS:Name="DelayValidation">0</DTS:Property>
<DTS:Property DTS:Name="ObjectName">SourceConnectionOLEDB</DTS:Property>
<DTS:Property DTS:Name="DTSID">{4603763B-06CF-40CD-A847-D29E114B468A}</DTS:Property>
<DTS:Property DTS:Name="Description"></DTS:Property>
<DTS:Property DTS:Name="CreationName">OLEDB</DTS:Property><DTS:ObjectData><DTS:ConnectionManager>
<DTS:Property DTS:Name="Retain">0</DTS:Property>

<DTS:Property DTS:Name="ConnectionString">Data Source=THESISWIN7\SQL2011WIN7;
Initial Catalog=thesis_db;Provider=SQLNCLI10;Integrated Security=SSPI;Auto Translate=false;
</DTS:Property>

</DTS:ConnectionManager></DTS:ObjectData></DTS:ConnectionManager>

<DTS:Property DTS:Name="LastModifiedProductVersion">10.50.1600.1</DTS:Property>
<DTS:Property DTS:Name="ForceExecValue">0</DTS:Property>
<DTS:Property DTS:Name="ExecValue" DTS:DataType="3">0</DTS:Property>
<DTS:Property DTS:Name="ForceExecutionResult">-1</DTS:Property>
<DTS:Property DTS:Name="Disabled">0</DTS:Property>
<DTS:Property DTS:Name="FailPackageOnFailure">0</DTS:Property>
<DTS:Property DTS:Name="FailParentOnFailure">0</DTS:Property>
<DTS:Property DTS:Name="MaxErrorCount">0</DTS:Property>
<DTS:Property DTS:Name="ISOLevel">1048576</DTS:Property>
<DTS:Property DTS:Name="LocaleID">1035</DTS:Property>
<DTS:Property DTS:Name="TransactionOption">1</DTS:Property>
<DTS:Property DTS:Name="DelayValidation">0</DTS:Property>
<DTS:LoggingOptions>
<DTS:Property DTS:Name="LoggingMode">0</DTS:Property>
<DTS:Property DTS:Name="FilterKind">1</DTS:Property>
<DTS:Property DTS:Name="EventFilter" DTS:DataType="8"></DTS:Property></DTS:LoggingOptions>
```

KUVA 1. SSIS-paketin sisältämä XML-tieto.



KUVA 2. SSIS-paketin rakenne BIDS-kehitysympäristössä. Kuvassa esitetty SSIS-paketti käsittää yhden Data Flow -tehtäväkomponentin.

4.1.2 Task

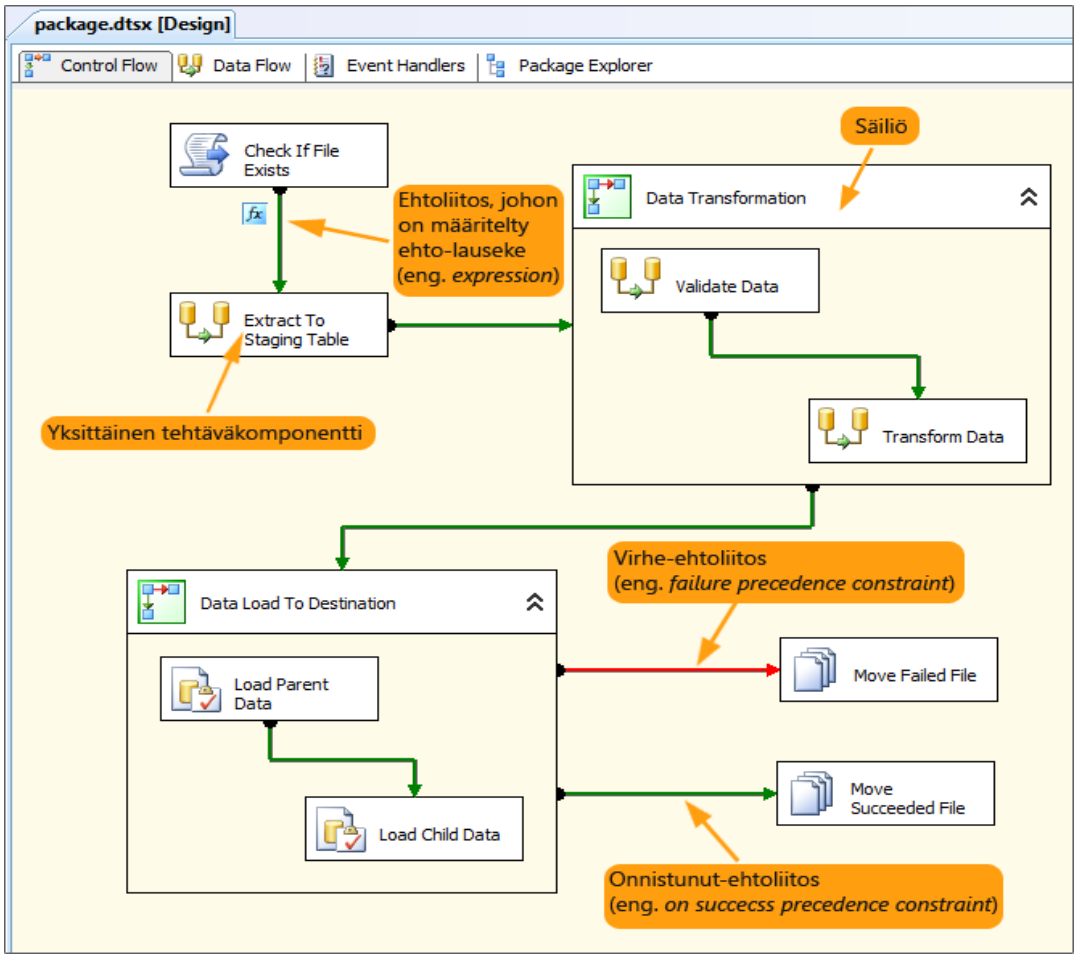
SSIS-kehitysympäristön avulla toteutetun tiedonsiirtorajapinnan toiminnot muodostetaan *task*-nimisten tehtäväkomponenttien avulla. SSIS-tehtäväkomponenttia voidaan verrata esimerkiksi jonkin ohjelmointikielen yksittäiseen metodiin, joka muodostaa osan ohjelmoidun sovelluksen toiminnallisuudesta. Suurin ero yksittäisen ohjelmointikielen metodin ja SSIS-tehtäväkomponentin välillä on kuitenkin se, että SSIS-tehtäväkomponenttien hyödyntäminen SSIS-paketissa ei edellytä ohjelmointia. Kukin SSIS-tehtäväkomponentti käsittää siis valmiiksi tiettyä toiminnallisuutta, joka voidaan ottaa käyttöön BIDS-kehitysympäristössä siirrä-pudota-menetelmällä (eng. *drag and drop*) ja konfiguroimalla. (Knight ym. 2008, 8.) Microsoft:n (s.a.) SQL Server 2008 Integration Services -tuotedokumentaation mukaan SSIS-kehitysympäristön sisältämät tehtäväkomponentit voidaan luokitella yhdeksään eri ryhmään. Nämä ryhmät ja niiden sisältämien tehtäväkomponenttien toiminnan yleinen kuvaus on esitetty seuraavassa luettelossa.

- *Data Flow Task*: käsittää yhden tehtäväkomponentin, jonka avulla voidaan suorittaa tiedon poiminta tiedonsiirron lähteestä, tiedon muunnokset ja tiedon lataus tiedonsiirron kohteeseen.

- *Data Preparation Tasks*: käsittää tehtäväkomponentit, joiden avulla voidaan käsitellä tiedostoja ja hakemistoja, ladata tiedostoja FTP-protokollan kautta, suorittaa web service -rajapintojen web-metodeja, käsitellä XML-tiedostoja ja profiloida tiedonsiirtorajapinnassa käsiteltävää tietoa.
- *Workflow Tasks*: käsittää tehtäväkomponentit, joiden avulla on mahdollista kutsua SSIS-paketin ulkopuolella olevia käyttöjärjestelmän sovelluksia tai prosesseja.
- *SQL Server Tasks*: käsittää tehtäväkomponentit, joiden avulla voidaan käsitellä SQL Server -tietokantaohjelmiston objekteja ja tietoa.
- *Scripting Tasks*: käsittää yhden tehtäväkomponentin, jonka avulla voidaan suorittaa C#.NET tai VB.NET kielillä ohjelmoitua toiminnallisuutta SSIS-paketissa.
- *Analysis Service Tasks*: käsittää tehtäväkomponentit, joiden avulla on mahdollista käsitellä SQL Server Analysis Services -objekteja.
- *Maintenance Tasks*: käsittää tehtäväkomponentit, joiden avulla pystytään suorittamaan SQL Server -tietokantoja ylläpitäviä toimenpiteitä, kuten tietokantojen varmistamista, tietokanta-hakemistojen (ts. *index*) ylläpitoa ja SQL Server Agent -tehtävien suorittamista.
- *Backward Compatibility Tasks*: käsittää tehtäväkomponentit DTS-teknologian tuelle SSIS-kehitysympäristössä.
- *Custom Tasks*: käsittää sovelluskehittäjien itse luomat tehtäväkomponentit, jotka eivät ole saatavilla suoraan SSIS-kehitysympäristön oletusasennuksessa.

4.1.3 Control Flow

SSIS-paketin osaa, missä hallitaan SSIS-rajapinnan suorituksen kulkua, kutsutaan nimellä *Control Flow*. Tässä osassa käsitellään SSIS-paketin suorituksen kulkuun vaikuttavia tehtäväkomponentteja, säiliöitä (eng. *container*), joiden sisään voidaan eristää haluttua toiminnallisuutta, ja ehtoliitoksia, joiden avulla tehtäväkomponentit sidotaan toisiinsa. Control Flow -osa voidaan ymmärtää siis SSIS-paketin rakenteen määrittelyn keskeisimpänä alueena. (Microsoft s.a.; Knight ym. 2008, 37.) Kuvassa kolme on havainnollistettu SSIS-paketin Control Flow -osaa ja siinä käytettäviä komponentteja. Kuvasta kolme voidaan huomata muun muassa, miten SSIS-paketin tehtäväkomponentit sidotaan toisiinsa ehtoliitoksien avulla ja, miten ehtoliitokset määrittävät SSIS-paketin suorituksen etenemisen.



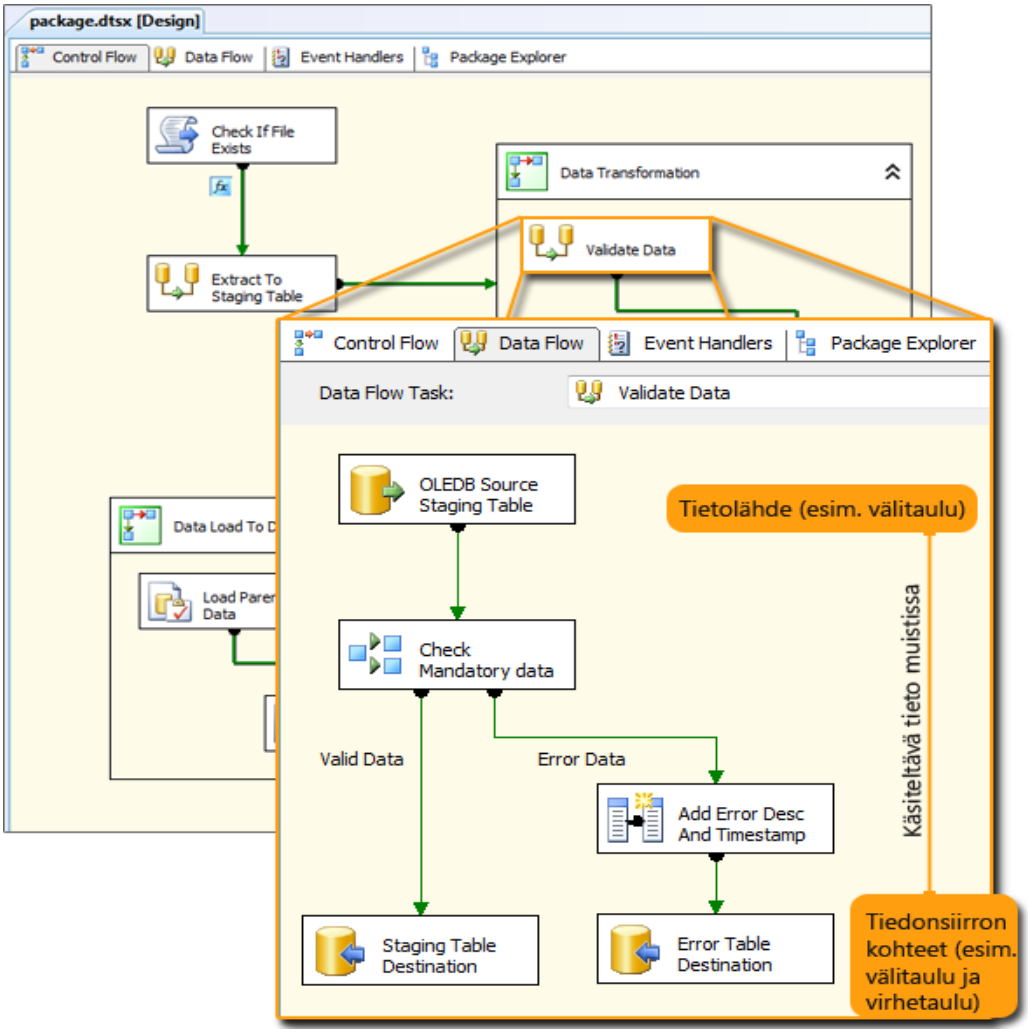
KUVA 3. SSIS-paketin Control Flow -osa.

4.1.4 Data Flow

Data Flow -osan tehtävänä on kapseloida SSIS-paketin toiminnallisuus, jonka avulla tietoa siirretään tiedonsiirtorajapinnan lähteiden ja kohteiden välillä. SSIS-paketin Data Flow -osassa suoritetaan myös SSIS-tiedonsiirtorajapinnan tiedon muuntaminen. Data Flow -osaa voidaan pitää ETL-tiedonsiirtorajapinnan toteutuksen näkökulmasta katsottuna SSIS-paketin merkittävimpänä osana, sillä se mahdollistaa tiedon poiminnan, muunnosten tekemisen ja latauksen tiedonsiirron kohteeseen, kun SSIS-tiedonsiirtorajapinnassa käsitellään suuria tietomääriä. (Microsoft s.a. .)

SSIS-kehitysympäristön oikeaoppinen ja tehokas hyödyntäminen tietojärjestelmien ja tietovarastojärjestelmien välisten tiedonsiirtojen toteutuksessa edellyttää SSIS-paketin Control Flow - ja Data Flow -osan toiminnan ymmärtämistä ja niiden erottamista toisistaan. Data Flow -osan tiedon prosessointi perustuu muistin käyttöön, jolloin tässä osassa tehdyt tiedon muunnokset suoritetaan siis muistissa. Tämä mahdollistaa käytännössä tiedon nopean käsittelyn ETL-tiedonsiirroissa. Mikäli tiedon muunnokset tehdään SSIS-rajapinnassa Data Flow -osan sijaan Control Flow -

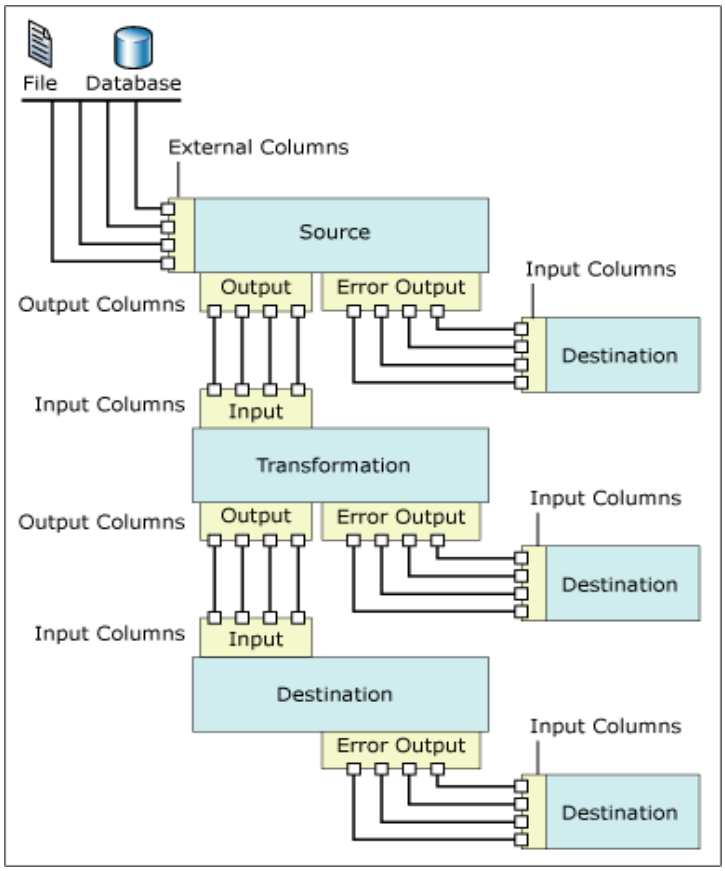
osassa, tarkoittaa se sitä, että tiedonsiirron välitauluja, joissa siirrettävää tietoa mahdollisesti käsitellään, joudutaan päivittämään toistuvasti. (Knight ym. 2008, 131.) Tässä tapauksessa tietoa joudutaan siis käsittelemään toistuvasti SQL-kyselyiden avulla, joka on hidasta erityisesti suuria tietomassoja käsiteltäessä. ETL-tiedonsiirtoja kehittäessä on kuitenkin huomattava, että SSIS-paketin Data Flow -osan toimintalogiikkaan lukeutuva muistin käyttö vaatii runsaasti muistiresursseja siltä palvelimelta, missä SSIS-prosessi suoritetaan. Kuviossa neljä havainnollistetaan SSIS-paketin yksittäistä Data Flow -tehtäväkomponenttia ja tiedon siirtymistä lähteestä kohteeseen sen sisässä.



KUVIO 4. Data Flow -tehtäväkomponentissa tiedonsiirron lähde- ja kohdetietokannan resursseja kulutetaan ainoastaan tiedon poiminta- ja latausvaiheessa, koska tietoon kohdistettavat muunnokset suoritetaan SSIS-ympäristön käyttämässä muistissa.

SSIS-ympäristön Data Flow -osan voidaan katsoa muodostuvan kolmesta eri kokonaisuudesta: tietolähteet (eng. *source*), muunnokset (eng. *transformation*) ja tiedonsiirron kohteet (eng. *destination*). Data Flow -osan tietolähteet mahdollistavat tiedon poiminnan tiedonsiirron lähteestä, muunnosten avulla suoritetaan tiedon

puhdistaminen, koonti ja muokkaus ja tiedonsiirron kohteet mahdollistavat tiedon latauksen kohteena olevaan tietokantaan tai tiedonsiirtorajapinnan käyttämään muistiin. Tiedon siirtyminen Data Flow -osan muistissa perustuu erityyppisiin reitteihin, jotka muodostuvat Data Flow:ssa olevien komponenttien välille ja liittyvät komponenttien sisääntulo- (eng. *input buffer*) ja ulostulo-puskureihin (eng. *output buffer*). Tieto siirtyy Data Flow:n erityyppisissä reiteissä sarakkeittain ja tiedon siirtyminen Data Flow:n yksittäisen komponentin sisässä sisääntulo-puskurista ulostulo-puskuriin perustuu sarakeliitoksiin (eng. *column mapping*). (Microsoft s.a. .) Kuviossa viisi havainnollistetaan Data Flow -tehtäväkomponentin tiedonsiirrossa olevia eri tyyppisiä reittejä ja komponenttien sisääntulo- ja ulostulo-puskureita.

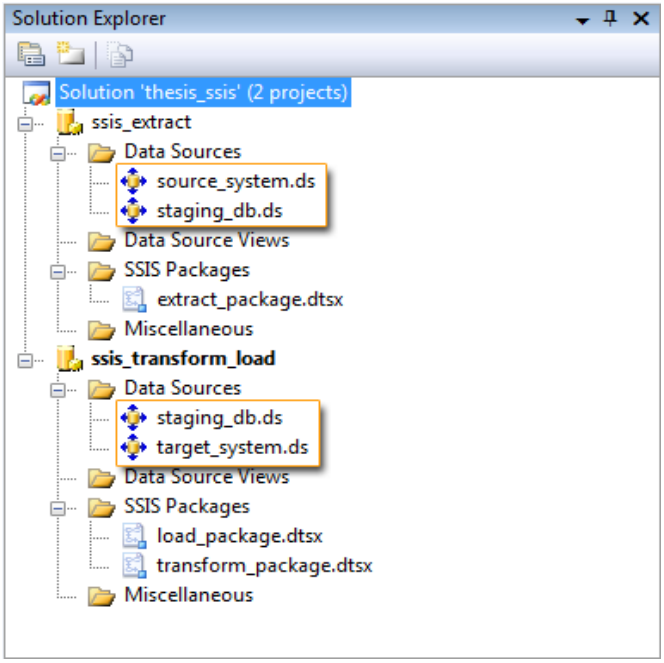


KUVIO 5. Microsoft:n SSIS-ympäristön dokumentaatiossa esitetty kuva Data Flow:n eri tyyppisistä reiteistä, joita pitkin tieto siirtyy lähteestä kohteeseen, ja komponenttien sisääntulo- ja ulostulo-puskureista. Kuvasta voidaan huomata myös käsitteet *External Columns*, *Output Columns* ja *Input Columns,* joiden välillä komponenttien sarakeliitokset luodaan. (Microsoft s.a. .)

4.1.5 Data Source Element

Tietolähteisiin ja -kohteisiin kytkeytyminen voidaan perustaa SSIS-paketissa ennalta määritettyjen tietoyhteyskomponenttien (eng. *data source*) käyttöön. Tietoyhteyskomponentti voidaan ymmärtää SSIS-paketin toteutuksen ulkopuolisena komponenttina, joka viittaa tiettyyn tietolähteeseen, kuten esimerkiksi tietokantaan. Yksittäinen tietoyhteyskomponentti pitää sisällään tietolähteen, johon komponentti on kytketty, tietomallin ja tiedon. (Microsoft s.a. .)

Yksittäistä tietoyhteyskomponenttia voidaan hyödyntää BIDS-kehitysympäristön samassa projekti-kokonaisuudessa olevien kaikkien SSIS-pakettien kesken tietoyhteyksien (eng. c*onnection manager*) - tai tietoyhteyskomponenttien näkymien (eng. d*ata source view*) muodostamisessa. Tämän lisäksi yksittäisen tietoyhteyskomponentin pohjalta on mahdollista luoda uusi tietoyhteyskomponentti muihin projektikokonaisuuksiin, jotka on sijoitettu samaan *solution*-kokonaisuuteen BIDS-kehitysympäristössä. Tekniikka, jonka avulla SSIS 2008 -ympäristön tietoyhteyskomponenttien avulla kytkeydytään eri tyyppisiin tietolähteisiin, perustuu OLE DB -standardiin ja ADO.NET-teknologian tietoyhteyksien luontitekniikkaan. (Knight ym. 2008, 9-10.) Kuvassa neljä havainnollistetaan samassa solution-kokonaisuudessa olevan kahden eri SSIS-projektin tietoyhteyskomponentteja.



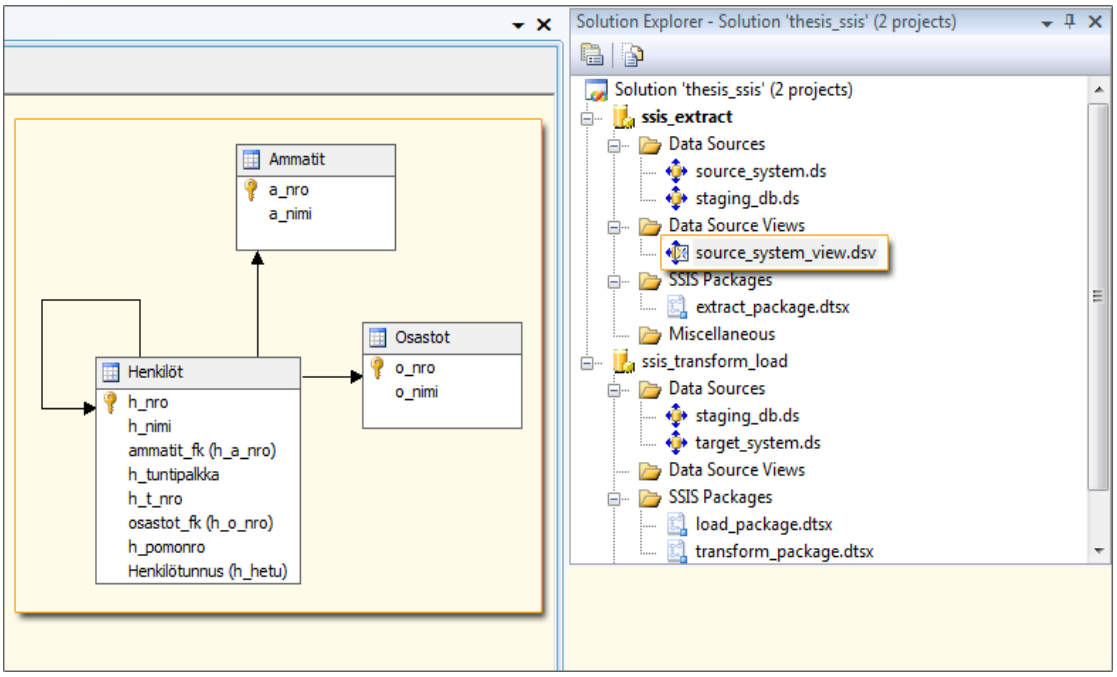
KUVA 4. BIDS-kehitysympäristön SSIS-solution-kokonaisuuteen luotuja tietoyhteyskomponentteja voidaan tarkastella *Visual Studio* -kehitysympäristöstä tutun *Solution Explorer* -ikkunan kautta.

4.1.6 Data Source View

Tietoyhteyskomponenttien näkymien (eng. *data source view*) hyödyntäminen mahdollistaa SSIS-paketissa käsiteltävän tiedon koonnin loogisiin ryhmittymiin. Yksittäinen näkymä luodaan BIDS-kehitysympäristössä tietoyhteyskomponentin pohjalta. Luotavaan näkymään voidaan koota tietoa valikoidusti tietoyhteyskomponentin objekteista, kuten tauluista, tietokantanäkymistä ja -proseduureista. Näiden lisäksi tietoyhteyskomponentin näkymän sisältämät objektit voidaan sitoa toisiinsa viite-eheys-määritysten avulla. (Knight ym. 2008, 10.)

Tietoyhteyskomponenttien näkymien käyttö muodostuu hyödylliseksi esimerkiksi tilanteissa, missä tiedonsiirron tietolähteenä käytettävän tietojärjestelmän rakenne halutaan muuntaa helpommin ymmärrettävään muotoon. Tietoyhteyskomponentin pohjalta luotuun näkymään valitut taulut ja niiden kentät voidaan siis nimetä tarvittaessa uudelleen ja tietokokonaisuuksien välille voidaan muodostaa uusia riippuvuussuhteita. Tämän lisäksi näkymän tauluihin on mahdollista luoda uusia kenttiä, joihin voidaan suorittaa esimerkiksi oman liiketoimintalogiikan mukaista laskentaa. (Knight ym. 2008, 10; Microsoft s.a. .) Tietoyhteyskomponentin näkymä voidaan siis ymmärtää tässä tapauksessa tiedonsiirtorajapinnan räätälöitynä tietomallina, mihin tieto kootaan tietolähteestä oman liiketoimintalogiikan edellyttämän rakenteen mukaisesti.

Yksittäiseen SSIS-projektiin, jota kehitetään BIDS-kehitysympäristössä, luotua tietoyhteyskomponentin näkymää voidaan hyödyntää projektin jokaisen SSIS-paketin kesken esimerkiksi Data Flow -tehtäväkomponentin tietolähteiden ja -kohteiden määrittämisessä. On kuitenkin huomioitava, että ennen kuin yksittäinen näkymä on hyödynnettävissä SSIS-paketissa, on pakettiin muodostettava tietoyhteys, joka muodostuu siitä tietoyhteyskomponentista, jonka pohjalta näkymä on muodostettu. Huomioitavaa on lisäksi se, että SSIS-paketin ja tietoyhteyskomponentin näkymän välillä ei ole fyysistä riippuvuutta. Tietoyhteyskomponentin näkymän määritys liittyy siis osaksi SSIS-pakettia, kun näkymä otetaan käyttöön paketissa ja paketti käännetään (eng. *build*) BIDS-kehitysympäristössä. (Microsoft s.a. .) Kun SSIS-paketti asennetaan esimerkiksi tuotantoympäristöön, ei tietoyhteyskomponentin näkymää tarvitse siis fyysisesti siirtää SSIS-paketin mukana tuontantoympäristöön. Kuvassa viisi on havainnollistettu yksittäiseen SSIS-projektiin BIDS-kehitysympäristössä luotua tietoyhteyskomponentin näkymää.

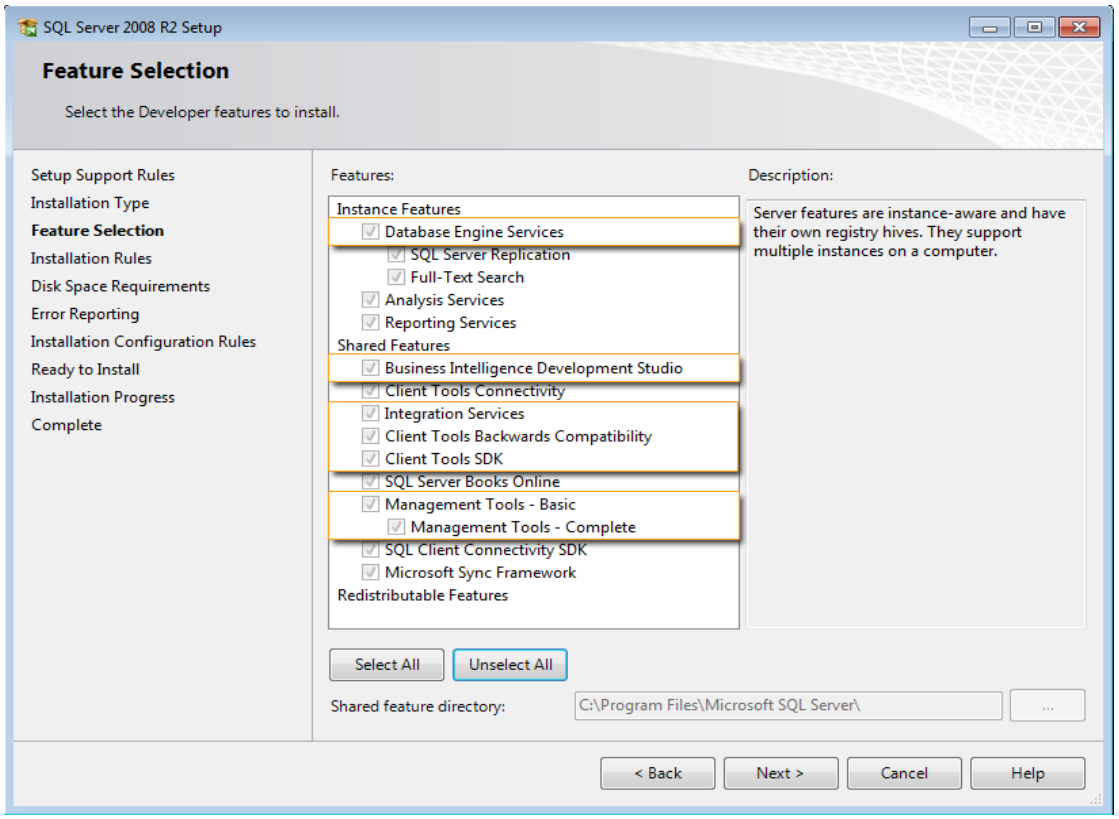


KUVA 5. Kuvan oikeassa osassa olevasta solution explorer -ikkunasta voidaan huomata tietoyhteyskomponentin näkymä "*source_system_view.dsv*", jonka rakenne on esitetty kuvan vasemmassa osassa.

4.2 Ohjelmistovaatimukset ja asennus

SSIS 2008 -ympäristön kaikkien ominaisuuksien tuotantokäyttöä edellyttää, että SSIS asennetaan vähintään SQL Server 2008 R2 -ohjelmiston *Enterprise*-julkaisuversioon. Tämän lisäksi SSIS 2008 -ympäristön ominaisuudet ovat saatavilla kokonaisuudessaan kehitys-, testaus- ja esittelykäyttöön SQL Server 2008 R2 -ohjelmiston *Developer*-erikoisversiossa. SSIS 2008 -ympäristön sisältämiä ominaisuuksia on mahdollista käyttää myös rajoitetusti SQL Server 2008 R2 Standard -ohjelmistoversiossa. Tässä tapauksessa SSIS-ympäristössä ei pystytä kuitenkaan hyödyntämään muun muassa seuraavia ETL-prosessin muunnosten tekoon tarkoitettuja komponentteja: *Data Mining Query Component*, *Fuzzy Grouping*, *Fuzzy Lookup*, *Term Extraction*, *Term Lookup*. (Knight ym. 2008, 19; Policht 2010.)

Kaikki SQL Server 2008 R2 -tietokantaohjelmiston ominaisuudet, kuten SSIS, voidaan asentaa yksittäiselle työasemalle tai palvelimelle valikoidusti erillisen SQL Server -asennussovelluksen avulla. Asennussovelluksen avulla on mahdollista valita myös erillisenä ne SSIS-ympäristön komponentit, jotka tietokoneelle halutaan asentaa. (Microsoft s.a. .) Kuvasta kuusi voidaan nähdä SQL Server -asennussovelluksessa valittavissa olevat komponentit, jotka vaikuttavat SSIS-ympäristön toimintaan.



KUVA 6. SQL Server 2008 R2 -ohjelmiston ominaisuudet voidaan asentaa tietokoneelle erillisen SQL Server -asennussovelluksen avulla. SSIS 2008 -ympäristön toimintaan vaikuttavia komponentteja, joiden asentaminen on valittavissa, on yhteensä kuusi kappaletta.

Yleinen tilanne on, että SSIS-ympäristö asennetaan yksittäiselle työasemalle ainoastaan SSIS-pakettien kehittämistä varten. Tässä tapauksessa työasemaan tarvitaan ainoastaan BIDS-kehitysympäristön asennus. Tällöin kehitettäviä SSIS-paketteja on mahdollista suorittaa ainoastaan BIDS-kehitysympäristön sisässä. Mikäli SSIS-paketteja halutaan suorittaa sen sijaan BIDS-kehitysympäristön ulkopuolella esimerkiksi tuotantojärjestelmässä, tulee tietokoneelle asentaa vähintään SQL Server:n *Integration Services* -komponentti. SSIS-pakettien suorittamista ei siis edellytä, että tietokoneelle on asennettuna SQL Server:n tietokantamoottorin instanssi. On kuitenkin huomioitava, että mikäli SSIS-paketteja halutaan hallita tietokoneen käyttöjärjestelmän tiedostojärjestelmän sijasta SQL Server -ohjelmiston tietokantainstanssin yhteydessä, on tietokoneelle asennettava SQL Server:n *Management Tools - Complete* - ja *Database Engine Services* -komponentti. Näistä Management Tools - Complete -komponentti käsittää *SQL Server Management Studio* -sovelluksen asennuksen ja Database Engine Services -komponentti SQL Server -tietokantamoottorin lisäksi muun muassa *SQL Server Agent* -sovelluksen, jonka avulla SSIS-paketteja voidaan suorittaa ajastetusti. Muita SSIS-ympäristön toimintaan vaikuttavia komponentteja ovat *Client Tools Backward Compatibility* -

komponentti, joka käsittää tuen SSIS-ympäristössä DTS-teknologialle, ja *Client Tools SDK* -komponentti, joka käsittää .NET-luokkakirjastot SSIS-ohjelmointirajapinnan käyttöön. (Microsoft s.a. .)

4.3 BIDS-kehitysympäristö

Business Intelligence Development Studio (ts. *BIDS*) on SSIS-pakettien ensisijainen kehitysympäristö. SSIS-projektien ohella BIDS-kehitysympäristössä on mahdollista kehittää myös muita SQL Server Business Intelligence -projekteja, kuten SQL Server Reporting Services - ja SQL Server Analysis Services -projekteja. BIDS-kehitysympäristö mahdollistaa kaikentyyppisten SQL Server Business Intelligence -projektien hallinnan yhden solution-kokonaisuuden sisässä, jonka johdosta esimerkiksi yksittäiseen tietovarastoprojektiin liittyviä BI-projekteja voidaan hallita yhden kokonaisuuden sisässä. (Microsoft s.a.; Knight ym. 2008, 30.)

BIDS-kehitysympäristön toiminta pohjautuu Microsoft:n kehittämän *Visual Studio 2008* -sovelluskehittimen toimintaan. SSIS-tiedonsiirtojen kehittäminen on mahdollista suoraan myös Visual Studio 2008:n kokonaisversiossa. Tämä ohjelmistoversio ei kuitenkaan sisällä ominaisuuksia, joita BIDS-kehitysympäristössä ei olisi, joten sen hyödyntäminen ei tuo lisäarvoa SSIS-pakettien kehittämiseen. (Knight ym. 2008, 30.) Huomattavaa kuitenkin on, että SSIS-pakettien kehittäminen ei ole mahdollista Visual Studio 2010 -kehitysympäristössä, joka on tämän opinnäytetyön kirjoittamiseen mennessä uusin Visual Studio -ohjelmiston julkaisu. Tästä johtuen SSIS-pakettien kehittämistä edellyttää, että BIDS-kehitysympäristö asennetaan työasemalle erillisenä, mikäli Visual Studio 2008 -sovelluskehitin ei ole asennettuna.

5 SQL SERVER INTEGRATION SERVICES -RAJAPINNAN LUONTI

5.1 SSIS-tiedonsiirtorajapinnan rakennemallin suosituksia

Selkeä vahvuus tiedonsiirtorajapintojen luonnissa SSIS-teknologian avulla on, että SSIS mahdollistaa rajapintojen luonnin noudattaen itse suunniteltua rakennemallia (ts. *arkkitehtuuri*). Tästä johtuen SSIS-teknologiaa voidaan hyödyntää tehokkaasti suuriin tietovarastoihin kohdistuvien ETL-tiedonsiirtojen lisäksi myös pienempien operatiivisten tietojärjestelmien välisten integraatioiden kehittämisessä. Kehitettävän SSIS-paketin rakennemallia suunniteltaessa tulisi siis aina ottaa huomioon, että mihin tarkoitukseen SSIS-pakettia kehitetään. Tämän lisäksi rakennemallin suunnitellussa tulisi ottaa huomioon SSIS-paketin ylläpidon vaatimukset sen jälkeen, kun kehitettävä SSIS-paketti on saatu valmiiksi ja asennettu tuotantoympäristöön.

Suorituskyky on yksi niistä seikoista, joka tulee ottaa usein huomioon SSIS-liittymien kehityksessä. SSIS-paketin kehittämisen näkökulmasta katsottuna suorituskyvyn huomioon ottaminen tarkoittaa esimerkiksi sitä, että hyödynnetäänkö SSIS-paketissa tiedonsiirtoon osallistuvien tietokantojen tietokantamoottorin vai SSIS-kehitysympäristön muistinkäyttöön perustuvan tiedonsiirtomoottorin resursseja. Näistä vaihtoehdoista kumpikaan ei erotu selkeästi jokaiseen tilanteeseen paremmin soveltuvaksi vaihtoehdoksi, sillä SQL Server -ohjelmiston tietokantamoottori on luotu myös tehokkaasti toimivaksi ja tietyissä tilanteissa sen hyödyntäminen on järkevä vaihtoehto (Mundy ym. 2011, 197).

Taulukossa yksi havainnollistetaan kahden erityyppisen SSIS-tiedonsiirron suoritusnopeuksia, kun tiedonsiirron lähde- ja kohde-tietokanta ovat samassa SQL Server -instanssissa yhdellä tietokoneella. SSIS-paketit, joiden avulla taulukon yksi testitulokset on luotu, on toteutettu rakenteellisesti eri tavalla. Toisessa SSIS-paketissa tiedonsiirto tehdään täysin hyödyntäen tiedonsiirtoon osallistuvien SQL Server -tietokantojen tietokantamoottoreita. Toisessa SSIS-paketissa tiedonsiirto tehdään sen sijaan hyödyntäen enemmän SSIS-ympäristön omaa tiedonsiirtomoottoria.

Tarkemmin tarkasteltuna taulukon yksi testien SSIS-pakettien avulla siirretään tiedonsiirron lähdetietokannasta kohdetietokantaan kolmen eri luokittelutaulun sisältämä tieto. Tiedonsiirrossa suoritetaan uusien tietokantarivien lisäys ja vanhojen päivittäminen luokittelutiedoista löytyvän yksilöivän tunnistekoodin perusteella.

Päivitystilanteessa luokittelutietojen tunnistekoodit ovat yhtenäiset sekä tiedon lähteessä että kohteessa ja vastaavasti uusi luokittelutieto lisätään tiedonsiirron kohteeseen, mikäli siirrettävän luokittelutiedon tunnistekoodille ei löydy vastinetta tiedonsiirron kohteesta. On huomioitava lisäksi, että taulukon yksi testien SSIS-paketit suoritettiin SSIS-ympäristöön liittyvän *DTEXEC*-sovelluksen avulla.

Tämän opinnäytetyön liitteessä yksi on esitetty taulukon yksi testien SQL Server -tietokantamoottoria hyödyntävän SSIS-paketin rakenne BIDS-kehitysympäristössä ja tiedonsiirron suorittavan tietokantaproseduurin SQL-koodi. Liitteessä kaksi selvitetään sen sijaan SSIS-tiedonsiirtomoottoria hyödyntävän SSIS-paketin rakenne BIDS-kehitysympäristössä.

Taulukko 1. SQL Server -tietokantamoottorin ja SSIS-tiedonsiirtomoottorin nopeustestien tulokset.

Testin numero	Käsiteltävien rivien määrä	Tiedonsiirtomoottori	Tiedonsiirron kesto (sekuntia)
1	Kuhunkin 3 tauluun 10 uutta riviä ja kustakin 3 taulusta 3 rivin päivitys	SQL Server	0.281
		SSIS	0.625
2		SQL Server	0.188
		SSIS	0.593
3	Kuhunkin 3 tauluun 19787 uutta riviä ja kustakin 3 taulusta 3 rivin päivitys	SQL Server	1.547
		SSIS	0.922
4		SQL Server	0.594
		SSIS	0.922
5		SQL Server	0.516
		SSIS	0.922
6	Kuhunkin 3 tauluun 19787 uutta riviä ja kustakin 3 taulusta 253 rivin päivitys	SQL Server	0.797
		SSIS	2.89
7		SQL Server	0.64
		SSIS	3.00

Taulukossa yksi esitetyistä testituloksista voidaan huomata, että testitulosten luonnissa käytettyjen SSIS-pakettien toteutus ja käsiteltävien tietokantojen rakenne olivat hyvin yksinkertaisia. Tämä voidaan huomata erityisesti siitä, että tiedonsiirron kesto ei kasvanut merkittävästi, vaikka tiedonsiirrossa käsiteltävien rivien määrä kasvoikin suuremmaksi. Taulukon yksi testituloksista voidaan kuitenkin huomata, että SQL Server -tietokantamoottorin hyödyntäminen tiedonsiirrossa oli tässä tapauksessa tehokkaampaa kuin SSIS-tiedonsiirtomoottorin hyödyntäminen. Testin

tulokset saattavat johtua siitä, että testin SSIS-tiedonsiirtomoottoria hyödyntävän SSIS-paketin Data Flow -tehtäväkomponenteissa oli tarpeellista suorittaa ainoastaan yksi muunnos. Tämän johdosta SSIS-ympäristön tiedonsiirtomoottorista ei ollut siis käytännössä hyötyä. SSIS-paketti muodosti tässä tapauksessa oikeastaan ylimääräisen tiedonsiirtoa hidastavan kerroksen siirrettävän tiedon lähteen ja kohteen välille.

SSIS-pakettien rakkennemallille on yleistä, että tiedonsiirrossa hyödynnetään tiedon lähde- tai kohdetietokantojen moottoreiden resursseja. Tämä rakennemalli saadaan aikaiseksi hyvin helposti esimerkiksi silloin, kun tiedonsiirrossa käytetään hyväksi tietokantaproseduureja. Tämän lisäksi tiedonsiirron lähde- tai kohdetietokannan resursseja käytetään hyväksi, mikäli SQL-koodia sulautetaan esimerkiksi SSIS-paketin Data Flow -tehtäväkomponentissa tietolähde- tai -kohdekomponenttiin (Mundy ym. 2011, 197).

Tiedonsiirtoon osallistuvien tietokantamoottoreiden resurssien hyödyntämistä voidaan pitää tehokkaana vaihtoehtona, mikäli tietoa käsittelevä SQL-koodi ei sisällä runsaasti alikyselyitä tai monimutkaista logiikkaa. Myös suurten tiedonsiirtojen yhteydessä tietokantamoottoreiden resursseja voidaan hyödyntää esimerkiksi yksinkertaisten SQL-kielen *WHERE*-rajausten ja NULL- tai tietotyyppimuunnosten suorittamiseen. Suurissa ETL-tiedonsiirroissa tiedon pääsääntöinen muuntaminen kannattaa kuitenkin suorittaa aina SSIS-ympäristön resursseja hyödyntäen, sillä suorituskyvyn ohella tämän avulla tiedonsiirtorajapintaan saadaan aikaiseksi esimerkiksi parempi virhetilanteiden hallinta ja tehokkampi tiedonsiirron lokitus. (Mundy ym. 2011, 197, 416-417.)

SSIS-ympäristössä suurten ETL-tiedonsiirtojen rakennemallille on yleistä, että yksittäisten tietokannan taulujen käsittelyssä hyödynnetään erillisiä SSIS-paketteja. Tällöin kokonaisen SSIS-tiedonsiirtorajapinnan suorittamista hallitaan yhden "pää"-SSIS-paketin (eng. *master package*) sisässä, missä erillinen ali-paketti muodostaa itsenäisenä suoritettavan prosessin. Tämä rakennemalli tukee tiedonsiirtorajapinnan osien suorittamista yhdenaikaisesti. Tällöin on kuitenkin varmistettava, että tietokannan taulut, joita käsitellään yhdeaikaisesti, eivät ole riippuvaisia toisistaan. On myös ymmärrettävä, että SSIS-ympäristö ei mahdollista ali-pakettien ja pää-paketin keskinäistä tiedonvälitystä kovin tehokkaasti. Kun liittymän suoritus palautuu siis ali-paketilta takaisin pää-paketille, välittyy pää-paketille tieto ainoastaan siitä, että oliko ali-paketin suoritus onnistunut. Tiedonsiirtorajapintaan lukeutuvien SSIS-pakettien välinen edistyneempi tiedonvälitys voidaan kuitenkin toteuttaa esimerkiksi SSIS-

ympäristön konfiguraatio-tiedostojen (ts. *dtsConfig*-tiedosto), XML-tiedostojen tai tietokannassa olevien auditointi-taulujen (eng. *audit table*) avulla. (Mundy ym. 2011, 197-198.)

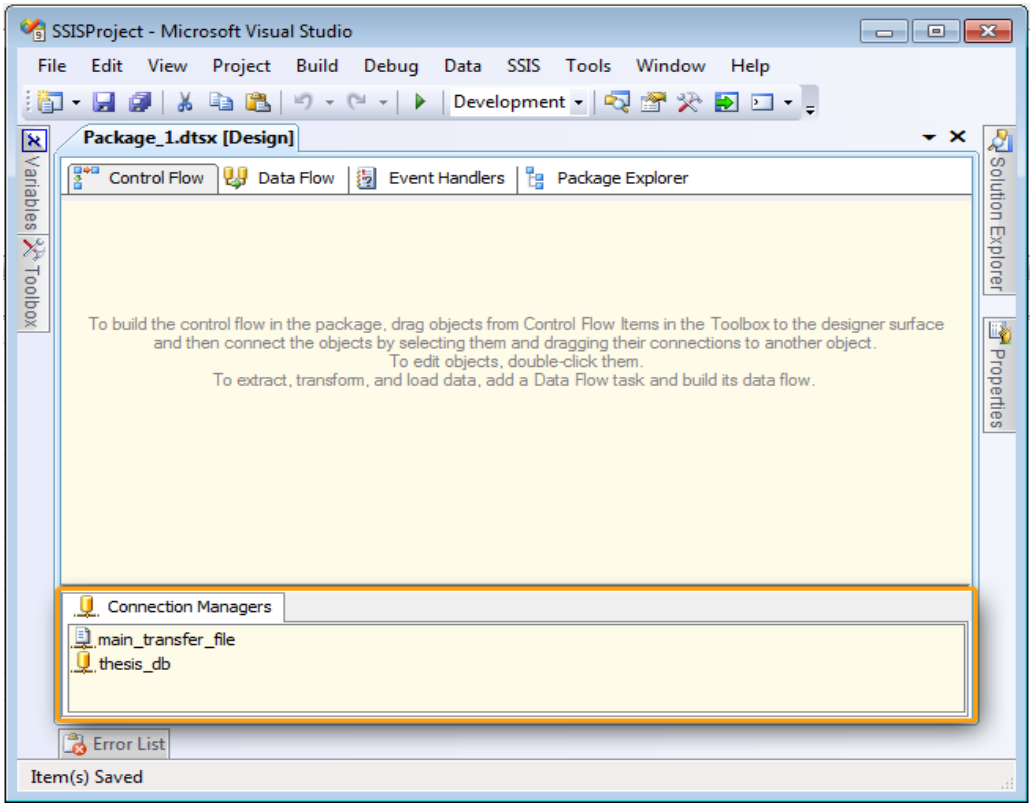
Tietokonesovellusten yleiseen rakennemalliin on kuulunut jo pitkään modulaarisuus (vrt. *monoliittinen rakenne*). Myös SSIS-pakettien kehittäminen on mahdollista modulaarisen rakenteen mukaisesti ja sitä voidaan pitää SSIS-tiedonsiirtorajapinnan ylläpidettävyyden kannalta erittäin suositeltavana. SSIS-pakettien kohdalla modulaarisuus voidaan jakaa kolmeen eri kokonaisuuteen, joita ovat: tiedonsiirtoprosessin modulaarisuus, SSIS-paketin modulaarisuus ja SSIS-komponentti-modulaarisuus. (Duffy 2008, 38.)

Tiedonsiirtoprosessin modulaarisuus voidaan ymmärtää rakennemallina, missä tiedonsiirtorajapinnan toiminnan loogiset osat kootaan esimerkiksi erillisiin SSIS-paketteihin. Tässä tapauksessa yksittäinen SSIS-paketti toimii itsenäisenä prosessina rajapinnan muun toteutuksen yhteydessä. SSIS-paketin modulaarisuus tarkoittaa sen sijaan yksittäisen SSIS-paketin sisässä suoritettavia ali-prosesseja. Yksittäinen ali-prosessi voi muodostua esimerkiksi erillisestä tietokantaproseduurista, jota kutsutaan suoritettavasta paketista, tai paketin toteutukseen lukeutuvasta säiliöstä, jonka sisään on eristetty loogisesti toisiinsa sidoksissa olevat tehtäväkomponentit. SSIS-tiedonsiirtorajapinnan modulaarisen rakenteen jaottelun viimeinen kokonaisuus eli SSIS-komponentti-modulaarisuus voidaan ymmärtää osana, missä erikoisempien ongelmien ratkaisemisessa käytetään apuna räätälöityä toiminnallisuutta. Räätälöidyn toiminnallisuuden kehittäminen tarkoittaa SSIS-ympäristössä yleisesti Script Task tai Script Component -tehtäväkomponenttien hyödyntämistä. Koska kyse on tässä tapauksessa VB.NET- tai C#.NET-ohjelmointikielen käytöstä, voidaan itse luotu toiminnallisuus eristää erilliseen *DLL*-luokkakirjastoon (ts. *Dynamic Link Library*) tai itse luotuun tehtäväkomponenttiin. Tämä tehostaa räätälöidyn toiminnallisuuden uudelleenkäytettävyyttä esimerkiksi tulevissa SSIS-tiedonsiirtorajapintojen kehitysprojekteissa. (Duffy 2008, 38.)

5.2 Tietoyhteyksien määrittäminen

Tietoyhteyksien määrittäminen on yleisesti ensimmäinen tehtävä, joka suoritetaan, kun uusi SSIS-projekti aloitetaan BIDS-kehitysympäristössä. Tietolähteisiin ja -kohteisiin kytkeytyminen suoritetaan SSIS-ympäristössä *Connection Manager* -komponenttien avulla. Yksittäiseen SSIS-pakettiin luodut tietoyhteydet voidaan löytää

BIDS-kehitysympäristön käyttöliittymän alaosasta (katso kuva seitsemän), kun SSIS-paketti on avattu muokkaustilaan kehitysympäristössä. (Knight ym. 2008, 40.) On siis huomattava, että yksittäistä tietoyhteyttä voidaan hyödyntää vain siinä SSIS-paketissa, mihin se on luotu. Uusi tietoyhteys voidaan luoda SSIS-pakettiin esimerkiksi BIDS-kehitysympäristön *Connection Managers* -ikkunan kontekstivalikon kautta.

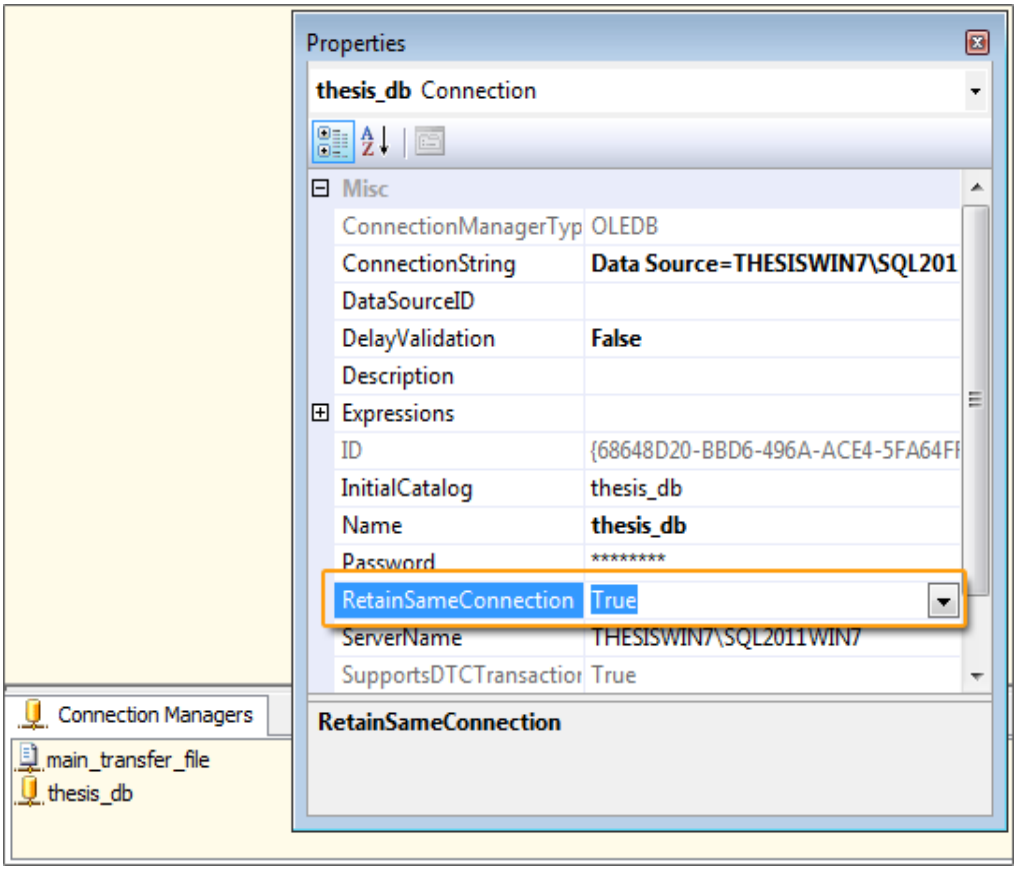


KUVA 7. BIDS-kehitysympäristön Connection Managers -ikkuna, missä yksittäiseen SSIS-pakettiin luodut tietoyhteydet sijaitsevat. SSIS-pakettiin, jota kuva ilmentää, on luotu kaksi Connection Manager -komponenttia: *Flat File Connection* ja *OLE DB Connection*.

Kun tietoyhteydet on luotu SSIS-pakettiin, niitä voidaan hyödyntää paketin Control Flow -osassa tehtäväkomponenttien tietoyhteyksien määrittämisessä ja Data Flow -tehtäväkomponenteissa tietolähteiden ja -kohteiden määrittämisessä (Knight ym. 2008, 40). Myös tietyt Data Flow -tehtäväkomponentissa olevat muunnoskomponentit hyödyntävät SSIS-pakettiin määriteltyjä tietoyhteyksiä.

SSIS-paketin suorituksen aikana SSIS-ympäristön ajonaikainen moottori tutkii, että mitä tietoyhteyttä suoritusvuorossa seuraavana olevassa komponentissa on hyödynnetty, ja suorittaa esimerkiksi tietokantayhteyden avaamisen tietoyhteyteen määritellyn connection string -merkkijonon pohjalta (Microsoft s.a.). On syytä ottaa

huomioon, että SSIS-paketin eri osissa hyödynnetyn saman tietoyhteyden pohjalta avataan oletuksena paketin suorituksen aikana aina uusi erillinen tietokantayhteys samaan tietokantaan. Tietokantayhteyksien avaaminen voidaan kuitenkin määrittää SSIS-pakettiin myös siten, että paketissa olevan tietoyhteyden pohjalta avataan ainoastaan yksi tietokantayhteys kohteena olevaan tietokantaan. Tässä tapauksessa tätä samaa yhteyttä hyödynnetään siis aina, kun yhteyden mukaiseen tietokantaan suoritetaan toimenpiteitä. Kuvassa kahdeksan on havainnollistettu, miten OLE DB Connection -tietoyhteyteen voidaan määritellä asetus, jonka johdosta tietoyhteyden mukaiseen tietokantaan avataan SSIS-paketin suorituksen aikana ainostaan yksi tietokantayhteys. Tämä toiminnallisuus voidaan ottaa siis käyttöön määrittämällä tietoyhteyden ominaisuuteen *RetainSameConnection* arvo *True*.



KUVA 8. SSIS-pakettiin luodun tietoyhteyden RetainSameConnection-ominaisuus.

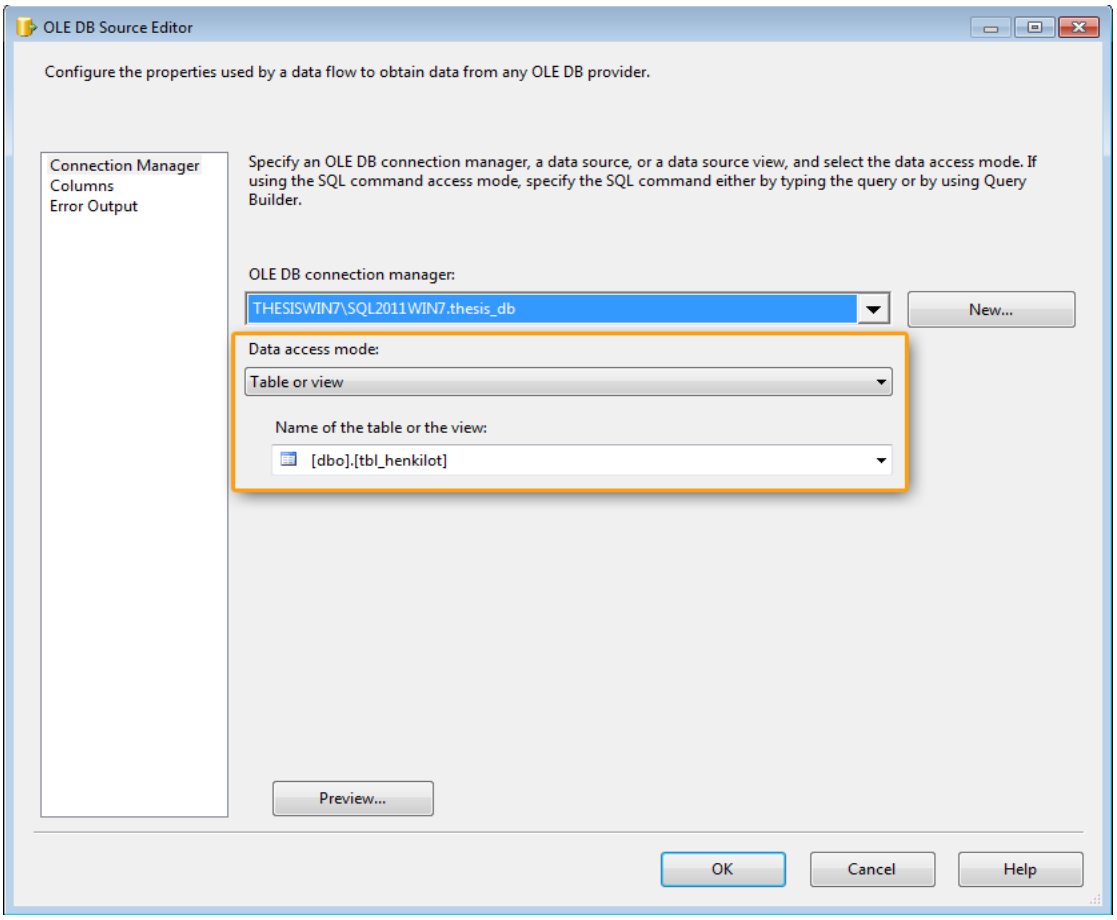
Mikäli SSIS-paketissa suoritetaan tietokantoihin kohdistuvia toimenpiteitä saman tietokantayhteyden kautta ja SSIS-paketissa hyödynnetään transaction-käsittelyä, on huomioitava, että tämä saattaa johtaa transaction-käsittelyn lukkotilanteisiin (eng. *transaction deadlock*). Lukkotilanteet ilmenevät erityisesti tilanteissa, joissa samaa tietokantayhteyttä hyödynnetään SSIS-paketin eri osissa, jotka suoritetaan yhdenaikaisesti.

5.3 Tiedon poiminta tietolähteestä

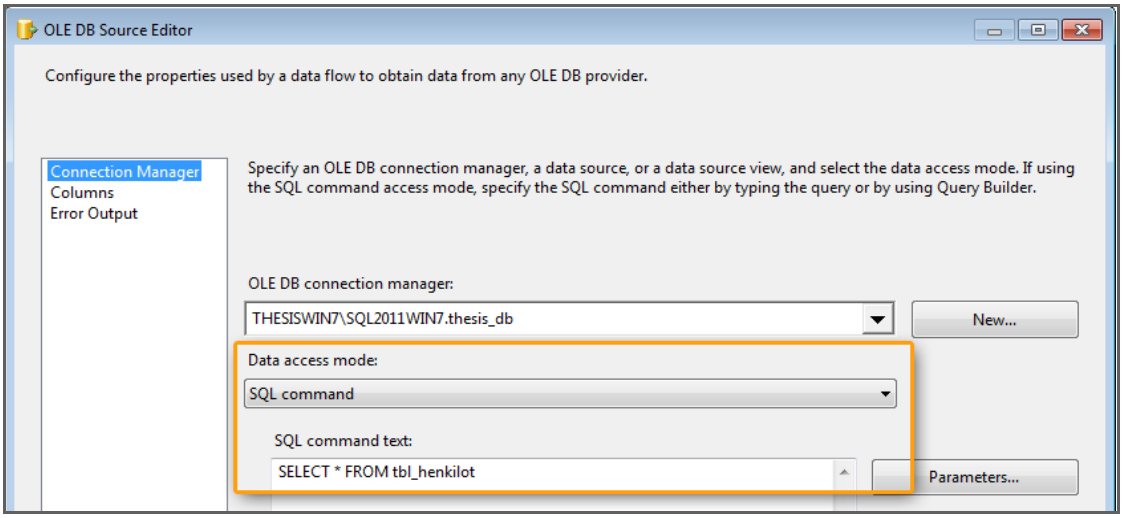
5.3.1 Tiedon poiminta tietojärjestelmästä

Mikäli ETL-tiedonsiirron poimintavaihe kohdistetaan suoraan operatiiviseen tietojärjestelmään, on hyvin yleistä, että poiminta suoritetaan esimerkiksi lähdejärjestelmään toteutetun tietokantanäkymän kautta tai SSIS-projektiin toteutetun tietoyhteyskomponentin näkymän avulla. Näkymien hyödyntämistä poiminnassa voidaan pitää hyvänä käytäntönä, koska tässä tapauksessa tieto voidaan poimia ETL-rajapintaan yksinkertaisesti esimerkiksi siten, että SSIS-paketin Data Flow -tehtäväkomponentin OLE DB Source -lähdekomponenttiin poimitaan kaikki tieto, minkä poiminnan apuna käytettävä näkymä käsittää. On kuitenkin huomioitava, että tämä menettely on kannattava ainoastaan, mikäli poiminnan suorittamiseen käytettävä näkymä on kehitetty tietyn ETL-tiedonsiirtorajapinnan tarpeita ajatellen.

Kaiken tiedon poimimista ilman rajauksia tietolähteestä ETL-rajapintaan voidaan pitää huonona käytäntönä, mikäli tieto poimitaan suoraan esimerkiksi operatiivisen tietojärjestelmän tietokannan yksittäisestä taulusta. Tällöin vaarana on, että tiedonsiirtoon poimitaan ylimääräisiä tietokannan taulun sarakkeita, joiden käytölle ei ole tarvetta. Kuvissa yhdeksän ja kymmenen on havainnollistettu sitä, miten tieto voidaan poimia OLE DB Source -lähdekomponentin avulla kokonaisuudessa yksittäisestä tietokannan taulusta SSIS-paketin Data Flow -tehtäväkomponentin käyttämään muistiin. Tiedon poimiminen OLE DB Source -lähdekomponenttiin voidaan suorittaa siis esimerkiksi määrittelemällä suoraan tietolähteenä käytettävä taulu valittavissa olevien taulujen ja näkymien valintalistasta. Tässä tapauksessa OLE DB Source -lähdekomponentin asetusten määritysikkunan Connection Manager -välilehdellä, missä poiminnan kohdetaulu määritellään, *Data Access Mode* -valintalistaan tulee valita arvo *Table or view*. Mikäli Data Access Mode -valintalistaan valitaan sen sijaan arvo *SQL Command*, voidaan tieto poimia kokonaisuudessaan tietolähteestä määrittelemällä lähdekomponenttiin "*SELECT **"-alkuinen SQL-komento.



KUVA 9. Tiedon poiminta kokonaisuudessaan tietolähteessä olevasta yksittäisestä taulusta, kun OLE DB Source -komponentin asetusten määritysikkunan Connection Manager -välilehdellä olevaan Data access mode -valintalistaan on valittu arvo Table or view.



KUVA 10. Tiedon poiminta kokonaisuudessaan tietolähteessä olevasta yksittäisestä taulusta, kun OLE DB Source -komponentin asetusten määritysikkunan Connection Manager -välilehdellä olevaan Data access mode -valintalistaan on valittu arvo SQL command.

Kaiken tiedon poiminnan huonoutta voidaan perustella sillä, että tiedonsiirron lähdetietokannan taulun jokainen sarake, joka poimitaan SSIS-paketin Data Flow -tehtäväkomponentin käyttämään muistiin, varaa oman osansa tiedonsiirron käytettävissä olevista muistiresursseista. Tämän lisäksi mikäli tiedonsiirrossa tietolähteenä käytettävän tietokannan taulun rakennetta muutetaan ETL-rajapinnan käyttöönoton jälkeen, saattaa tämä johtaa virhetilanteisiin tiedonsiirrossa. Kaikki uudet sarakkeet, jotka lisätään siis tiedonsiirron tietolähteenä käytettävään tauluun, liittyvät tässä tapauksessa myös SSIS-pakettiin toteutettuun tiedonsiirtoon, koska tiedonsiirtoon poimitaan kaikki sarakkeet tietolähteenä olevasta taulusta. Jokaisella SSIS-paketin tiedonsiirtoon poimitulla sarakkeella tulee olla siis oma tarkoitus, eikä ylimääräisiä sarakkeita tule sisällyttää käsiteltävien tietueiden joukkoon. (Knight ym. 2008, 414; Noor s.a. .)

BIDS-kehitysympäristö pyrkii huomauttamaan SSIS-paketin kehittäjää, mikäli paketin Data Flow -osassa käsitellään tiedonsiirron lähdetietokannan taulusta peräisin olevia ylimääräisiä sarakkeita. Mikäli SSIS-paketti suoritetaan siis BIDS-kehitysympäristössä ja pakettiin lukeutuvan Data Flow -tehtäväkomponentin muistissa on mukana sarakkeita, joita ei käsitellä tiedonsiirron aikana, tulostuu BIDS-ympäristön *output*-ikkunaan varoitusteksti: "*Warning: 0x80047076 at Data Flow Task, SSIS.Pipeline: The output column "h_a_nro" (23) on output "OLE DB Source Output" (11) and component "OLE DB Source" (1) is not subsequently used in the Data Flow task. Removing this unused output column can increase Data Flow task performance.*". Tässä varoitustekstissä selvitetään siis SSIS-paketin Data Flow -tehtäväkomponentti ja sen muistissa olevan sarakkeen nimi, joka on ylimääräisenä mukana tiedonsiirrossa.

Mikäli tiedonsiirron lähdejärjestelmässä olevissa tiedoissa ylläpidetään aikaleima-kenttiä, jotka määrittävät järjestelmässä olevan tiedon viimeisimmän muokkausajankohdan, on näitä tietoja järkevää hyödyntää ETL-rajapinnan tiedon poimintavaiheessa. Lähdejärjestelmän tietokannassa olevia aikaleima-kenttiä voidaan käyttää siis apuna niiden tietojen tunnistamisessa, jotka ovat muuttuneet edellisen suorituskerran jälkeen. Poimittavan tiedon rajaaminen aikaleima-kenttien avulla on tehokasta suorittaa suoraan Data Flow -osassa olevassa OLE DB Source -lähdekomponentissa. Tässä tapauksessa tiedon poiminta suoritetaan lähdekomponentissa SQL-kielen SELECT-komennon avulla, johon lisätään WHERE-ehto, missä poimittavaa tietoa rajataan lähdejärjestelmässä olevan aikaleima-kentän avulla. Poimittavien tietojen rajaaminen näin parantaa huomattavasti ETL-rajapinnan poimintavaiheen suorituskykyä. (Knight ym. 2008, 416-417.) Tämän lisäksi mikäli

poimintavaiheessa hyödynnetään SQL-kielen SELECT-komentoa, voidaan vaiheen suorituskykyä parantaa entisestään käyttämällä komennossa apuna T-SQL-kielen *NOLOCK*-avainsanaa. NOLOCK-avainsanaa käytettäessä SQL-kielen SELECT-kysely jättää siis huomioimatta mahdollisen transaction-käsittelyn olemassaolon, jonka vuoksi kysely on huomattavasti nopeampi. NOLOCK-avainsanaa käytettäessä on kuitenkin tiedostettava se, että tieto, jota SELECT-komennolla poimitaan, ei välttämättä ole eheässä muodossa. (Noor s.a. .)

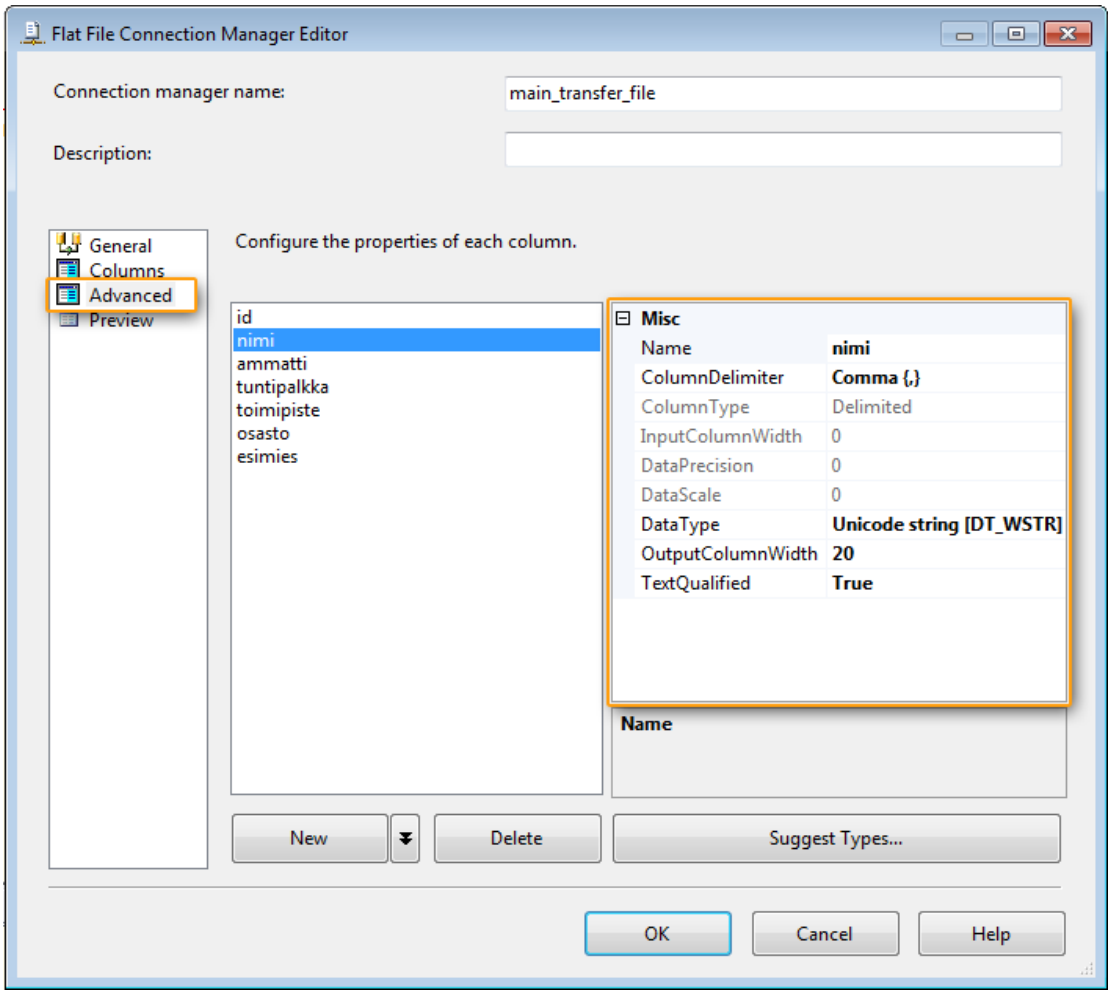
5.3.2 Tiedon poiminta siirtotiedostosta

Mikäli ETL-rajapinnan tietolähteenä käytetään siirtotiedostoa ja tiedon poiminta halutaan suorittaa SSIS-paketin sisässä, tulee poiminta toteuttaa paketin Data Flow -tehtäväkomponenttiin käyttäen apuna SSIS-ympäristön sisältämiä ominaisuuksia. Tiedon poimiminen SSIS-paketin sisältämiä ominaisuuksia hyödyntäen on yleisesti tehokkain ratkaisu, kun poimintaprosessi kohdistetaan siirtotiedostoon. On kuitenkin huomattava, että tiedon siirtäminen siirtotiedostosta tiedonsiirtorajapintaan voidaan toteuttaa myös SQL Server -tietokantaohjelmiston ominaisuuksien avulla ja tietyissä tilanteissa tämä on myös tehokas vaihtoehto. (Knight ym. 2008, 425.)

Siirtotiedostossa oleva tieto voidaan poimia SSIS-paketissa Data Flow -tehtäväkomponentin käyttämään muistiin *Flat File Source* -lähdekomponentin avulla. Jotta SSIS-tiedonsiirtomoottori osaa lukea tietyn siirtotiedoston oikein SSIS-paketin suorituksen aikana, on Flat File Source -lähdekomponenttiin määriteltävä aina, että mitä *Flat File Connection* -tietoyhteyttä lähdekomponentissa hyödynnetään. Mikäli tiedonsiirtorajapinnan tietolähteenä käytetään useampaa kuin yhtä siirtotiedostoa, tulee Flat File Source -lähdekomponentti määrittää hyödyntämään *Multiple Flat Files Connection* -tietoyhteyttä. (Knight ym. 2008, 136.)

Yleisesti tiedon poiminta siirtotiedostosta tiedonsiirtorajapintaan SSIS-paketin suorituksen aikana on hyvin nopea toimenpide. Tietolähteenä käytettävän siirtotiedoston rakenne määritellään siis melko tarkasti Flat File Connection -tietoyhteyteen jo SSIS-paketin kehityksen aikana. Siirtotiedoston rakenteen määrittely Flat File Connection -tietoyhteyteen ja sarakeliitosten luonti poiminnan kohteeseen vaativat kuitenkin yleisesti runsaasti työtä, sillä siirtotiedostoa koskevat määritykset tulee asettaa sarakekohtaisesti SSIS-pakettiin. (Knight ym. 2008, 136.) Tiedonsiirtorajapinnan tietolähteenä käytettävän siirtotiedoston rakenne on järkevää määritellä SSIS-pakettiin mahdollisimman tarkasti, sillä tämän avulla mahdollistetaan muun muassa virheellisten rivien havaitseminen siirtotiedostosta ilman erillisten

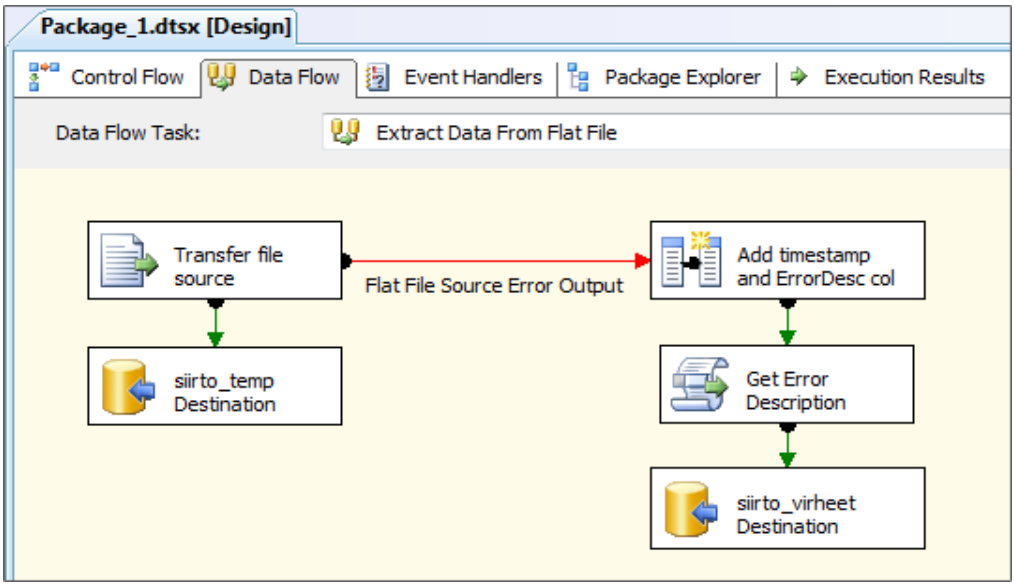
tiedon kelvollisuustarkastusten luontia. Tämän lisäksi on huomioitava, että esimerkiksi turhan suuret siirtotiedoston sarakkeiden enimmäismerkkimäärät kuluttavat turhaan tiedonsiirron käytettävissä olevia muistiresursseja. (Knight ym. 2008, 139.) Kuvasta yksitoista voidaan huomata BIDS-kehitysympäristön ikkuna, jonka kautta Flat File Connection -tietoyhteyden sarakemäärityksiä on mahdollista muokata. Kuvan yksitoista mukainen ikkuna aukaistaan BIDS-kehitysympäristön Connection Managers -ikkunassa olevan yksittäisen Flat File Connection -tietoyhteyden nimen kautta. Siirtotiedoston tarkkojen sarakemääritysten kannalta tärkein välilehti Flat File Connection -tietoyhteyden asetusten määritysikkunassa on *Advanced*-välilehti. Tämän välilehden oikeassa laidassa olevasta listasta voidaan löytää kulloinkin aktivoidun sarakkeen määritykset, kuten tietotyyppi (*DataType*) ja kentässä olevan arvon enimmäismerkkimäärä (*OutputColumnWidth*).



KUVA 11. Flat File Connection -tietoyhteyden asetusten määritysikkunan Advanced-välilehti.

Kun tiedonsiirtorajapinnan tietolähteenä käytettävän siirtotiedoston rakenne on määritelty Flat File Connection -tietoyhteyteen, voidaan itse tiedon poiminta toteuttaa

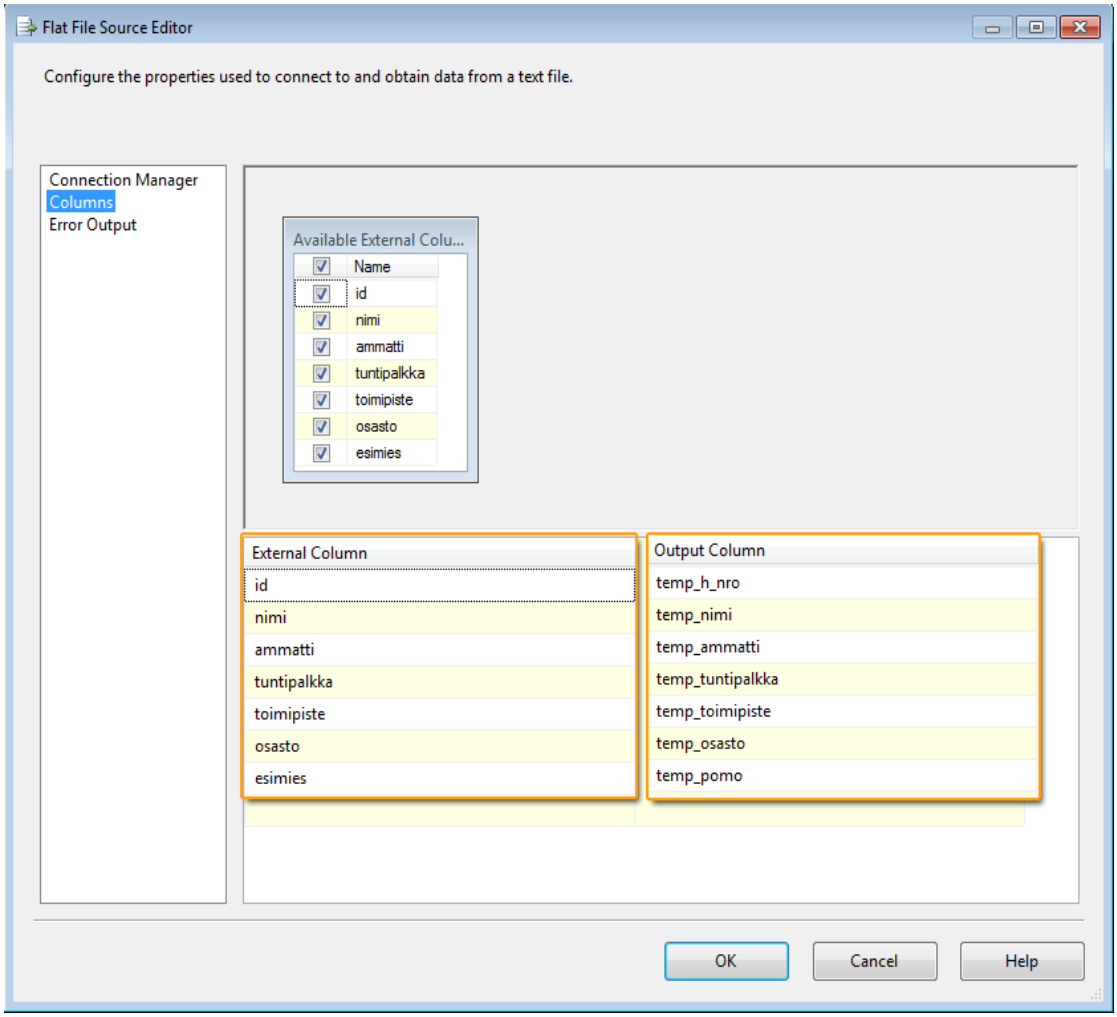
paketin Data Flow -tehtäväkomponenttiin. Kuvassa kaksitoista on havainnollistettu yksinkertaista esimerkkiä siitä, miten tiedon poiminta voidaan suorittaa SSIS-paketin Data Flow -tehtäväkomponentissa, kun tietolähteenä käytetään siirtotiedostoa. Kuvasta kannattaa huomata erityisesti se tapa, miten virheelliset rivit ohjataan suoraan ilman erillisiä kelvollisuustarkastuksia *Transfer file source* -nimisestä Flat File Source -lähdekomponentista kohti tiedonsiirron virheellisille riveille tarkoitettua taulua. Virheelliset rivit ohjautuvat siis *Flat File Source Error Ouput* -nimistä ulostulopuskuria pitkin kohti *siirto_virheet Destination* -kohdekomponenttia, missä rivit ladataan tietokannassa olevaan erilliseen tauluun. Kelvolliseksi havaitut rivit siirtyvät sen sijaan Transfer file source -komponentin oletus-ulostulopuskurista *siirto_temp Destination* -kohdekomponenttiin, missä tieto ladataan tiedonsiirtorajapinnan toteutukseen lukeutuvaan tiedonsiirron välitauluun.



KUVA 12. Tiedon poiminta siirtotiedosta SSIS-paketin Data Flow -tehtäväkomponentissa.

Kuvassa kolmetoista on havainnollistettu esimerkkiä siitä, miten Flat File Source -lähdekomponentin sarakkeet on järkevää nimetä, jotta sarakkeiden liitosten luonti Data Flow:n kohdekomponentissa olisi helpompaa. Tässä kuvassa esiintyvät *External Column* -sarakkeessa olevat nimet ilmentävät siis siirtotiedoston sarakkeiden nimiä, jotka on määritelty lähdekomponentin hyödyntämään Flat File Connection -tietoyhteyteen. *Output Column* -sarakkeessa olevat nimet ilmentävät sen sijaan nimiä, joilla sarakkeet esiintyvät Data Flow:n muistissa. Data Flow:n muistissa oleva yksittäinen sarake kannattaa nimetä siis siten, että se on samanniminen kuin se tiedonsiirron kohteena olevan taulun sarake, mihin tieto on tarkoitus siirtää siirtotiedoston sarakkeesta. Näin tiedonsiirron lähde- ja kohdesarakkeet liittyvät

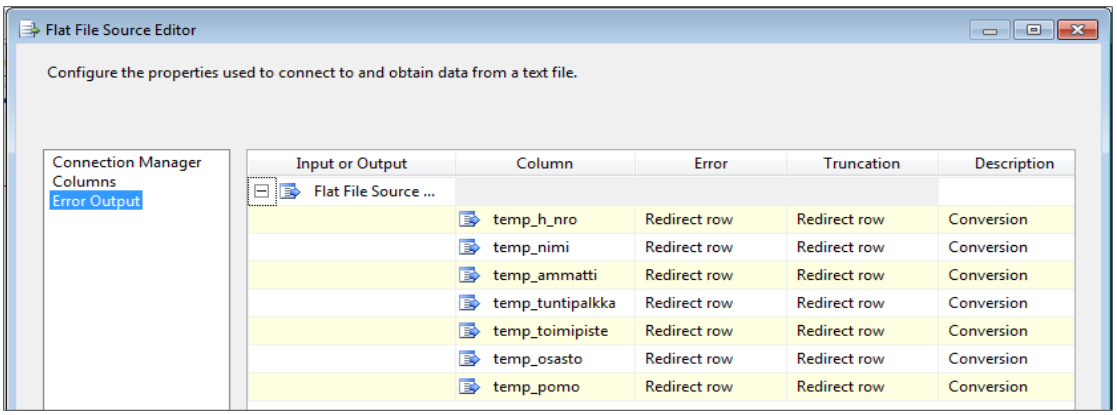
toisiinsa automaattisesti Data Flow:n tiedonsiirron kohdekomponentin sarakeliitoksissa.



KUVA 13. Flat File Source -komponentin Output Column -sarakenimien määrittäminen.

Kuvasta neljätoista voidaan huomata, miten Flat File Source -lähdekomponentin asetukset tulee määritellä, jotta tiedonsiirron tietolähteenä käytettävästä siirtotiedostosta voidaan havaita virheelliset rivit ilman erillisiä siirrettävän tiedon kelvollisuustarkastuksia. Flat File Source -lähdekomponentin asetusten määritysikkunan *Error Output* -välilehdellä voidaan siis määritellä sarakekohtaisesti, miten komponentti käsittelee rivejä, joissa on virheellinen arvo (*error*) tai arvo, joka ylittää sarakkeelle asetetun enimmäismerkkimäärän (*truncation*). Virheelliseksi arvoksi voidaan luokitella tässä tapauksessa esimerkiksi siirtotiedostossa oleva arvo, jota ei voida muuttaa siihen tietotyyppiin, mikä tiedolle on määritetty. Mahdollisia toimintoja, joita Flat File Source -lähdekomponentti voi suorittaa virheelliseksi havaitulle riville, on kolme erityyppistä. Komponentti voi siis ohjata virheellisen rivin erilliseen virheellisille riveille tarkoitettuun ulostulopuskuriin (*Redirect row*).

Komponentti voidaan määrittää myös hylkäämään virheelliseksi havaittu rivi (*Ignore failure*), jolloin rivissä olevaa tietoa ei käsitellä Data Flow:ssa. Kolmas mahdollinen toimenpide on, että mikäli komponentti havaitsee virheellisen rivin, merkitsee se komponentin suorituksen epäonnistuneeksi (*Fail component*). Tämä määritys on aina oletuksena aktiivisena Flat File Source -lähdekomponentin kaikissa sarakkeissa. On huomioitava, että mikäli Data Flow:n suorituksen aikana yksittäinen komponentti merkataan epäonnistuneesti suoritetuksi merkitsee se oletuksena kyseisen Data Flow:n suorituksen päättymistä.

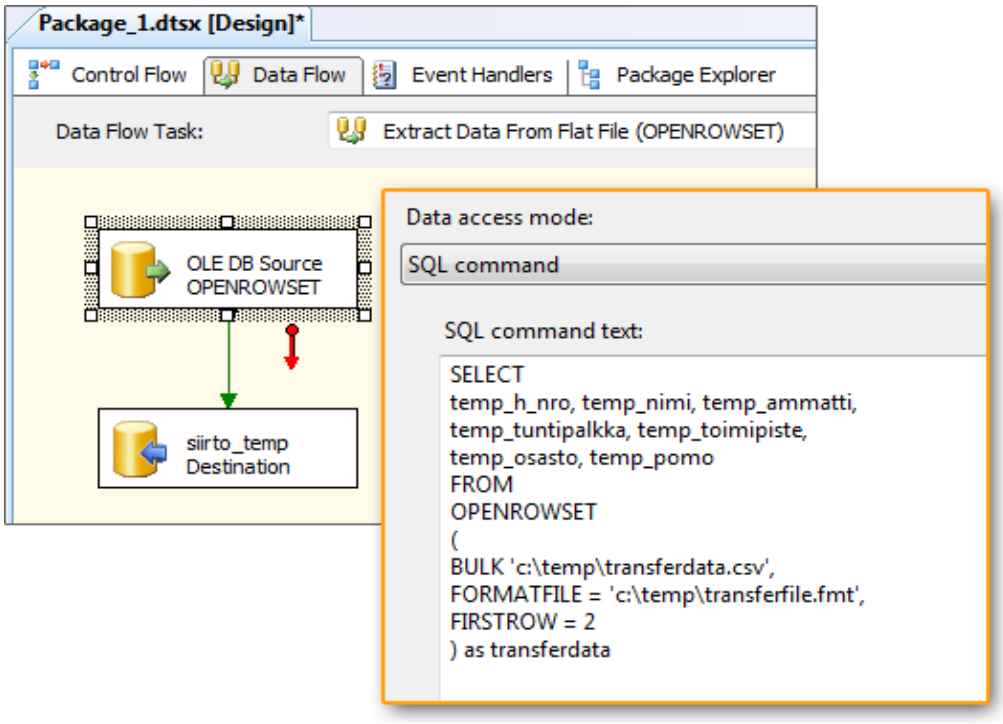


KUVA 14. Flat File Source -komponentin käsittelemän siirtotiedoston virheellisten tietojen hallinta Error Output -välilehdellä.

SSIS-ympäristön sisältämien siirtotiedostojen käsittelyominaisuuksien lisäksi on syytä huomioida, että myös SQL Server -tietokantaohjelmisto käsittää ominaisuuksia, joiden avulla on mahdollista toteuttaa tiedon poiminta siirtotiedostosta SSIS-tiedonsiirtorajapintaan. SQL Server -ohjelmisto käsittää siis T-SQL-kyselyn nimeltä *OPENROWSET*, jonka avulla esimerkiksi teksti-tyyppisten tiedostojen käsittely on mahdollista suoraan SQL Server -tietokantamoottorin yhteydessä. Mikäli tiedon poiminta suoritetaan SSIS-paketin Data Flow -tehtäväkomponentissa käyttäen apuna *OPENROWSET*-kyselyä, on Data Flow:n lähdekomponenttina käytettävä *OLE DB Source* - tai *ADO.NET Source* -komponenttia. (Knight ym. 2008, 425, 428.)

Kuvassa viisitoista on havainnollistettu, miten yksittäisessä siirtotiedostossa oleva tieto voidaan poimia Data Flow -tehtäväkomponentin käyttämään muistiin OLE DB Source -lähdekomponentissa käyttäen apuna OPENROWSET-kyselyä. OPENROWSET-kyselyn *BULK*-argumentti määrittää tiedoston, josta tieto poimitaan. Kyselyn argumentissa *FORMATFILE* määritellään *fmt*-tyyppinen tiedosto, joka pitää sisällään kyselyn tietolähteenä käytettävän siirtotiedoston rakenteen. Siirtotiedoston rakenne määritellään XML-muodossa ja rakenteen määrittelevän fmt-tiedoston luonnissa voidaa käyttää apuna SQL Server -ohjelmistoon liittyvää *bcp utility* -

sovellusta. Mikäli OPENROWSET-kyselyn avulla käsitellään teksti-tyyppistä tiedostoa, on tiedoston rakenne määriteltävä aina fmt-tiedostoon. Kuvassa viisitoista esitetyn OPENROWSET-kyselyn viimeinen argumentti *FIRSTROW* määrittää, että miltä siirtotiedoston riviltä tiedon poiminta aloitetaan. (Microsoft s.a. .)



KUVA 15. Siirtotiedostossa olevan tiedon poimiminen Data Flow -tehtäväkomponentin käyttämään muistiin käyttäen apuna OPENROWSET-kyselyä ja OLE DB Source -lähdekomponenttia.

OPENROWSET-kyselyn hyödyntäminen SSIS-paketissa siirtotiedostojen käsittelyssä on tehokasta erityisesti silloin, kun siirtotiedostossa oleva tieto täytyy lajitella tai liittää tietokannan muussa taulussa olevaan tietoon välittömästi tiedon poiminnan jälkeen. Mikäli tiedon poiminta suorittaisiin esimerkiksi Flat File Source -lähdekomponentin avulla, tulisi tiedon lajittelu tai liittäminen muuhun tietokannassa olevaan tietoon toteuttaa erillisten muunnoskomponenttien avulla. OPENROWSET-kysely muodostaa siis siirtotiedostossa olevasta tiedosta ikään kuin virtuaalisen näkymän, missä olevaa tietoa voidaan käsitellä samalla tavalla kuin relaatiotietokannan taulussa olevaa tietoa. Tästä voidaan siis ymmärtää, että OPENROWSET-kyselyn poimimaa tietoa voidaan käsitellä suoraan esimerkiksi SQL-kielen *ORDER BY, JOIN* tai *UNION* lausekkeiden avulla. Näiden lausekkeiden avulla tieto voidaan siis tarvittaessa lajitella ja liittää muuhun tietoon yhden SQL-kielen SELECT-kyselyn sisässä, joka on huomattavasti tehokkaampaa kuin näiden samojen toimenpiteiden suorittaminen erillisten SSIS-paketin Data Flow -muunnosten avulla. (Knight ym. 2008, 428.)

5.4 Tiedon muuntaminen

SSIS-ympäristö käsittää kahden tyyppisiä muunnoskomponentteja: synkronisia (eng. *synchronous*) ja asynkronisia (eng. *asynchronous*). Yleisimmin SSIS-ympäristössä käytettyjä muunnoskomponentteja ovat synkroniset komponentit. Esimerkiksi SSIS-ympäristön *Data Conversion* -muunnoskomponentti on synkroninen, sillä komponenttiin sisääntulevien rivien määrä ja järjestys on sama kuin siitä ulostulevien. Toisin sanottuna komponentin ulostulo-puskuri on siis synkroninen sisääntulo-puskurin kanssa. Sykroniset muunnoskomponentit toimivat tarkemmin tarkasteltuna siten, että jokainen rivi, joka kuuluu komponentin käsittelemään muistipuskuriin, käsitellään komponentissa itsenäisesti. Kun komponentti käsittelee lisäksi yksittäistä muistipuskurissa olevaa riviä, se ei ole tietoinen muista riveistä, joita muistipuskurissa on. Tämän vuoksi synkroniset muunnoskomponentit ovat usein nopeita ja varaavat pienemmän määrän muistia käyttöönsä. (Microsoft s.a. ;Knight ym. 2008, 147.)

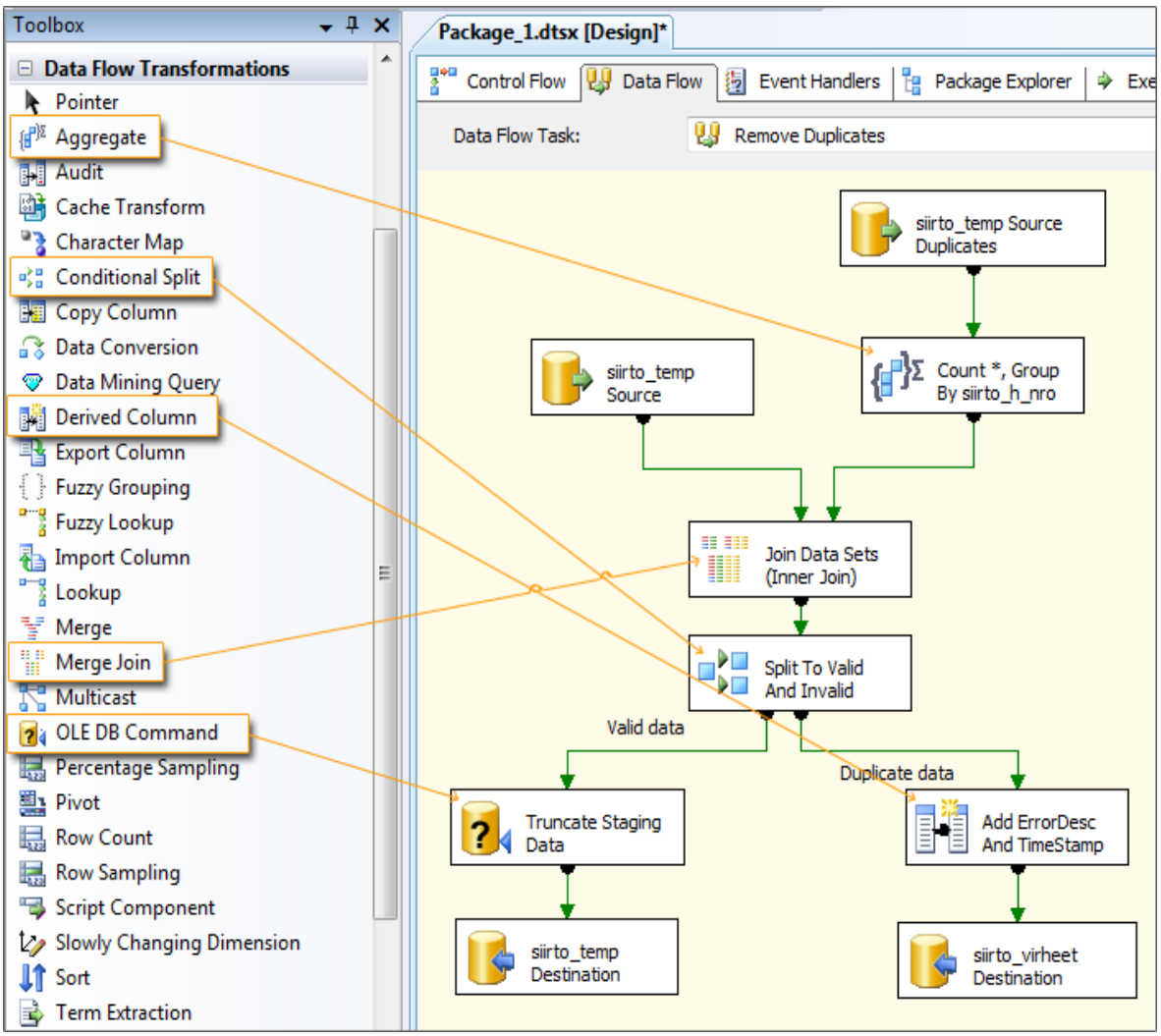
Asynkroniset muunnoskomponentit ovat suorituskyvyltään usein hitaampia ja varaavat huomattavan määrän muistia käyttöönsä. Tämä johtuu siitä, että tämän tyyppiset muunnoskomponentit luovat käsiteltävästä muistipuskurista osittaisen (eng. *partial blocking*) tai täyden (eng. *fully blocking*) kopion omaan käyttöönsä. Tämän lisäksi tietyt asynkroniset muunnoskomponentit luovat uusia ulostulo-puskureita siitä tiedosta, mitä komponentin käsiteltäväksi siirtyy sisääntulo-puskurin kautta. SSIS-ympäristön *Union All* -muunnoskomponentti on esimerkiksi asynkroninen, sillä se muodostaa käsitteltävästä tiedosta uusia erillisiä ulostulo-puskureita. Myös *Sort*-muunnoskomponentti on asynkroninen, sillä sen ulostulo-puskurin kautta siirtyvä tieto poikkeaa siitä tiedosta, mikä tulee sisään komponenttiin sen sisääntulo-puskurin kautta. *Sort* -komponentti muodostaa lisäksi täyden kopion käyttöönsä tiedosta, jota se käsittelee. Tästä on syytä huomioida, että mikäli Sort-muunnoskomponenttiin sisääntulo-puskurin kautta tuleva tieto on kooltaan 250MB, kuluttaa komponentti käsitellessään tietoa RAM-muistia yhteensä 500MB verran. (Microsoft s.a. ;Knight ym. 2008, 147.)

SSIS-tiedonsiirtorajapinnan tiedon muunnokset suoritetaan SSIS-paketin Control Flow -pinnalle lisättävien Data Flow -tehtäväkomponenttien sisässä. Jotta SSIS-pakettiin kuuluvassa Data Flow -tehtäväkomponentissa voidaan suorittaa muunnoksia, tulee Data Flow:n sisältää vähintään yksi lähdekomponentti, jonka avulla Data Flow:n käyttämään muistiin poimitaan tietoa. Tämän jälkeen Data Flow -tehtäväkomponenttiin voidaan lisätä muunnoksia suorittavia komponentteja. Yhden

muunnoskomponentin avulla voidaan suorittaa yhden tyyppinen muunnos käsiteltävään tietoon (Knight ym. 2008, 146).

Kun uusi muunnoskomponentti lisätään BIDS-kehitysympäristössä osaksi Data Flow:ta, tulee lisättyyn muunnoskomponenttiin ensin määrittää se tieto, jota komponentin on tarkoitus käsitellä. Tämä tehdään liittämällä aiemmin Data Flow -tehtäväkomponenttiin luotu lähdekomponentti tai muunnoskomponentti ulostulo-puskurin kautta lisätyn muunnoskomponentin sisääntulo-puskuriin. Tämän jälkeen lisätyn komponentin toiminta voidaan konfiguroida halutun mukaiseksi sen sisältämien ominaisuuksien avulla. (Knight ym. 2008, 146-147.)

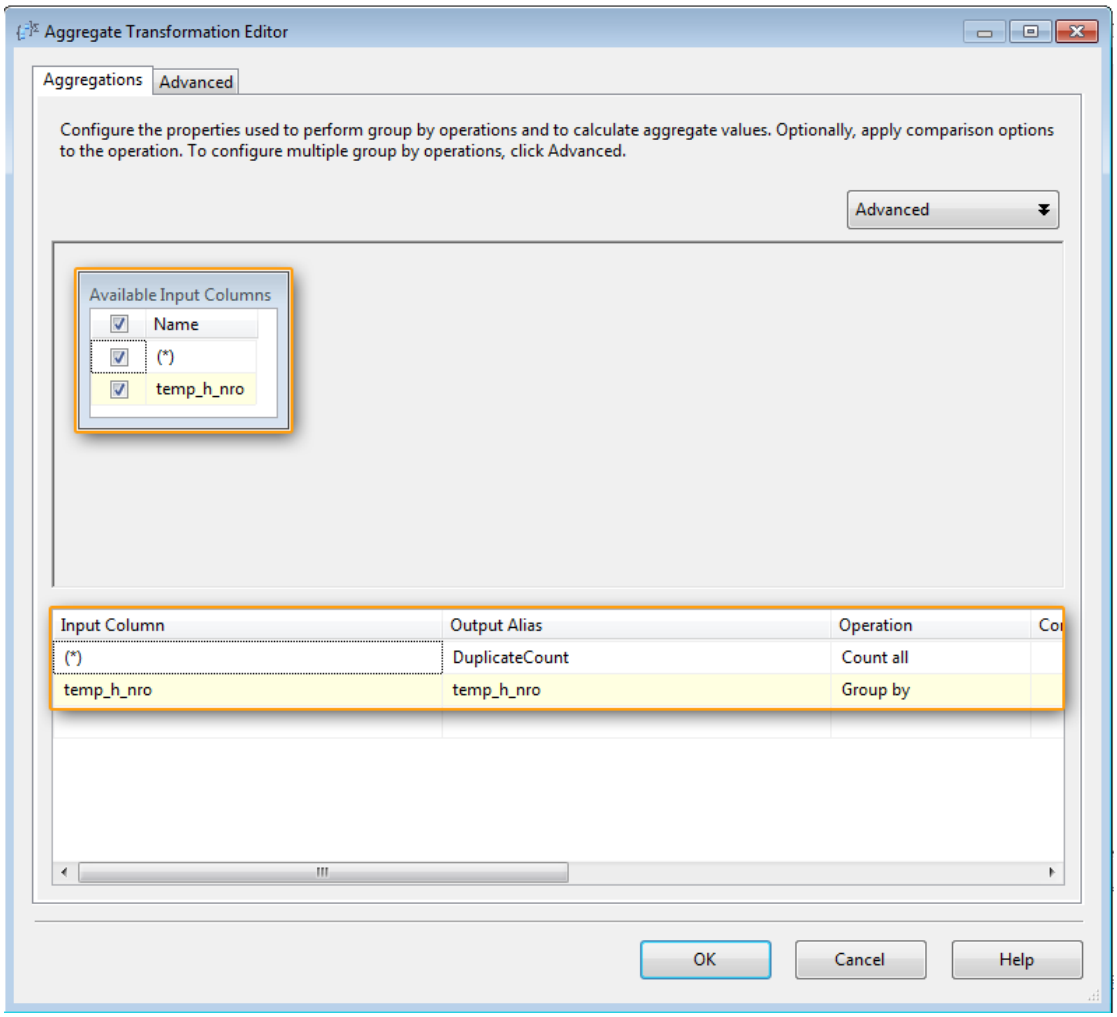
Seuraavassa esimerkissä havainnollistetaan sitä, miten SSIS-paketin Data Flow -tehtäväkomponentiin lisättävien muunnoskomponenttien avulla voidaan suorittaa siirrettävien tietojen duplikaattitarkastus. Tässä esimerkissä on tavoitteena, että duplikaattitarkastuksessa virheelliseksi havaitut tiedot ohjataan omaan tauluun tietokannassa, jolloin virheellisiä tietoja on mahdollista tutkia myös tiedonsiirron päättymisen jälkeen. Duplikaattitarkastuksessa kelvolliseksi havaitut rivit ohjataan sen sijaan tiedonsiirron välitauluna käytettävään kohteeseen, mistä tiedot myös poimitaan tarkastettavaksi Data Flow:n alussa. Kuvassa kuusitoista on esitetty Data Flow -tehtäväkomponentin, missä duplikaattitarkastus suoritetaan, rakenne kokonaisuudessaan.



KUVA 16. SSIS-paketin Data Flow -tehtäväkomponentissa suoritettava siirrettäviin tietoihin kohdistuva duplikaattitarkastus. Kuvan vasemmassa osassa olevasta *Toolbox*-ikkunasta voidaan huomata kaikki ne muunnoskomponentit, jotka kuvan Data Flow käsittää.

Kuvassa kuusitoista esitetyn Data Flow -tehtäväkomponentin rakenteesta voidaan todeta, että jopa melko yksinkertaisen duplikaattitarkastuksen suorittaminen vaatii useamman eri tyyppisen muunnoskomponentin lisäämistä osaksi SSIS-paketin Data Flow:ta. Eri tyyppisten muunnoskomponenttien asetusten määrittäminen BIDS-kehitysympäristössä on kuitenkin yleisesti melko yksinkertaista. Kuvassa seitsemäntoista on esitetty ikkuna, jonka kautta SSIS-ympäristön *Aggregate*-muunnoskomponentin asetukset voidaan määrittää. Kuvassa esiintyvästä *Available Input Columns* -paneelista voidaan huomata Data Flow:n muistissa olevat sarakkeet, jotka saapuvat sisääntulo-puskurin kautta Aggregate-komponenttiin. Tästä paneelista aktivoidut sarakkeet siirtyvät kuvan alaosassa esitettyyn listaan, missä jokaisen valitun sarakkeen kohdalla voidaan määritellä, että onko sarake ryhmittelevä tekijä (*Group by*) vai halutaanko sarakkeessa oleviin arvoihin kohdistaa jokin laskuoperaatio. Kuvan seitsemäntoista alaosassa olevan listan sarakkeeseen *Output*

Alias voidaan määritellä nimi, joka edustaa saraketta, kun se siirtyy komponentin ulostulo-puskurin kautta takaisin Data Flow:n muistiin.



KUVA 17. SSIS-ympäristön Aggregate-muunnoskomponentin asetusten määrittämisikkuna.

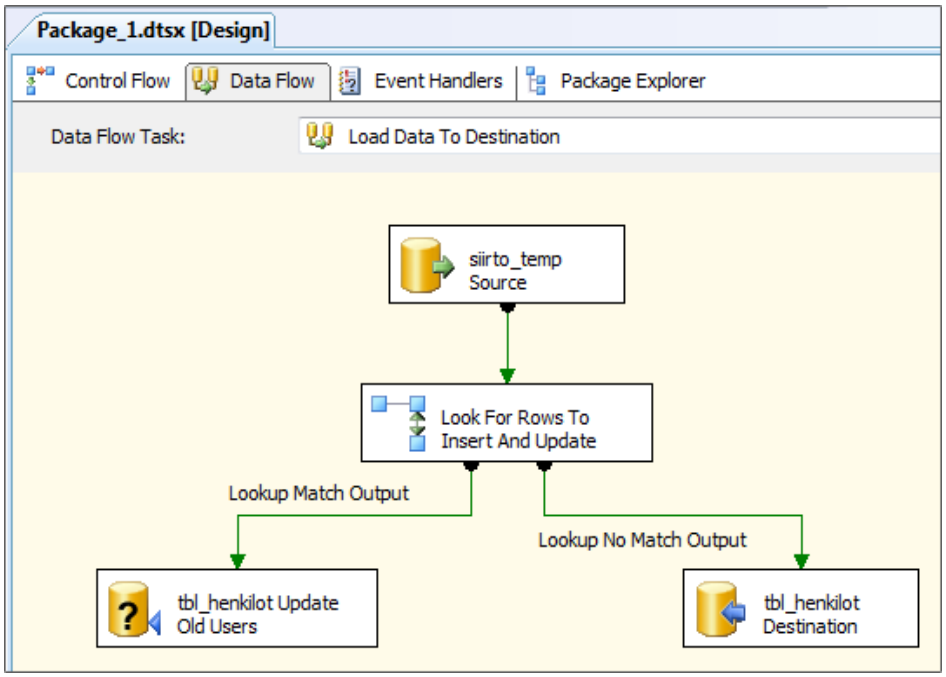
Aggregate-muunnoskomponentin avulla selvitetään kuvan kuusitoista esimerkissä, että kuinka monta kertaa rivi esiintyy Data Flow:n muistissa, kun muistissa olevat rivit ryhmitellään tietyn sarakkeen mukaisesti. Duplikaattitarkastuksen kannalta tämä tarkoittaa siis sitä, että mikäli tietty rivi esiintyy muistissa useamman kuin yhden kerran, voidaan tämä luokitella duplikaattitiedoksi. Varsinainen tutkimus siitä, että esiintyykö tietty arvo useamman kuin yhden kerran, tehdään kuvan kuusitoista Data Flow:ssa kuitenkin vasta *Conditional Split* -muunnoskomponentissa. Ennen tämän muunnoskomponentin suorittamista Data Flow:ssa suoritetaan *Merge Join* -muunnoskomponentti, jonka avulla ryhmitelty tieto ja tiedonsiirron välitaulusta noudettu varsinainen tieto yhdistetään toisiinsa *Inner Join* -liitoksella. Tämän avulla muodostetaan siis tietokokonaisuus, mikä käsittää kaiken tarvittavan tiedon tiedonsiirron välitaulusta ja lisäksi tiedon siitä, että kuinka monta kertaa yksittäisessä

rivissä oleva yksilöiväksi tarkoitettu tieto esiintyy tietokokonaisuudessa. Esimerkkinä käytetyn Data Flow:n lopussa suoritetaan vielä *OLE DB Command* -muunnoskomponentti, jonka avulla tyhjennetään tiedonsiirron välitaulussa oleva tieto ennen uuden tiedon latausta siihen, ja *Derived Column* -muunnoskomponentti, jonka avulla Data Flow:n muistissa oleviin virheellisiin riveihin lisätään aikaleima- ja virheen selite -sarake.

5.5 Tiedon lataus tiedonsiirron kohteeseen

Tiedon lataaminen tiedonsiirron kohteeseen suoritetaan SSIS-ympäristössä yleisesti SSIS-paketin Data Flow -tehtäväkomponentin sisässä käyttäen apuna tilanteeseen soveltuvaa tiedonsiirron kohdekomponenttia (eng. *destination*). Se, minkä tiedonsiirron kohdekomponentin avulla tiedon lataus suoritetaan, määräytyy siis tiedonsiirron latausprosessin luonteen mukaan – halutaanko tieto ladata esimerkiksi SQL Server -tietokantaan vai käyttöjärjestelmän tiedostojärjestelmään muodostettavaan siirtotiedostoon. Mikäli tieto ladataan SSIS-paketin Data Flow -osassa SQL Server -tietokantaan, voidaan lataus suorittaa esimerkiksi *OLE DB Destination* - tai *ADO NET Destination* -kohdekomponentin avulla. Tiedon lataaminen SSIS-paketin Data Flow -osassa esimerkiksi CSV-muotoiseen siirtotiedostoon suoritetaan sen sijaan *Flat File Destination* -kohdekomponentin avulla. SSIS-ympäristö mahdollistaa tiedon laatamisen muun muassa OLE DB -yhteensopiviin tietolähteisiin, tiedostoihin ja Analysis Services -kohteisiin (Knight ym. 2008, 142).

Kuvassa kahdeksantoista on esitetty SSIS-pakettiin kuuluvan Data Flow -tehtäväkomponentin, jonka avulla suoritetaan tiedon lataus tiedonsiirtorajapintaan lukeutuvasta välitaulusta varsinaiseen tiedonsiirron kohdetauluun tietokannassa, rakenne. Kuvan kahdeksantoista Data Flow:ssa käytetään apuna *Lookup*-muunnoskomponenttia sen selvittämiseen, että mitkä tiedonsiirron välitaulussa olevat rivit päivitetään ja mitkä lisätään uutena tiedonsiirron latauksen kohteena olevaan tietokannan tauluun. Tiedon konkreettinen lataus suoritetaan kuvan kahdeksantoista Data Flow:ssa siten, että uudet rivit lisätään tiedonsiirron kohdetauluun OLE DB Destination -kohdekomponentin avulla ja rivien päivitys suoritetaan OLE DB Command -muunnoskomponentin avulla.



KUVA 18. Yksinkertainen esimerkki SSIS-paketin Data Flow -tehtäväkomponentissa suoritettavasta tiedon latausprosessista.

Vaikka kuvassa kahdeksantoista esitetyn Data Flow:n rakenne näyttää selkeältä ja toimivalta, voidaan siitä silti löytää yksi ongelmakohta, joka saattaa aiheuttaa ongelmia, mikäli ladattavan tiedon määrä Data Flow:ssa on suuri. OLE DB Command -muunnoskomponenttia ei siis ole tehokasta hyödyntää Data Flow:n lopussa tiedon lataamisessa tiedonsiirron kohteeseen. OLE DB Command -muunnoskomponentti toimii tarkemmin tarkasteltuna siten, että se suorittaa jokaista riviä, joka siirtyy komponentin sisääntulo-puskurin kautta komponentin käsiteltäväksi, kohden erillisen SQL-kyselyn (Micorosft s.a.). ETL-rajapintojen kehityksessä tulisi siis aina muistaa, että tietoa tulee pyrkiä käsitelemään aina ensisijaisesti joukoittain (eng. *batch processing*) eikä niinkään rivi kerrallaan (Mundy ym. 2011, 197). Kuvassa kahdeksantoista esitetyn Data Flow:n tapauksessa tehokkaampaa olisi siis, että latausvaiheen rivien päivitysprosessi toteutettaisiin esimerkiksi SQL-kyselyn avulla, joka suoritettaisiin SSIS-paketin Control Flow -pinnalle lisättävässä *Execute SQL Task* -tehtäväkomponentissa. Vaikka tämä tapa on tehokkaampi, on kuitenkin huomioitava, että SQL-kyselyn suorittamista SSIS-paketin Control Flow -pinnalla edellyttää, että käsiteltävä tieto on siirretty aiemmin erilliseen tiedonsiirron välitauluun, mistä tiedon päivittäminen tiedonsiirron kohteeseen suoritetaan (Knight ym. 2008, 175). Välitaulun hyödyntäminen tiedonsiirrossa ei välttämättä ole kuitenkaan aina mahdollista.

Kuten Data Flow:n lähdekomponenttien myös kohdekomponenttien käyttöä edellyttää tietoyhteyden lisääminen SSIS-pakettiin. Tarkemmin tarkasteltuna ainoa suuri

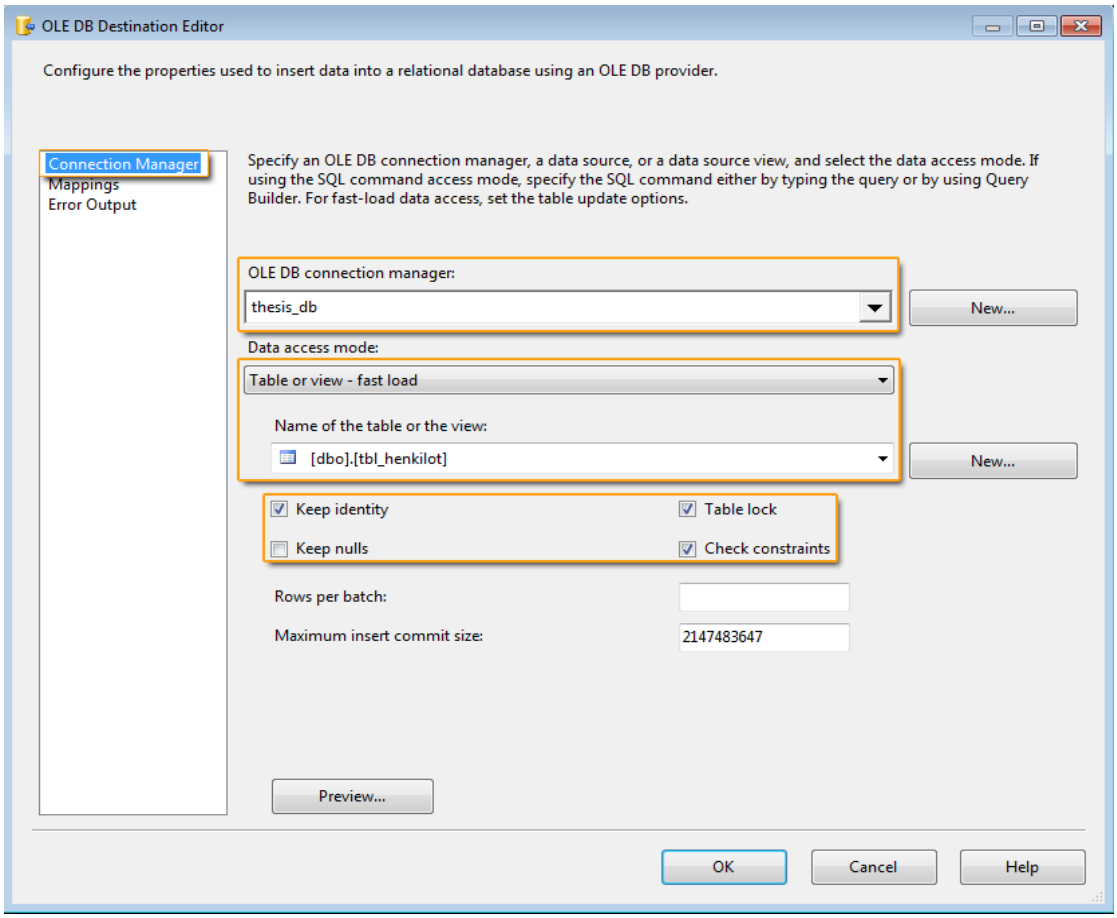
eroavaisuus Data Flow:n lähdekomponenttien ja kohdekomponenttien välillä on, että kohdekomponentteihin tulee määritellä aina sarakeliitokset, joiden mukaan tieto ladataan Data Flow:n muistista tiedonsiirron kohteeseen. (Knight ym. 2008, 142.) Mikäli tiedonsiirron latauksen kohteena on siis tietokanta, sarakeliitokset luodaan Data Flow:n muistissa olevien sarakkeiden ja tiedon latauksen kohteena olevan tietokannan taulun sarakkeiden välillä. Jos tietoa ladataan sen sijaan CSV-siirtotiedostoon, luodaan sarakeliitokset siirtotiedoston sarakkeiden, jotka määritellään Flat File Connection -tietoyhteyteen, ja Data Flow:n muistissa olevien sarakkeiden välillä.

SSIS-ympäristön OLE DB Destination -kohdekomponentin asetusten määrittämisessä on kaksi keskeistä vaihetta. Ensimmäisessä vaiheessa kohdekomponentin asetukset määritellään komponentin asetusten määrittelyikkunan *Connection Manager* -välilehdellä (katso kuva yhdeksäntoista). Komponentin asetusten määrittelyn toinen vaihe käsittää sarakeliitosten luonnin Data Flow:n muistissa olevien sarakkeiden ja tiedon latauskohteessa olevien sarakkeiden välille (katso kuva kaksikymmentä).

OLE DB Destination -kohdekomponentin Connection Manager -välilehdellä määritellään, että miten kohdekomponentin avulla on tarkoitus ladata tietoa, ja mihin kohteeseen tieto ladataan. Tämän välilehden *OLE DB connection manager* -valintalistaan aktivoidaan, että minkä SSIS-pakettiin perustetun tietoyhteyden kautta kohdekomponentti kytkeytyy tiedon latauskohteeseen. *Data access mode* -valintalistaan tulee aktivoida lisäksi, että millä tavalla tiedon latauksen kohdetaulu määritellään komponenttiin. Tiedon latauksen kohde voidaan määritellä SSIS-paketissa olevan muuttujan välityksellä, jolloin muuttujassa välitetään latauksen kohteena olevan taulun tai näkymän nimi kohdekomponentille. Tämän lisäksi latauksen kohde voidaan määritellä valitsemalla Data access mode -valintalistaan suoraan latauksen kohdetaulu tai -näkymä valittavissa olevista vaihtoehdoista. Latauksen kohde on mahdollista määritellä myös SQL-kyselyn avulla, jolloin kysely määritää yhden taulun tai näkymän ja tarvittavat sarakkeet latauksen kohteeksi.

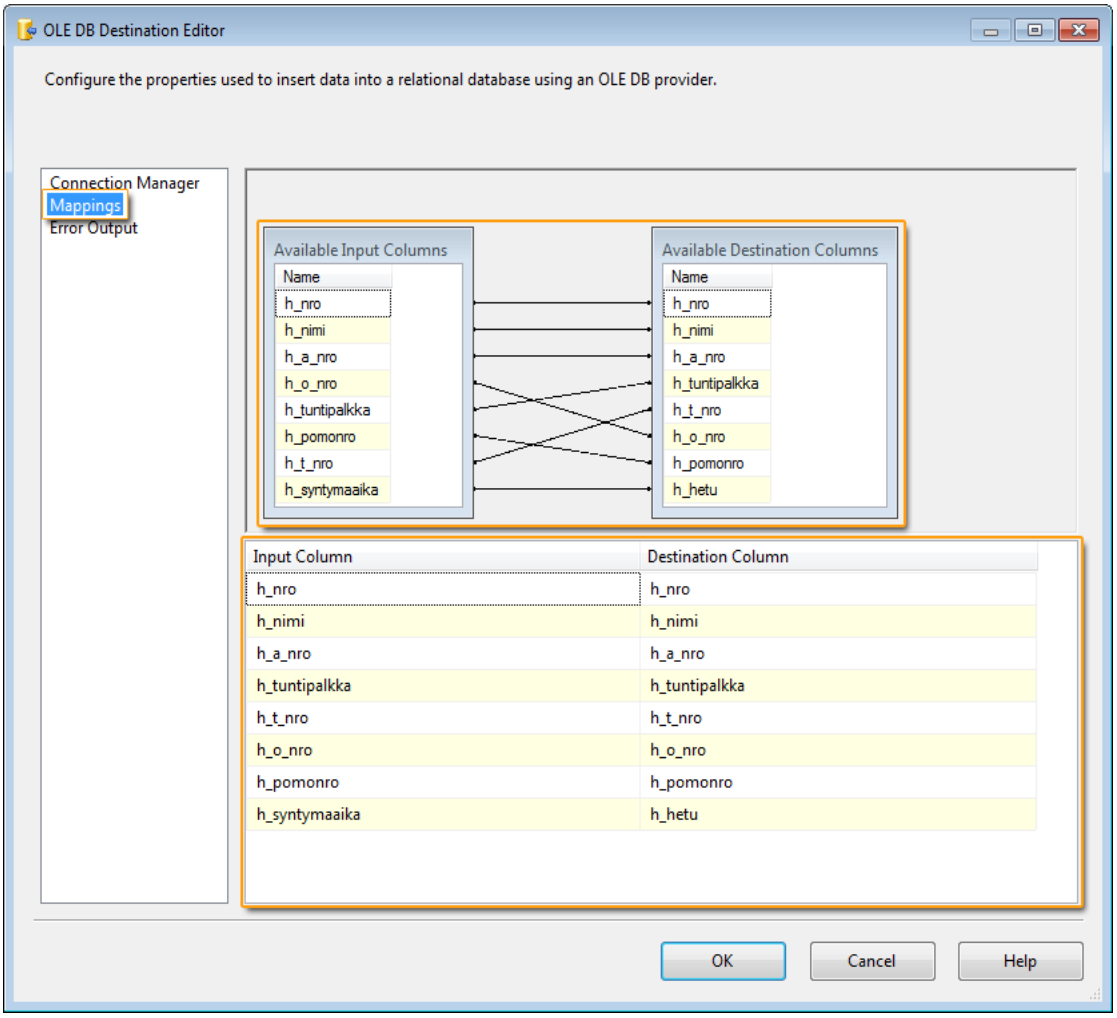
OLE DB Destination -kohdekomponentin asetusten määritysikkunan Connection Manager -välilehdellä, on mahdollista määritellä, että käytetäänkö latauksessa apuna SSIS-ympäristön sisältämää *fast load* -ominaisuutta. Mikäli latauksessa käytetään apuna tätä ominaisuutta, voidaan kohdekomponenttiin määritellä suoraan, että ladataanko komponentin avulla tietoa tietokannan taulun automaattisesti kasvavaan perusavain-kenttään (*Keep identity*), muodostaako kohdekomponentti transaction-

lukon latauksen kohteena olevaan tauluun (*Table lock*), säilyttääkö kohdekomponentti NULL-arvot tiedoissa kaikissa tapauksissa (*Keep nulls*) tai tarkastaako kohdekomponentti aina mahdollisten kohdetauluun luotujen eheys-sääntöjen olemassaolon (*Check constraints*).



KUVA 19. SSIS-paketin Data Flow -tehtäväkomponenttiin kuuluvan OLE DB Destination -kohdekomponentin perusasetusten määrittäminen.

OLE DB Destination -kohdekomponentin käsittämät sarakeliitokset luodaan kohdekomponentin asetusten määrittelyikkunan *Mappings*-välilehdellä. Sarakeliitokset luodaan OLE DB Destination -kohdekomponentissa Data Flow:n muistissa olevien sarakkeiden (*Input Columns*) ja latauksen kohteena olevan tietokannan taulun sarakkeiden (*Destination Columns*) välillä. Yksittäisten sarakkeiden liittäminen toisiinsa voidaan suorittaa raahaa-pudota-menetelmällä komponentin asetusten määritysikkunan Mappings-välilehden yläosassa olevien *Available Input Columns* ja *Available Destination Columns* listojen välillä. Sarakeliitokset voidaan luoda myös välilehden alaosassa olevassa listassa valitsemalla Input Column -sarakkeessa olevalle riville se Data Flow:n muistissa oleva sarake, joka halutaan liittää samalla rivillä olevaan latauksen kohdesarakkeeseen.



KUVA 20. SSIS-paketin Data Flow -tehtäväkomponenttiin kuuluvan OLE DB Destination -kohdekomponentin sarakeliitosten määrittäminen.

5.6 .NET-skriptien suorittaminen SSIS-paketissa

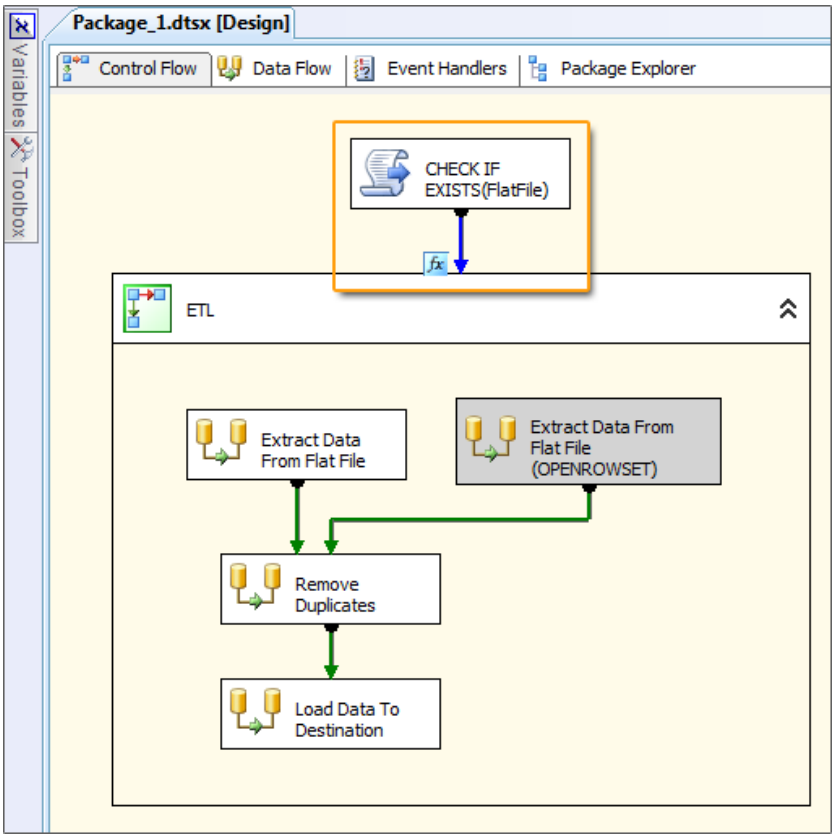
SSIS-ympäristö mahdollistaa räätälöidyn toiminnallisuuden kehittämisen ETL-tiedonsiirtorajapinnan yhteyteen C#.NET ja VB.NET ohjelmointikielten avulla. Ohjelmoimalla kehitettävä toiminnallisuus sijoitetaan SSIS-paketin Control Flow -osaan *Script Task* -tehtäväkomponenttiin tai Data Flow -osaan *Script Component* -komponenttiin. Ohjelmakoodi kirjoitetaan BIDS-kehitysympäristössä käyttäen apuna *Visual Studio Tools For Applications* (VSTA) -kehitysympäristöä, joka on käytännössä kevennetty versio Visual Studio 2008 -kehitysympäristöstä. SSIS-pakettiin kehitettävän ohjelmakoodin kirjoittamisen kannalta merkittävää on, että VSTA-kehitysympäristö tukee esimerkiksi Microsoft:n kehittämää .NET-ohjelmointikielten IntelliSense-ominaisuutta. (Knight ym. 2008, 295.) Tämän lisäksi VSTA-ympäristö mahdollistaa esimerkiksi *DLL*- ja *Web Service* -viittausten lisäämisen kehitetävään

ohjelmakoodiin Visual Studio 2008 -kehitysympäristöstä tutun toiminnallisuuden välityksellä.

SSIS-ympäristön Script Task -tehtäväkomponentin ja Script Component -komponentin hyödyntäminen ETL-tiedonsiirrossa on kannattavaa, mikäli SSIS-pakettiin halutaan lisätä toiminnallisuutta, jota ei ole mahdollista luoda valmiina SSIS-ympäristössä olevien muiden komponenttien avulla (Knight ym. 2008, 294). Tämän lisäksi SSIS-ympäristön Script Task ja Script Component komponentien käyttö on erityisen hyödyllistä, mikäli ohjelmakoodin avulla halutaan suorittaa useampia toimintoja samalla kertaa sen sijasta, että toiminnot suoritettaisiin useamman erilaisen tehtävä- tai muunnoskomponentin avulla (Microsoft s.a.). On kuitenkin huomioitava, että ETL-rajapinnan kehittäminen on järkevää suorittaa ensisijaisesti SSIS-ympäristön tarjoamien valmiiden tehtävä- ja muunnoskomponenttien avulla. Tätä periaatetta noudattamalla voidaan vaikuttaa esimerkiksi siihen, että kehitettävää ETL-rajapintaa on helpompaa ylläpitää sen käyttöönoton jälkeen.

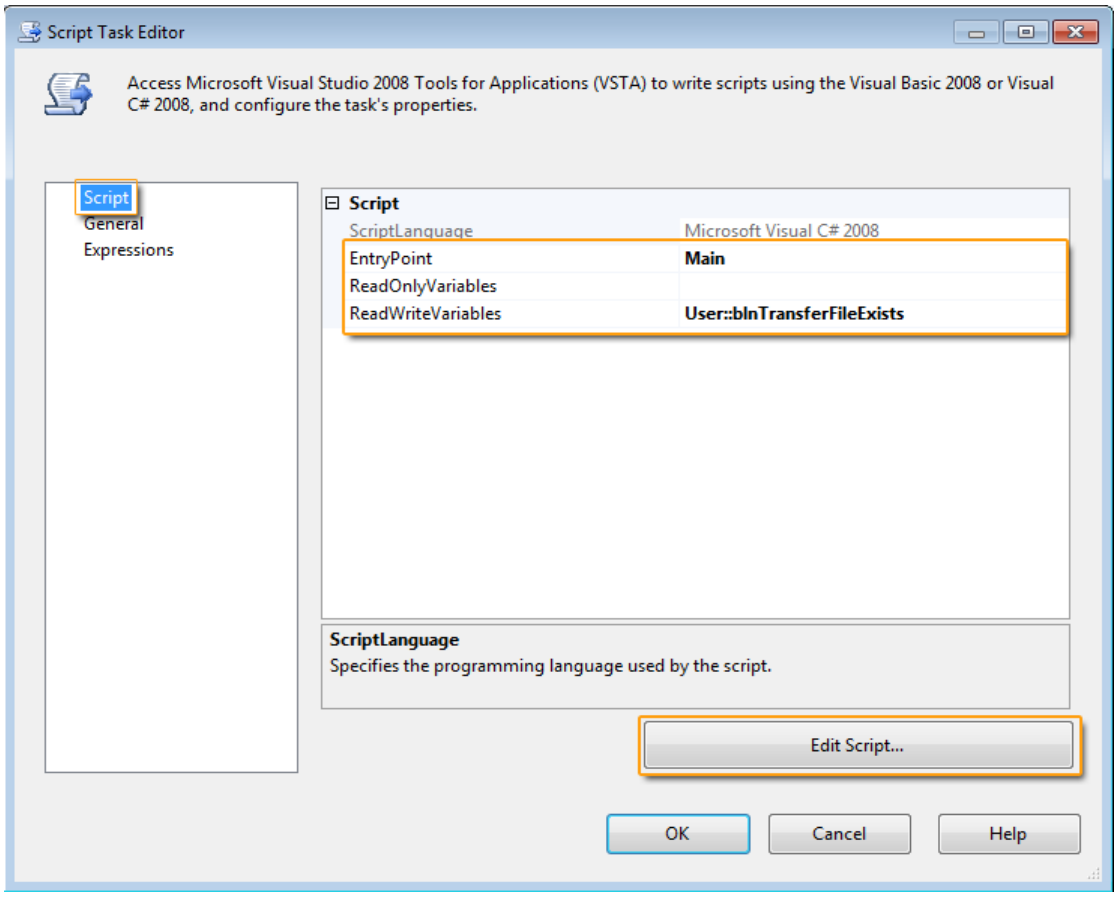
5.6.1 Skriptien suorittaminen Script Task -tehtäväkomponentissa

Script Task -tehtäväkomponenttia on kannattavaa hyödyntää SSIS-paketissa, mikäli paketin Control Flow -osassa halutaan suorittaa monimutkaisempia toimenpiteitä, jotka vaikuttavat paketin suorituksen etenemiseen. Tämän lisäksi Script Task -tehtäväkomponentin avulla voidaan hallita tehokkaasti SSIS-paketissa olevien muuttujien (eng. *variable*) arvoja. (Knight ym. 2008, 295.) Hyvin yleinen ETL-tiedonsiirroissa suoritettava toimenpide, minkä suorittamisessa SSIS-ympäristön Script Task -tehtäväkomponenttia voidaan hyödyntää, on esimerkiksi tiedonsiirron tietolähteenä käytettävän siirtotiedoston esiintyvyyden tarkastaminen ennen tiedonsiirron eri vaiheiden aloittamista. Kuvassa kaksikymmentäyksi on havainnollistettu, miten Script Task -tehtäväkomponentti, jonka avulla suoritetaan tiedonsiirtorajapinnan tietolähteenä käytettävän siirtotiedoston esiintyvyyden tarkastaminen, voidaan sijoittaa SSIS-pakettiin. Kuvassa kaksikymmentäyksi oleva Script Task -tehtäväkomponentti muuntaa SSIS-paketissa olevan *Boolean*-tyyppisen muuttujan arvon perustuen tiedonsiirron tietolähteenä käytettävän siirtotiedoston esiintyvyyteen. Kuvassa kaksikymmentäyksi havainnollistetun Script Task -tehtäväkomponentin jälkeisessä ehtoliitoksessa suoritetaan tarkastus siitä, että onko tietolähteenä käytettävän siirtotiedoston esiintyvyyden osoittaman Boolean-muuttujan arvo *True*. Mikäli tämän muuttujan arvo on *True*, etenee SSIS-paketin suoritus *ETL*-nimiseen säiliöön. Muussa tapauksessa SSIS-paketin suoritus päättyy, koska paketin suoritus ei voi enää edetä.



KUVA 21. Tiedonsiirron tietolähteenä käytettävän siirtotiedoston esiintyvyyden tarkastavan Script Task -tehtäväkomponentin sijoittuminen SSIS-pakettiin.

Kuvassa kaksikymmentäkaksi on esitetty BIDS-kehitysympäristön ikkunan osa, jonka kautta Script Task -tehtäväkomponentin tärkeimmät asetukset voidaan määrittää. Kuvasta voidaan huomata *ReadOnlyVariables*- ja *ReadWriteVariables*-kentät, joihin voidaan määrittää SSIS-paketin muuttujat, joita Script Task -tehtäväkomponenttiin lisättävässä ohjelmakoodissa halutaan käsitellä. Mikäli muuttuja lisätään ReadOnlyVariables-kenttään, on muuttujassa oleva tieto ainoastaan luettavissa komponentin ohjelmakoodissa. Vastaavasti ReadWriteVariables-kenttään lisättyjen muuttujien arvoja voidaan lukea ja muuttaa. Kuvasta kaksikymmentäkaksi voidaan huomata, että Script Task -tehtäväkomponentin ReadWriteVariables-kenttään on asetettu yksi SSIS-paketin muuttuja, jonka arvoa muokataan tiedonsiirtorajapinnan hyödyntämän siirtotiedoston esiintyvyyden perusteella. Script Task -tehtäväkomponentin asetusten määritysikkunan *EntryPoint*-kenttään voidaan sen sijaan määritellä sen metodin nimi, mistä tehtäväkomponenttiin lisätyn ohjelmakoodin suorittaminen aloitetaan. *Edit Script* -painiketta klikkaamalla voidaan siirtyä VSTA-kehitysympäristöön kirjoittamaan komponentiin lisättävää ohjelmakoodia.



KUVA 22. Script Task -tehtäväkomponentin asetusten määritysikkunan *Script*-välilehti.

Kuvassa kaksikymmentäkolme on havainnollistettu kuvassa kaksikymmentäyksi esiintyvän Script Task -tehtäväkomponentin sisältämää ohjelmakoodia. Ohjelmakoodi on sijoitettu tässä tapauksessa Script Task -tehtäväkomponentin *Main*-metodiin, mistä ohjelmakoodin suoritus aloitetaan oletuksena. Kuvassa kaksikymmentäkolme olevasta ohjelmakoodista kannattaa huomioida erityisesti, miten ohjelmakoodissa esiintyvän *Dts*-objektin eri ominaisuuksia on käytetty hyväksi. Dts-objekti on siis SSIS ohjelmointirajapinnan *Microsoft.SqlServer.Dts.Tasks.ScriptTask.ScriptObjectModel*-luokan instanssi. Tarkemmin tarkasteltuna Dts-objekti on *Microsoft.SqlServer.Dts.Tasks.ScriptTask.VSTARTScriptObjectModelBase*-luokan ominaisuus ja sen tarkoituksena on toimia rajapintana Script Task -tehtäväkomponentin ja suoritettavan SSIS-paketin eri ominaisuuksien, kuten tietoyhteyksien, tapahtumien, muuttujien, logien ja transaction-yhteyden, välillä. (Knight & ym. 2008, 309-310; Microsoft s.a. .) Kuvasta kaksikymmentäkolme voidaan huomata, että miten Dts-objektin *Variables*-ominaisuutta on käytetty hyväksi, kun SSIS-paketissa olevan muuttujan arvoa halutaan muokata perustuen SSIS-tiedonsiirtorajapinnan tietolähteenä käytettävän siirtotiedoston esiintyvyyteen.

```
public void Main()
{
    // TODO: Add your code here

    try
    {
        // "Dts.Connections"-kokoelma käsittää kaikki tietoyhteydet,
        // jota SSIS-pakettiin on määritelty. Tietoyhteyden nimellä
        // voidaan viitata haluttuun tietoyhteyteen.
        string strFlatFilePath =
            Dts.Connections["main_transfer_file"].ConnectionString;

        // Skripteissä voidaan hyödyntää .NET-luokkia.
        if (!System.IO.File.Exists(strFlatFilePath))
        {
            // "Dts.Variables"-kokoelma käsittää kaikki muuttujat,
            // jotka Script Task -tehtäväkomponenttiin on asetettu.
            // Muuttujaan viitataan sen nimellä.
            Dts.Variables["blnTransferFileExists"].Value = false;

            // "Dts.Events"-kokoelma käsittää kaikki tapahtumat,
            // joita SSIS-paketissa voidaan laukaista.
            Dts.Events.FireError(0, "CHECK IF EXISTS(FlatFile)",
                "The flat file used as a data source was not found.", "", 0);
        }
        else
            Dts.Variables["blnTransferFileExists"].Value = true;
    }
    catch (Exception ex)
    {
        Dts.TaskResult = (int)ScriptResults.Failure;
        throw (ex);
    }

    // "Dts.TaskResult"-ominaisuuteen voidaan asettaa tieto
    // skriptin suorituksen onnistumisesta, joka välittyy SSIS-paketin
    // Control Flow -osaan.
    Dts.TaskResult = (int)ScriptResults.Success;
}
```

KUVA 23. Tiedonsiirtorajapinnan tietolähteenä käytettävän siirtotiedoston esiintyvyyden tarkastavan Script Task -tehtäväkomponentin Main-metodin sisältämä ohjelmakoodi.

5.6.2 Skriptien suorittaminen Script Component -komponentissa

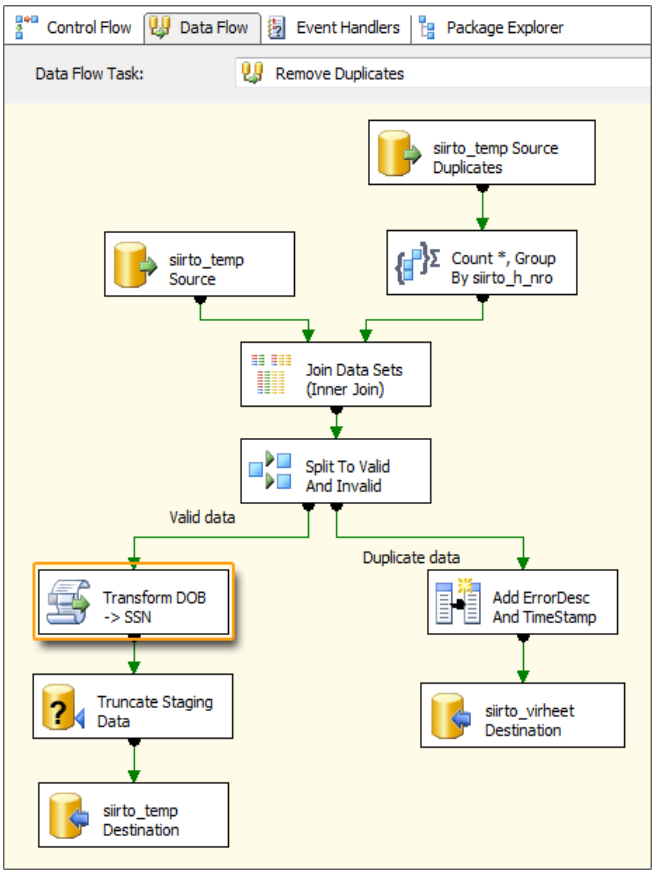
Script Component -komponenttia voidaan hyödyntää SSIS-paketin Data Flow -osassa tiedon poiminta-, muunto- tai lataustoimintojen suorittamiseen .NET-ohjelmakoodin avulla. Script Component -komponentin toimintaroolin jaottelu ilmenee heti, kun komponentti lisätään osaksi SSIS-paketin Data Flow -tehtäväkomponenttia. Ensimmäisenä tulee määritellä siis, että toimiiko komponentti tietolähteenä, muunnoskomponenttina vai tiedonsiirron kohdekomponenttina. Mikäli komponentti toimii tietolähteenä, voidaan siihen luoda useita ulostulopuskureita, joihin tietoa poimitaan komponentin ohjelmakoodin avulla. Kun komponenttia hyödynnetään

muunnosten tekoon, on komponentilla yksi sisääntulopuskuri ja tarvittaessa useampia ulostulopuskureita. Tiedonsiirron kohdekomponenttina käytettynä Script Component -komponenttiin voidaan määritellä yksi sisääntulopuskuri. (Microsoft s.a. ; Knight ym. 2008, 330.)

Mikäli Script Component -komponentin toiminnan luonnetta verrataan Script Task -tehtäväkomponentin toimintaan, on suurin eroavaisuus näiden komponenttien välillä selkeästi se, että Script Component -komponentti käsittelee sen muistipuskurissa olevaa tietoa rivi kerrallaan. Script Task -tehtäväkomponenttiin määritelty ohjelmakoodi suoritetaan siis SSIS-paketin Control Flow -osassa ainoastaan yhden kerran, mikäli komponenttia ei ole sijoitettu Loop Container -säiliön sisään. Kun Script Component -komponentin toimintarooliksi määritellään siis muunnoskomponenttina tai tiedonsiirron kohdekomponenttina toimiminen, suoritetaan komponenttiin määritelty ohjelmakoodi kerran jokaista komponentin muistipuskurissa olevaa riviä kohden. (Knight ym. 2008, 330.) On kuitenkin ymmärrettävä, että Script Component -komponentti käsittelee tietoa riveittäin, joka saattaa hidastaa SSIS-paketin tiedonsiirron suorituskykyä. Script Task -tehtäväkomponenttiin verrattuna tiedon käsitteleminen Script Component -komponentissa on kuitenkin selkeästi tehokkaampaa johtuen ensisijaisesti siitä, että tietoa käsitellään näin muistissa. Tehokkuutta voidaan perustella myös sillä, että Script Component -komponenttiin voidaan määritellä helposti esimerkiksi eri tyyppisiä ulostulopuskureita, joissa tieto siirtyy yhdenaikaisesti uusien muunnoskomponenttien käsiteltäväksi tai erillisiin tiedonsiirron kohteisiin. Jos tietoa käsitellään sen sijaan Script Task -tehtäväkomponentissa, tulee tieto ladata aina ohjelmakoodin suorituksen päätteeksi tietokantaan tai SSIS-paketissa olevaan muuttujaan. On myös huomioitava, että mikäli tietoa ladataan .NET-ohjelmakoodissa suoritettavien SQL-kyselyiden avulla suoraan tietokantaan, muodostaa itse ohjelmakoodi yhden suorituskykyresursseja kuluttavan kerroksen SSIS-paketin ja tietokantamoottorin lisäksi. Tämä hidastaa luonnollisesti tiedonsiirron suorituskykyä.

ETL-tiedonsiirrossa suoritettavia toimenpiteitä, joihin Script Component -komponenttia voidaan tehokkaasti hyödyntää, ovat esimerkiksi XML-muodossa olevan tiedon tai tiedostojen käsittely, monimutkaisten tiedon kelvollisuustarkastusten tai muunnosten suorittaminen ja tiedon poimiminen tai lataaminen web service -rajapinnan kautta (Knight ym. 2008, 330). Kuvassa kaksikymmentäneljä havainnollistetaan SSIS-paketin Data Flow -tehtäväkomponentissa olevaa Script Component -komponenttia, jonka avulla suoritetaan Data Flow:n muistissa olevan syntymäaika-tiedon muuntaminen suomalaisen sosiaaliturvatunnuksen alkuosan

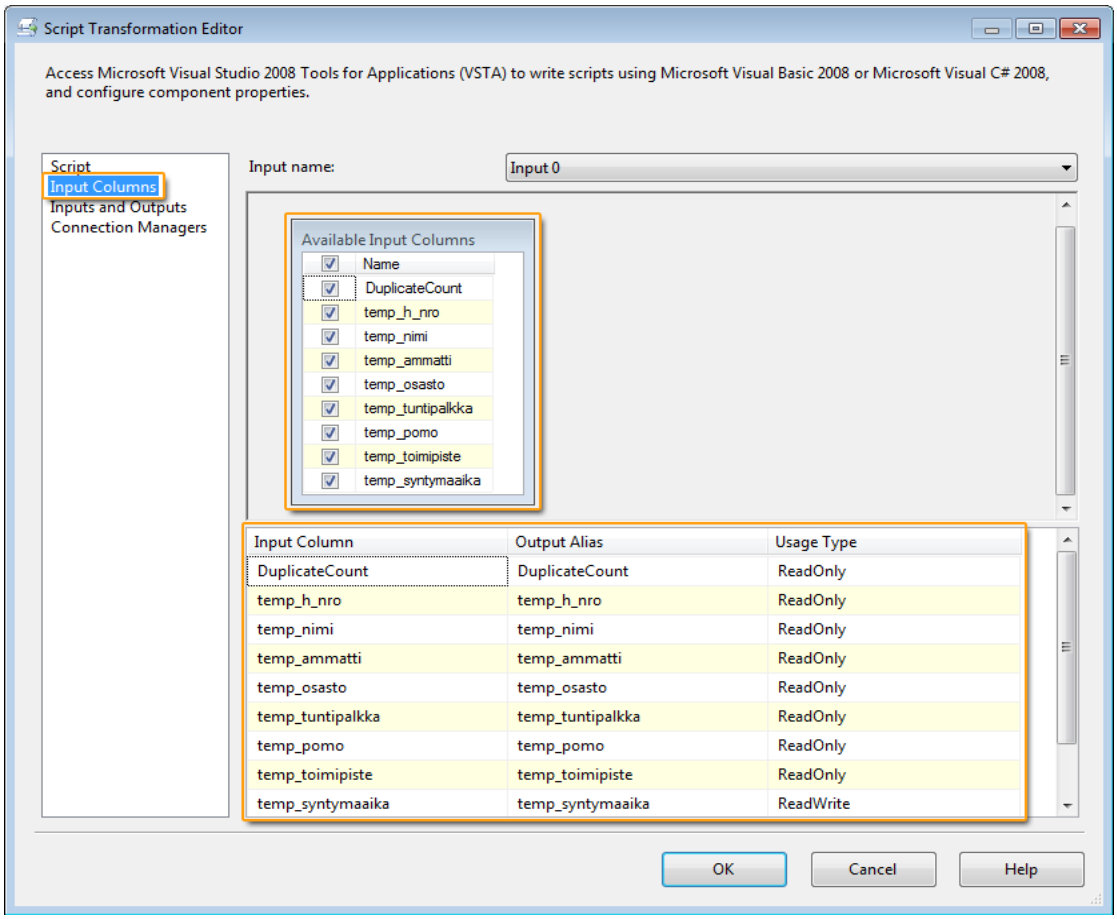
edellyttämään muotoon. Esimerkin Script Component -komponentilla on yksi sisääntulo- ja ulostulopuskuri ja ne ovat synkronisia keskenään. Näin siis ne sarakkeet, jotka siirtyvät kuvan kaksikymmentäneljä Script Component -komponentille käsiteltäväksi sisääntulopuskurin kautta, siirtyvät myös komponentin suorituksen jälkeen ulostulopuskurin kautta kohti tiedonsiirron kohdekomponenttia *siirto_temp Destination*.



KUVA 24. SSIS-paketin Data Flow -osaan lisätty Script Component -komponentti.

Kuvassa kaksikymmentäviisi havainnollistetaan Script Component -komponentin asetusten määritysikkunan *Input Columns* -välilehteä, jonka kautta määritetään muun muassa, että mitkä Data Flow:n muistissa olevat sarakkeet on tarkoitus sisällyttää komponentin muistipuskuriin. Tämän välilehden kautta voidaan määrittää lisäksi, miten sarakkeissa olevia arvoja on tarkoitus käsitellä komponentin ohjelmakoodissa. Mikäli komponentin asetusten määritysikkunan *Input Columns* -välilehdellä olevaan *Usage Type* -sarakkeeseen valitaan siis arvo *ReadOnly*, on rivillä olevan sarakkeen arvo ainoastaan luettavissa komponentin ohjelmakoodissa. Jos tähän sarakkeeseen valitaan sen sijaan arvo ReadWrite, voidaan rivillä olevan sarakkeen arvoa lisäksi muuttaa komponentin ohjelmakoodissa. Kuvasta kaksikymmentäviisi voidaan huomata, että sarakkeelle *temp_syntymaaika* on asetettu Usage Type -arvoksi ReadWrite, koska tässä sarakkeessa oleva syntymäaika-tieto on tarkoitus muuntaa

uuteen esitysmuotoon. Script Component -komponentin muistipuskuriin valittaville sarakkeille voidaan antaa myös oma alias-nimi, jonka kautta sarakkeessa olevaan tietoon voidaan viitata komponentin ohjelmakoodissa. Alias-nimi määrittää myös nimen, joka edustaa saraketta Data Flow:n muistissa Script Component -komponentin suorituksen jälkeen. Alias-nimi voidaan määrittää Script Component -komponentin muistipuskurissa olevalle sarakkeelle komponentin asetusten määritysikkunan Input Columns -välilehdellä olevaan *Output Alias* -sarakkeeseen.

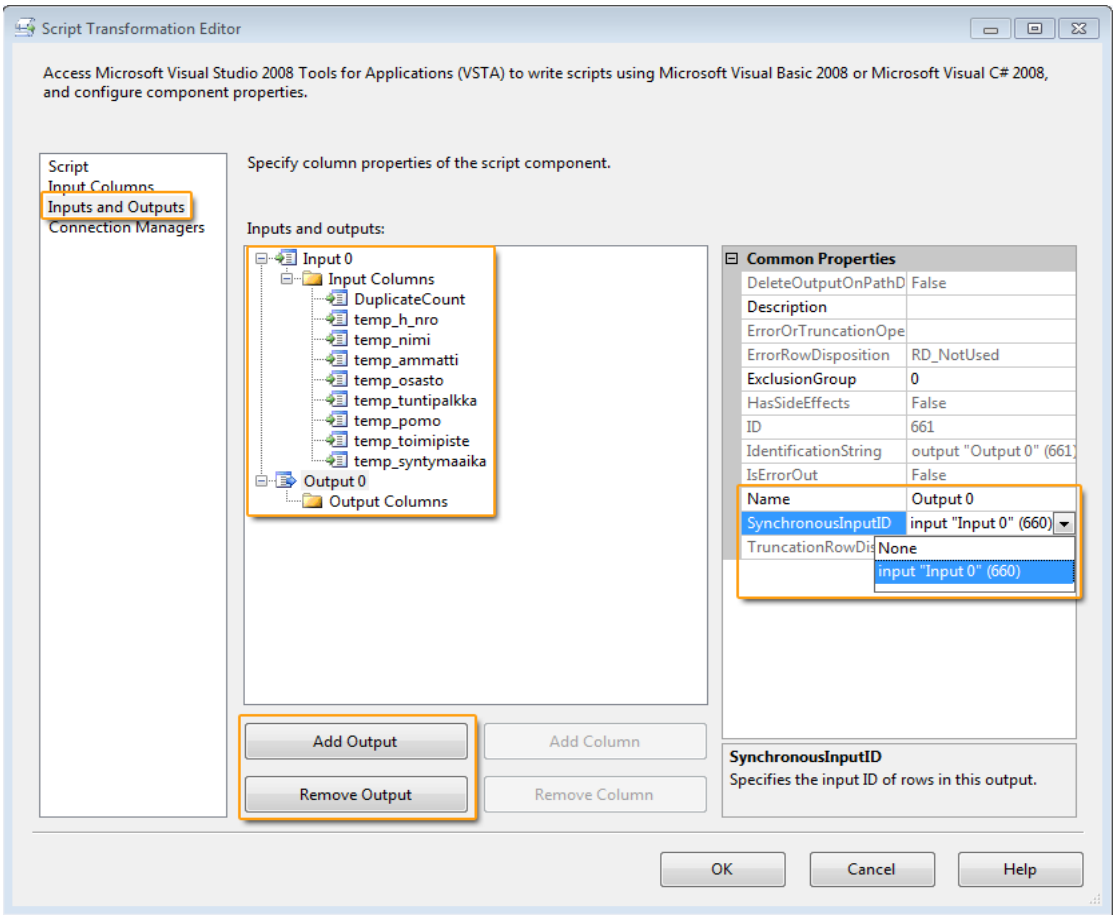


KUVA 25. Script Component -komponentin asetusten määritysikkunan Input Columns -välilehti.

Script Component -komponentin asetusten määritysikkunan välilehdellä *Inputs and Outputs* voidaan määritellä komponentin sisään- ja ulostulopuskureita koskevat tarkemmat asetukset. Kuvassa kaksikymmentäkuusi on esitetty Script Component -komponentin asetusten määritysikkunan Inputs and Outputs -välilehti. Script Component -komponentin, jonka asetuksia kuvassa kaksikymmentäkuusi on havainnollistettu, toimintarooliksi on valittu muunnoskomponenttina toimiminen, joten komponentilla on käytettävissä yksi sisääntulopuskuri ja tarvittaessa useampia ulostulopuskureita. Tämä voidaan huomata kuvan kaksikymmentäkuusi alaosasta, jossa sijaitsee painike *Add Output*. Tämän painikkeen kautta komponenttiin voidaan

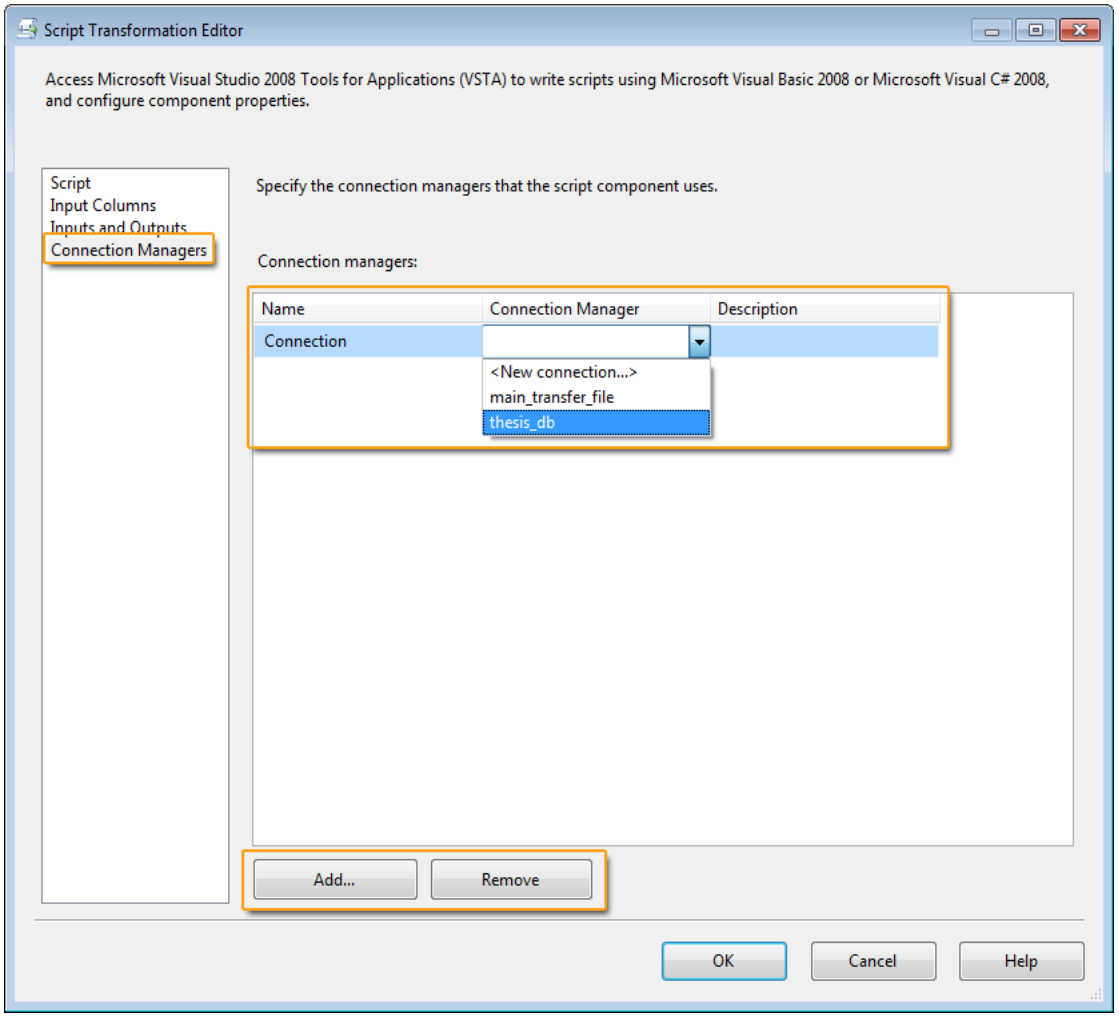
siis lisätä uusi ulostulopuskuri, johon voidaan määritellä uusia sarakkeita. Kuvasta kaksikymmentäkuusi voidaan huomata, että sisääntulopuskurin lisäämiselle ei ole saatavilla painiketta.

Kuvaa kaksikymmentäkuusi tarkastelemalla voi huomata, että Script Component -komponentin ainoan ulostulopuskurin *Ouput 0* alla ei esiinny lainkaan sarakkeita. Tämä johtuu siitä, että komponentin sisään- ja ulostulopuskuri ovat synkronisia eli rakenteeltaan samanlaisia keskenään. Script Component -komponentin sisään- ja ulostulopuskuri voidaan luoda synkroniseksi määrittämällä komponentin asetusten määritysikkunan Inputs and Outputs -välilehdellä ulostulopuskurin *SynchronousInputID*-ominaisuuteen se sisääntulospuskuri, jonka kanssa ulostulopuskurin halutaan olevan synkroninen. Esimerkki SynchronousInputID-ominaisuuden asettamisesta ulostulopuskurille voidaan huomata kuvan kaksikymmentäkuusi oikeasta laidasta. On vielä syytä huomioida, että vaikka Script Component -komponentin sisään- ja ulostulopuskuri ovat synkronisia niin ulostulopuskuriin voidaan silti lisätä uusia sarakkeita, joita ulostulopuskurin vastinparina olevassa sisääntulopuskurissa ei ole.



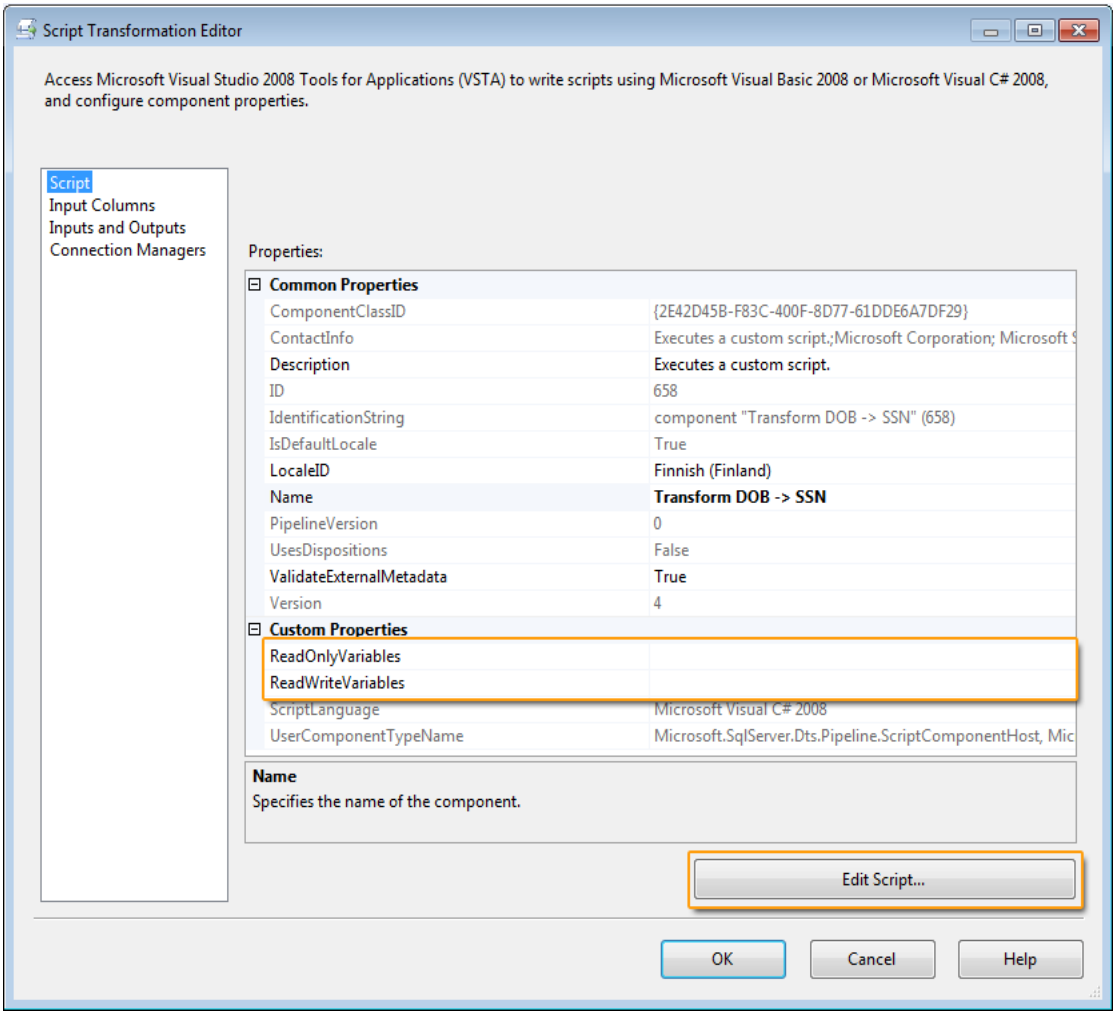
KUVA 26. Script Component -komponentin asetusten määritysikkunan Inputs and Outputs -välilehti.

Mikäli Script Component -komponentissa on tarve käsitellä SSIS-pakettiin luotuja tietoyhteyksiä, voidaan tietoyhteys lisätä komponenttiin sen asetusten määritysikkunan *Connection Managers* -välilehdellä. Kun tietoyhteys on asetettu Script Component -komponenttiin, voidaan lisättyyn tietoyhteyteen viitata komponentin ohjelmakoodissa nimellä, jolla tietoyhteys esiintyy komponentissa. Kuvassa kaksikymmentäseitsemän on havainnollistettu Script Component -komponentin asetusten määritysikkunan Connection Managers -välilehteä. Kuvasta voidaan huomata painikkeet *"Add.."* ja *"Remove"*, joiden kautta uusi tietoyhteys voidaan määritellä tai poistaa. Lisätyt tietoyhteydet on listattu *Connection Managers* -listaan, jonka *Name*-sarakkeeseen määritellään nimi, jolla tietoyhteys esiintyy komponentissa. *Connection Manager* -pudotuslistaan määritellään sen sijaan SSIS-paketissa oleva tietoyhteys, joka on tarkoitus lisätä Script Component -komponenttiin.



KUVA 27. Script Component -komponentin asetusten määritysikkunan Connection Managers -välilehti.

Script Component -komponentin asetusten määritysikkunan *Script*-välilehdeltä löytyvät komponenttia koskevat perusasetukset. Tämän välilehden kautta komponenttiin voidaan määrittää esimerkiksi SSIS-pakettiin luodut muuttujat, joissa olevia arvoja Script Component -komponentin ohjelmakoodissa halutaan käsitellä. Tälle välilehdelle on sijoitettu myös *Edit Script* -painike, jonka kautta voidaan siirtyä VSTA-kehitysympäristöön kirjoitamaan komponenttiin lisättävää ohjelmakoodia. Kuvasta kaksikymmentäkahdeksan voidaan tarkastella Script Component -komponentin asetusten määritysikkunan Script-välilehden käyttöliittymää. Kuvasta voidaan huomata Edit Script -painike sekä *ReadOnlyVariables* ja *ReadWriteVariables* kentät, joihin komponentissa käsiteltävät muuttujat voidaan asettaa siten, että muuttujassa oleva arvo on ainoastaan luettavissa tai myös muutettavissa komponentin ohjelmakoodissa.



KUVA 28. Script Component -komponentin asetusten määritysikkunan Script-välilehti.

Ohjelmakoodin kirjoittamisen näkökulmasta katsottuna Script Component -komponentti eroaa Script Task -tehtäväkomponentista muun muassa siinä, että

Script Component -komponentin ohjelmakoodissa ei ole käytettävissä Dts-objektia, jota Script Task -tehtäväkomponentin ohjelmakoodissa voidaan hyödyntää. Tämän johdosta esimerkiksi tapahtumien laukaisu suoritetaan komponentin ohjelmakoodissa *ComponentMetaData*-nimisen objektin käsittämien metodien kautta. Script Component -komponentin ohjelmakoodissa ComponentMetaData-objektiin voidaan viitata C#-ohjelmointikielen *this*-avainsanan kautta. this-avainsanan välityksellä voidaan viitata myös objekteihin *Variables* ja *Connections*, joiden kautta komponenttiin asetettuja muuttujia ja tietoyhteyksiä voidaan käsitellä. Yksittäiseen muuttujaan tai tietoyhteyteen viitataan niiden nimillä esimerkiksi syntaksilla *this.Variables.MuuttujanNimi* tai *this.Connections.TietoyhteydenNimi*. (Knight ym. 2008, 334-335.)

Toinen huomattava ero, joka ohjelmakoodin kirjoittamisen näkökulmasta katsottuna Script Component -komponentin ja Script Task -tehtäväkomponentin väliltä voidaan löytää, on, että Script Component -komponentti käsittää valmiita metodeja, jotka suoritetaan automaattisesti tietyissä tilanteissa komponentin suorituksen edetessä. Näitä metodeja ovat siis *PreExecute*, *PostExecute*, *CreateNewOuputRows* ja *ProcessInputRow*.

Script Component -komponentin tapahtumametodeista PreExecute suoritetaan siten, että metodi käynnistyy yhden kerran komponentin ohjelmakoodin suorittamisen alkaessa. PostExecute-metodi käynnistyy sen sijaan aina, kun yksittäinen rivi, joka on Script Component -komponentin sisääntulopuskurissa, on käsitelty ProcessInputRow-metodissa. (Knight ym. 2008, 333.)

Script Component -komponentin valmiiden tapahtumametodien CreateNewOutputRows-metodia on erityisen tehokasta hyödyntää, mikäli komponentin toimintarooliksi on valittu lähdekomponenttina toimiminen ja komponentin sisääntulopuskurissa ei ole tietoa. CreateNewOutputRows-metodi suoritetaan PreExecute-metodin jälkeen yhden kerran, jolloin siis metodiin kirjoitetun ohjelmakoodin avulla on tehokasta poimia tietoa komponentin eri ulostulopuskureihin. (Microsoft s.a. .)

ProcessInputRow-metodia voidaan pitää Script Component -komponentin keskeisimpänä metodina. Tämä metodi saa parametrina itselleen rivin, joka kuuluu komponentin sisääntulopuskuriin ja on käsittelyvuorossa tässä muistipuskurissa olevista riveistä. ProcessInputRow-metodi suoritetaan siis kerran jokaista Script Component -komponentin sisääntulopuskurissa olevaa riviä kohden. (Microsoft s.a. .)

Kuvassa kaksikymmentäyhdeksän on esitetty Script Component -komponenttiin, jota havainnollistetaan kuvassa kaksikymmentäneljä, lisätty ohjelmakoodi. Ohjelmakoodin avulla muunnetaan komponentin sisääntulopuskurissa oleva syntymäaika-tieto suomalaisen sosiaaliturvatunnuksen alkuosan edellyttämään muotoon. Kuvasta kaksikymmentäyhdeksän tulee erityisesti huomata, miten muistipuskurissa oleviin rivin sarakkeisiin viitataan ja, miten rivin sarakkeeseen voidaan kohdistaa NULL-tutkimus. Muistipuskurissa olevien rivien sarakkeisiin voidaan siis viitata yksinkertaisesti sarakkeen nimellä. Jokaiselta muistipuskurissa olevan rivin sarakkeelta löytyy lisäksi *_IsNull*-päätteinen tietotyypiltään Boolean-tyyppinen ominaisuus, joka osoittaa, että onko sarakkeessa oleva arvo NULL. Kuvasta kaksikymmentäyhdeksän voidaan huomata lisäksi tapa, millä muunnettu arvo sijoitetaan rivissä olevaan sarakkeeseen. Kuvan tapauksessa muunnettu arvo sijoitetaan samaan kenttään, missä syntymäaika oli alkuperäisessä muodossaan. Tavasta, jolla muunnetun arvon sijoittaminen tehdään tässä tapauksessa, tulee ymmärtää, että sisääntulopuskurin kautta Script Component -komponentin käsiteltäväksi tuleva tieto siirtyy synkronisesti komponentin ulostulopuskuriin. Tällöin muunnettua arvoa ei tarvitse asettaa erikseen ulostulopuskuriin, eikä ulostulopuskuriin tarvitse luoda erikseen uusia rivejä.

```
public override void Input0_ProcessInputRow(Input0Buffer Row)
{
    if (!Row.tempsyntymaaika_IsNull)
    {
        DateTime dob = new DateTime();

        System.Globalization.CultureInfo culture =
            new System.Globalization.CultureInfo("fi-FI");

        if (DateTime.TryParse(Row.tempsyntymaaika,
            culture, System.Globalization.DateTimeStyles.None, out dob))
        {
            Row.tempsyntymaaika =
                ("0" + dob.Day.ToString()).
                    Substring(("0" + dob.Day.ToString()).Length - 2, 2) +
                ("0" + dob.Month.ToString()).
                    Substring(("0" + dob.Month.ToString()).Length - 2, 2) +
                dob.Year.ToString().Substring(dob.Year.ToString().Length - 2, 2);
        }
    }
}
```

KUVA 29. Syntymäaika-tiedon suomalaisen sosiaaliturvatunnuksen alkuosan edellyttämään muotoon muuntavan Script Component -komponentin ohjelmakoodi. Komponenttiin luotu ohjelmakoodi on sijoitettu komponentin ProcessInputRow-metodiin.

5.7 SSIS-paketin virheiden hallinta

Kappaleessa *5.3.2 Tiedon poiminta siirtotiedostosta* havainnollistettiin esimerkin avulla sitä, miten SSIS-paketin Data Flow:ssa olevassa Flat File Source -lähdekomponentissa voidaan hallita siirrettävässä tiedossa olevia virheitä rivin tarkkuudella. Tiedonsiirtorajapinnassa siirrettävän tiedon virheiden hallinta on mahdollista suorittaa SSIS-paketissa ainoastaan paketin Data Flow -osassa. Suurimmalla osalla Data Flow -osan lähde-, muunnos ja kohdekomponenteista on oma virheellisten rivien ulostulopuskuri, mihin tiedonsiirrossa virheelliseksi havaitut rivit voidaan ohjata. Kaikilla Data Flow -osan komponenteilla virheellisten rivien puskuria ei kuitenkaan ole, sillä tämän tyyppisen puskurin olemassa olo johtuu yleisesti siitä, mitä virheitä yksittäisessä komponentissa on ylipäätään mahdollista tapahtua. (Knight ym. 2008, 648.)

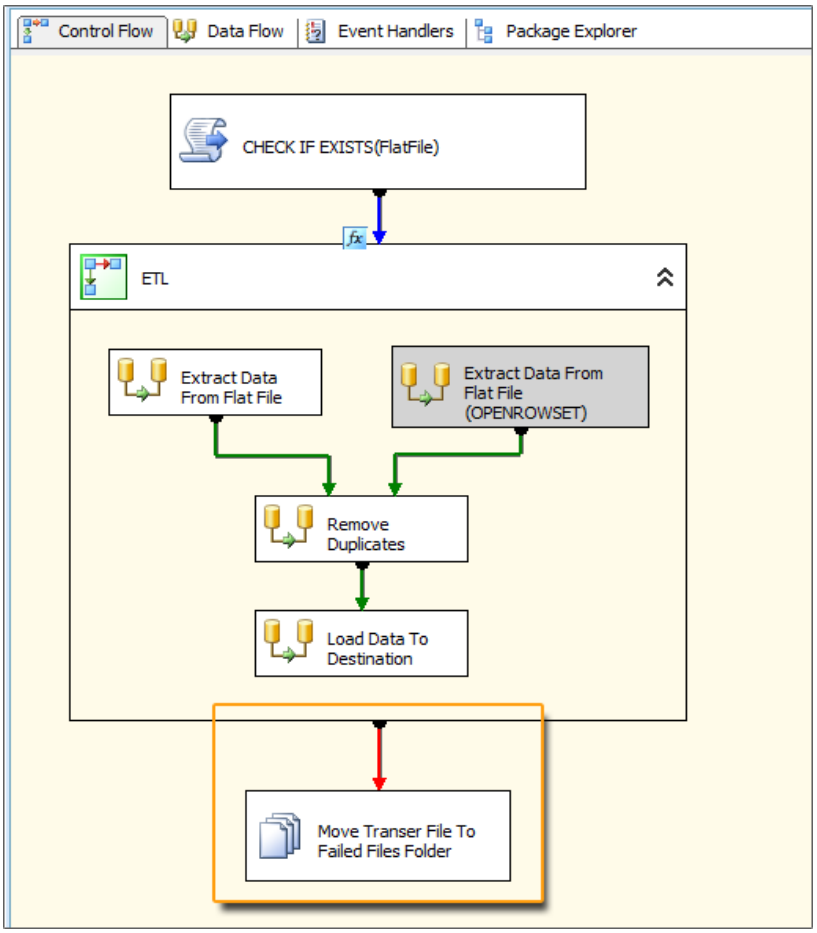
SSIS-paketin Data Flow:ssa suoritettavan virheiden hallinan avulla ei kuitenkaan voida vaikuttaa SSIS-paketin ajonaikaisten virheiden hallintaan. On siis ymmärrettävä, että Data Flow -osassa suoritettava virheiden hallinta kohdistuu ainoastaan tiedonsiirtorajapinnassa siirrettävään tietoon, jolloin SSIS-paketin suorituksen aikana tapahtuvia virheitä tulee hallita erikeinoin. SSIS-paketin suorituksen aikana tapahtuvia virheitä hallitaan ensisijaisesti SSIS-paketin Control Flow -osassa virhe-ehtoliitosten avulla. Control Flow -osassa suoritettavan virheiden hallinnan lisäksi SSIS-paketissa tapahtuvia virheitä voidaan hallita paketin *OnError*- ja *OnTaskFailed*-tapahtumassa. Oikein toteutetun virheiden hallinan avulla voidaan varmistua esimerkiksi siitä, että virhetilanteiden seurauksena järjestelmät, joita tiedonsiirtorajapinnassa käsiteltiin, jäävät eheään tilaan. Virheiden hallinan avulla voidaan varmistaa myös, että mikäli tiedonsiirto päättyy virheeseen, on tiedonsiirtorajapinta valmiina suorittamaan seuraavaksi ajastettua eräajoa normaaliin tapaan.

5.7.1 Virheiden hallinta SSIS-paketin Control Flow -osassa

Ehto-liitosten avulla määritetään SSIS-paketissa suoritettavien tehtäväkomponenttien suoritusjärjestys. Siirtyminen tehtäväkomponentista toiseen SSIS-paketin suorituksen aikana määräytyy sen mukaisesti, että edellyttääkö kahden Control Flow -osassa olevan komponentin välille luotu ehto-liitos edellisen komponentin onnistunutta vai epäonnistunutta suoritusta. Ehto-liitos voidaan määrittää toimivaksi myös siten, että

suoritusvuorossa seuraavaan komponenttiin edetään huolimatta edellisen komponentin suoritustuloksesta. (Knight ym. 2008, 624.)

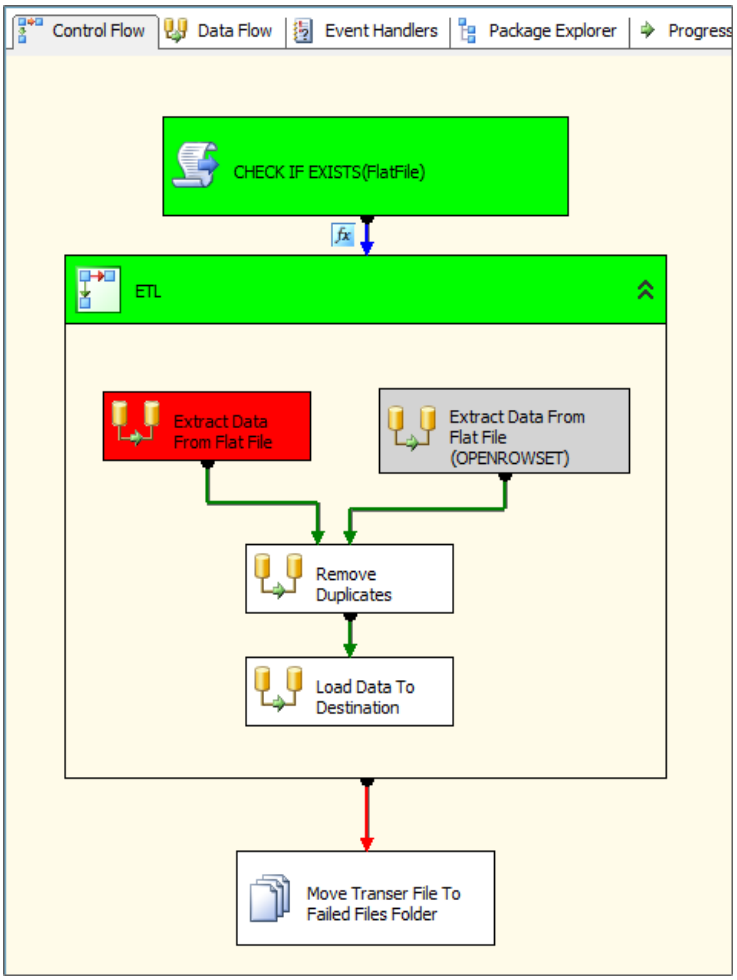
Kuvassa kolmekymmentä on havainnollistettu esimerkkiä siitä, miten SSIS-paketin ajonaikainen suoritus voidaan ohjata erillisiin toimenpiteisiin Control Flow -osassa, mikäli paketin suorituksessa kohdataan virhe siirrettävien tietojen käsittelyvaiheessa. Kuvasta voidaan siis huomata, että mikäli ETL-niminen säiliö tulkitaan SSIS-paketin suorituksen aikana epäonnistuneesti suoritetuksi, edetetään paketin suorituksessa säiliöön liitetyn virhe-ehtoliitoksen mukaisesti tehtäväkomponenttiin *Move Transfer File To Failed Files Folder*. Tässä tehtäväkomponentissa suoritetaan siirtotiedoston, jonka käsittelyssä SSIS-paketin suorituksessa kohdattiin virheitä, arkistointi erilliseen kansioon käyttöjärjestelmän tiedostojärjestelmässä. Kuvassa kolmekymmentä havainnollistetusta esimerkistä tulee ymmärtää erityisesti se, että etenemistä ETL-säiliöstä virhe-ehtoliitoksen mukaisesti suoritusjärjestyksessä seuraavana olevaan tehtäväkomponenttiin edellyttää, että ETL-säiliö tulkitaan SSIS-paketin suorituksen aikana epäonnistuneesti suoritetuksi. Jotta luotua virhe-ehtoliitosta voidaan hyödyntää oikein tulee siis tietää se, että missä tapauksessa SSIS-paketin suorituksen aikana säiliön suoritus tulkitaan epäonnistuneeksi.



KUVA 30. Virhe-ehtoliistoksen hyödyntäminen SSIS-paketin Control Flow -osassa.

SSIS-paketilta ja myös sen Control Flow -osan komponenteilta voidaan löytää ominaisuus *MaximumErrorCount*. Tämä ominaisuus määrittää, että kuinka monta suorituksen aikaista virhettä komponentin tai siihen kuuluvien alikomponenttien suorituksessa on sallittua tapahtua ennen kuin komponentin suoritus tulkitaan epäonnistuneeksi. Oletuksena kaikilla SSIS-paketin Control Flow -osaan lisättävillä komponenteilla tämän ominaisuuden arvo on 1. Tämän johdosta esimerkiksi yksittäisen säiliön suoritus tulkitaan siis epäonnistuneeksi, jos yhdenkin siihen kuuluvan alikomponentin suorituksessa tapahtuu virhe. On huomattava, että mikäli yhden Control Flow -osassa olevan tehtäväkomponentin ajonaikainen suoritus on epäonnistunut, ei tämä tarkoita yksiselitteisesti sitä, että suorituksessa olisi kohdattu ainoastaan yksi virhe. Yhden tehtäväkomponentin suorituksen epäonnistumisen yhteydessä ajonaikaisia virheitä on saattanut tapahtua siis useampia.

Kuvassa kolmekymmentäyksi on havainnollistettu esimerkkiä SSIS-paketin suorituksen etenemisestä BIDS-kehitysympäristössä, kun paketissa olevan säiliön MaximumErrorCount-ominaisuuden arvoksi on asetettu 5 ja yhden säiliöön lukeutuvan alikomponentin suoritus on epäonnistunut. Kuvasta voidaan siis huomata, että vaikka tehtäväkomponentin *Extract Data From Flat File* suorituksen tulos on epäonnistunut, ei ETL-nimisen säiliön suoritus ole silti epäonnistunut. Tämän johdosta suoritus ei siis etene ETL-säilioon liitettyä virhe-ehtoliitosta pitkin Move Transfer File To Failed Files Folder -tehtäväkomponenttiin.



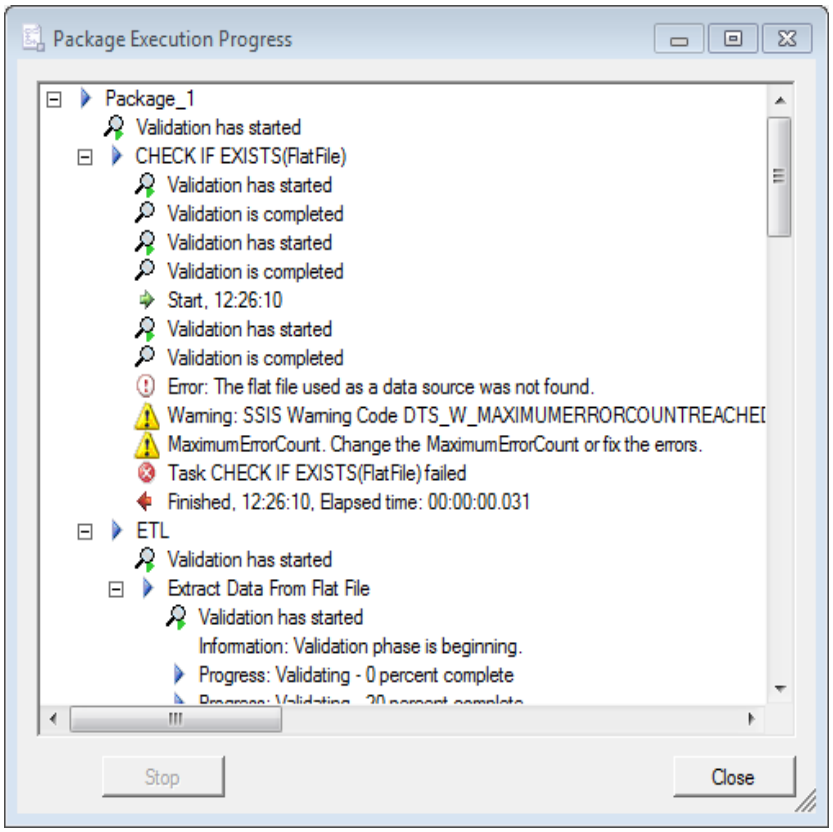
KUVA 31. SSIS-paketin suorituksen eteneminen, kun pakettiin kuuluvan säiliön MaximumErrorCount-ominaisuuden arvoksi on asetettu 5 ja säiliöön lukeutuvan tehtäväkomponentin ajonaikainen suoritus on epäonnistunut.

SSIS-paketin Control Flow -osan komponenttien toimintalogiikkaa virhetilanteissa voidaan muokata myös komponenttien *FailPackageOnFailure* ja *FailParentOnFailure* ominaisuuksien avulla. Oletuksena kaikilla Control Flow -osan komponenteilla näiden ominaisuuksien arvo on *False*. Mikäli SSIS-pakettiin kuuluvalle komponentille asetetaan esimerkiksi FailPackageOnFailure-ominaisuuden arvoksi *True*, aiheuttaa tämä SSIS-paketin suorituksen epäonnistumisen, jos komponentin, jolle asetus määritettiin, suoritus on epäonnistunut. Tässä tapauksessa SSIS-paketin suoritus on siis epäonnistunut huolimatta SSIS-paketin MaximumErrorCount-ominaisuudessa olevasta arvosta. FailParentOnFailure-ominaisuus muuttaa komponentin toimintaa samalla tavalla kuin FailPackageOnFailure, mutta tässä tapauksessa suorituksen hylkäys kohdistuu ainoastaan siihen komponenttiin, missä konfiguroitu komponentti toimii alikomponenttina.

5.7.2 Virheiden hallinta SSIS-paketin tapahtumien avulla

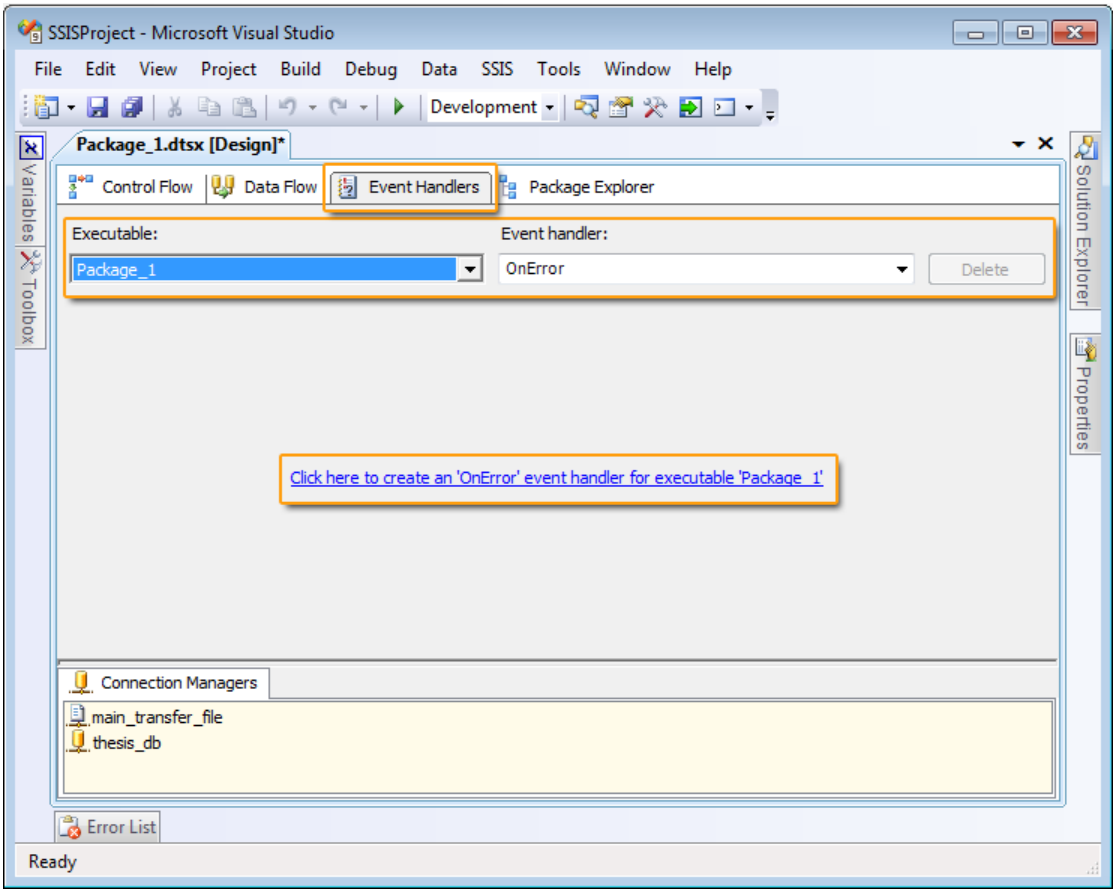
SSIS-ympäristön sisään rakennetut tapahtumankäsittelijät (eng. *Event handler*) muodostavat yhden osan SSIS-ympäristön arkkitehtuurista. Aina, kun SSIS-paketti suoritetaan, aiheuttaa SSIS-ympäristön ajonaikainen moottori useiden erityyppisten tapahtumien laukeamisen (Knight ym. 2008, 634). SSIS-paketin suorituksen aikana laukeavia tapahtumia voidaan seurata esimerkiksi BIDS-kehitysympäristön Progress-välilehdeltä tai SQL Server -ohjelmistoon liittyvän *Execute Package Utility* -sovelluksen (ts. *dtexecui*) kautta.

Kuvassa kolmekymmentäkaksi on esitetty Execute Package Utility -sovelluksen *Package Execution Progress* -ikkuna, joka avautuu sovelluksen käyttöliittymään, kun SSIS-paketti käynnistetään. Kuvasta voidaan huomata esimerkiksi *CHECK IF EXISTS(Flat File)* -tehtäväkomponentissa tapahtunut virhe "*Error: The flat file used as a data source was not found*", joka laukaisee SSIS-paketissa OnError-tapahtuman, ja virheen johdosta aiheutunut tehtäväkomponentin epäonnistunut suoritus "*Task CHECK IF EXISTS(flat file) failed*", joka puolestaan laukaisee SSIS-paketissa OnTaskFailed-tapahtuman.



KUVA 32. SQL Server -ohjelmistoon liittyvän Execute Package Utility -sovelluksen Package Execution Progress -ikkuna, josta voidaan tarkastella SSIS-paketin suorituksen aikana laukeavia tapahtumia.

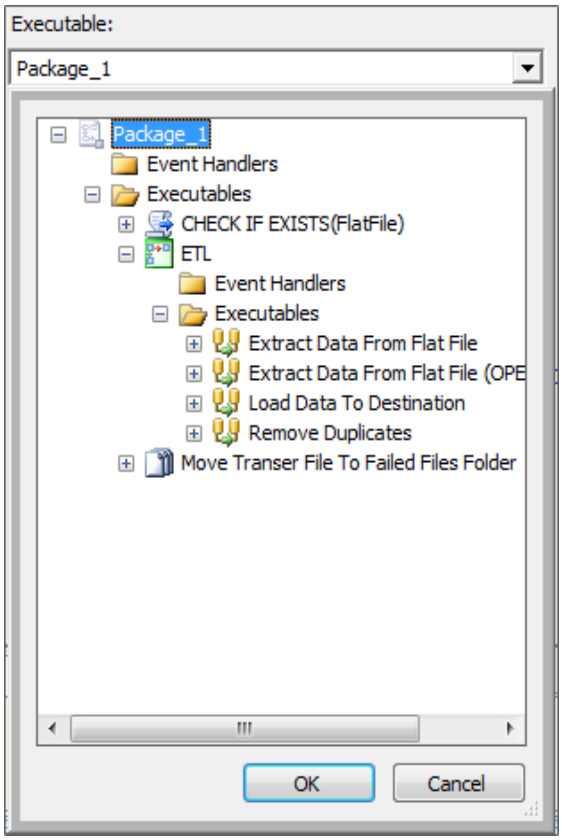
SSIS-paketille ja sen Control Flow -osassa oleville komponenteille voidaan asettaa tapahtumankäsittelijä BIDS-kehitysympäristön *Event Handlers* -osassa. Kuvassa kolmekymmentäkolme on havainnollistettu BIDS-kehitysympäristön käyttöliittymän Event Handlers -osaa ja sitä, miten uusi tapahtumankäsittelijä voidaan luoda SSIS-pakettiin. Kuvasta voidaan huomata muun muassa *Executable*-valintalista, mistä voidaan valita SSIS-paketti tai siihen kuuluva komponentti, jolle tapahtumankäsittelijä halutaan asettaa. Valintalistaan *Event Handler* aktivoidaan sen sijaan sen tapahtuman nimi, jonka mukainen tapahtumankäsittelijä halutaan luoda. Kun halutut arvot on valittu Executable- ja Event Handler -valintalistaan, voidaan tapahtumankäsittelijä aktivoida Event Handlers -osion näkymän keskellä olevasta tekstistä.



KUVA 33. BIDS-kehitysympäristön Event Handlers -osio, jonka kautta SSIS-paketin tapahtumankäsittelijät luodaan.

Yksittäinen tapahtumankäsittelijä voidaan asettaa tietylle SSIS-pakettiin kuuluvalle komponentille tai itse SSIS-paketille. Tapahtumankäsittelijöiden avulla suoritettavan virheiden hallinnan kannalta on tärkeää ymmärtää, että miten SSIS-paketti ja siihen kuuluvat komponentit asemoituvat toisiinsa nähden paketin komponenttihierarkiassa.

SSIS-paketti on siis komponenttihierarkian korkeimmalla tasolla, jolloin siihen kuuluvat muut komponentit ovat alisteisia SSIS-pakettiin nähden. SSIS-paketissa olevan säiliön sisään sijoitetut komponentit ovat sen sijaan alisteisia säiliölle, mihin ne kuuluvat. Tapahtumankäsittelijöiden näkökulmasta tarkasteltuna tämä tarkoittaa sitä, että komponenttihierarkiassa ylemmällä tasolla olevat komponentit reagoivat oletuksena niihin tapahtumiin, joita alisteisissa komponenteissa laukeaa. Mikäli siis tapahtumankäsittelijä OnError asetetaan SSIS-paketille itselleen, laukeaa tämä tapahtumankäsittelijä aina, kun SSIS-paketissa tapahtuu hallitsematon virhe. (Knight ym. 2008, 634, 643.) Kuva kolmekymmentäneljä havainnollistaa SSIS-paketin komponenttihierarkiaa. Kuvassa esiintyvä valintalista avautuu BIDS-kehitysympäristön Event Handlers -osiossa olevasta Executable-valintalistasta, mistä valitaan aina komponentti, jolle tietty tapahtumankäsittelijä halutaan aktivoida.

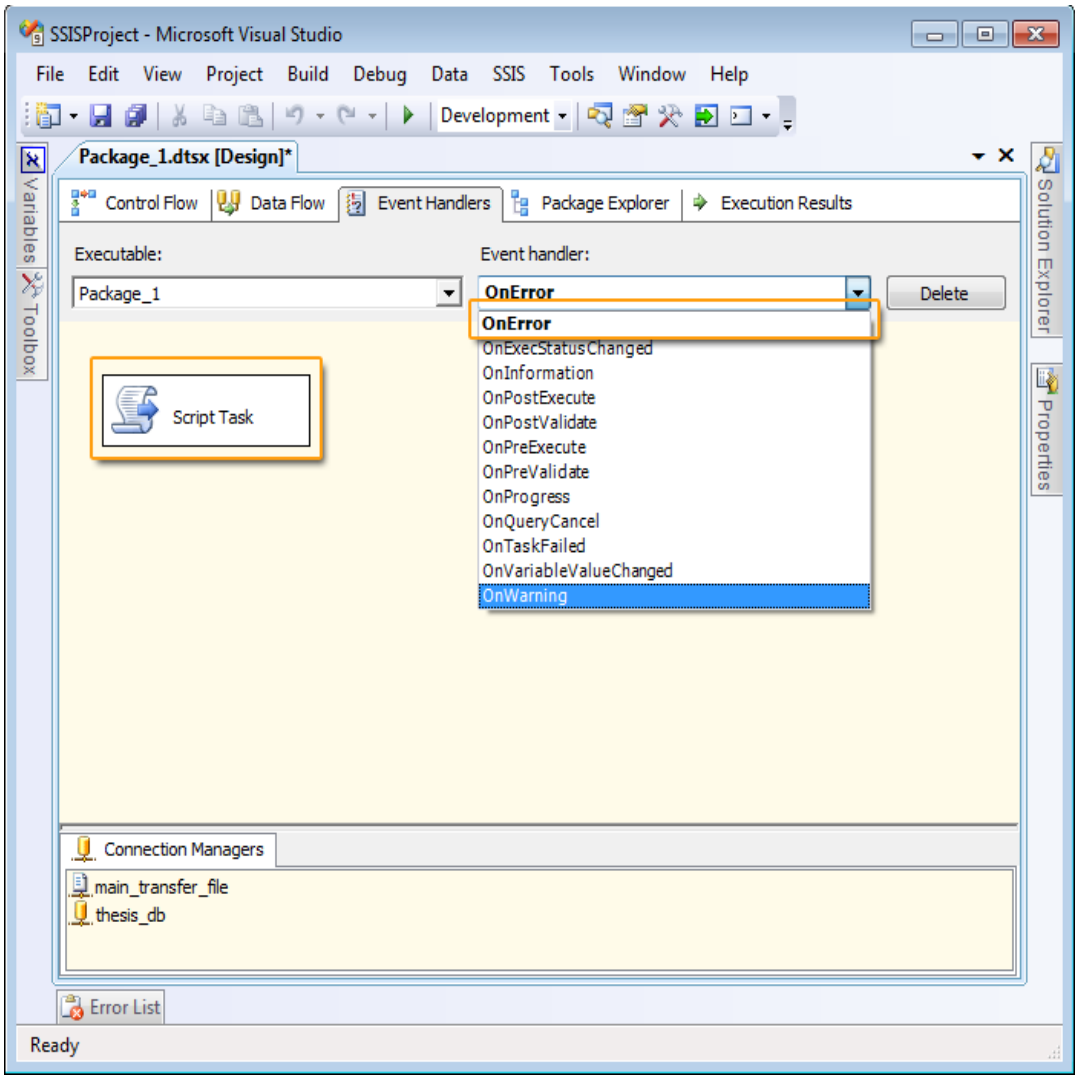


KUVA 34. BIDS-kehitysympäristön Event Handlers -osiossa oleva Executable-valintalista, mistä voidaan tarkastella SSIS-paketin komponenttihierarkiaa.

Mikäli kuvan kolmekymmentäneljä mukaiseen SSIS-pakettiin lisätään OnError-tapahtumankäsittelijä sekä SSIS-paketin tasolle että ETL-nimisen säiliön tasolle saattaa tämä aiheuttaa yllättäviä tilanteita virheiden tapahtuessa paketin suorituksen aikana. Voidaan siis helposti olettaa, että mikäli jokin ETL-säiliön alikomponenteista aiheuttaa virheen, käsitellään virhe ainoastaan ETL-säiliön tapahtumankäsittelijässä.

Näin ei kuitenkaan ole, sillä ylemmän tason komponenttiin asetettu tapahtumankäsittelijä reagoi oletuksena alikomponenttien tapahtumiin. Tämän johdosta, mikäli ETL-säiliössä oleva alikomponentti aiheuttaa SSIS-paketin ajonaikaisen virheen, sekä ETL-säiliön että SSIS-paketin tasolle asetettu OnError-tapahtumankäsittelijä laukeaa. Yksittäiselle tapahtumankäsittelijälle voidaan kuitenkin määrittää asetus, jonka mukaan tapahtumankäsittelijän lauetessa, eivät komponenttihierarkiassa ylemmälle tasolle sijoitettujen komponenttien vastaavat tapahtumankäsittelijät laukea. Tämä asetus on määritetty tapahtumankäsittelijän *Propagate*-järjestelmämuuttujan (eng. *system variable*) Boolean-arvoon. Mikäli tämän muuttujan arvo on siis *True*, aiheuttaa tapahtumankäsittelijä komponenttihierarkiassa ylemmällä tasolla olevien vastaavien tapahtumankäsittelijöiden laukeamisen. Muuttujan arvon ollessa *False* on tapahtumankäsittelijän toiminta päinvastainen. (Knight ym. 2008, 644.)

Kuvassa kolmekymmentäviisi on esitetty BIDS-kehitysympäristön Event Handlers -osion näkymä, kun osiossa on aktivoitu SSIS-paketin tasolle OnError-tapahtumankäsitellijä. Kuvasta voidaan huomata, että tapahtumankäsittelijän, joka Executable-valintalistaan valitulle komponentille on aktivoitu, nimi näkyy lihavoituna Event handler -valintalistassa. Kun yksittäinen tapahtumankäsittelijä on aktivoitu SSIS-paketille tai siihen kuuluvalle komponentille, on tapahtumankäsittelijässä mahdollista suorittaa toimenpiteitä samojen komponenttien ja periaatteiden avulla kuin SSIS-paketin Control Flow -osassa (Knight ym. 2008, 633).



KUVA 35. BIDS-kehitysympäristön Event Handlers -osion näkymä, kun SSIS-pakettiin on aktivoitu OnError-tapahtumankäsittelijä.

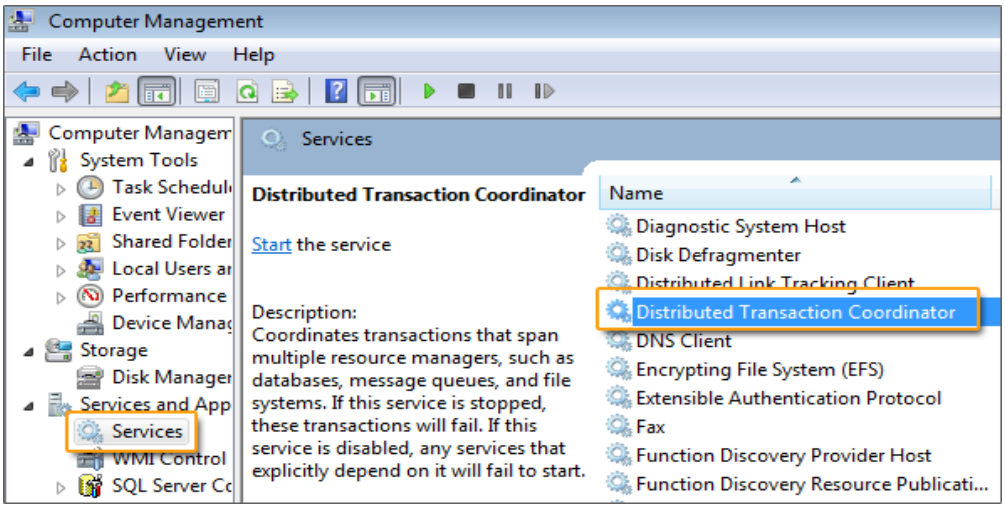
5.8 SSIS-paketin transaction-käsittely

5.8.1 Hajautetun transaction-käsittelyn käyttö

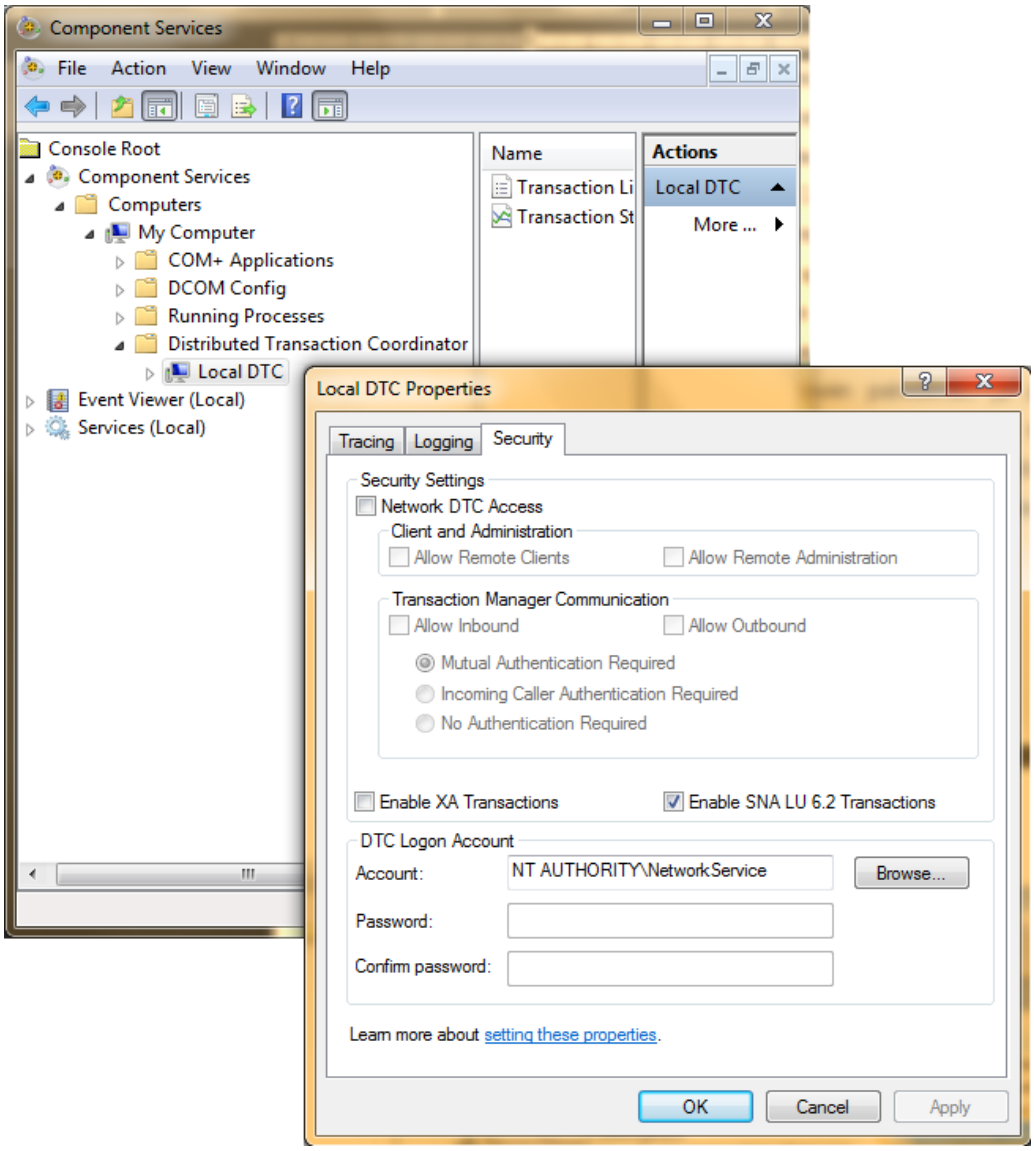
Yksittäisessä SSIS-paketissa suoritettavaa transaction-käsittelyä voidaan kutsua hajautetuksi (eng. *distributed transaction*), kun yhteen SSIS-paketissa vallitsevaan transaction-viitekehykseen lukeutuu useampia tietokantayhteyksiä, jotka liittyvät fyysisesti eri paikoissa oleviin tietokantoihin. Transaction-viitekehyksellä tarkoitetetaan tässä tapauksessa SSIS-paketissa suoritettavan transaction-käsittelyn kehystä, jonka sisässä eri tietokantapalvelimilla sijaitseviin tietokantoihin tehdyt muutokset joko hyväksytään (eng. *commit*) tai perutaan (eng. *rollback*). Hajautettua transaction-käsittelyä hallitsee yleisesti erillinen transaction-manageri (Kuronen 2008, 2). SSIS-paketissa suoritettavaa hajautettua transaction-käsittelyä hallitsee *The*

Microsoft Distributed Transaction Coordinator (ts. *MS DTC*) -niminen transaction-manageri (Knight ym. 2008, 503).

Fyysisesti MS DTC -transaction-manageri esiintyy esimerkiksi *Microsoft Windows Server 2008 R2* -palvelinkäyttöjärjestelmässä yhtenä palveluna (eng. *service*), jonka nimi on *Distributed Transaction Coordinator*. Jotta hajautetun transaction-käsittelyn hyödyntäminen SSIS-pakettiin toteutetussa tiedonsiirrossa on mahdollista, tulee Distributed Transaction Coordinator -palvelu olla käynnissä niillä palvelimilla tai työasemilla, joissa tiedonsiirron suorittava SSIS-paketti ja transaction-käsittelyyn lukeutuvat tietokannat sijaitsevat. Tämän lisäksi eri palvelimilla sijaitsevien transaction-managerien on kyettävä kommunikoimaan keskenään. Tämä tarkoittaa esimerkiksi sitä, että transaction-managerin on sallittua olla yhteydessä fyysisen sijaintinsa ulkopuolella oleviin palvelimiin ja palvelimien tietoliikennettä valvovat palomuurit eivät saa estää transaction-managerien keskinäistä yhteydenpitoa. Kuvassa kolmekymmentäkuusi on havainnollistettu Microsoft Transaction Coordinator -palvelua *Microsoft Windows 7* -käyttöjärjestelmässä. Kuvassa kolmekymmentäseitsemän on havainnollistettu sen sijaan ikkunaa, jonka kautta MS DTC -palvelun toimintaan vaikuttavia asetuksia voidaan hallita.



KUVA 36. Microsoft Windows 7 -käyttöjärjestelmään lukeutuva Distributed Transaction Coordinator -palvelu.

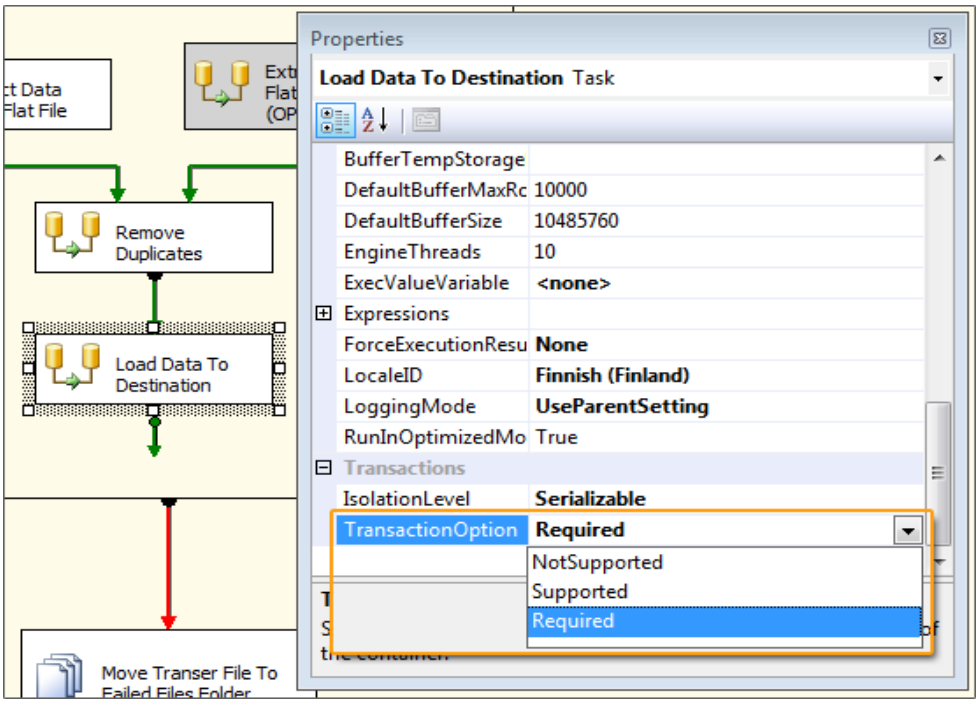


KUVA 37. Microsoft Windows 7 -käyttöjärjestelmän Component Services -osio, jonka kautta käyttöjärjestelmän MS DTC -palveluun liittyviä asetuksia voidaan hallita.

Jokaiselta SSIS-paketin Control Flow -osaan lisätyltä tehtäväkomponentilta tai säiliöltä voidaan löytää *TransactionOption*-ominaisuus. Tämän ominaisuuden avulla voidaan määrittää, että missä vaiheessa hajautettu transaction-käsittely tulisi aloittaa. TransactionOption-ominaisuuden avulla määritetään myös, että mitä SSIS-pakettiin lukeutuvia komponentteja hajautettuun transaction-käsittelyyn halutaan liittää.

SSIS-pakettiin lukeutuvien tehtäväkomponenttien ja säiliöiden TransactionOption-ominaisuuden arvoksi on mahdollista asettaa kolme erityyppistä arvoa, joiden avulla hajautetun transaction-käsittelyn viitekehys määritellään. Näitä arvoja ovat *Supported*, *Not Supported* ja *Required*. Supported-arvo määrittää yksittäiselle komponentille, että mikäli komponentin isäntä kuuluu transaction-viitekehykseen, liittyy komponentti myös tähän samaan viitekehykseen. Not Supported -arvo

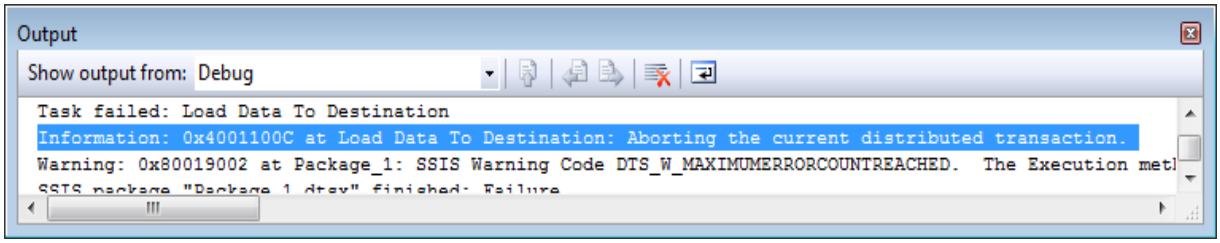
määrittää sen sijaan, että yksittäinen komponentti ei liity transaction-viitekehykseen, vaikka se olisikin olemassa. Required-arvon avulla voidaan määrittää, että mikäli komponentin isäntä ei kuulu transaction-viitekehykseen, aloitetaan transaction-käsittely tästä komponentista, muussa tapauksessa komponentti liittyy isäntänsä transaction-viitekehykseen. (Knight ym. 2008, 503.) Kuvassa kolmekymmentäkahdeksan on havainnollistettu TransactionOption-ominaisuuden asettamista SSIS-paketin Control Flow -osassa olevalle Data Flow -tehtäväkomponentille.



KUVA 38. TransactionOption-asetuksen määrittäminen SSIS-paketin Control Flow -osan Data Flow -tehtäväkomponentille.

Mikäli jonkin SSIS-paketin Control Flow -osaan kuuluvan komponentin, joka lukeutuu hajautetun transaction-käsittelyn viitekehykseen, suoritus on epäonnistunut SSIS-paketin suorituksen aikana, aiheuttaa tämä automaattisesti kaikkien transaction-viitekehyksen sisässä suoritettujen tietokantamuutosten peruuntumisen. Mikäli SSIS-paketin suorituksenaikaisia virheitä ei kuitenkaan tapahdu tiettyyn transaction-viitekehykseen lukeutuvissa komponenteissa, tulevat viitekehyksen sisässä suoritetut tietokantamuutokset automaattisesti hyväksytyksi. MS DTC -palvelun toiminta on siis täysin automaattista, jonka vuoksi sen toiminta on myös suurelta osin täysin näkymätöntä. BIDS-kehitysympäristön Output-ikkunasta voidaan kuitenkin tarkastella MS DTC -palvelun tekemiä toimintoja SSIS-paketin suorituksen aikana. Kuvasta kolmekymmentäyhdeksän voidaan huomata Output-ikkunaan SSIS-paketin

suorituksen aikana tulostunut ilmoitus, kun transaction-viitekehykseen kuuluneen tehtäväkomponentin suorituksessa on kohdattu virhe.



KUVA 39. BIDS-kehitysympäristön Output-ikkunaan tulostunut ilmoitus, kun SSIS-paketin suorituksessa on kohdattu virhe, jonka johdosta vallitsevan transaction-viitekehyksen sisässä suoritetut tietokantamuutokset on peruttu.

5.8.2 Paikallisen transaction-käsittelyn käyttö

SSIS-pakettien transaction-käsittely voidaan suorittaa myös käyttäen apuna SQL Server -tietokantaohjelmiston sisältämiä transaction-ominaisuuksia, jotka eivät hyödynnä MS DTC -palvelua toiminnassaan (Knight ym 2008, 509). SQL Server -ohjelmiston käsittämien ominaisuuksien avulla suoritettavaa transaction-käsittelyä voidaan kutsua paikalliseksi, koska transaction-viitekehykseen on mahdollista sisällyttää tässä tapauksessa ainoastaan yksi tietokantayhteys yhdellä kertaa.

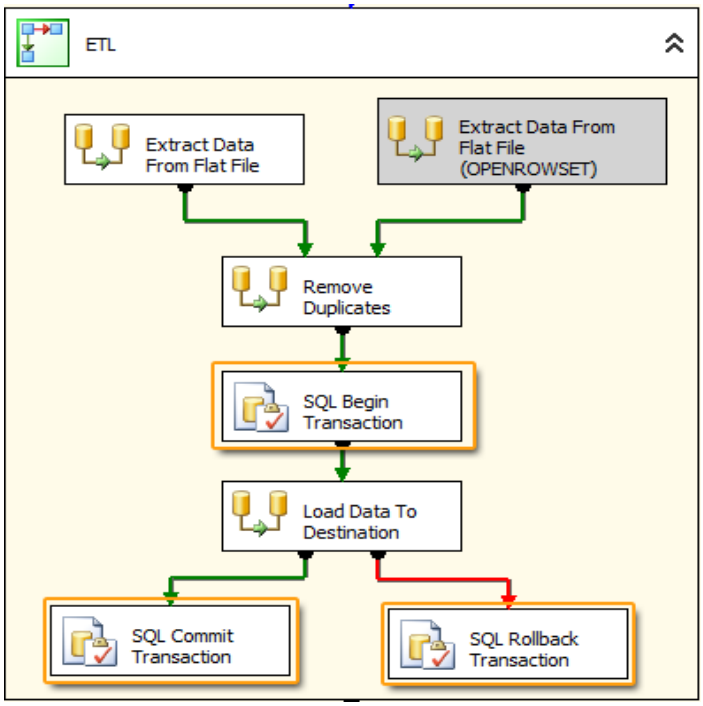
SQL Server -ohjelmiston käsittämän paikallisen transaction-käsittelyn hyödyntämistä SSIS-paketissa edellyttää, että SSIS-paketin Control Flow -pinnalla suoritetaan SQL-komento, joka muodostaa transaction-käsittelyn viitekehyksen eksplisiittisesti tietyn SSIS-pakettiin luodun tietoyhteyden mukaiseen tietokantaan. Tämän lisäksi muodostettu transaction-viitekehys tulee joko hyväksyä tai perua ennen SSIS-paketin suorituksen päättymistä erillisen SQL-komennon avulla. Transaction-käsittely aloitetaan SQL Server 2008 R2 -ohjelmistossa SQL-komennon *BEGIN TRANSACTION* avulla. Muodostettu transaction-käsittely voidaan sen sijaan hyväksyä komennon *COMMIT TRANSACTION* ja perua komennon *ROLLBACK TRANSACTION* avulla. Näiden komentojen käytön lisäksi on huomioitava, että tietoyhteyden, minkä mukaiseen tietokantaan transaction-viitekehys luodaan, ominaisuuden RetainSameConnection arvo tulee olla *True* (katso kappale *5.2 Tietoyhteyksien määrittäminen*).

Kuvassa neljäkymmentä on havainnollistettu esimerkkiä siitä, miten SQL Server -ohjelmiston paikallista transaction-käsittelyä voidaan hyödyntää SSIS-paketissa. Kuvasta voidaan huomata oranssin värisillä kehyksillä vahvistetut Execute SQL Task -tehtäväkomponentit, joiden avulla suoritetaan paikallisen transaction-viitekehyksen

luonti ja vastaavasti sen hyväksyminen tai peruminen perustuen *Load Data To Destination* -tehtäväkomponentin suorituksen lopputulokseen. Mikäli Load Data To Destination -tehtäväkomponentin ajonaikainen suoritus on siis onnistunut, etenee SSIS-paketin suoritus onnistunutta suoritusta havainnollistavan ehtoliitoksen mukaisesti *SQL Commit Transaction* -tehtäväkomponenttiin, missä luotu transaction-viitekehys hyväksytään COMMIT TRANSACTION -SQL-komennon avulla. Mikäli Load Data To Destination -tehtäväkomponentin suoritus on sen sijaan epäonnistunut, edetään SSIS-paketin suorituksessa virhe-ehtoliitoksen mukaisesti tehtäväkomponenttiin *SQL Rollback Transaction*, missä transaction-viitekehys perutaan SQL-komennon ROLLBACK TRANSACTION avulla.

Paikallisen transaction-käsittelyn hyödyntämisen ehdottomana etuna voidaan pitää sitä, että transaction-viitekehyksen hyväksyminen tai peruminen voidaan suorittaa tässä tapauksessa täysin oman logiikan mukaisesti (Knight ym 2008, 509). Tämä tarkoittaa sitä, että transaction-käsittelyn viitekehys voidaan esimerkiksi perua, vaikka mitään ajonaikaista virhettä ei tapahtuisikaan SSIS-paketin suorituksen aikana. Transaction-viitekehyksessä tehtyjen tietokantamuutosten peruminen voidaan suorittaa siis perustuen esimerkiksi SSIS-paketin tietyssä muuttujassa olevaan arvoon.

Kuvan neljäkymmentä mukaisessa esimerkissä transaction-käsittely hyväksytään tai perutaan perustuen yhden Data Flow -tehtäväkomponentin suorituksen lopputulokseen. SSIS-paketissa suoritettava paikallinen Transaction-käsittely voidaan kuitenkin kohdistaa myös esimerkiksi tiettyyn säiliöön, missä suoritetaan useita eri tyyppisiä tehtäväkomponentteja. Tässä tapauksessa transaction-viitekehys hyväksytään siis perustuen säiliön suorituksen lopputulokseen, jolloin transaction-viitekehys voi käsittää useampia erillisiä tietokantaan kohdistuvia toimenpiteitä. On kuitenkin huomioitava, että kun transaction-viitekehys luodaan, kattaa se ainoastaan siihen tietokantaan suoritetut toimenpiteet, johon transaction-viitekehys luotiin BEGIN TRANSACTION -komennolla.



KUVA 40. Esimerkki paikallisen transaction-käsittelyn suorittamisesta SSIS-paketissa.

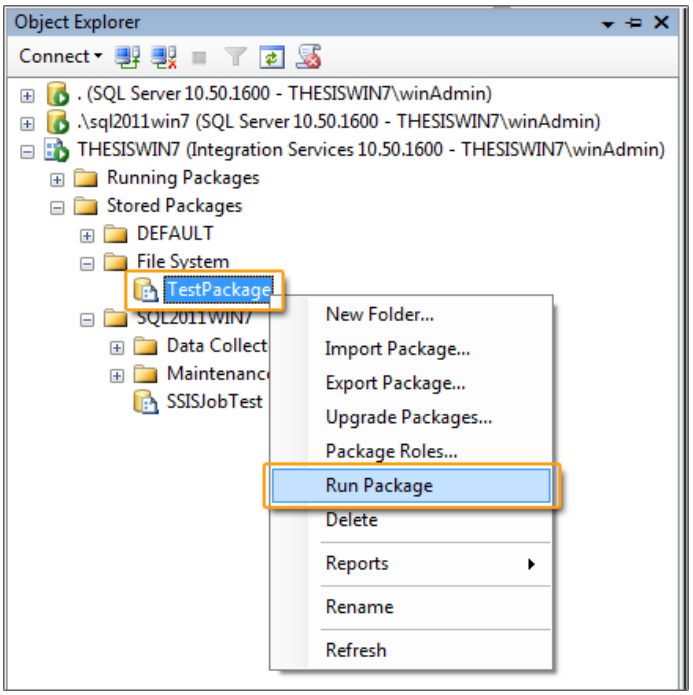
5.9 SSIS-paketin suorittaminen BIDS-kehitysympäristön ulkopuolella

SSIS-ympäristö tarjoaa useita erityyppisiä vaihtoehtoja SSIS-pakettien suorittamiseen BIDS-kehitysympäristön ulkopuolella. Yksinkertaisin tapa SSIS-paketin suorittamiseen on hyödyntää SQL Server -ohjelmistoon liittyvää *DTExecui*-sovellusta tai vaihtoehtoisesti *DTExec*-komentorivisovellusta. Näiden vaihtoehtojen lisäksi SSIS-paketti on mahdollista suorittaa ajastetusti esimerkiksi SQL Server -tietokantamoottoriin liittyvän SQL Server Agent -sovelluksen avulla tai käyttäen apuna Windows käyttöjärjestelmään liittyvää *Windows Task Scheduler* -sovellusta. (Knight ym. 2008, 826, 834, 836.)

5.9.1 SSIS-paketin suorittaminen DTExecui-sovelluksen avulla

DTExecui-sovellus voidaan käynnistää esimerkiksi avaamalla Windows-komentojonoikkuna, kirjoittamalla tähän *dtexecui.exe* ja painamalla näppäimistön *enter*-painiketta. Sovelluksen käynnistystiedosto sijaitsee oletuksena Windows-käyttöjärjestelmän tiedostojärjestelmän kansiossa "*%program files%\Microsoft SQL Server\100\Tools\Binn*". Vaihtoehtoisesti DTExecui-sovellus voidaan käynnistää myös SQL Server Management Studio -sovelluksen kautta, mikäli suoritettava SSIS-paketti on tallennettu SQL Server -tietokantamoottoriin liittyvään *MSDB*-tietokantaan

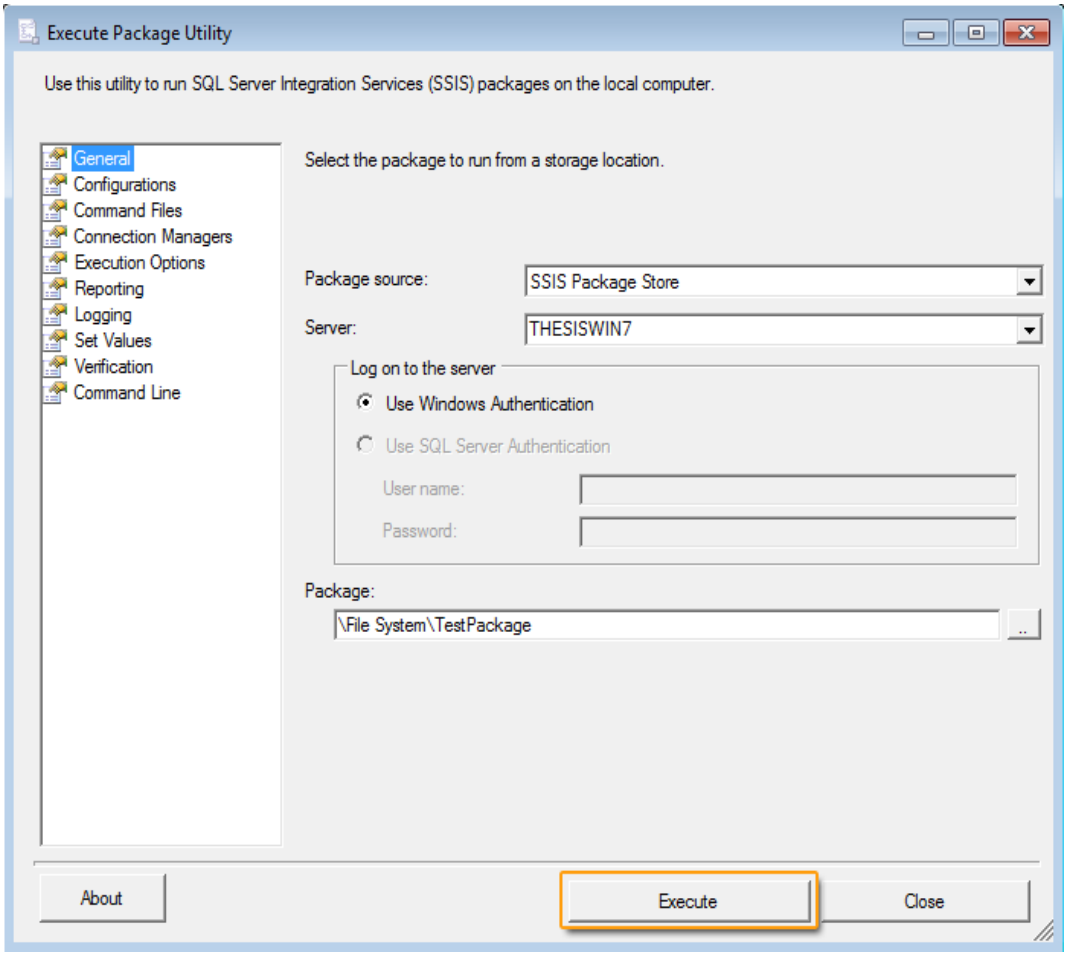
tai Windows-tiedostojärjestelmässä olevaan kansioon, mihin aktiivisena oleva Integration Service -palvelu on määritetty viittaamaan. DTExecui-sovellus käynnistetään tässä tapauksessa yksittäiseen SSIS-paketiin liittyvän kontekstivalikon kautta kohdasta *Run Package* (katso kuva neljäkymmentäyksi).



KUVA 41. DTExecui-sovelluksen käynnistäminen SQL Server Management Studion kautta.

Kuvassa neljäkymmentäkaksi on havainnollistettu DTExecui-sovelluksen käyttöliittymää, kun sovellus on avattu SQL Server Management Studiossa olevan yksittäisen SSIS-paketin kontekstivalikon Run Package -toiminnon kautta. Tässä tapauksessa kaikki tarvittavat asetukset, joiden avulla SSIS-paketti voidaan suorittaa, ovat valmiina DTExecui-sovelluksen käyttöliittymässä. Kuvasta neljäkymmentäkaksi voidaan huomata *Execute*-painike, jonka kautta *Package*-tekstikenttään määritetyn polun osoittama SSIS-paketti voidaan suorittaa. DTExecui-sovelluksen *General*-välilehdellä olevan *Package source* -pudotuslistan arvo määrittää sen sijaan, että sijaitseeko suoritettava SSIS-paketti yksittäiseen Integration Services -palveluun liittyvässä SSIS-paketti-varastossa (*SSIS Package Store*), yksittäisen tietokantamoottori-instanssin *msdb*-tietokannassa (*SQL Server*) vai Windows-käyttöjärjestelmän tiedostojärjestelmässä (*File system*). Mikäli suoritettavan SSIS-paketin sijainniksi on määritelty Integration Services -palveluun liittyvä SSIS-paketti-varasto tai tietokantamoottori-instanssin msdb-tietokanta, tulee DTExecui-sovelluksen General-välilehden *Server*-pudotusvalikkoon määritellä sen tietokoneen

nimi, missä haluttu Integration Services -palvelu sijaitsee, tai sen tietokantamoottori-instanssin nimi, missä oikea msdb-tietokanta sijaitsee.

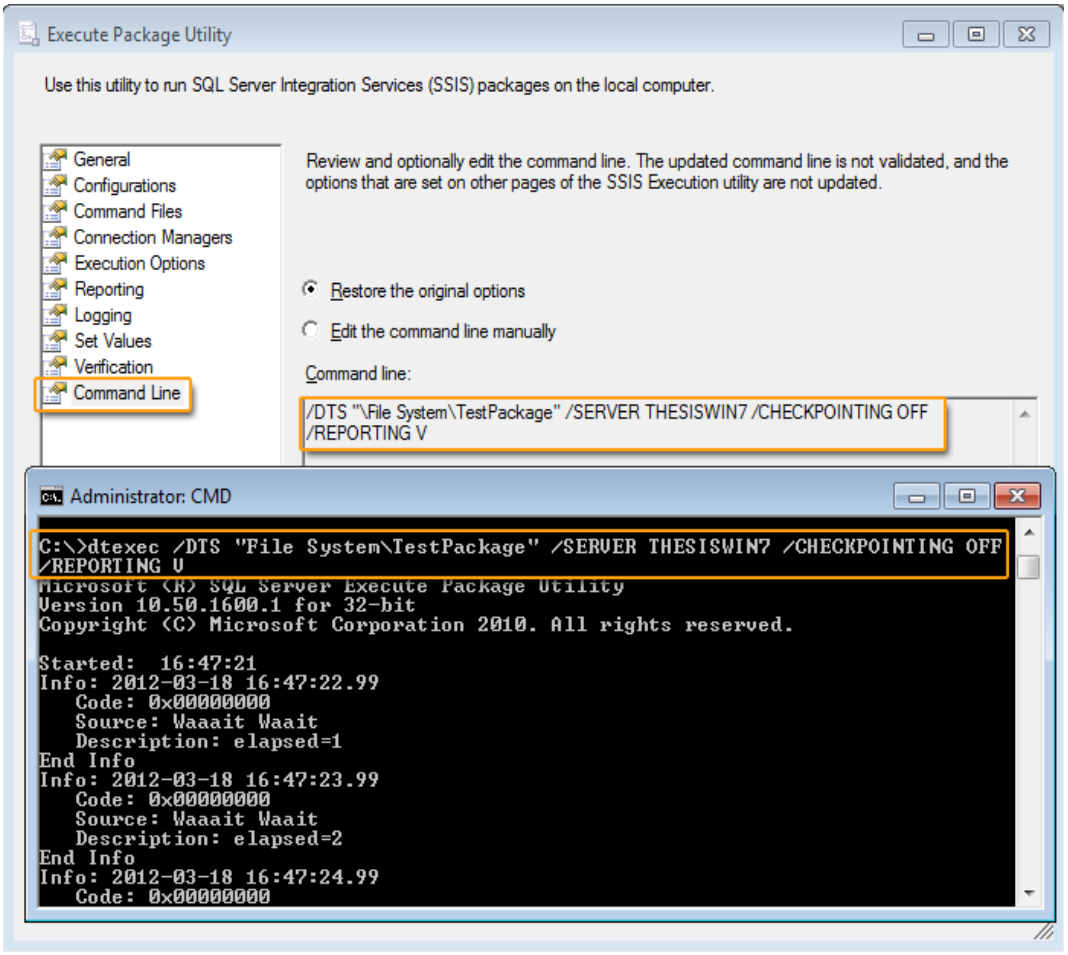


KUVA 42. DTExecui-sovelluksen General-välilehti, missä suoritettava SSIS-paketti määritellään.

5.9.2 SSIS-paketin suorittaminen DTExec-komentorivisovelluksen avulla

DTExec-komentorivisovelluksen perus toimintaperiaate on sama, mitä DTExecui-sovelluksen. DTExec-sovelluksen avulla voidaan siis suorittaa SQL Server tietokantamoottori-instanssin msdb-tietokantaan, Integration Services -palvelun SSIS-paketti-varastoon tai Windows-käyttöjärjestelmän tiedostojärjestelmään talennettuja SSIS-paketteja. Tämän lisäksi DTExec-sovellus mahdollistaa muun muassa suoritettavaan SSIS-pakettiin liittyvien ominaisuuksien, muuttujien, tietoyhteyksien tai logien asetusten muuttamisen tilapäisesti SSIS-paketin suorituksen ajaksi. (Microsoft s.a. .) Tämä tarkoittaa siis esimerkiksi sitä, että SSIS-paketissa oleva tietoyhteys voidaan muuttaa paketin suorituksen ajaksi viittaamaan tilapäisesti eri tietokantaan kuin, mihin se normaalisti viittaisi.

SSIS-paketti voidaan suorittaa DTExec-sovelluksen avulla siirtymällä Windows-käyttöjärjestelmässä olevaan komentoriviin ja kirjoittamalla tähän komennon "*dtexec*" sekä tarvittavat lisäparametrit. Lisäparametrit määrittävät, että mikä SSIS-paketti tulisi suorittaa ja, miten tämä paketti suoritetaan. Lista niistä parametreista, joita DTExec-sovelluksen käynnistykseen voidaan määrittää, saadaan selville suorittamalla komentorivissä komento "*dtexec /?*". Komento, jonka avulla tietty SSIS-paketti voidaan suorittaa, on mahdollista selvittää myös DTExecui-sovelluksen avulla. Mikäli tähän sovellukseen on määritetty asetukset tietyn SSIS-paketin suorittamiseen, voidaan sovelluksen *Command Line* -välilehdeltä löytää komento, minkä avulla määritetty SSIS-paketti voidaan suorittaa DTExec-sovelluksen avulla. (Knight ym. 2008, 834.) Kuvassa neljäkymmentäkolme on havainnollistettu SSIS-paketin suorittamista DTExec-sovelluksen avulla, kun suoritettava SSIS-paketti määritellään DTExecui-sovelluksen kautta noudettujen lisäparametrien avulla.



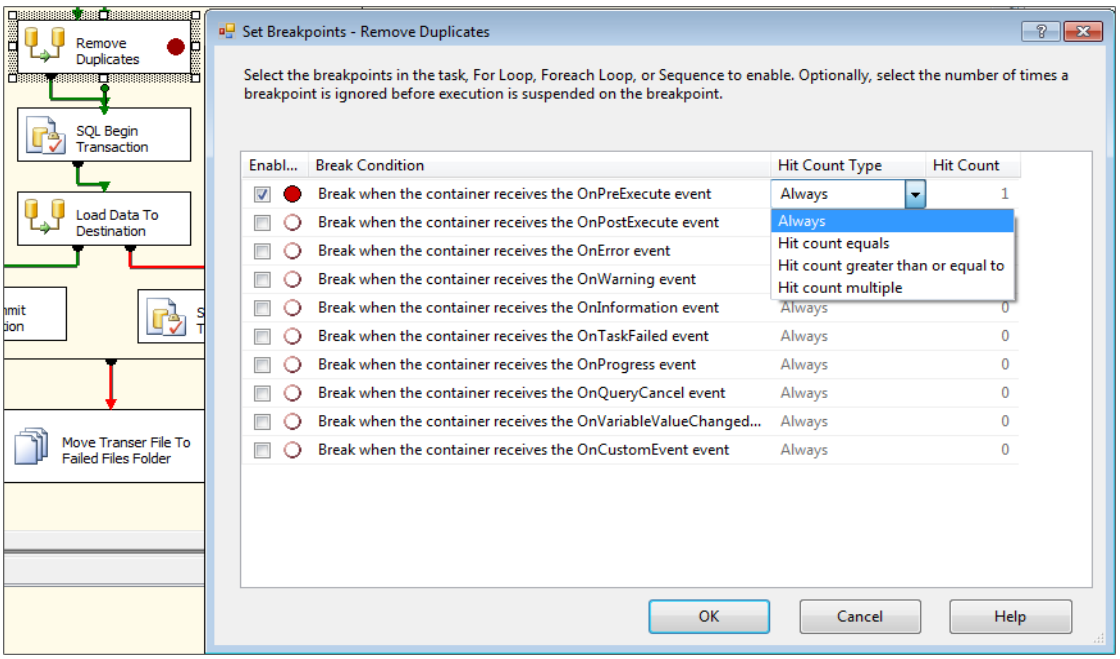
KUVA 43. SSIS-paketin suorittaminen DTExec-sovelluksen avulla, kun sovelluksen lisäparametrit on noudettu DTExecui-sovelluksen kautta.

6 SQL SERVER INTEGRATION SERVICES -RAJAPINNAN TESTAUS BIDS-KEHITYSYMPÄRISTÖSSÄ

BIDS-kehitysympäristö mahdollistaa SSIS-pakettien suorittamisen vaivattomasti SSIS-tiedonsiirtorajapinnan kehitysvaiheessa. SSIS-paketin suorituksen onnistumista on myös helppoa seurata kehitysympäristön graafisesta käyttöliittymästä. SSIS-paketin suorituksen aikana suoritettavat komponentit värjäytyvät siis BIDS-kehitysympäristön käyttöliittymässä joko vihreällä tai punaisella värillä riippuen siitä, että onko tietyn komponentin suorituksen lopputulos onnistunut vai epäonnistunut. BIDS-kehitysympäristö tarjoaa SSIS-paketin suorituksen lopputulosta havainnollistavan graafisen käyttöliittymän lisäksi myös monia muita toimintoja, joiden avulla kehitettävän SSIS-paketin testaus BIDS-kehitysympäristössä voidaan suorittaa.

6.1 Pysähdyspisteiden asettaminen

BIDS-kehitysympäristö mahdollistaa pysähdyspisteiden (eng. *breakpoint*) asettamisen SSIS-paketin Control Flow -osassa oleviin komponentteihin. Mikäli yksittäiseen SSIS-paketin Control Flow -osassa olevaan komponenttiin asetetaan pysähdyspiste, keskeytyy SSIS-paketin ajonaikainen suoritus tähän komponenttiin hetkellisesti kunnes paketin suoritusta halutaan taas jatkaa. Pysähdyspiste voidaan asettaa komponentille siihen kuuluvan kontekstivalikon "*Edit breakpoint...*"-toiminnon kautta tai painamalla näppäimistön painiketta *F9*, kun komponentti on aktivoituna BIDS-kehitysympäristössä. Mikäli pysähdyspiste asetetaan Edit breakpoint -toiminnon kautta, voidaan pysähdypisteeseen määritellä tarkemmin, että missä tilanteissa SSIS-paketin ajonaikainen suoritus pysäytetään pysähdyspisteen kohdalla. Kun pysähdyspiste on asetettu yksittäiseen komponenttiin BIDS-kehitysympäristössä, voidaan sen kohdalla huomata punainen ympyrä. Kuvassa neljäkymmentäneljä on havainnollistettu BIDS-kehitysympäristön *Set Breakpoints* -ikkunaa, minkä kautta komponenttiin asetettavaan pysähdyspisteeseen liittyvät tarkemmat pysähdysehdot voidaan määrittää. Kyseinen ikkuna avautuu siis SSIS-paketin Control Flow -osassa olevan komponentin kontekstivalikon Edit breakpoint -painikkeen kautta.



KUVA 44. BIDS-kehitysympäristön Set Breakpoints -ikkuna, jonka kautta SSIS-paketin Control Flow -osassa olevaan komponenttiin voidaan määritellä pysähdyspiste.

SSIS-paketin Control Flow -osassa olevien komponenttien lisäksi BIDS-kehitysympäristö mahdollistaa pysähdyspisteiden asettamisen .NET-ohjelmakoodiin, joka on kirjoitettu Script Task -tehtäväkomponenttiin. Tässä tapauksessa pysähdyspiste voidaan asettaa, kun Script Task -tehtäväkomponentin ohjelmakoodi on avattu VSTA-kehitysympäristössä muokattavaksi. Kun SSIS-paketin ajonaikainen suoritus pysähtyy Script Task -tehtäväkomponentin ohjelmakoodiin asetetun pysähdyspisteen kohdalle, voidaan ohjelmakoodin suorituksessa edetä Visual Studio 2008 -kehitysympäristöstä tuttujen toimintojen avulla. Ohjelmakoodin suorituksessa voidaan siis edetä esimerkiksi rivi kerrallaan, joka tarjoaa mahdollisuuden esimerkiksi ohjelmakoodissa olevien muuttujien arvojen tarkasteluun suorituksen aikana. Hieman yllättävää on kuitenkin se, että BIDS-kehitysympäristö mahdollistaa ainoastaan yhden pysähdyspisteen asettamisen yhteen Script Task -tehtäväkomponenttiin SSIS-paketin suorituksen ajaksi. (Microsoft s.a. .) Mikäli SSIS-paketin Control Flow -osassa olevan kahden erillisen Script Task -tehtäväkomponentin ohjelmakoodiin asetetaan siis pysähdyspisteet, tulostuu SSIS-paketin suorituksen aikana BIDS-kehitysympäristön Output-ikkunaan varoitusteksti "*Warning: 0x6 at Script Task: Debugging of more than one script task is not currently supported*".

BIDS-ympäristön selkeäksi puutteeksi voidaan laskea se, että se ei mahdollista pysähdyspisteiden asettamista Data Flow -tehtäväkomponentissa suoritettavan Script

Component -komponentin ohjelmakoodiin. Mikäli Script Component -komponentin ohjelmakoodissa käsiteltävää tietoa halutaan kuitenkin tarkastella ohjelmakoodin suorituksen aikana, voidaan tämä toteuttaa käyttäen apuna esimerkiksi .NET-luokkakirjaston *System.Windows.Forms.MessageBox*-luokan *Show*-metodia. Tämän lisäksi Script Component -komponentin ohjelmakoodissa voidaan kutsua esimerkiksi ScriptComponent-luokan *ComponentMetaData*-ominaisuuden *FireProgress*- tai *FireInformation*-metodia, jotka kirjoittavat tietoa SSIS-paketin suorituksen aikana BIDS-kehitysympäristön *Progress*-välilehdelle ja Output-ikkunaan. (Microsoft s.a. .)

6.2 SSIS-paketin suorittaminen osissa

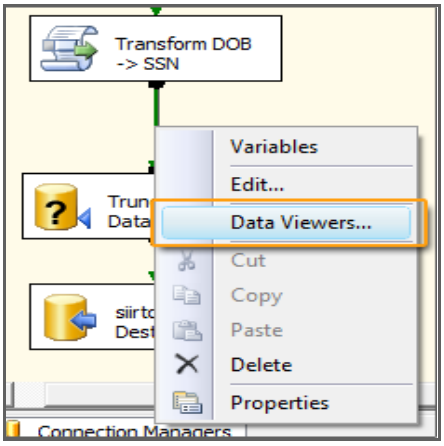
Mikäli kehitettävä SSIS-paketti sisältää runsaasti tiedonsiirtoa ohjaavaa logiikkaa, on tämän logiikan toimivuutta järkevää testata aluksi pienemmissä osissa. BIDS-kehitysympäristö mahdollistaa siis SSIS-paketin Control Flow -osassa olevien komponenttien poistamisen käytöstä tilapäisesti. On kuitenkin huomattava, että mikäli SSIS-pakettiin kehitetty tiedonsiirtologiikka on sijoitettu esimerkiksi yksittäiseen Script Task - tai Execute SQL Task -tehtäväkomponenttiin, on tätä logiikkaa vaikeampaa testata osissa. Tämä johtuu luonnollisesti siitä, että BIDS-kehitysympäristö mahdollistaa Control Flow -osassa olevan toiminnallisuuden poistamisen käytöstä tarkimmillaan komponenttitasolla. Tämän vuoksi kehitettävän SSIS-paketin rakenteessa on tehokasta suosia mallia, missä yksittäisessä tehtäväkomponentissa suoritetaan aina yhden tyyppinen toimenpide. Hyvänä sääntönä voisi pitää sitä, että mikäli tehtäväkomponentin nimessä ei pystytä kuvaamaan selkeästi niitä toimenpiteitä, joita komponentissa suoritetaan, tulisi komponentissa oleva toiminnallisuus mahdollisesti eriyttää erillisiin tehtäväkomponentteihin.

SSIS-paketin Control Flow -osassa oleva komponentti voidaan poistaa käytöstä tilapäisesti komponentin kontekstivalikossa olevan *Disable*-painikkeen kautta, kun SSIS-paketti on avattu muokkaustilaan BIDS-kehitysympäristössä. Kun yksittäinen komponentti poistetaan käytöstä, muuttuu sen väri BIDS-kehitysympäristön käyttöliittymässä harmaaksi. Tämän lisäksi käytöstä poistetun komponentin kontekstivalikkossa olleen Disable-painikkeen tekstiksi vaihtuu teksti *Enable*. Mikäli tätä painiketta painetaan, on komponentti jälleen aktiivisena SSIS-paketissa.

6.3 Data Viewers

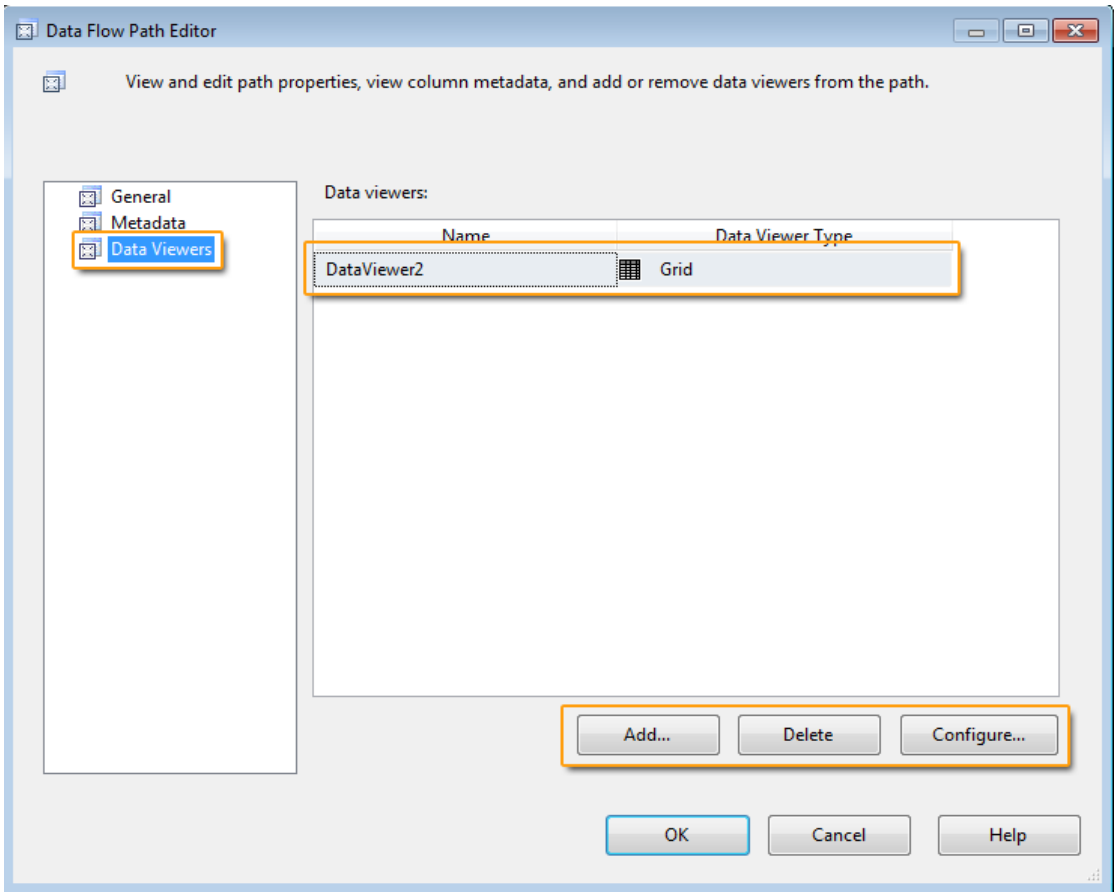
BIDS-kehitysympäristö mahdollistaa *Data Viewer* -pysähdyspisteiden asettamisen SSIS-paketin Data Flow -tehtäväkomponenttiin sijoitettujen komponenttien välille.

Data Viewer -pysähdyspisteet mahdollistavat Data Flow -tehtäväkomponentissa siirtyvän tiedon tarkastelun SSIS-paketin suorituksen aikana, kun tieto on poimittu tietolähteestä Data Flow:n käyttämään muistiin, ennen tai jälkeen yksittäisen komponentin suorittamisen ja ennen kuin tieto ladataan tiedonsiirron kohteeseen (Microsoft s.a.). Yksittäisen Data Viewer -pysähdyspisteen asettamiseen voidaan siirtyä kahden Data Flow:ssa olevan komponentin liitoskohdan kontekstivalikossa olevan ”*Data Viewers...*”-painikkeen kautta (katso kuva neljäkymmentäviisi). Tämän painikkeen kautta BIDS-kehitysympäristön käyttöliittymään avautuu *Data Flow Path Editor* -niminen ikkunan, jonka kautta asetettavan Data Viewer -pysähdyspisteen asetukset määritellään.

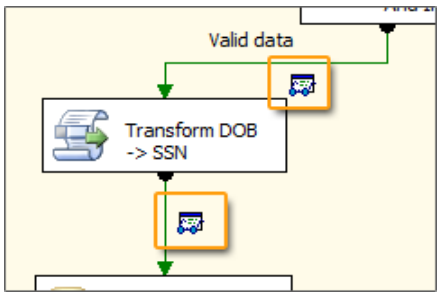


KUVA 45. Siirtyminen Data Viewer -pysähdyspisteen asettamiseen.

Kuvassa neljäkymmentäkuusi on havainnollistettu Data Flow Path Editor -ikkunan käyttöliittymää. Kyseisen ikkunan *Data Viewers* -välilehden kautta voidaan määritellä asetettavaan Data Viewer -pysähdyspisteeseen liittyvät asetukset. Kuvasta neljäkymmentäkuusi voidaan huomata, että liitoskohtaan, minkä asetuksia kuvan mukaisessa ikkunassa hallitaan, on asetettu *DataViewer2*-niminen Data Viewer, jonka tyyppi on *Grid*. Grid-tyyppi tarkoittaa sitä, että kun SSIS-paketin suorituksen aikana edetään Data Viewer -pysähdyspisteen kohdalle, näytetään Data Flow:n muistissa olevat ja Data Viewer:iin valitut tiedot yksinkertaisessa taulukossa sarakkeittain ja riveittäin. Perinteisen taulukkomuodon lisäksi Data Flow:ssa siirtyvää tietoa voidaan havainnollistaa Data Viewer:n avulla myös muissa muodoissa. Tietoa voidaan tarkastella Data Viewer:n avulla siis eri tyyppisten kaavioiden muodossa, kuten esimerkiksi histogrammi-, hajonta- tai pylväskaaviona. (Microsoft s.a. .) Kuvasta neljäkymmentäseitsemän voidaan huomata ikoni, joka ilmestyy SSIS-paketin Data Flow -tehtäväkomponentin muokkauspinnalle BIDS-kehitysympäristössä, kun Data Viewer on lisätty muunnoskomponenttien väliseen liitokseen.



KUVA 46. BIDS-kehitysympäristön Data Flow Path Editor -ikkunan käyttöliittymä.

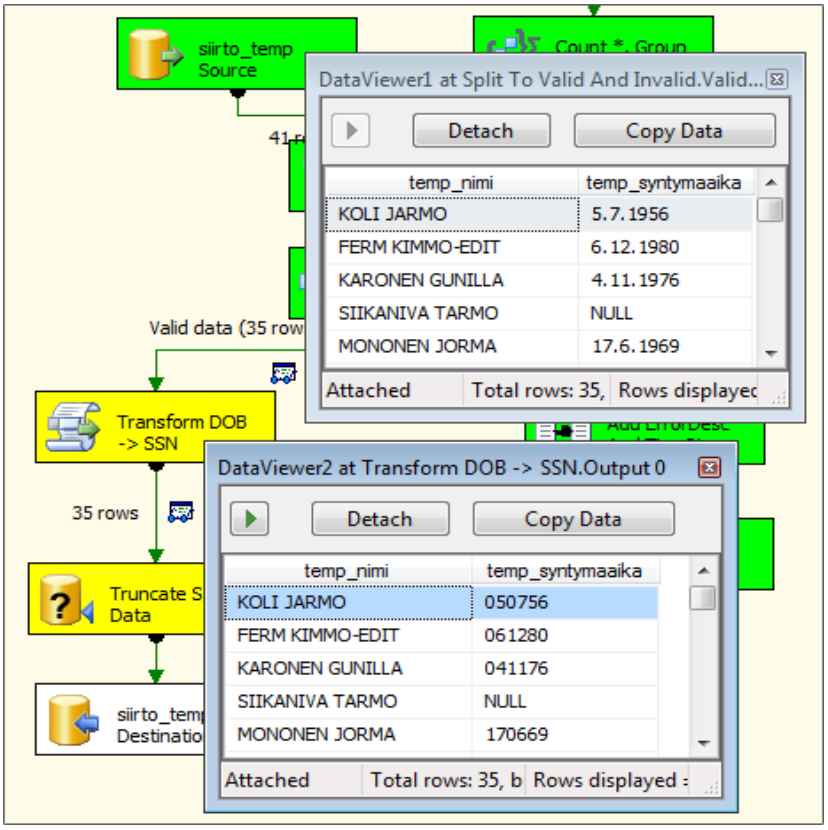


KUVA 47. Ikonit Data Flow -tehtäväkomponentin muokkauspinnalla BIDS-kehitysympäristössä, kun muunnoskomponenttien välisiin liitoksiin on asetettu Data Viewer -pysähdyspisteet.

Kuvassa neljäkymmentäkahdeksan on esitetty tilanne, missä SSIS-paketin Data Flow -tehtäväkomponenttiin on lisätty kaksi Data Viewer -pysähdyspistettä ja paketin suoritus on pysähtynyt jälkimmäisenä suoritettavan Data Viewer:n kohdalle. Kuvasta voidaan huomata kaksi erillistä ikkunaa, jotka ovat avautuneet BIDS-kehitysympäristön käyttöliittymään, kun SSIS-paketin suoritus on edennyt Data Flow -tehtäväkomponenttiin asetetun Data Viewer:n kohdalle. Tieto, joka näytetään Data Viewer -pysähdyspisteistä avautuneissa ikkunoissa, esitetään taulukkomuodossa eli Data Viewer -pysähdyspisteiden tyypiksi on valittu tässä tapauksessa tyyppi Grid.

Mikäli avautuneita ikkunoita tarkastellaan tarkemmin, voidaan ikkunoiden käyttöliittymän vasemmasta yläkulmasta huomata ”nuolipainike”, jota painamalla SSIS-paketin suoritusta voidaan jatkaa sen jälkeen, kun suoritus on pysähtynyt Data Viewer:n kohdalle.

Kuvasta neljäkymmentäkahdeksan voidaan todeta, miten ScriptComponent-muunnoskomponentti ”*Transform DOB -> SSN*” on muuntanut Data Flow:n muistissa olevassa sarakkeessa *temp_syntymaaika* olleen tiedon. Tämän muunnoskomponentin avulla suoritetaan siis Data Flow:ssa siirtyvään henkilötietoon liittyvän syntymäaika-tiedon muuntaminen suomalaisen sosiaaliturvatunnuksen alkuosan edellyttämään muotoon. Tiedon muuntautumisen tarkastelu suoraan BIDS-kehitysympäristössä heti SSIS-paketin suorituksen aikana on tehokas tapa varmistaa, että yksittäinen muunnoskomponentti toimii odotetulla tavalla. Tiedon oikeellinen muuntautuminen on siis huomattavasti helpompaa todeta näin kuin, että se todettaisiin tutkimalla tietoja tiedonsiirron välitaulussa tai kohteessa.



KUVA 48. SSIS-paketin suorituksen aikana Data Viewer -pysähdyspisteistä BIDS-kehitysympäristön käyttöliittymään avautuvat ikkunat.

7 POHDINTA

Mikäli yritysorganisaation järjestelmäympäristössä hyödynnetään Microsoft:n teknologioita, kuten esimerkiksi SQL Server -tietokantaohjelmistoa, on SSIS järkevä valinta ohjelmistoksi, jonka avulla yrityksen tietojärjestelmien sisältämiä tietomassoja hallitaan. Tätä voidaan perustella sillä, että SSIS on saatavilla kokonaisuudessaan suoraan yritysorganisaation käyttöön, mikäli yritys omistaa lisenssin SQL Server -ohjelmiston kokonaisversion käyttöön. SSIS-teknologian käytön voidaan katsoa olevan itsessään siis edullista, koska erillisten ohjelmistolisenssimaksujen suorittaminen ei ole edellytyksenä sen käytölle tuotantoympäristössä.

Mikäli SSIS-teknologiaa hyödynnetään yrityksen järjestelmäympäristössä olevien tietomassojen hallinnassa eri tyyppisten tietolähteiden välillä, on yrityksen järkevää pohtia myös muiden Microsoft:n BI-ohjelmistojen käyttöönottoa tuotantokäyttöön. SSIS-teknologian selkeäksi eduksi voidaan katsoa siis se, että sen käyttö sitoutuu hyvin tiiviisti myös muihin Microsoft:n BI-ohjelmistoihin. Valmiita SSIS-paketteja on siis mahdollista käyttää esimerkiksi SSRS-teknologian avulla luotujen raporttien tietolähteinä, jolloin tiedon raportoinnin edellytyksenä ei ole sen sijainti tietokonassa. Tämän lisäksi SSIS mahdollistaa tiedon käsittelyn esimerkiksi SSAS-teknologian avulla luoduissa kohteissa. SSIS-teknologiaa ei voida siis pitää erillisenä teknologiana, joka soveltuu ainoastaan ETL-rajapintojen kehittämiseen.

SSIS-teknologian vahvuudeksi voidaan katsoa myös se, että se soveltuu hyvin erityyppisten ja erikokoisten ETL-rajapintojen toteuttamiseen. Mikäli SSIS-teknologialla toteutetussa ETL-rajapinnassa käsitellään siis esimerkiksi suuria tietomääriä, voidaan tämän tietomassan hallinnassa hyödyntää SSIS-arkkitehtuuriin lukeutuvan tiedonsiirtomoottorin tapaa käsitellä siirrettävää tietoa palvelimen muistissa. Toisaalta mikäli toteutettava ETL-rajapinta on yksinkertainen ja siinä käsitellään pieniä tietomääriä, voidaan rajapinnan toteutuksessa hyödyntää SQL Server -ohjelmiston käsittämiä ominaisuuksia, kuten tietokantaproseduureja, näkymiä ja SQL-ohjelmointikieltä. SSIS mahdollistaa siis ETL-rajapintojen kehittämisen noudattaen itse suunniteltua toteutustapaa, mikä voidaan katsoa sen selkeäksi eduksi. Tämän lisäksi SSIS käsittää runsaasti erityyppisiä tehtäväkomponentteja, joiden avulla on mahdollista suorittaa ETL-tiedonsiirroille tyyppillisiä toimenpiteitä. Mikäli tiettyä toimintoa, kuten esimerkiksi tiedoston siirtäminen käyttäen apuna SFTP-protokollaa, ei ole saatavilla suoraan SSIS-ympäristön valmiiden komponenttien joukosta, voidaan tämä ongelma kiertää luomalla oma komponentti tarvittavan

toiminnon suorittamista varten. On huomioitava myös, että SSIS-teknologian yhteydessä on mahdollista hyödyntää .NET-ohjelmointiympäristön sisältämiä ominaisuuksia, jonka avulla myös kehitettäville ETL-rajapinnoille asetettavat erikoisemmat vaatimukset voidaan täyttää.

SSIS-teknologian hyödyntäminen ETL-rajapintojen kehittämisessä saattaa herättää ristiriitaisia tuntemuksia sovelluskehittäjien keskuudessa. Tällä tarkoitetaan sitä, että SSIS-teknologiaa on helppoa hyödyntää perustason ETL-rajapintojen kehittämiseen, mutta mikäli tarpeena on monimutkaisempien tiedonsiirtojen luonti, on SSIS-ympäristön toimintaa ymmärrettävä syvällisemmällä tasolla. Perustason ETL-rajapinnoilla tarkoitetaan tässä yhteydessä rajapintoja, joissa tiedonsiirto suoritetaan pääasiassa SQL-ohjelmointikielen avulla. Monimutkaisempien ETL-rajapintojen, joissa käsitellään tietovarastoissa olevia tietoja, kehittäminen vaatii kuitenkin myös SSIS:n Data Flow -tehtäväkomponentin ja erityyppisten muunnoskomponenttien käyttöä ja sen myötä tuntemusta SSIS:n tiedonsiirtomoottorin toiminnasta.

SSIS-ympäristön Data Flow:n hyödyntämistä tiedonsiirroissa ei voida siis missään nimessä pitää yksinkertaisena. Merkittävä tekijä Data Flow -osan tiedonsiirtomoottorin toiminnassa on esimerkiksi metatieto, mikä määrittää Data Flow:ssa siirrettävän tiedon rakenteen. Mikäli esimerkiksi tiedonsiirtorajapinnan taustalla olevat tietokantarakenteet muuttuvat, on myös SSIS-rajapinnan sisältämät metatiedot päivitettävä vastaamaan muuttunutta tietokantarakennetta. Mikäli tämän tyyppisistä seikoista ei olla tietoisia, saattavat ne aiheuttaa yllättäviäkin virhetilanteita SSIS-rajapintojen tuotantokäytössä. Tämän lisäksi merkittävä osa Data Flow:n toiminnasta muodostuu esimerkiksi sarakeliitoksista, erityyppisistä muistipuskureista ja virheellisten tietojen hallinnasta. Kaikkien näiden osa-alueiden toiminnan ymmärtäminen ja hallitseminen on tärkeää, jotta Data Flow:ta pystytään hyödyntämään oikein ja tehokkaasti.

SSIS on melko uusi teknologia ja sen vuoksi esimerkiksi BIDS-kehitysympäristön käsittämistä ominaisuuksista voidaan löytää paikoin puutteita. Puutteita voidaan havaita siis esimerkiksi BIDS-kehitysympäristön ominaisuuksissa, joiden avulla kehitettäviä SSIS-paketteja voidaan testata. Tässä tutkielmassa havaittiin muun muassa, että BIDS-kehitysympäristö ei tue kuin yhden pysähdyspisteen asettamista yhden Script Task -tehtäväkomponentin sisältämään ohjelmakoodiin yhdellä kertaa. Tämän lisäksi pysähdyspisteiden asettaminen ScriptComponent-muunnoskomponentin sisältämään ohjelmakoodiin ei ole mahdollista. Tämän tyyppiset puutteet saattavat vaikuttaa pieniltä, mutta mikäli tarkoituksena on

esimerkiksi etsiä virheitä kooltaan hyvin suuresta ETL-tiedonsiirrosta, joka sisältää runsaasti Script Task - ja ScriptComponent-komponenttiin sijoitettua ohjelmakoodia, saattaa tämän suorittaminen olla hankalaa ja monimutkaista. Tämän lisäksi on huomioitava, että SSIS-teknologian avulla luotujen tiedonsiirtojen testaamiseen ei ole kehitetty merkittäviä yksikkötestausohjelmistoja.

LÄHTEET

Albrecht, A. & Naumann, F. 2008. *Managing ETL Processes* [elektroninen aineisto]. Potsdam: University of Potsdam, Hasso-Plattner-Institute [viitattu 20.10.2011]. Saatavissa: http://scholar.google.fi

Arshad, A. *SSIS - An Inside View Part 1*. SQLServerPerformance.com [blogi]. 29.1.2009 [viitattu 14.9.2011]. Saatavissa: http://www.sql-server-performance.com/2009/ssis-an-inside-view-part-1/

Brian, K., Veerman, E., Dickinson, G., Hinson, D. & Herbold, D. 2008. *Professional Microsoft SQL Server 2008 Integration Services*. Indianapolis: Wiley Publishing, Inc.

Brobst, S. & Pareek, A. 2009. New Trends in Data Acquisition Services for the Real-Time Enterprise. *Business Intelligence Journal* [elektroninen aineisto]. 2009 nro 1 [viitattu 13.10.2011]. Saatavissa: http://search.proquest.com

Dieter, D. 2009. *Quick Table Transfers (Imports) using SSIS, Bulk Insert or BCP*. SQL Server Planet [blogi]. 1.6.2009 [viitattu 27.10.2011]. Saatavissa: http://sqlserverplanet.com/data-warehouse/quick-table-transfers-imports-using-ssis-bulk-insert-or-bcp/

Duffy, B. 2008. *SSIS 2008 Performance Best Practices* [elektroninen aineisto]. Irish SQL Academy 2008, Microsoft [viitattu 28.12.2011]. Saatavissa: http://download.microsoft.com/

Henry, S., Hoon, S., Hwang, M., Lee, D. & DeVore, M. 2005. *Engineering Trade Study: Extract, Transform, Load Tools for Data Migration* [elektroninen aineisto]. Charlottesville: University of Virginia, Department of Systems and Information Engineering [viitattu: 20.10.2011]. Saatavissa: http://ieeexplore.ieee.org/

Hovi, A. 2009. *Tietovarastot ja business intelligence* [Elektroninen aineisto]. Helsinki: WSOYpro [viitattu 21.9.2011]. Saatavissa: http://www.wsoypro.fi

Johansson, R. & Hotti, M. 2010. *SQL Server 2008 R2; tietovarastointi ja Business Intelligence* [video]. Microsoft Oy [viitattu 6.10.2011]. Saatavissa: http://www.microsoft.com/showcase/fi/fi/

Karppinen, J. 2007. *Tietovarasto automaattisten tarkastusjärjestelmien keräämälle datalle.* Espoo: Teknillinen korkeakoulu, sähkö- ja tietoliikennetekniikan osasto. Diplomityö [viitattu 13.10.2011]. Saatavilla: http://www.cs.hut.fi/Research/COMPSER/TAPAS/diplomityo_jouni_karppinen.pdf

Krudop, M. 2007. Maximizing Your ETL Tool Investment. *Information Mangement* [elektroninen aineisto]. 2007 nro 3 [viitattu 6.11.2011]. Saatavissa: http://search.proquest.com

Krueger, T. *The History of SQL Server Integration Services*. Less Than Dot [blogi]. 10.11.2010 [viitattu 7.9.2011]. Saatavissa: http://blogs.lessthandot.com/index.php/DataMgmt/DBAdmin/history-of-ssis

Kuronen, V. 2008. *X/Open Distributed Transaction Processing Model*. Helsinki: Helsingin yliopisto, tietojenkäsittelytieteen laitos. Seminaariraportti [viitattu 13.3.2012]. Saatavilla: http://www.cs.helsinki.fi/u/vkuronen/seminaarit/tki/dtp-malli-raportti.pdf

Larsen, D. & Garden, E. 2000. *Data Transformation Services (DTS) in SQL Server 2000* [verkkodokumentti]. Microsoft Corporation [viitattu 5.9.2011]. Saatavissa: http://technet.microsoft.com/en-us/library/cc917688.aspx

Martin, P. *Considerations for High Volume ETL Using SQL Server Integration Services* [verkkodokumentti]. Microsoft Corporation [viitattu 4.1.2012]. Saatavissa: http://technet.microsoft.com/en-us/library/cc671624.aspx

Microsoft 2005. *Integration Services Development Enhancements* [verkkodokumentti]. Microsoft Corporation [viitattu 10.9.2011]. Saatavissa: http://msdn.microsoft.com/en-us/library/ms170843%28v=sql.90%29.aspx

Microsoft. *SSIS Package Essentials* [verkkodokumentti]. Microsoft Corporation [viitattu 15.11.2011]. Saatavissa: http://msdn.microsoft.com/en-us/library/cc280511.aspx

Microsoft. *Integration Services Tasks* [verkkodokumentti]. Microsoft Corporation [viitattu 17.11.2011]. Saatavissa: http://msdn.microsoft.com/en-us/library/ms139892.aspx

Microsoft. *Designing Package Control Flow* [verkkodokumentti]. Microsoft Corporation [viitattu 23.11.2011]. Saatavissa: http://msdn.microsoft.com/en-us/library/ms140234.aspx

Microsoft. *Data Flow Task* [verkkodokumentti]. Microsoft Corporation [viitattu 24.11.2011]. Saatavissa: http://msdn.microsoft.com/en-us/library/ms141122.aspx

Microsoft. *Data Source (SSIS)* [verkkodokumentti]. Microsoft Corporation [viitattu 29.11.2011]. Saatavissa: http://msdn.microsoft.com/en-us/library/ms141792.aspx

Microsoft. *Using Data Source Views In Packages* [verkkodokumentti]. Microsoft Corporation [viitattu 1.12.2011]. Saatavissa: http://msdn.microsoft.com/en-us/library/ms141097.aspx

Microsoft. *Considerations for Installing Integration Services* [verkkodokumentti]. Microsoft Corporation [viitattu 3.12.2011]. Saatavissa: http://msdn.microsoft.com/en-us/library/ms143731.aspx

Microsoft. *Introducing Business Intelligence Development Studio* [verkkodokumentti]. Microsoft Corporation [viitattu 12.12.2011]. Saatavissa: http://msdn.microsoft.com/en-us/library/ms173767.aspx

Microsoft. *Integration Services Connections* [verkkodokumentti]. Microsoft Corporation [viitattu 29.12.2011]. Saatavissa: http://msdn.microsoft.com/en-us/library/ms140203.aspx

Microsoft. *OPENROWSET (Transact-SQL)* [verkkodokumentti]. Microsoft Corporation [viitattu 18.1.2012]. Saatavissa: http://msdn.microsoft.com/en-us/library/ms190312.aspx

Microsoft. *Data Flow Elements* [verkkodokumentti]. Microsoft Corporation [viitattu 22.1.2012]. Saatavissa: http://msdn.microsoft.com/en-us/library/ms140080.aspx

Microsoft. *Understanding Synchronous and Asynchronous Transformations* [verkkodokumentti]. Microsoft Corporation [viitattu 26.1.2012]. Saatavissa: http://msdn.microsoft.com/en-us/library/aa337074.aspx

Microsoft. *OLE DB Command Transformation* [verkkodokumentti]. Microsoft Corporation [viitattu 4.2.2012]. Saatavissa: http://msdn.microsoft.com/en-us/library/ms141138.aspx

Microsoft. *Script Task* [verkkodokumentti]. Microsoft Corporation [viitattu 9.2.2012]. Saatavilla: http://msdn.microsoft.com/en-us/library/ms141752.aspx

Microsoft. *VSTARTScriptObjectModelBase Class* [verkkodokumentti]. Microsoft Corporation [viitattu 18.2.2012]. Saatavissa: http://social.msdn.microsoft.com/search/

Microsoft. *Script Component* [verkkodokumentti]. Microsoft Corporation [viitattu 20.2.2012]. Saatavissa: http://msdn.microsoft.com/en-us/library/ms137640.aspx

Microsoft. *Creating an Asynchronous Transformation with the Script Component* [verkkodokumentti]. Microsoft Corporation [viitattu 26.2.2012]. Saatavissa: http://msdn.microsoft.com/en-us/library/ms136133.aspx

Microsoft. *dtexec Utility (SSIS Tool)* [verkkodokumentti]. Microsoft Corporation [viitattu 20.3.2012]. Saatavissa: http://msdn.microsoft.com/en-us/library/ms162810.aspx

Microsoft. *Debugging Script* [verkkodokumentti]. Microsoft Corporation [viitattu 26.3.2012]. Saatavissa: http://msdn.microsoft.com/en-us/library/ms142157.aspx

Microsoft. *Coding and Debugging the Script Component* [verkkodokumentti]. Microsoft Corporation [viitattu 26.3.2012]. Saatavissa: http://msdn.microsoft.com/en-us/library/ms136033.aspx

Microsoft. *Debugging Data Flow* [verkkodokumentti]. Microsoft Corporation [viitattu 1.4.2012]. Saatavissa: http://msdn.microsoft.com/en-us/library/ms137944.aspx

Mimno, P. 2001. How to Select an Extraction/Transformation/Loading (ETL) Tool. *Federal Computer Week* [elektroninen aineisto]. 2001 nro 21 [viitattu 10.11.2011]. Saatavissa: http://search.proquest.com

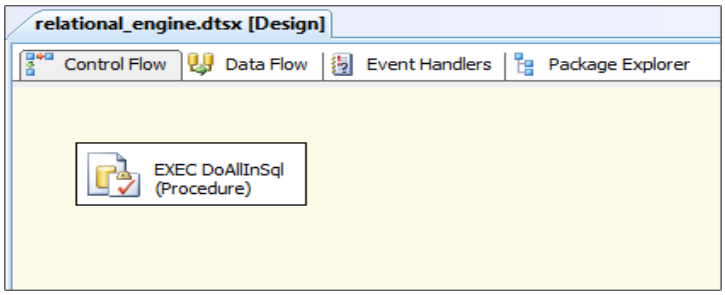
Mundy, J., Thornthwaite, W. & Kimball, R. 2011. *The Microsoft Data Warehouse Toolkit: With SQL Server 2008 R2 and the Microsoft Business Intelligence Toolset* [elektroninen aineisto]. Indianapolis: Wiley Publishing, Inc. [viitattu 6.10.2011]. Saatavissa: http://books.google.com

Noor, D. *Tuning Your SSIS Package Data Flow in the Enterprise* [video]. Microsoft Corporation [viitattu 6.1.2012]. Saatavissa: http://technet.microsoft.com/en-us/sqlserver/ff686901.aspx

Payne, A. 2005. Business Intelligence gets a make over. *Intelligent Enterprise* [elektroninen aineisto]. 2005 nro 7 [viitattu 1.10.2011]. Saatavissa: http://search.proquest.com/

Policht, M. 2010. *Introducing SQL Server 2008 and 2008 R2 Integration Services* [verkkodokumentti]. *Database Journal* [viitattu 3.12.2011]. Saatavissa: http://www.databasejournal.com/

SQL Server -tietokantamoottoria hyödyntävän SSIS-paketin rakenne BIDS-kehitysympäristössä ja tiedonsiirron suorittavan tietokantaproseduurin SQL-koodi

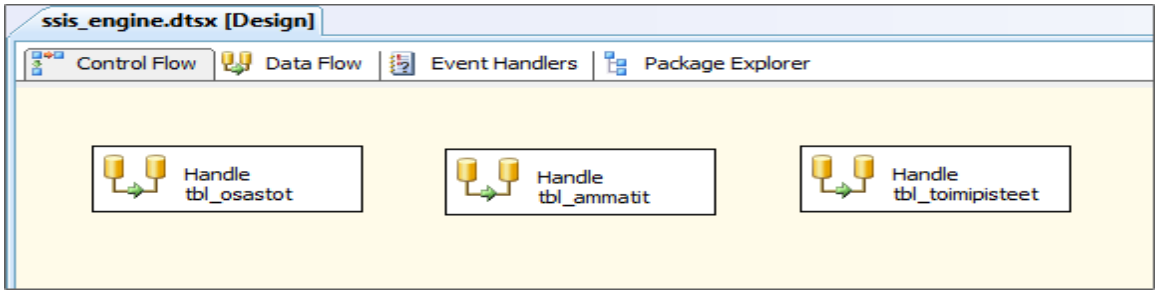


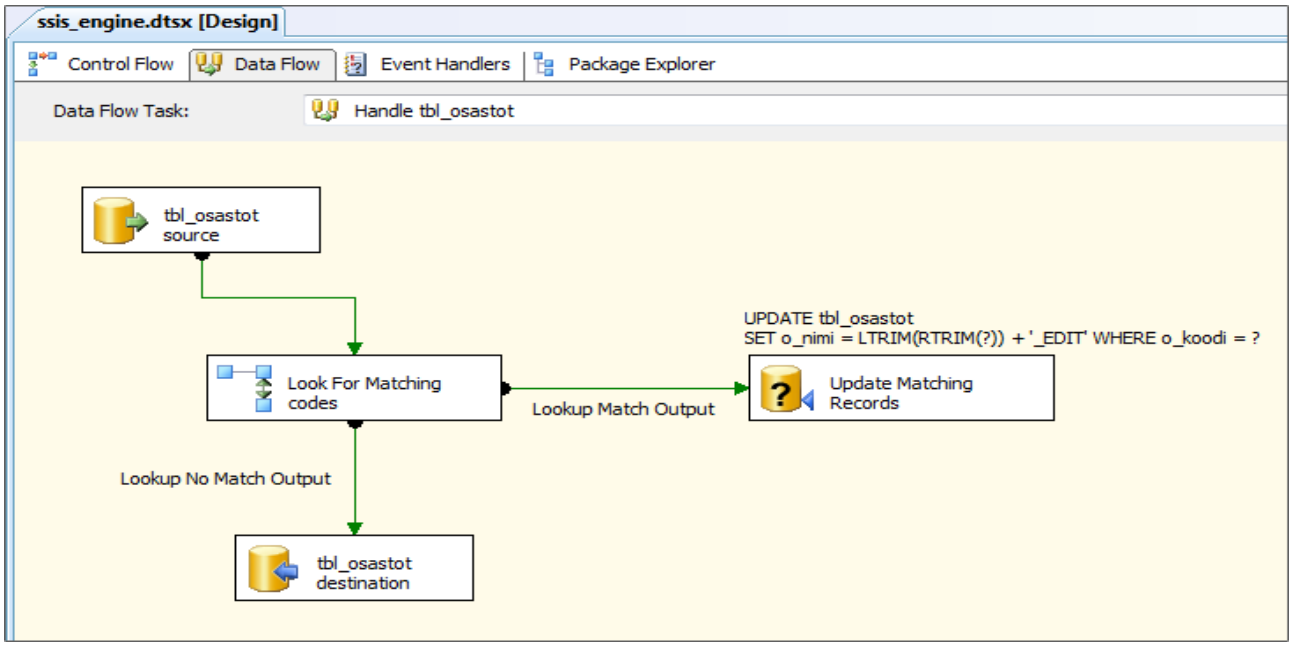

```
CREATE PROCEDURE [dbo].[DoAllInSql]
AS

--Päivitetään vanhat ammatit
UPDATE
        thesis_db.dbo.tbl_ammatit
SET
        a_nimi = LTRIM(RTRIM(db2.a_nimi)) + '_EDIT'
FROM
        thesis_db2.dbo.tbl_ammatit db2
WHERE
        tbl_ammatit.a_koodi = db2.a_koodi


--Lisätään uudet ammatit
INSERT INTO thesis_db.dbo.tbl_ammatit (a_nro, a_nimi, a_koodi)

SELECT
        db2.a_nro, db2.a_nimi, db2.a_koodi
FROM
        thesis_db2.dbo.tbl_ammatit db2
WHERE
        NOT EXISTS
        (
                SELECT 'exists' FROM thesis_db.dbo.tbl_ammatit db1
                WHERE db1.a_koodi = db2.a_koodi
        )
```

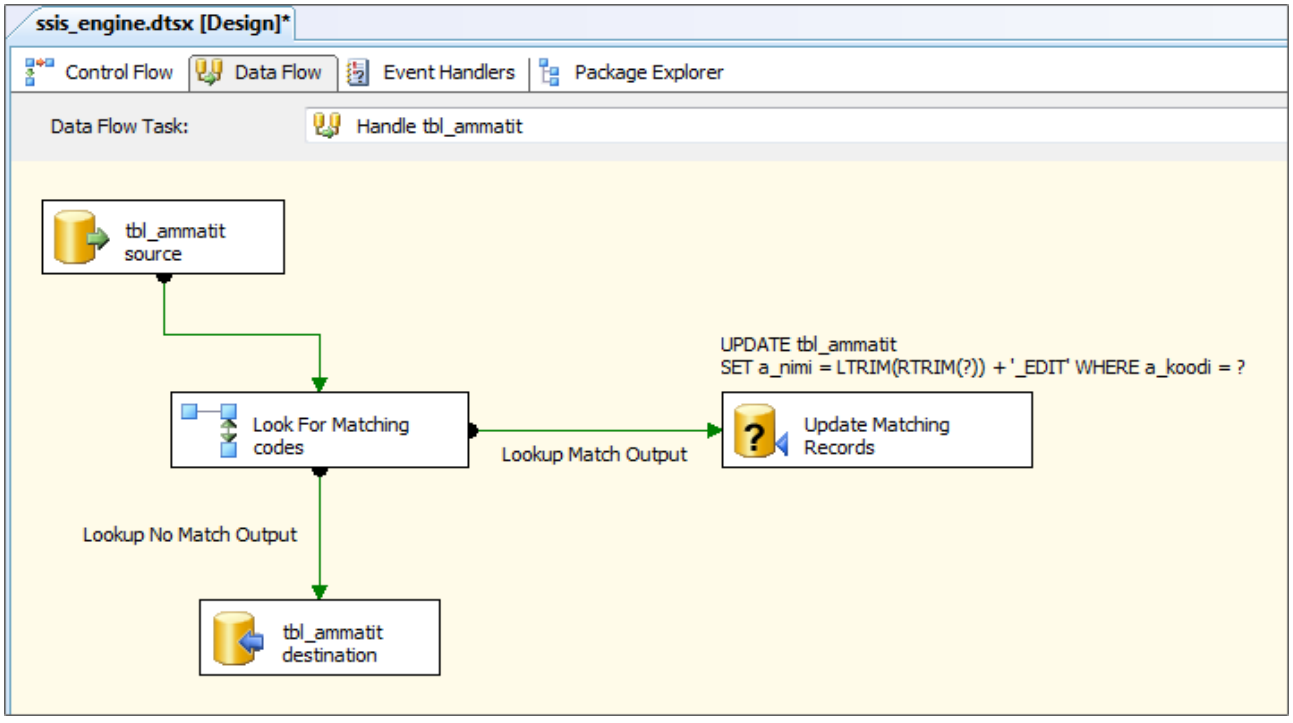
```
--Päivitetään vanhat osastot
UPDATE
		thesis_db.dbo.tbl_osastot
SET
		o_nimi = LTRIM(RTRIM(db2.o_nimi)) + '_EDIT'
FROM
		thesis_db2.dbo.tbl_osastot db2
WHERE
		tbl_osastot.o_koodi = db2.o_koodi


--Lisätään uudet osastot
INSERT INTO thesis_db.dbo.tbl_osastot (o_nro, o_nimi, o_koodi)

SELECT
		db2.o_nro, db2.o_nimi, db2.o_koodi
FROM
		thesis_db2.dbo.tbl_osastot db2
WHERE
		NOT EXISTS
		(
			SELECT 'exists' FROM thesis_db.dbo.tbl_osastot db1
			WHERE db1.o_koodi = db2.o_koodi
		)

--Päivitetään vanhat toimipisteet
UPDATE
		thesis_db.dbo.tbl_toimipisteet
SET
		t_nimi = LTRIM(RTRIM(db2.t_nimi)) + '_EDIT'
FROM
		thesis_db2.dbo.tbl_toimipisteet db2
WHERE
		tbl_toimipisteet.t_koodi = db2.t_koodi


--Lisätään uudet toimipisteet
INSERT INTO thesis_db.dbo.tbl_toimipisteet (t_nro, t_nimi, t_koodi)
```

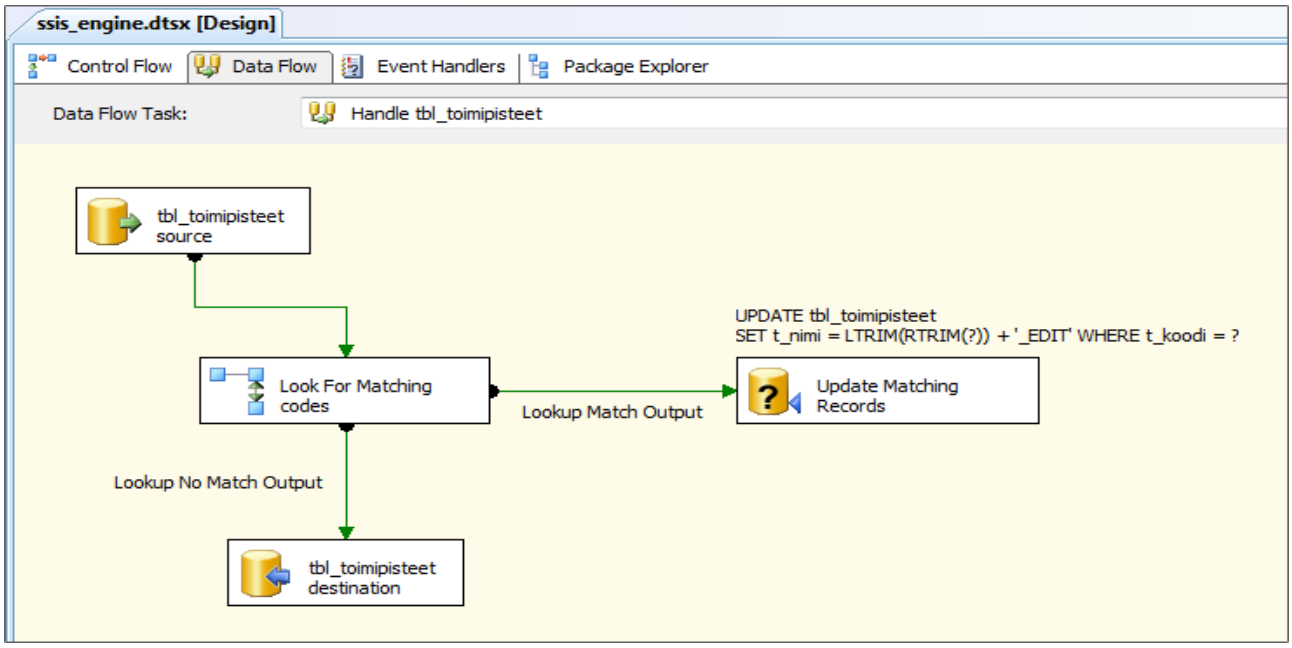
```
SELECT
        db2.t_nro, db2.t_nimi, db2.t_koodi
FROM
        thesis_db2.dbo.tbl_toimipisteet db2
WHERE
        NOT EXISTS
        (
                SELECT 'exists' FROM thesis_db.dbo.tbl_toimipisteet db1
                WHERE db1.t_koodi = db2.t_koodi
        )
```

SSIS-tiedonsiirtomoottoria hyödyntävän SSIS-paketin rakenne BIDS-kehitysympäristössä








ssis_engine.dtsx [Design]
Control Flow
Data Flow
Event Handlers
Package Explorer
Data Flow Task:
Handle tbl_toimipisteet
tbl_toimipisteet source
UPDATE tbl_toimipisteet
SET t_nimi = LTRIM(RTRIM(?)) + '_EDIT' WHERE t_koodi = ?
Look For Matching codes
Lookup Match Output
Update Matching Records
Lookup No Match Output
tbl_toimipisteet destination

www.savonia.fi

Savonia
ammattikorkeakoulu