pmOne

Smart Start Dataload (SSDL)

Technische Dokumentation

• Datum: 14.03.2016 • Version: 2.0 •

• Ansprechpartner: Thanh Tung Nguyen

• Telefon: +43 676 838 52 338

Inhaltsverzeichnis

1 Überblick DWH Architektur

1.1 Technisches Konzept

Aus den Vorsystemen werden Daten als 1:1 Abbild in eine sog. „Staging Area“ auf dem SQL-Server eingelesen, von dort erfolgt eine Weiterverarbeitung von technischen Daten zum fachlichen Zielmodell in ein Core Datawarehouse (CDW), eventuelle Berechnungen können ebenfalls in diesem Ladeprozess erfolgen. Eine weitere Architektur-Schicht dient zur relationalen Auswertung und als Basis für den Analyse-Cube, welcher ad-hoc oder mit Standard-Reporting abgefragt wird.

Alle Architekturschichten befinden sich auf dem SQL-Server, die Ladeprozesse (ETL) werden mittels „Integration Services“ (SSIS) umgesetzt, eine Zeitsteuerung der Ladeprozesse ist mittels „SQL Server Agent“ möglich, der Analysecube läuft auf einer Analysis Datenbank (SSAS), für die ad-hoc Analyse ist Excel notwendig, die Standardreports werden mittels cMORE/XL erstellt.

Weitere Abfragesysteme können ab dem Layer 3 angebunden werden.

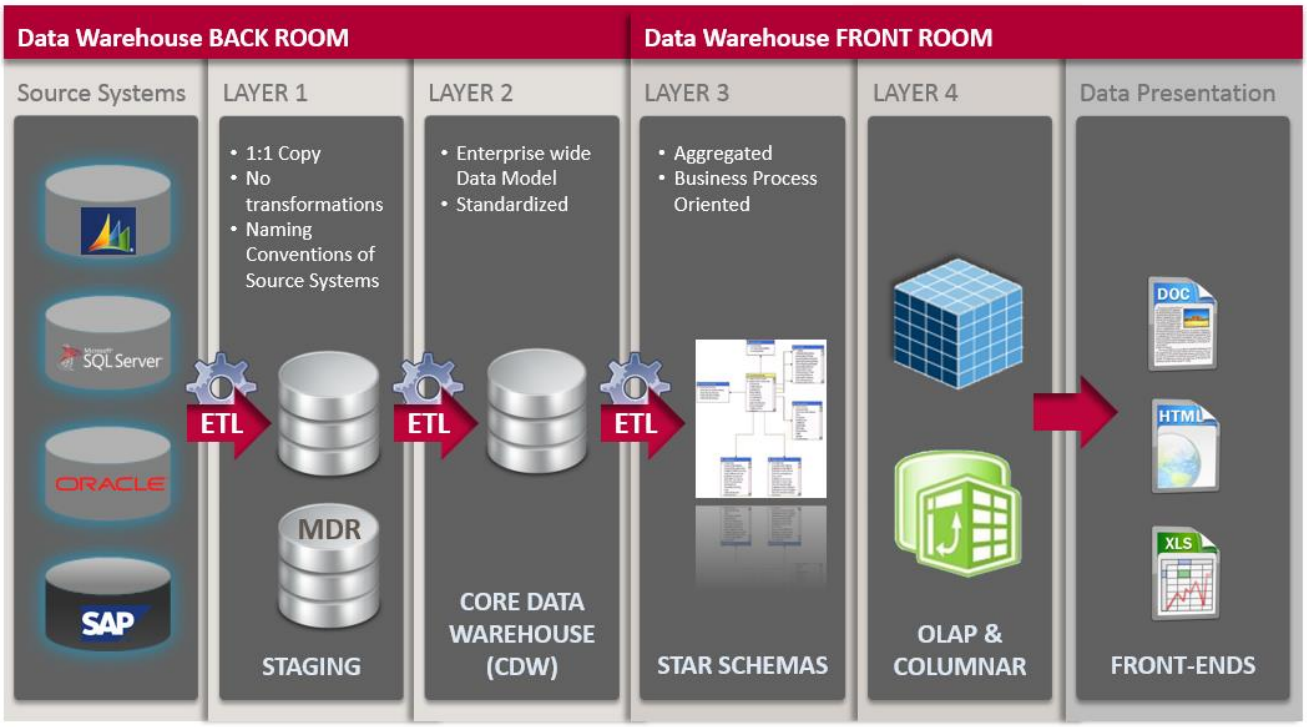


Abbildung 1: Überblick DWH Architektur

1.2 Tools und Datenbanken

Die Architektur setzt den Einsatz von Microsoft Technologien voraus, die im Genaueren Folgendes beinhalten müssen:

- Entwicklung: SQL Data Tools (SSDT) oder Visual Studio mit Business Intelligence Unterstützung
- Betrieb: Microsoft SQL Server mit SSIS Catalog
- Reports (optional): Microsoft Excel, cMORE/XL

Als Mindeststandard empfehlen wir ab der Version 2010 für die Tools (bei SQL Server 2012).

Wenn Sie das pmOne Framework “Smart Start Dataload” (SSDL) mit dem Entwicklungstool ab der Version 2013 einsetzen möchten, dann ist ein Upgrade der Pakete erforderlich. Die Pakete sind aufwärtskompatibel, aber nicht abwärtskompatibel. D.h. Sie können beispielsweise die SSIS Pakete auf die Version 2013 upgraden, aber nicht zurück auf die Version 2012 downgraden.

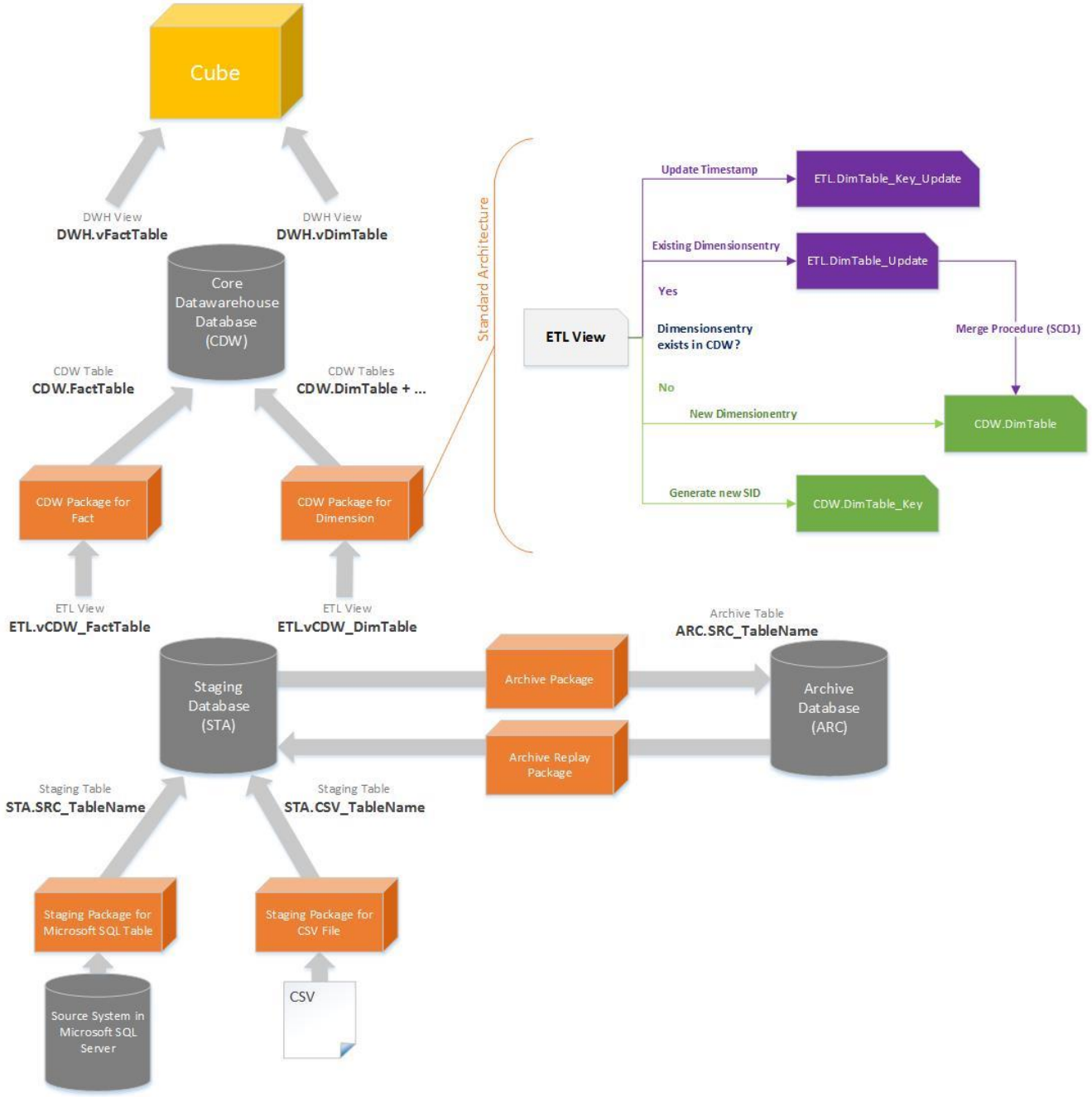
Das Framework setzt auf das Project Deployment Model auf, sowie einen SSIS Catalog für das Deployment auf dem SQL Server.

Des Weiteren benötigen Sie folgende Datenbanken, die in SSDL als Backupdatei und als Datenbankprojekt enthalten sind:

- StagingDB OneTemplateStaging
- CoreDwDB OneTemplateCoreDW
- LoggingDB OneTemplateLogging
- (optional) ArchivDB OneTemplateArchiv

Zur Initialisierung empfehlen wir die Datenbanken aus den Backup Dateien zu erstellen und für weitere Änderungen können Sie es über das Datenbankprojekt pflegen, das besonders geeignet für die Übertragung von Datenbankstruktur vom Testserver auf den Produktivserver ist.

1.3 Standardarchitektur



1.4 ETL Prozess und Logging

Für den Ladeprozess (ETL) kommen SSIS Pakete mit vorbereiteter Ablauflogik für Archiv und Historisierung, Datensatzzähler, Logging und Error-Handling durch Eventhandler zum Einsatz, die dem pmOne Best Practice entsprechen.

Dazu wurde ein Template mit allen relevanten Komponenten erstellt, welches dann im Folgenden für alle Pakete angewendet werden kann. Eine eigene Logging-Datenbank empfängt die Ablaufinformationen und kann für den Betrieb ausgewertet und bei Bedarf als Report dargestellt werden. Im Framework haben Sie verschiedene Anwendungsfälle ein passendes Template. Weiteres enthält SSDL ein gesamtes Logging-

Framework („OneLog“) und entsprechende Reports für den Betrieb. Die Steuerung der Ladeprozesse erfolgt mit einem Masterpaket und Agent-Job, inklusive Logging.

1.5 Staging Area (STA)

Die Daten von den Vorsystemen werden in eine zentrale Sammelstelle abgelegt, die sogenannte Staging Area. Es ist 1:1 Abbild des Quellsystems, wobei nicht alle Tabellen vorhanden sind, sondern nur für die Überleitung ins Datenmodell (ETL Views) relevant sind. Das fachliche Datenmodell wird in der Staging Area abgebildet, das eventuell Berechnungen, Business Logik, usw. beinhalten können, im Weiteren die Dimensionen und Fakten darstellen.

Meistens können die Datenquellen sehr vielfältig sein, wie z.B. Oracle, Navision, SAP, usw. Das pmOne Framework stellt ein Template für die Beladung von einer Microsoft SQL Tabelle oder CSV Datei in die Staging Area zur Verfügung. In anderen Fällen brauchen Sie eventuell einen zusätzlichen Treiber oder Software. Wir bieten für SAP als Vorsystem unser Produkt cMORE Connect mit vorgefertigten Standardmodellen für verschiedene Module an.

Folgende Schemen und Namenskonvention sind zu benutzen:

- **STA** für Staging Tabellen aus dem Vorsystem
  - Namenskonvention: **[STA].[Kürzel vom Vorsystem + „_“ + Tabellenname vom Vorsystem]**
  - z.B. STA.AW_Customer und STA.CSV_KundenZusatzinformation
- **MAN** für manuelle Stammdaten
  - Typische Charakteristik
    - Beladung durch CSV-Dateien (keine Bewegungsdaten), die für eine CDW Dimension benötigt werden, z.B. Hierarchie
    - Kaum Veränderung in der Struktur
    - Manuelle Pflege der Daten
  - Namenskonvention: **[MAN].[Tabellenname]**
  - z.B. MAN.SalesTerritory
- **ETL** (Views) für die Beladung ins CDW (Überleitung in das fachliche Datenmodell)
  - Namenskonvention für Dimension: **„vCDW_Dim“ + Dimensionsname**
  - Namenskonvention für Fakt: **„vCDW_Fact“ + Faktenname**
  - z.B. [ETL].[vCDW_FactBalances] und [ETL].[vCDW_DimKunde]

1.5.1 Aufbau Staging View (ETL)

Die Zusammenführung von Vorsystemdaten zu einem einheitlichen fachlichen Datenmodell wird in einer Staging View abgebildet und es dient im Weiteren als Datenbasis für das DWH.

Der Aufbau einer Staging View für Standarddimensionen setzt sich aus:

- Business Key oder aus mehreren Business Keys
- Attribute
  - empfehlenswert mit expliziter Konvertierung, z.B. CONVERT(nvarchar(50), ATTR) AS Attribut
- SystemCode

Und bei den Fakten:

- Datum oder mehrere Datumsfelder (Roleplaying)
- Business Keys (für die Verknüpfung der Dimensionen)
- Measures
- SystemCode

In jeder ETL View muss am Ende eine „**SystemCode**“ Spalte (Kürzel vom Vorsystem) angeführt sein, um den Herkunft zu identifizieren:

z.B. `CONVERT(nvarchar(10),'AW') AS SystemCode`

Der SystemCode ist in der Tabelle „MAN.System“ (CDW DB) ebenfalls zu pflegen.

1.6 Core Data Warehouse (CDW)

Das Core Datawarehouse (CDW) ist die zentrale Datenbank für das DWH und enthält die transformierten Staging-Daten in Form eines fachbezogenen Modells, welches dem Konzept des Starschemas verfolgt. Eine weitere wichtige Funktion des CDW ist die Historisierung nach SCD2.

Nach unserem pmOne Best Practice unterscheiden wir in der CDW-Datenbank folgende Arten von Tabellen:

- **Dimensions-Tabellen (CDW Schema)**: Die CDW Haupttabellen sind die Grundlage für die Dimensionen im Cube. Sie enthalten die generierte SID (Surrogate Key), alle Business Keys des Quellsystems, alle Dimensionsattribute sowie Audit-Spalten.
- **Key-Tabellen (CDW Schema)**: Diese Tabellen dienen zur Generierung der SIDs (eines kurzen, eindeutigen, Integer-Schlüssels). Eine SID ist ein eindeutiger Integer-Schlüssel in DWH, der in SSAS performanter abgefragt werden kann als mit (potentiell alphanumerischen) Natural Keys des Quellsystems.
- **Update-Tabellen (ETL Schema)**: Im ETL Prozess agieren diese Tabellen als Zwischenspeicher, damit eine Aktualisierung in einer Gesamttransaktion über die MERGE Prozedur vom Zwischenspeicher auf die CDW Haupttabelle erfolgen kann. Dabei werden die Attribute in der Dimensionstabelle durch neue Werte von der Update Tabelle überschrieben (SCD1). Der Zwischenspeicher wird über das Paket von der ETL View (Staging Daten) beladen.
- **Fakten-Tabellen (CDW Schema)**: Diese Tabellen enthalten die Bewegungsdaten (Fakten) des Quellsystems, bspw. die Finanzkennzahlen, die Buchungskennzahlen oder die Personalkennzahlen.
- **Steuerungs-Tabellen (MAN Schema)**: Manuelle Tabellen sind händisch zu pflegen. Hier findet keine automatische Beladung oder Datenänderung statt. Sie werden bspw. genutzt um den Datenherkunft bzw. Quellsysteme zu ermitteln oder die speziellen Kostenarten für die Forecast-Berechnung zu vermerken.
- **History-Tabellen (optional):** In der History-Tabelle werden die SCD2 Datensätze mit entsprechenden Gültigkeiten gespeichert mit jener Attribut-Kombination die historisiert werden soll. Die Tabelle hat eine eigene SID als Primary Key (History SID), eine SID mit der Referenz zur CDW Haupttabelle, SCD2-Attribute, ein Current Flag und Audit Fields.

Um die verschiedenen Arten von CDW-Tabellen zu unterscheiden, verfolgt die Namensvergebung nach folgender Namenskonvention, damit bestimmte Einrichtungen des Ladeprozesses automatisiert werden können:

- **Dimensionen** beginnen mit „**Dim**“
  z.B. CDW.DimCustomer
- **Fakten** beginnen mit „**Fact**“
  z.B. CDW.FactBalances
- **Key-Tabelle**: Dimension + „**_Key**“
  z.B. CDW.DimCustomer_Key
- **History-Tabelle**: Dimension + „**_History**“
  z.B. CDW.DimCustomer_History
- **Update-Tabelle**: Dimension + „**_Update**“ + Dimension + „**_Key_Update**“
  z.B. ETL.DimCustomer_Update und ETL.DimCustomer_Key_Update

☝ Hinweis:
Mithilfe des pmOne Frameworks können Sie die oben genannten CDW Tabellen über eine Prozedur generieren lassen. Nähere Information über die Verwendung der Prozedur können Sie unter dem Punkt 2 CDW Generierungsprozeduren nachschlagen.

Manuelle Stammdaten in CDW

Folgende Standardtabellen werden im **Schema „MAN“ in CDW** abgelegt und nicht durch die Staging Area beladen, weil diese in der Regel keine Business Logik haben bzw. vom Quellsystem unabhängig sind:

- MAN.DimDatenart — Englisch: MAN.DimDataType
- MAN.DimZeitvergleich — Englisch: MAN.DimTimeAnalysis
- MAN.DimZeittyp — Englisch: MAN.DimTimeType
- MAN.DimSystem — Englisch: MAN.DimSystem

1.6.1 Roleplaying

In vielen Fällen kann eine Dimension verschiedene Ausprägungsarten in den Fakten annehmen (Roleplaying), z.B. eine Zeitdimension kann für ein Buchungs- und Rechnungsdatum verwendet werden. Zur Unterscheidung ist folgende Namenskonvention notwendig:

SID Feld + „_“ + Names des Roleplaying

Beispiele: DimDate_SID_Buchung und DimDate_SID_Rechnung
DimCustomer_SID_Lieferant und DimCustomer_SID_Rechnung

1.7 DWH Views

Basierend auf die CDW Tabelle wird eine DWH View erstellt, die als Basis für den Analyse-Cube dient. Die neue View soll nach der unten angeführten Namenskonvention im Schema DWH abzulegen. Die View beinhaltet alle vorhandene Spalten der CDW Tabelle, ausgenommen der Audit Spalten, und die SID Spalte wird nach folgender Namenskonvention umbenannt, damit es in SSAS beim Dimensionsmapping die Auswahl vorselektiert wird:

Anpassung:

SID Spalte => Dimensions-/Faktenobjekt (mit Suffix) + „_SID“

DWH View Namenskonvention =>

Dimension „[DWH].[vDim“ + Dimensionsname + „]“

Fakten „[DWH].[vFakt“ + Faktenname + „]“

z.b. „DimProject_SID“

```
CREATE VIEW [DWH].[vDimProject]
AS
SELECT [SID] AS DimProject_SID
      ,[ProjectID]
      ,[ProjectName]
      ,[LanguageID]
      ,[ProjectID_External]
      ...
FROM [CDW].[DimProject]
```

1.7.1 Historisierung

Im Gegensatz zur nicht historisierten Dimension werden alle Dimensionsveränderungen ausgegeben und diese mit einer History SID identifiziert, die im Cube mit den Fakten verknüpft werden. In der DWH Faktenview wird der Stand der historisierten Dimension zum führenden Datum ermittelt. Der gültige Datensatz, der durch die History SID eindeutig ist, wird zu dem Zeitpunkt in der Faktenview ausgegeben und später im Cube mit der historisierten Dimension verknüpft:

Beispiel:

DWH.vFaktVerkaufsauftrag (Ausschnitt):

```
…
,COALESCE(vgh.[HISTORY_SID],-1) AS DimVerkaufsgebiet_History_SID
…
LEFT JOIN [CDW].[DimVerkaufsgebiet_History] vgh ON va.DimVerkaufsgebiet_SID = vgh.[SID]
AND va.Bestelldatum BETWEEN vgh.VALID_FROM AND vgh.VALID_TO
```

DWH.vDimVerkaufsgebiet:

```
SELECT
       vgh.[DimVerkaufsgebiet_History_SID]
                                        -- Eindeutige ID für Cube Dimension Mapping
       ,vg.[SID] AS [DimVerkaufsgebiet_SID]
       ,vg.[Verkaufsgebietnummer]
       ,vg.[Verkaufsgebietname]         -- Nicht Hist. von der Haupttabelle (akt. Wert)
       ,vgh.[Land]                      -- Historisiertes Feld von History Tabelle
       ,vgh.[Gruppe]                    -- Historisiertes Feld von History Tabelle
FROM [CDW].[DimVerkaufsgebiet] vg
INNER JOIN [CDW].[DimVerkaufsgebiet_History] vgh
ON vg.[SID] = vgh.[SID]
```

2 CDW Generierungsprozeduren

Um den Aufbau des CDWs soweit wie möglich zu vereinfachen bzw. zu automatisieren, wurden einige Generierungsprozeduren in das Framework eingebaut, die dabei helfen die CDW Tabellen und die erforderlichen Prozeduren nach dem pmOne Best Practice automatisch zu erstellen.

Zur Erstellung von CDW Objekten können folgende Prozeduren verwendet werden:

- **[ETL].[GenerateCdwDim]**
  Hauptprozedur für die CDW Dimensionserstellung mit allen CDW Objekten (CDW Tabellen, Dummy, Merge und Merge History Prozedur bei Historisierung).

- **[ETL].[GenerateCdwTables]**
  Erstellt die CDW Tabellen (Update, Key, usw.), siehe 1.5 Core Data Warehouse (CDW)

- **[ETL].[GenerateDummyProc]** Generierungsprozedur für Dummy
- **[ETL].[GenerateMergeMain]** Generierungsprozedur für Merge
- **[ETL].[GenerateMergeCdwHistory]** Generierungsprozedur für Merge History (Historisierung in CDW)
- **[ETL].[GenerateMergeCdwHistoryOrigValidDates]** Generierungsprozedur für Merge History (Historisierung vom Quellsystem)

☝ Hinweis:
In der Regel benötigen Sie nur die erste CDW Prozedur „GenerateCdwDim“ für die Dimensionserstellung, wenn Sie nicht separate CDW Objekte erstellen wollen, denn diese Prozedur führt implizit die anderen Generierungsprozeduren für CDW Tabellen, Dummy-, Merge- und im Falle einer Historisierung Merge History Prozedur aus.

2.1 Generierung der CDW Objekte - [ETL].[GenerateCdwDim]

CDW Tabellen, Dummy-, Merge- und im Falle einer Historisierung Merge History Prozedur

Parameter	Beschreibung
@strViewName	Voller Name der ETL View, z.B. „ETL.vCDW_DimKunde“
@strDimName	Dimensionsname mit „Dim“ als Präfix, z.B. „DimKunde“
@strKeyCols	Liste von Business Keys, getrennt durch Komma – z.B. „Kundennummer“
@strHistcols	Liste von zu historisierenden Feldern – z.B. „Land, Gruppe“ „NULL“ - keine Historisierung [Default]
@DropKeyTbl	Flag für die Löschung der Keys „1“ - Löschen [Default] „0“ - Beibehalten
@DatabaseNameTarget	Zieldatenbank bzw. CDW Datenbank für das Anlegen der CDW Objekte
@DatabaseNameSource	Quelldatenbank bzw. Staging Datenbank für das Auslesen der ETL View (Dimensionsview)

@ExecProc ***[optional]***	Flag für die automatische Ausführung der Generate Prozeduren (Dummy, Merge und MergeHistory) „1“ - Ausführen [Default] „0“ - Generierte Codes anzeigen
@strDestinationSchema ***[optional]***	Schemaname für das Anlegen der CDW Tabellen „CDW“ [Default]
@strETLSchema ***[optional]***	Schemaname für das Anlegen der CDW Prozeduren „ETL“ [Default]

Tabelle 1: GenerateCdwDim Parameter

☝ Hinweis:
Optionale Parameter werden mit Default Werten ausgeführt, die dem pmOne Standard entsprechen, und daher müssen Sie es standardmäßig nicht setzen.

2.2 Generierung CDW Tabellen - [ETL].[GenerateCdwTables]

Auf Basis von der ETL View werden die CDW Tabellen (Update, History, usw.) in der angegeben Zieldatenbank erstellt.

Parameter	Beschreibung
@strViewName	Voller Name der ETL View, z.B. „ETL.vCDW_DimKunde“
@strDimName	Dimensionsname mit „Dim“ als Präfix, z.B. „DimKunde“
@strHistcols	Liste von zu historisierenden Feldern – z.B. „Land, Gruppe“ „NULL“ - keine Historisierung [Default]
@DropKeyTbl	Flag für die Löschung der Keys „1“ - Löschen [Default] „0“ - Beibehalten
@DatabaseNameTarget	Zieldatenbank bzw. CDW Datenbank für das Anlegen der CDW Objekte
@DatabaseNameSource	Quelldatenbank bzw. Staging Datenbank für das Auslesen der ETL View (Dimensionsview)

Tabelle 2: GenerateCdwTables Parameter

2.3 Generierung Dummy Prozedur - [ETL].[GenerateDummyProc]

Die Prozedur liest die Struktur aus der CDW Haupttabelle aus, damit jedes Feld vom Dummy Eintrag mit dem richtigen Datentyp befüllt werden kann, und erstellt eine neue Prozedur, die den Dummy Eintrag in die CDW Haupttabelle und bei eingeschalteter Historisierung auch in die History Tabelle einfügt.

Parameter	Beschreibung
@procSchema	Schemaname für das Ablegen der neu generierten Dummy Prozedur
@dstSchema	Schemaname der CDW Haupttabelle (standardmäßig „CDW“)
@tableName	Tabellenname der CDW Haupttabelle bzw. Dimensionsname, z.B. „DimKunde“

Tabelle 3: GenerateDummyProc Parameter

2.4 Generierung Merge Prozedur - [ETL].[GenerateMergeMain]

Die Prozedur liest die Struktur von der CDW Haupttabelle aus, damit für jedes Attribut ein Code Block generiert werden kann, der abfragt, ob bei dem Feld eine Veränderung zwischen der CDW Haupt- und der Update Tabelle (Daten von Staging) vorliegt. Im Falle einer Veränderung werden alle Attribute in der Haupttabelle von der Update Tabelle überschrieben (SCD1). Im Grunde besteht die neu generierte Prozedur ein MERGE Statement mit dem Code Block für die Überprüfung eines Attributs und das Update dazu.

Parameter	Beschreibung
@tableName	Tabellenname der CDW Haupttabelle bzw. Dimensionsname, z.B. „DimKunde“
@tableNameSource	Tabellenname der CDW Update Tabelle, z.B. „DimKunde_Update“
@businessKeys	Liste von Business Keys, getrennt durch Komma – z.B. „Kundennummer“
@dstSchema ***[optional]***	Schemaname der CDW Haupttabelle „CDW“ [Default]
@srcSchema ***[optional]***	Schemaname der CDW Update Tabelle „ETL“ [Default]

Tabelle 4: GenerateMergeMain Parameter

2.5 Generierung Merge History Prozedur (CDW Historisierung) – [ETL].[GenerateMergeCdwHistory]

Ähnlich zum Merge Main generiert es eine Prozedur mit einem MERGE Statement, aber mit dem Unterschied, dass historisierte Datensätze der IsCurrent Flag auf „nicht aktuell“ und das Gültigkeitsdatum auf den Vortag gesetzt wird, wenn sich dabei die historisierte Attribute geändert haben. Der neue bzw. aktuelle Datensatz wird als Output aus der Merge Prozedur an das CDW Paket übergeben und in die History Tabelle beladen.

Parameter	Beschreibung
@TableName	Tabellenname der CDW History Tabelle, z.B. „DimKunde_History“
@TableNameSource	Tabellenname der CDW History Update Tabelle, z.B. „DimKunde_History_Update“
@BusinessKeys	Liste von Business Keys, getrennt durch Komma – z.B. „Kundennummer“
@IncludeDeletePart	Flag für das Markieren von Datensätzen als nicht aktuell (IsCurrent), die nicht in den Staging Daten (History Update Tabelle) vorhanden sind, aber in CDW „0“ [Default] – IsCurrent beibehalten, wenn der Datensatz nicht in den Staging Daten vorkommt
@SCD2Columns	Liste von zu historisierenden Feldern – z.B. „Land, Gruppe“
@DestSchema ***[optional]***	Schemaname der CDW History Tabelle „CDW“ [Default]
@SourceSchema ***[optional]***	Schemaname der CDW History Update Tabelle „ETL“ [Default]
@IsCurrentName ***[optional]***	Feldname von IsCurrent Flag „IS_CURRENT“ [Default]

@SpecialColumns ***[optional]***	Felder zum Ausschließen „IS_CURRENT, VALID_FROM, VALID_TO, INSERT_AUDIT_ID, UPDATE_AUDIT_ID, INSERT_AUDIT_DATE" [Default]

Tabelle 5: GenerateMergeCdwHistory Parameter

2.6 Generierung Merge History Prozedur (Historisierung vom Quellsystem) - [ETL].[GenerateMergeCdwHistoryOrigValidDates]

Im Gegensatz zu Merge History werden die historisierenden Felder, IsCurrent Flag und Gültigkeitsdatum von der History Update Tabelle (Daten vom Staging/Quellsystem) bzw. die historische Entwicklung vom Quellsystem direkt übernommen. Auch hier werden neue Datensätze über das CDW Paket in die History Tabelle geschrieben.

Parameter	Beschreibung
@TableName	Tabellenname der CDW History Tabelle, z.B. „DimKunde_History"
@TableNameSource	Tabellenname der CDW History Update Tabelle, z.B. „DimKunde_History_Update"
@BusinessKeys	Liste von Business Keys, getrennt durch Komma – z.B. „Kundennummer"
@IncludeDeletePart	Flag für das Löschen von alten Datensätzen, die nicht in den Staging Daten (History Update Tabelle) vorhanden sind, aber in CDW „0" [Default] – Datensatz beibehalten, wenn der Datensatz nicht in den Staging Daten vorkommt
@SCD2Columns	Liste von zu historisierenden Feldern – z.B. „Land, Gruppe"
@DestSchema ***[optional]***	Schemaname der CDW History Tabelle „CDW" [Default]
@SourceSchema ***[optional]***	Schemaname der CDW History Update Tabelle „ETL" [Default]
@IsCurrentName ***[optional]***	Feldname von IsCurrent Flag „IS_CURRENT" [Default]
@SpecialColumns ***[optional]***	Felder zum Ausschließen „IS_CURRENT, VALID_FROM, VALID_TO, INSERT_AUDIT_ID, UPDATE_AUDIT_ID, INSERT_AUDIT_DATE" [Default]

Tabelle 6: GenerateMergeCdwHistoryOrigValidDates Parameter

2.7 Beispiele für die Verwendung der Hilfsprozeduren

Generate CDW Dimension

```
EXEC [ETL].[GenerateCdwDim] 'ETL.vCDW_Kunde', 'Kunde', 'Kundennummer', null, 1,
'OneTemplateCoreDW', 'OneTemplateStaging'

EXEC [ETL].[GenerateCdwDim] 'ETL.vCDW_Produkt', 'Produkt', 'Produktnummer', 'Produktnummer,
ProduktSubkategorieID, ProduktKategorieSID', 1, 'OneTemplateCoreDW', 'OneTemplateStaging'
```

Generate CDW Tables

```
EXEC [ETL].[GenerateCdwTables] 'ETL.vCDW_Kunde', 'Kunde', 'Kundennummer', null, 1,
'OneTemplateCoreDW', 'OneTemplateStaging'

EXEC [ETL].[GenerateCdwTables] 'ETL.vCDW_Produkt', 'Produkt', 'Produktnummer', 'Produktnummer,
ProduktSubkategorieID, ProduktKategorieSID', 1, 'OneTemplateCoreDW', 'OneTemplateStaging'
```

Generate Dummy Procedure

```
EXEC [ETL].[GenerateDummyProc] @procSchema = 'ETL', @dstSchema = 'CDW', @tableName =
'DimKunde'

EXEC [ETL].[GenerateDummyProc] @procSchema = 'ETL', @dstSchema = 'CDW', @tableName =
'DimProdukt'
```

Generate Merge Main

```
EXEC [ETL].[GenerateMergeMain] @tableName = 'DimKunde', @tableNameSource = 'DimKunde_Update',
@businessKeys = 'Kundennummer', @dstSchema = 'CDW',@srcSchema = 'ETL'


EXEC [ETL].[GenerateMergeMain] @tableName = 'DimProdukt', @tableNameSource =
'DimKunde_Update', @businessKeys = 'Produktnummer'
```

Generate Merge CDW History

```
EXEC [ETL].[GenerateMergeCdwHistory] @TableName = 'DimKunde_History', @TableNameSource =
'DimKunde_History_Update', @BusinessKeys = 'Kundennumer', @IncludeDeletePart = 0, @SCD2Columns
= 'Land, Gruppe'


EXEC [ETL].[GenerateMergeCdwHistory] @TableName = 'DimProdukt_History', @TableNameSource =
'DimProdukt_History_Update', @BusinessKeys = 'Produktnummer', @IncludeDeletePart = 0,
@SCD2Columns = 'Produktname'
```

Generate Merge CDW History Orig Valid Dates

```
EXEC [ETL].[GenerateMergeCdwHistoryOrigValidDates] @TableName = 'DimKunde_History',
@TableNameSource = 'DimKunde_History_Update', @BusinessKeys = 'Kundennumer',
@IncludeDeletePart = 0, @SCD2Columns = 'Land, Gruppe'


EXEC [ETL].[GenerateMergeCdwHistoryOrigValidDates] @TableName = 'DimProdukt_History',
@TableNameSource = 'DimProdukt_History_Update', @BusinessKeys = 'Produktnummer',
@IncludeDeletePart = 0, @SCD2Columns = 'Produktname'
```

3 SSIS Template

Für den Ladeprozess von Staging in das CDW steht eine Reihe von Templates zur Verfügung, die für verschiedene Anwendungsfälle wie z.B. CDW Dimensionsbeladung oder Archivierung eingesetzt werden können. Das Konzept des Templates erleichtert die Entwicklung neuerer Pakete und spart dadurch viel Zeit. Damit der Anpassungsaufwand möglichst gering ist, wurde im Paket vieles automatisiert. Natürlich können die Pakete auch individuell nach Bedarf geändert werden, es ist als Ausgangsbasis gedacht, die in vielen Fällen funktionieren. Der Standardpakete sind nach dem pmOne Best Practice aufgesetzt.

Die SSIS-Pakete richten sich nach dem Project-Deployment-Modell, damit die Nutzung des SSIS Catalogs möglich ist.

Tabelle 7: Template Packages

Paketname	Anwendungsfall
_TEMPLATE_STA_SQL_Table.dtsx	Staging Beladung von einer Microsoft SQL Tabelle
_TEMPLATE_STA_CSV_File.dtsx	Staging Beladung von einer CSV-Datei
_TEMPLATE_CDW_Dimension.dtsx	CDW Beladung für Dimension
_TEMPLATE_CDW_FactFullLoad.dtsx	CDW Beladung für Fakten in Full Load
_TEMPLATE_ARC_DimArchive.dtsx	Archivierung für Dimension
_TEMPLATE_ARC_FactArchive.dtsx	Archivierung für Fakten
_TEMPLATE_ARC2STA_DimArchiveReplay.dtsx	Replay Paket für Dimension
_TEMPLATE_ARC2STA_FactArchiveReplay.dtsx	Replay Paket für Fakten

Anpassung:
Zur übersichtlichen Darstellung der Paketen Liste in Visual Studio (VS) empfehlen wir nach folgender Namenskonvention zu benennen: **[Schema] + „_“ + [Zieltabellenname] + „.dtsx“**

Anmerkung: Der Zieltabellenname für STA enthält noch ein davorstehendes Kürzel vom Vorsystem, also gilt:
[Schema] + „_“ + [Kürzel vom Vorsystem] + „_“ + [Zieltabellenname] + „.dtsx“

Beispiele:
[STA].[AW_KUNDE] => STA_AW_KUNDE.dtsx
[STA].[CSV_Kundenzusatz] => STA_CSV_Kundenzusatz.dtsx
[CDW].[DimKunde] => CDW_DimKunde.dtsx
[CDW].[FactSalesOrders] => CDW_FactSalesOrders.dtsx
[ARC].[AW_KUNDE] => ARC_AW_KUNDE.dtsx

3.1 Project Connections

Connection Manager	Beschreibung
SourceDB	Quelldatenbank für die Befüllung der Staging Area In den meisten Fällen ist es die Datenbank vom operativen System, z.B. ERP, CRM, usw.

	Hinweis: Wenn ein anderes DBMS, wie z.B. Oracle, in Anwendung kommt, dann müssen Sie evtl. einen zusätzlichen Treiber installieren.
StagingDB	Staging Datenbank
CoreDwDB	Core Datawarehouse Datenbank
ArchivDB *[optional]*	Archiv Datenbank
LoggingDB	OneLog Datenbank fürs Logging

Tabelle 8: SSIS Project Connections

3.2 STA Template - Microsoft SQL Tabelle

Template Datei: **_TEMPLATE_STA_SQL_Table.dtsx**
Namenskonvention: **„STA_“ + Kürzel vom Vorsystem + „_“ + Tabellenname + „.dtsx“**

Das folgende Template dient für die Beladung von einer MS SQL Tabelle in die Staging Tabelle.

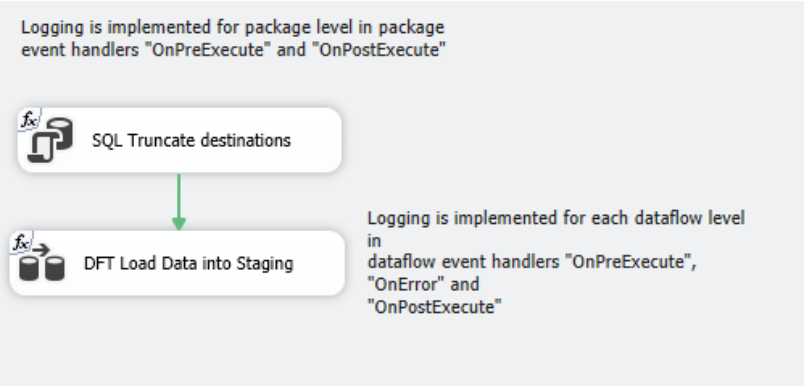


Abbildung 2: STA Template MS SQL Table - Control Flow

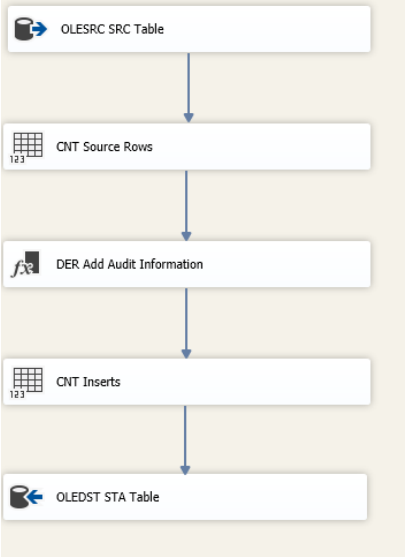


Abbildung 3: STA Template MS SQL Table - Data Flow

Die Beladung erfolgt in Full Load, d.h. die Zieltabelle bzw. Staging Tabelle wird im ersten Schritt geleert und dann von der Quelltabelle 1:1 befüllt.

Hinweis:
Insofern Sie die Staging Tabelle noch nicht angelegt haben, können Sie es aus VS einen CREATE Skript generieren lassen, indem Sie die Zielkomponente (OLEDST STA Table) öffnen, Data access mode auf „Table or view“ umstellen, auf „New“ klicken und dann erscheint ein Fenster mit einem CREATE Skript für die neue Staging Tabelle.

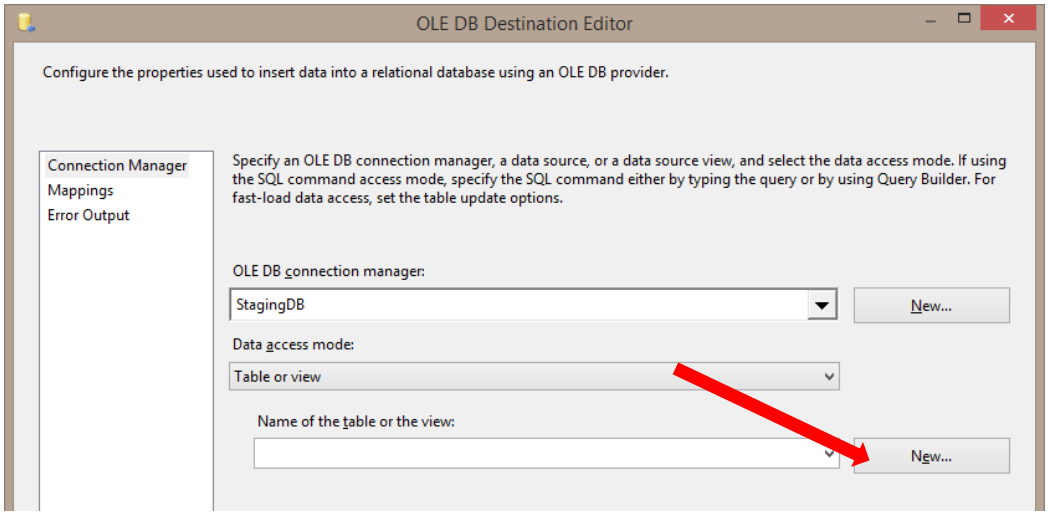


Abbildung 4: STA Template MS SQL Table - Generierung CREATE Skript

3.2.1 Variablen

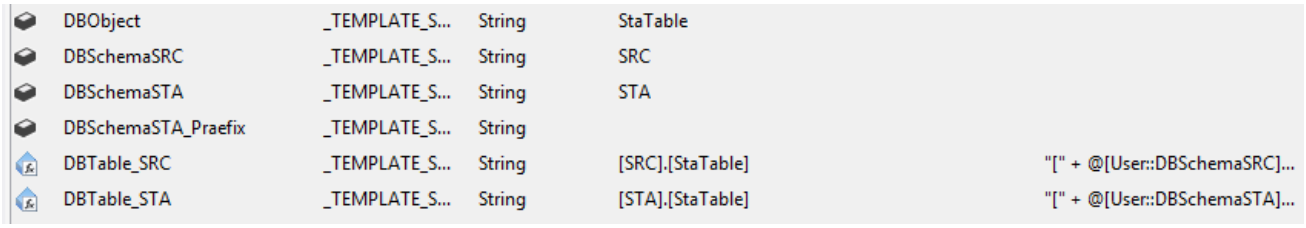


Abbildung 5: STA Template MS SQL Table - Variablen

Variable	Beschreibung
DBObject	Tabellenname von der Quelle
DBSchemaSRC	Schema von der Quelltabelle
DBSchemaSTA	Schema von der Staging Tabelle
DBSchemaSTA_Praefix	Abkürzung für die Quelle (Systemcode) in Großbuchstaben Präfix für den Staging Tabellenname und durch „_“ (Unterstrich) getrennt
DBTable_SRC (Expr.)	Vollständiger Name der Quelltabelle
DBTable_STA (Expr.)	Vollständiger Name der Staging Tabelle

Tabelle 9: STA Template MS SQL Table - Variablen

Anpassung:

In den meisten Fällen müssen Sie nur **DBObject, DBSchemaSRC** und **DBSchemaSTA_Praefix** setzen.

3.3 STA Template - CSV Datei

Template Datei: **_TEMPLATE_STA_CSV_File.dtsx**
Namenskonvention: **„STA_CSV_“ + Tabellenname + „.dtsx“**

Zur Beladung von einer oder mehreren CSV Dateien können Sie das folgende Template nehmen:

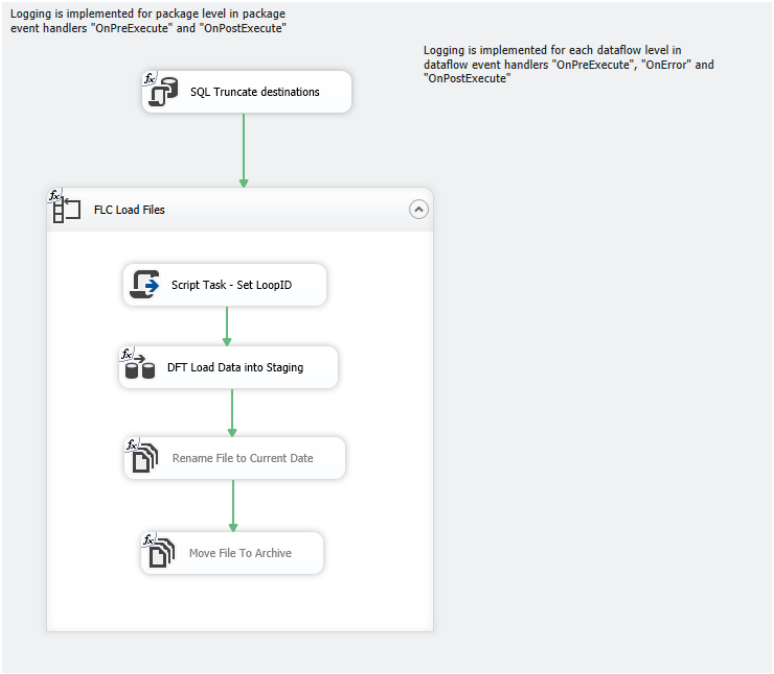


Abbildung 6: STA Template CSV - Control Flow

Wie beim STA Template erfolgt der Datentransfer in Full Load. Des Weiteren haben Sie die Möglichkeit mehrere Dateien hintereinander zu beladen oder optional an einem bestimmten Ordner zu archivieren, die im Foreach Container (FLC Load Files) iteriert werden. Über die „FileMask“ Einstellung können Sie mithilfe von Wildcards einstellen, welche Dateien verarbeitet werden sollen. Dabei können Sie es auf täglich oder monatlich bereitgestellte Dateien anwenden, die durch den Dateiname erkennbar sind. Beispiel dafür sind Währungskurse oder Budgetdaten.

Hinweis:
Sie können ein CREATE Skript auf dieselbe Methode wie im vorigen Kapitel (siehe 0 Tabelle 8: SSIS Project Connections

STA Template - Microsoft SQL Tabelle) aus der Zielkomponente in VS automatisch erstellen lassen. Jedoch ist aufgrund der Dynamisierung des CSV Connection Managers ein einmaliger Aufruf der Metadaten von der zu beladenden Datei erforderlich, weil die Metadaten anfangs unbekannt sind bzw. die eingehenden Input Felder in die Zieltabelle zum Zeitpunkt der Entwicklung für das Mapping nicht aktuell sind. Dabei gehen Sie auf die Verbindungseinstellung „CSV_FileConnection“, wählen die zu beladende Datei aus und wechseln zum Reiter „Columns“ und klicken auf „Reset Columns“. Als Nächstes aktualisieren Sie die Metadaten bei der Quellkomponente im Data Flow, indem Sie die Komponente öffnen und zum Reiter „Columns“ wechseln, damit die aktuellen Metadaten vom Connection Manager aufgerufen werden. Dann sollten Sie in der Lage sein, einen CREATE Skript in der Zielkomponente zu generieren.

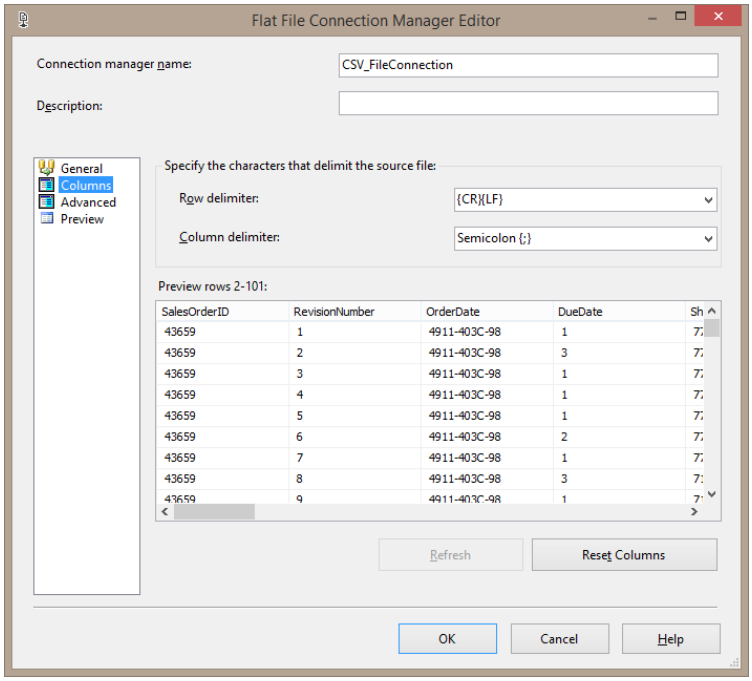


Abbildung 7: STA Template CSV - Metadaten Aktualisierung in CSV FileConnectionManager

3.3.1 Variablen

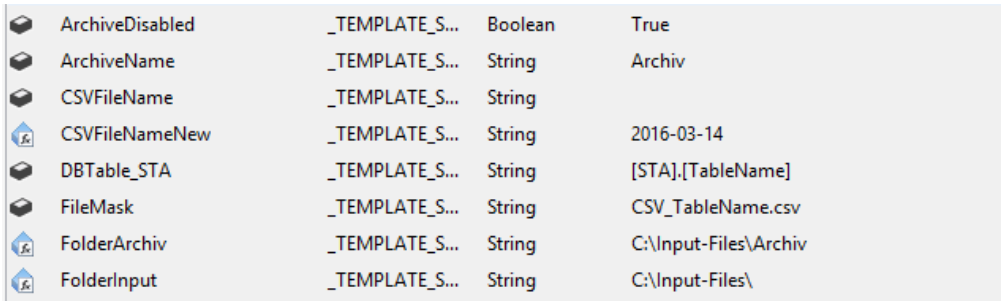


Abbildung 8: STA Template CSV - Variablen

Variable	Beschreibung
ArchiveDisabled	Flag für Archivierung
ArchiveName	Verzeichnisname des Archivs
CSVFileName	Wird automatisch in Foreach Loop neu gesetzt Enthält den Dateiname der aktuell verarbeiteten Datei
CSVFileNameNew (Expr.)	Alternativ zu CSVFileName in File Connection Manager mit aktuellem Datum in Dateiname: „YYYY-MM-DD Dateiname.csv“
DBTable_STA	Vollständiger Name der Staging Tabelle
FileMask	Dateiname Wildcard wird unterstützt (“*”), wenn mehrere Dateien auf einmal geladen werden sollen
FolderArchiv (Expr.)	Pfad für Archiv Verzeichnis
FolderInput (Expr.)	Verzeichnis für die Ablage der zu importierenden Dateien

Tabelle 10: STA Template CSV - Variablen

✋ Anpassung:

In den meisten Fällen müssen Sie nur **DBTable_STA** und **FileMask** setzen. Der Ablageort für die CSV Dateien können Sie beim **Projektparameter** „**CSVInputPathGeneral**“ ändern (Standardwert: „C:\Input-Files\“). Beachten Sie, dass Sie beim Deployen den neuen Wert auch in den Umgebungsvariablen einstellen müssen.

3.4 CDW Template - Dimension

Template Datei: **_TEMPLATE_CDW_Dimension.dtsx**
Namenskonvention: **„CDW_“ + Dimensionsname (mit „Dim“ als Präfix) + „.dtsx“**

Das Paket verarbeitet den normalen Ladeprozess ins CDW als auch SCD2 Historisierungen, die aber explizit eingeschaltet werden müssen.

Grober Ablauf:

- Prüfen des Referenzdatums bei „Historisierung“
- Einfügen eventuell fehlender Dummy-Werte
- Laden der BusinessKeys und Inferred-Member Handling
- Parallel: Keys Update für Inferred | Keys in den Cache übertragen
- Parallel: Historisierung (falls aktiviert) | Hauptladezweig:
  Key Lookup
  Über Hilfstabellen INSERT oder Update-Zwischenspeicher
  Batchupdate aller ermittelten Updates
  Bereinigung Hilfstabellen

☝ Hinweis:

Die Paketvariablen ersetzt nicht den Paketname, der im OneLog erscheinen wird. Dieser muss über die Package Properties (Rechter Mausklick auf den Hintergrund im Control Flow → Properties) im Feld „Name“ geändert werden, insofern VS beim Kopieren des Templates nicht richtig gesetzt hat.

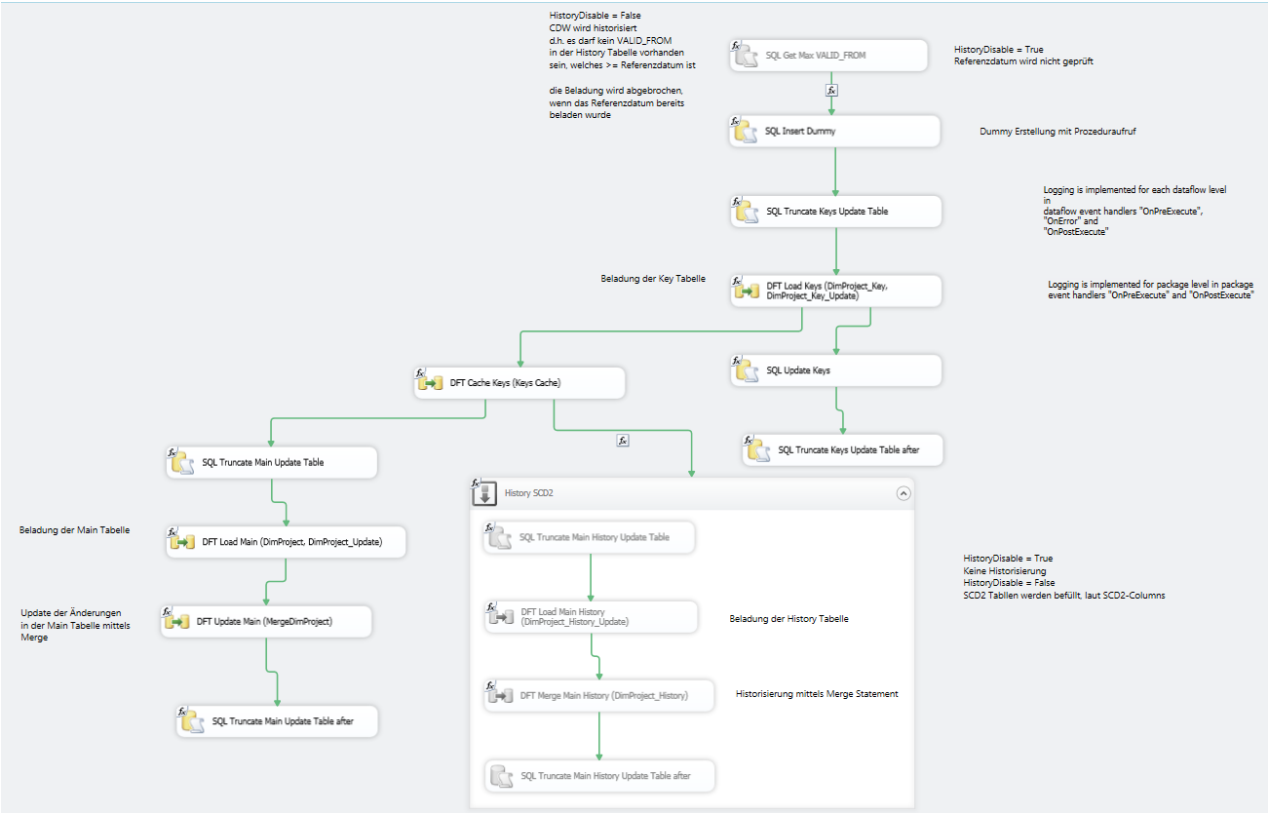


Abbildung 9: CDW Template Dimension - Control Flow

3.4.1 Variablen

DBBusinessKeyColumns	_TEMPLATE_C...	String	BKColumn1, BKColumn2
DBObject	_TEMPLATE_C...	String	DimTable
DBSCD2Columns	_TEMPLATE_C...	String	[Column1],[Column2],[Column3]
DBSchemaCDW	_TEMPLATE_C...	String	CDW
DBSchemaETL	_TEMPLATE_C...	String	ETL
HistoryDisabled	_TEMPLATE_C...	Boolean	True

Abbildung 10: CDW Template Dimension - Variablen

Aus den obigen Variablen leiten sich die anderen Variablen für CDW Tabellen, Prozeduren und ETL View ab, die nach einer einheitlichen Namenskonvention folgen. (siehe 1.6 Core Data Warehouse (CDW))

DBTable	_TEMPLATE_C...	String	[CDW].[DimTable]
DBTable_History	_TEMPLATE_C...	String	[CDW].[DimTable_History]
DBTable_History_Update	_TEMPLATE_C...	String	[ETL].[DimTable_History_Update]
DBTable_Key	_TEMPLATE_C...	String	[CDW].[DimTable_Key]
DBTable_Key_Update	_TEMPLATE_C...	String	[ETL].[DimTable_Key_Update]
DBTable_Update	_TEMPLATE_C...	String	[ETL].[DimTable_Update]
DBView	_TEMPLATE_C...	String	[ETL].[vCDW_DimTable]

Abbildung 11: CDW Template Dimension - Variablen mit Expression für CDW Tabellen und ETL View

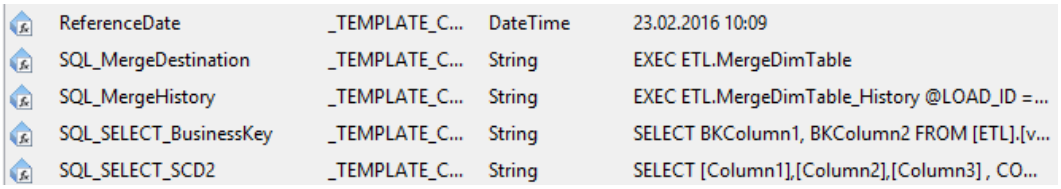


Abbildung 12: CDW Template Dimension - Variablen mit Expression für ReferenceDate, SELECT Abfragen und Prozeduraufrufe

Variable	Beschreibung
DBBusinessKeyColumns	Liste von Business Keys, getrennt durch Komma, z.B. “Kundennummer”
DBObject	Dimensionsname mit „Dim“ als Präfix, z.B. „DimKunde“
DBSCD2Columns	Liste von zu historisierenden Feldern, getrennt durch Komma – z.B. „Land, Gruppe“ wird ignoriert wenn die Historisierung nicht eingeschaltet ist (HistoryDisabled => False)
DBSchemaCDW	CDW Schema
DBSchemaETL	ETL Schema
HistoryDisabled	Flag für Historisierung bzw. Historisierungszweig im Paket Default: True für das Deaktivieren der Historisierung
DBTable (Expr.)	CDW Haupttabelle
DBTable_History (Expr.)	CDW History Tabelle Historisierte Felder mit Gültigkeitszeitraum
DBTable_History_Update (Expr.)	CDW History Update Tabelle Dient als temporärer Speicher für historisierte Staging Daten
DBTable_Key (Expr.)	CDW Key Tabelle Zur Generierung von SIDs
DBTable_Key_Update (Expr.)	CDW Key Update Tabelle Zeichnet die Updates bzw. Änderungen auf
DBTable_Update (Expr.)	CDW Update Tabelle Dient als temporärer Speicher für Staging Daten
DBView (Expr.)	ETL View (von Staging DB)
ReferenceDate (Expr.)	Referenzdatum wenn das Referenzdatum in den Parametern nicht gesetzt wird, dann wird automatisch das Ausführungsdatum bzw. das aktuelle Datum genommen
SQL_MergeDestination (Expr.)	Merge Prozedur Aktualisiert die Attribute in der CDW Haupttabelle durch die neuen Werte von der CDW Update Tabelle
SQL_MergeHistory (Expr.)	History Merge Prozedur Aktualisiert die Attribute und Gültigkeitszeiträume in der CDW History Tabelle durch die neuen Werte von der CDW History Update Tabelle
SQL_SELECT_BusinessKey (Expr.)	SELECT Statement auf die Business Keys von ETL View
SQL_SELECT_SCD2 (Expr.)	SELECT Statement auf die historisierte Felder von ETL View

Nähere Informationen über die einzelnen CDW Tabellen finden Sie unter dem Kapitel 1.6 Core Data Warehouse (CDW).

Hinweis:
Vor der Bearbeitung am Paket müssen alle entsprechenden Datenbankobjekte vorhanden sein, die Sie über die CDW Generierungsprozedur automatisch erstellt haben sollten.

Anpassung:
In den meisten Fällen müssen Sie nur **DBObject** und **DBBusinessKeyColumns** einstellen. Wenn Historisierung erwünscht ist, dann müssen Sie zusätzlich **HistoryDisabled auf True** und **DBSCD2Columns** setzen.

3.4.2 Key Cache

Die interne Connection für die Keys bzw. „Keys Cache“ sollte mit dem richtigen Business Key und der dazugehörige Datentyp angepasst werden.

3.4.3 Mapping

Auch wenn viele Objektnamen und SQL automatisch gesetzt werden und auch Beschriftungen, so müssen dennoch die Mappings an allen Dataflow Tasks manuell angepasst werden:

- DFT Load Keys
- DFT Cache Keys
- DFT Load Main
- DFT Update Main
- Bei Historisierung
    - DFT Load Main History + DFT Merge Main History

Dabei öffnen Sie die Quellkomponente um die Metadaten zu aktualisieren und dann die Zielkomponente um die Zielspalten zu verknüpfen, die Sie durch die Funktion „Matching Items by Matching Names“ automatisch verbinden können.

3.5 CDW Template Fakten

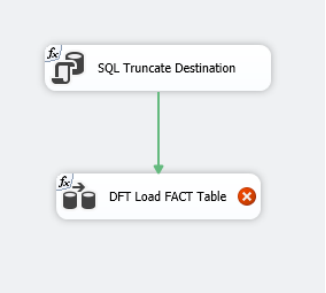
Wie die Fakten ins CDW befüllt werden, existieren zwei Beladungsarten:

- Full Load: Flush and fill, komplete Neubeladung aller Daten
- Delta Load: Neue Daten werden hinzugefügt, bestehende werden nicht gelöscht

Das ist je nach Projekt und Faktentabelle unterschiedlich. Zusätzlich kann ein Delta Load unterschiedliche Beladungslogik haben, wie z.B. stichtagsbezogen oder nur veränderte Datensätze anhand einer bestimmten Flag Spalte. Ein Full Load dagegen ist im Regelfall immer gleich.

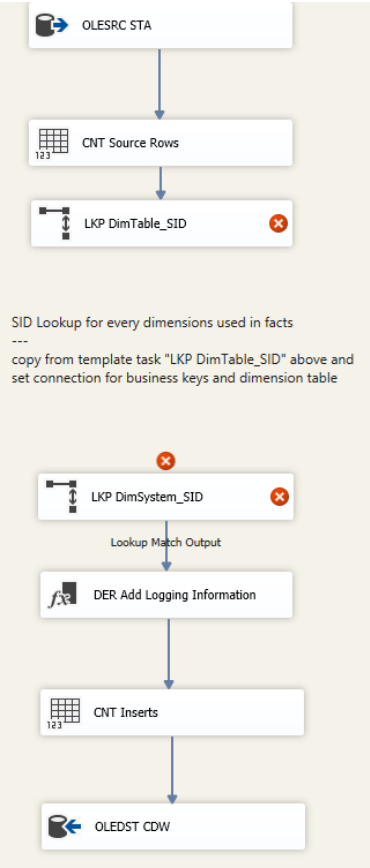
3.5.1 Full Load

Template Datei: **_TEMPLATE_CDW_Fact.dtsx**
Namenskonvention: **„CDW_“ + Faktenname (mit „Fact“ als Präfix) + „.dtsx“**



Im Ladeprozess werden die SIDs von allen angehängten Dimensionen mithilfe von Lookups ermittelt und mit den bestehenden Feldern von der Quelle in die CDW Faktentabelle geladen.

Abbildung 13: CDW Template Fakten FullLoad - Control Flow



Grundsätzlich sollte eine ETL Faktenview nach folgender Reihenfolge aufgebaut sein (siehe 1.5.1 Aufbau Staging View (ETL)):

- Datum
  - Wenn mehrere Datumsfelder existieren und für das DWH relevant sein sollten, dann wird Roleplaying angewendet.
- Business Key oder Business Keys einer Dimension
- Measures
- SystemCode (Kürzel vom Vorsystem)

In der Faktenbeladung werden die Dimension SIDs, die an den Fakten hängen, mittels Lookup auf dem entprechende Business Key der Dimension ermittelt.

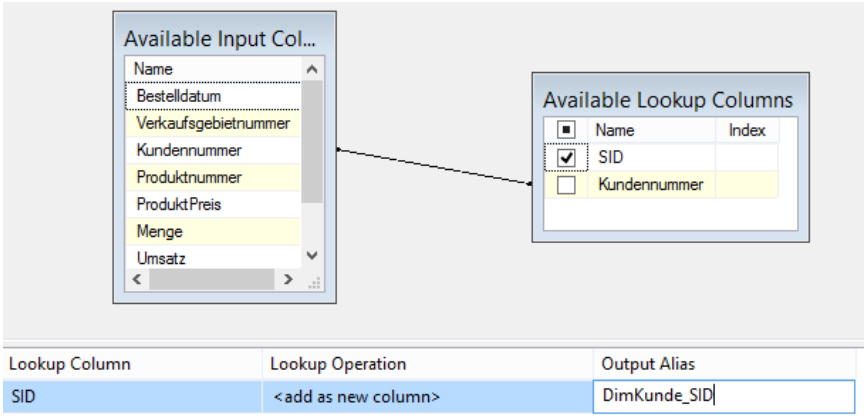
Anpassung:

Für jede Dimension, die später auf die Fakten verknüpft werden soll, ist ein Lookup zu erstellen. Am besten kopieren Sie sich die Beispielkomponente „LKP DimTable_SID“ und benennen es nach der neuen Dimension um, wie z.B. für DimKunde „LKP DimKunde_SID“.

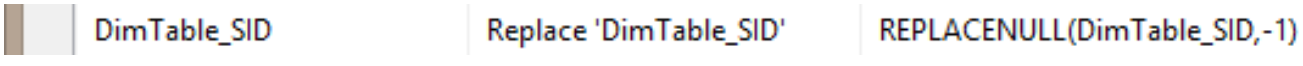
Als Nächstes wechseln Sie zum Reiter „Connection“ und fügen Ihre Abfrage auf die Key Tabelle der neuen Dimension hinzu. Das wäre zum Beispiel für DimKunde folgende Abfrage:
SELECT [SID], [Kundennummer] FROM CDW.DimKunde_Key with(nolock)

Abbildung 14: CDW Template Fakten FullLoad - DataFlow

Im Reiter „Columns“ verknüpfen Sie den Business Key in den Fakten mit dem Business Key von der Abfrage bzw. Key Dimensionstabelle. Den Output Alias ändern Sie nach folgender Namenskonvention:
[Dimensionsname] + „_“ + „SID“
z.B. DimKunde_SID, DimProdukt_SID, usw.



Nach dem Lookup öffnen Sie die Komponente „DER Add Logging Information“, die dazu dient, dass bei nicht gefundener Dimension SID durch „-1“ ersetzt, damit es auf den Dummy Datensatz bzw. „nicht zugeordnet“ zuweist. Das müssen Sie für jede Dimension SID machen.



Hinweis:
Sobald Sie alle Lookups und die Ersetzung der Dimension SIDs angepasst haben, können Sie den CREATE Skript aus der Zielkomponente generieren lassen.

3.5.1.1 Variablen

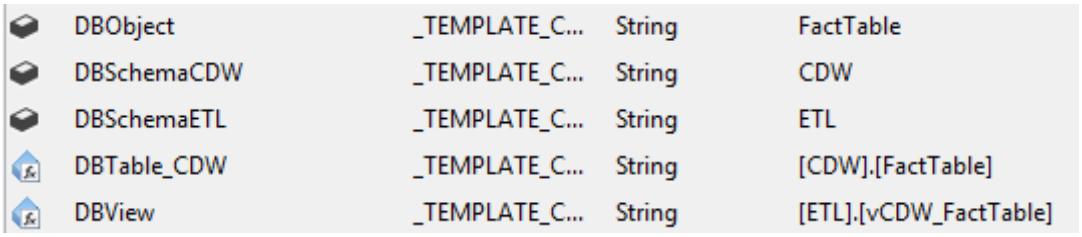


Abbildung 15: CDW Template Fakten FullLoad - Variablen

Variable	Beschreibung
DBObject	Faktenname mit „Fact“ als Präfix, z.B. „FactVerkaufsauftrag“
DBSchemaCDW	CDW Schema
DBSchemaETL	ETL Schema
DBTable_CDW (Expr.)	CDW Haupttabelle
DBView (Expr.)	ETL View (von Staging DB)

Tabelle 11: CDW Template Fakten FullLoad - Variablen

3.6 MASTER Pakete

Ein Masterpaket wird für die Ausführung mehrerer SSIS Pakete verwendet und zu einem SQL Agent Job angehängt. Damit können Sie beispielsweise die Staging Area über den Job neubeladen und über den Status im OneLog mitverfolgen. (siehe Kapitel 6 OneLog – Logging Framework)

Paketname	Beschreibung
_MASTER_STA.dtsx	Staging Beladung
_MASTER_CDW_Dimensions.dtsx	CDW Dimensionsbeladung
_MASTER_CDW_Facts.dtsx	CDW Faktenbeladung
_MASTER_ARC.dtsx	Archivbeladung
_MASTER_ARC2STA.dtsx	Replaybeladung (Archiv => Staging) Keine Anpassung notwendig, weil es über Expressions und Metadaten von der STA DB gesteuert wird
_MASTERJOB.dtsx	Tägliche Beladung Enthält standardmäßig die Staging Beladung und CDW Beladung für Dimension und Fakten Optional kann die Archivbeladung eingeschaltet werden (initial deaktiviert)
_CUBE_PROCESS.dtsx	Cube Verarbeitung (Dimensionen + Fakten) Muss auf den aktuellen Cube angepasst werden

Tabelle 12: MASTER Standardpakete

Anpassung:

Für jedes neue Paket muss es im jeweiligen Masterpaket hinzugefügt werden. Im jeden Master Template (ausgenommen MASTER ARC2STA) gibt es eine Beispielkomponente mit Standardparameter, die Sie kopieren und anpassen können. Dabei wählen Sie aus der Liste das neue Paket aus und benennen es nach „EPT „ + Paketname (z.b. „EPT CDW_DimKunde“) um.

Hinweis:

Wenn Sie eine komplett neue Art von Masterpaket erstellen wollen, das Sie über einen Job ausführen möchten, dann fügen Sie einen neuen Datensatz für die neue Kategorie in die OneLog Tabelle „oneLog.Job“ hinzu und passen den Parameter „Job_SID“ im Paket auf die neue SID an. Somit ist die neue Kategorie im OneLog Bericht ersichtlich.

SID	Description
-1	Test
1	MasterJob
2	Master Staging
3	Master Archiv
4	Master CoreDW Stammdaten
5	Master CoreDW Faktendaten
6	Master ARC2STA

Abbildung 16: OneLog Job Standardkategorie für Masterpakete

4 Deployment

In vielen Fällen werden Test- und Produktivserver bereitgestellt. Neue Änderungen werden hauptsächlich auf dem Testserver implementiert, die im späteren Verlauf auf dem Produktivserver übertragen werden. In den folgenden Kapiteln wird der Standardfall für das Deployment vom Test- auf dem Produktivserver aufgezeigt.

4.1 Datenbank Projekt

Zur Vereinfachung für die Übertragung von Datenbankstrukturen von einer Datenbank auf die Andere hilft Ihnen das Datenbank Projekt als Zwischenglied. Das Projekt sollte in der Regel den aktuellen Stand bzw. den Stand der Produktivdatenbank haben. Dadurch haben Sie die Möglichkeit mithilfe von Schema Compare die Änderungen nachvollzuziehen.

Auf der linken Seite der Auswahlliste wählen Sie die Quelle und auf der Rechten das Ziel. In folgender Reihenfolge ist der Abgleich vom Test auf Produktiv durchzuführen:

- **Test => Datenbank Projekt**
- **Datenbank Projekt => Produktiv**

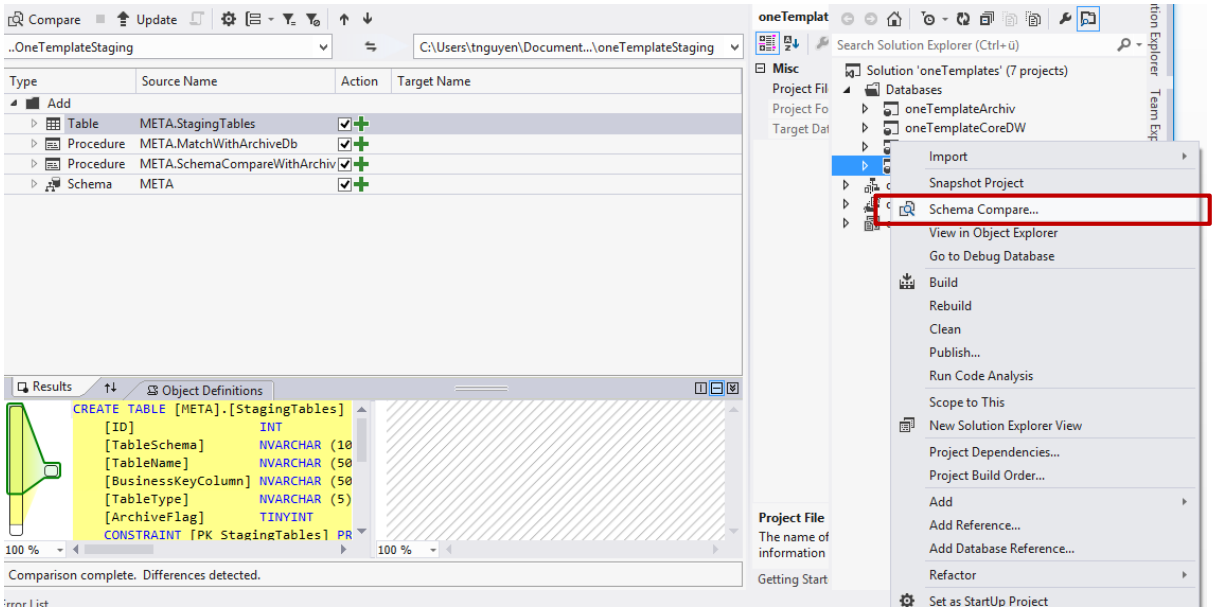


Abbildung 17: Datenbank Projekt - Schema Compare

Hinweis:

Wenn Sie auf der Produktivdatenbank Änderungen an einer Tabelle mit Daten durchführen wollen, dann müssen Sie es manuell anpassen, andernfalls könnten dadurch Daten verloren gehen. Mit der Schema Compare Standardeinstellung („Block on possible data loss“) sollte VS beim Update eine Fehlermeldung ausgeben.

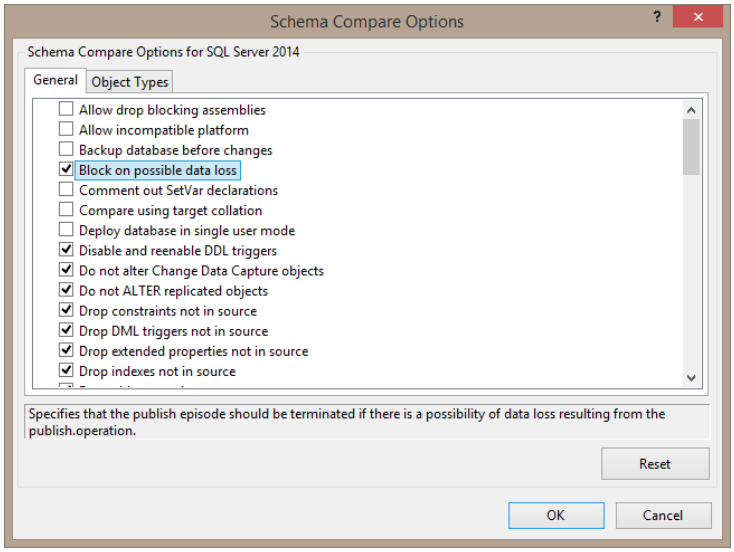


Abbildung 18: Datenbank Projekt - Schema Compare Options für Datenverlust bei bestehenden Daten

4.2 ETL Projekt

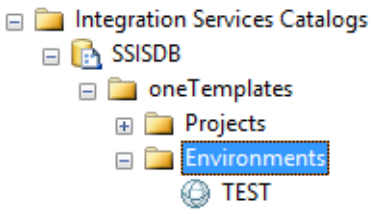
Integration Services unterstützt zwei Deployment Modelle:

- Project Deployment Model (ab SQL Server 2012 verfügbar)
- und Package Deployment Model

Das SSDL Framework setzt auf das Project Deployment Model auf. Beim Project Deployment wurde der Deployment Prozess gegenüber dem Package Deployment Model deutlich vereinfacht, weil Konfigurationen

wie Parameter und Verbindung zentral am Projekt hängen und nicht dezentralisiert am Paket. Weiteres bietet Ihnen das vereinfachte Bereitstellungsmodell die Pakete auf einen SSIS Catalog (Paketverwaltung) zu deployen, auf den Sie Serverkonfigurationen mithilfe von Umgebungsvariablen für Parameter und Verbindungen vornehmen können. Zum Beispiel wenn Sie Ihre Pakete auf den Testserver bereitstellen, dann werden die Einstellungen für die Testumgebung bei der Ausführung wie beispielsweise Connection Managers übernommen, insofern Sie die Umgebungsvariablen konfiguriert haben. D.h. Sie haben die Möglichkeit Ihre Konfiguration direkt auf den Test- und Produktivserver getrennt abzulegen, ohne dabei in VS für die jeweilige Umgebung beim Deployen einzustellen.

4.2.1 Umgebungsvariablen



Mithilfe von Umgebungsvariablen können Sie Projekt-, Paketparameter und Verbindungseinstellungen durch Konfiguration vom Server ersetzen. Folgende Umgebungen (Environments) legen wir standardmäßig fest:

- „TEST“ oder „DEV“ auf den Testserver bzw. Entwicklungsserver
- „PROD“ auf den Produktivserver

Abbildung 19: Deployment - Umgebungsvariablen

Folgende Umgebungsvariablen werden standardmäßig gesetzt:

Variablenname	Typ	Wert
BLOBTempStoragePath	Projektparameter	*(standardmäßig leer)*
BufferTempStoragePath	Projektparameter	*(standardmäßig leer)*
ConString_Archiv *(optional, wenn nicht gefordert => dann leer belassen)*	Projektverbindung	*(projektabhängig)*
ConString_CDW	Projektverbindung	*(projektabhängig)*
ConString_Cube	Projektverbindung	*(projektabhängig)*
ConString_Logging	Projektverbindung	*(projektabhängig)*
ConString_Source	Projektverbindung	*(projektabhängig)*
ConString_Staging	Projektverbindung	*(projektabhängig)*
CSVInputPathGeneral *(Standardwert)*	Projektparameter	D:\Input-Files\
DefaultBufferMaxRows *(Standardwert)*	Projektparameter	100000
DefaultBufferSize *(Standardwert)*	Projektparameter	104857600
EngineThreads *(Standardwert)*	Projektparameter	10
ReferenceDate	Paketparameter	31.12.9999 00:00:00

Tabelle 13: Deployment - Standardkonfiguration für Umgebungsvariablen

Anpassung:

Die Umgebungsvariablen, die in der Tabelle 13: Deployment - Standardkonfiguration für Umgebungsvariablen aufgelistet sind, müssen an den entsprechenden Parameter im Projekt referenziert werden. Alle Projektparameter und –Verbindungen können auf Projektebene eingestellt werden und Paketparameter müssen für alle Masterpakete bzw. auf Paketebene gesetzt werden.

4.2.1.1 *Projektreferenz*

Damit das bereitgestellte Projekt auf die Umgebungsvariablen zugreifen kann, müssen Sie die Umgebung freigeben, indem Sie ***„ETL Project (in SSIS Catalog) => Configure (context menu) => References (tab) => Add (button)“*** gehen und die konfigurierte Umgebung zum Projekt hinzufügen.

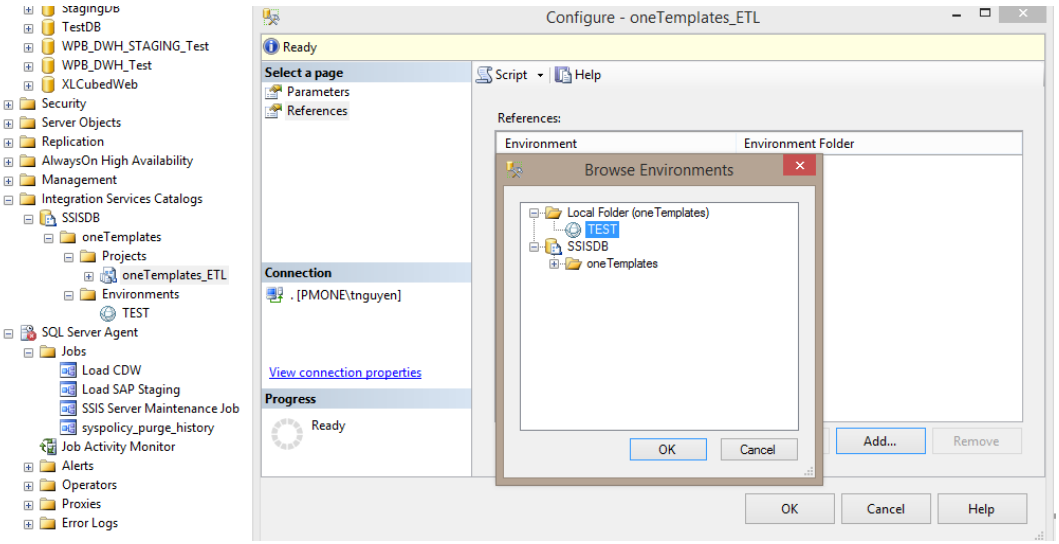


Abbildung 20: Deployment - Projektreferenz einer Umgebung

4.2.1.2 *Parameterbindung*

Danach können Sie für jeden Parameter und jede Verbindungseinstellung zu einer Umgebungsvariable referenzieren. In der folgender Abbildung sehen Sie beispielhaft aufgezeigt, wie Sie den Connection Manager für CDW an die Umgebungsvariable „ConString_CDW“ binden können.

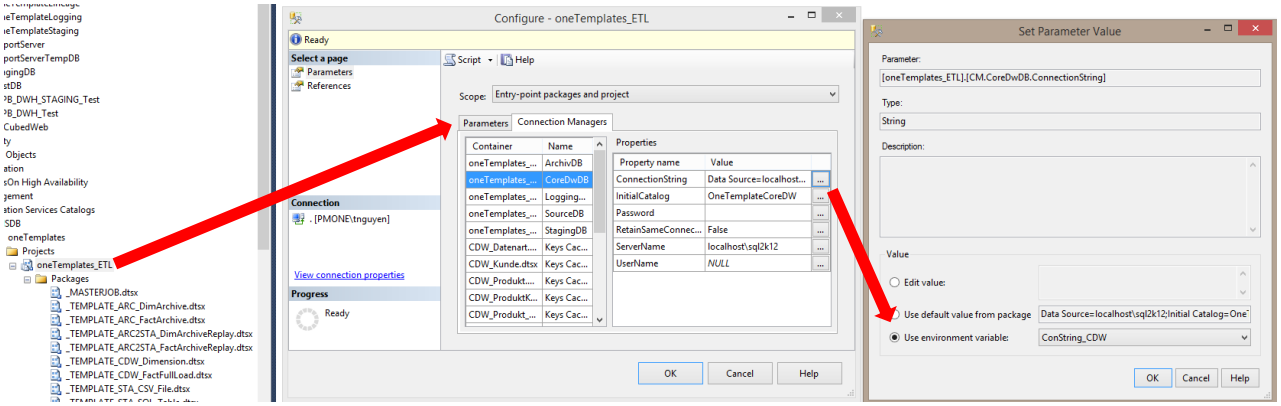


Abbildung 21: Deployment - Parameterbinding auf eine Umgebungsvariable

4.2.2 Deployment

Unter ***„Project => Properties => Configuration Properties => Deployment => Server Name“*** legen Sie fest, auf welchen Server Sie Ihr Projekt bereitstellen wollen.

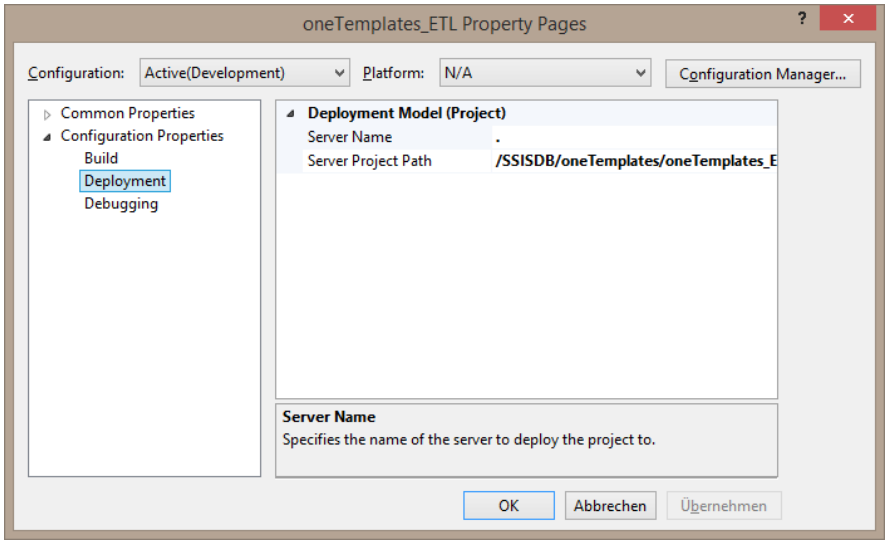


Abbildung 22: Deployment – ETL Deploymenteinstellung

4.3 Cube Projekt

Unter ***„Project => Properties => Configuration Properties => Deployment => Target => Server***“ legen Sie fest, auf welchen Server Sie Ihr Cube bereitstellen wollen und bei ***„Database“*** stellen Sie den Würfelname ein. Sie können auch mehrere Cubes auf demselben Server bereitstellen um z.B. den Produktivcube durch einen anderen Würfel auszutauschen oder einen Testwürfel mit Produktivdaten zu testen.

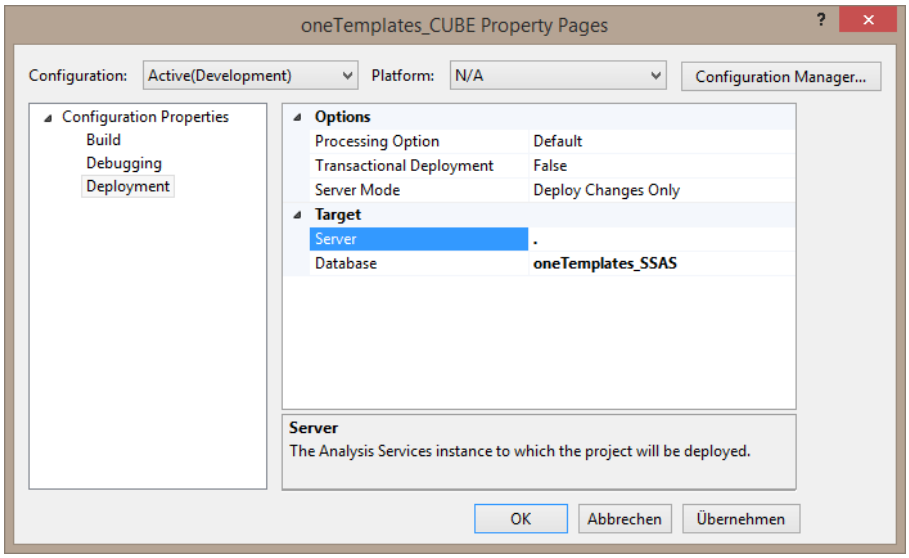


Abbildung 23: Deployment - Cube Deploymenteinstellung

Anpassung:

Der Cube Datenbankname muss auch im Paket „CUBE_PROCESS“ gesetzt werden, wenn es täglich prozessiert werden soll.

Hinweis:

Wenn der Produktivcube von mehreren Personen verwendet wird, dann ist die Erstellung eines zweiten Cubes empfehlenswert, insbesondere für große Cubes, die eine lange Verarbeitungszeit benötigen. Nach der erfolgreichen Verarbeitung des zweiten Cubes tauschen Sie es mit dem Produktivcube aus, indem Sie ein

Backup und Restore machen, damit Sie auf die Cube ID setzen können, die in der täglichen Beladung festgelegt wurde. Andernfalls bekommen Sie eine Fehlermeldung vom täglichen Job, weil die Cube ID nicht gefunden werden konnte, wenn Sie den Produktivcube löschen und den neuen Cube umbenennen. Durch die Umbenennung wird die ID nicht mitgeändert, sondern nur der Cube Datenbankname.

5 SQL Server Agent Jobsteuerung

5.1 Standardjobs

Standardjobs	Beschreibung und Konfiguration
Load_Staging	Beladung der Staging Area Step 1: Start Package => Masterpaket Staging [MASTER_STA.dtsx]
Load_CDW_Dimensions	Beladung von CDW Dimensionen Step 1: Start Package => Masterpaket CDW Dimensionen [MASTER_CDW_Dimensions.dtsx]
Load_CDW_Facts	Beladung von CDW Fakten Step 1: Start Package => Masterpaket CDW Fakten [MASTER_CDW_Facts.dtsx]
Load_CDW	Beladung von CDW Dimensionen und Fakten Step 1: Load CDW Dimensions => Masterpaket CDW Dimensionen [MASTER_CDW_Dimensions.dtsx] Step 2: Load CDW Facts => Masterpaket CDW Fakten [MASTER_CDW_Facts.dtsx]
Process_Cube	Cube Verarbeitung Step 1: Start Package => Cube Process Database Paket [CUBE_PROCESS.dtsx]
Masterjob	Beladung von Staging Area bis Cube Step 1: Start Masterjob Package => Masterjob Paket [MASTERJOB.dtsx] Step 2: Process Cube Database => Cube Process Database Paket [CUBE_PROCESS.dtsx]

Tabelle 14: SQL Server Agent Job - Standardjobs

5.2 Service Account Berechtigungen

Folgende Service Accounts müssen auf dem Datenbankserver von Test und Produktiv berechtigt werden:

- **NT SERVICE\SQLSERVERAGENT**
  - Für die Ausführung von SSIS Paketen über einen Job
- **NT SERVICE\MSSQLServerOLAPService**
  - Für die Cube Verarbeitung über einen Job

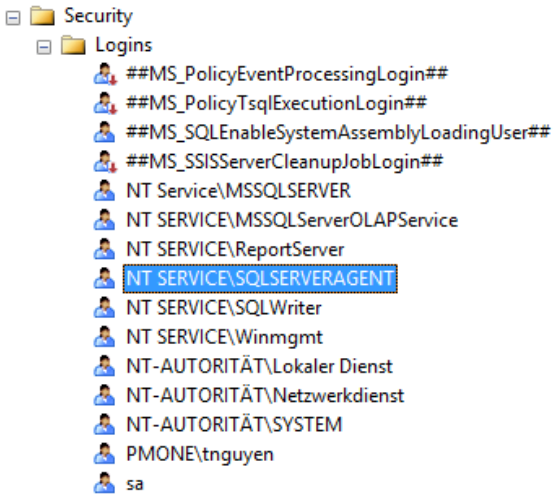


Abbildung 24: SQL Server Agent Job - Berechtigungen

Hinweis:

In der Regel können Sie den Service Benutzer in die Rolle „sysadmin“ setzen.

5.3 Jobeinrichtung

*Abbildung 25: SQL Server Agent Job - **Einstellung eines Job Steps***

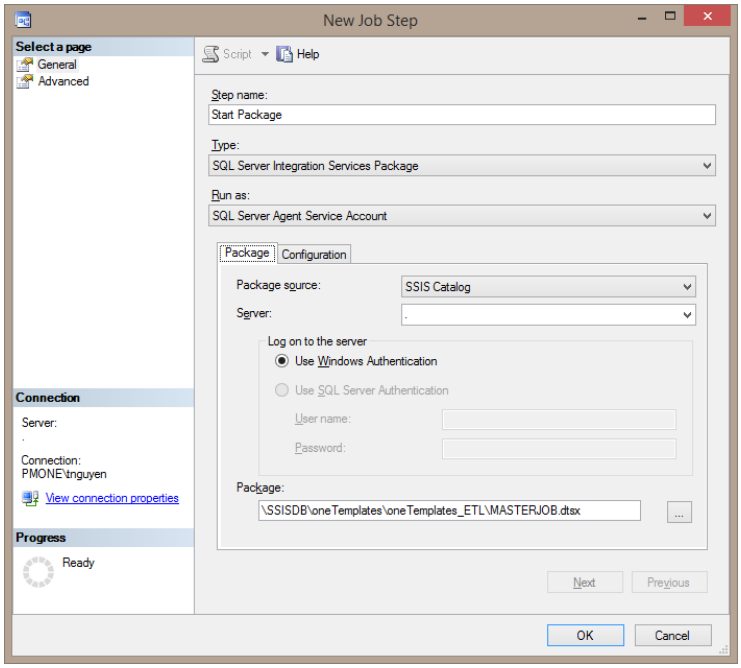
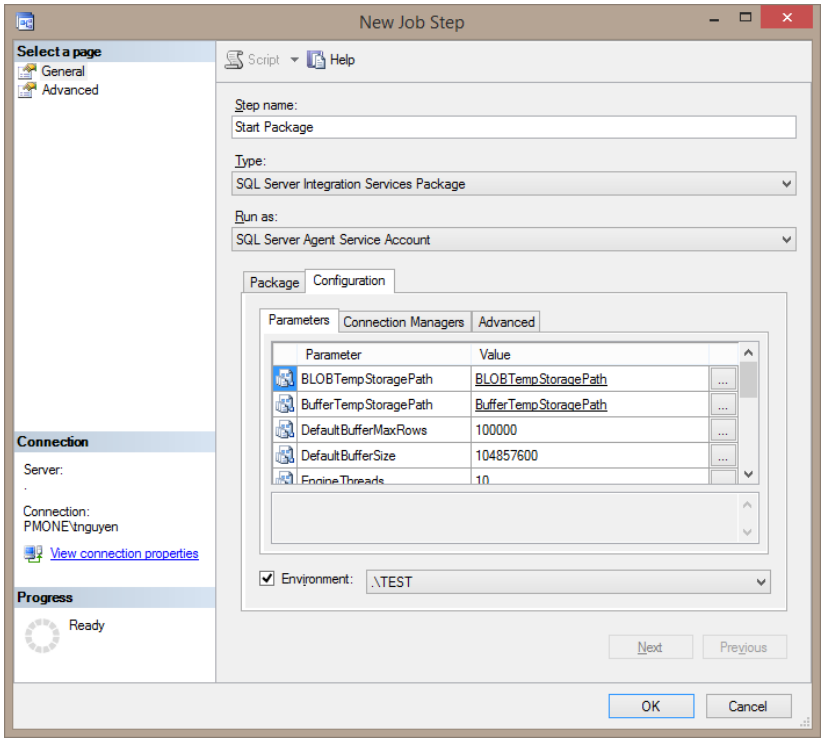


*Abbildung 26: SQL Server Agent Job - **Einbindung einer Umgebung zum Job und Paket***



5.4 Email Benachrichtigung im Fehlerfall

Wenn ein Job fehlschlägt, können Sie einstellen an welchen Empfänger es gesendet werden soll. Standardmäßig reicht die Emailaussendung am Masterjob bzw. der tägliche Job. Im Falle einer Änderung der E-Mail muss dies an dieser Stelle gemacht werden:

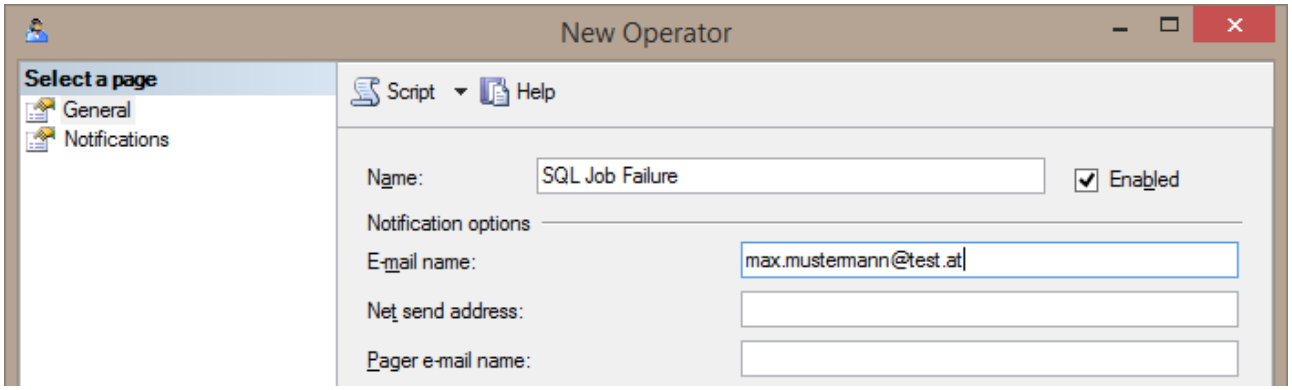


Abbildung 27: SQL Server Agent Job - Email Benachrichtigung im Fehlerfall

Über den ***„SQL Server Agent → Operators → SQL Job Failure (wenn nicht vorhanden, dann neu anlegen) → Properties“*** kann die Emailadresse geändert werden.

5.5 Job Status

Den Job Status kann man über den sogenannten „Job Activity Monitor“ des SQL Server Agents abrufen. Sie gibt eine Übersicht über die aktuell laufenden Jobs, fehlgeschlagene oder erfolgreich ausgeführte Jobs.

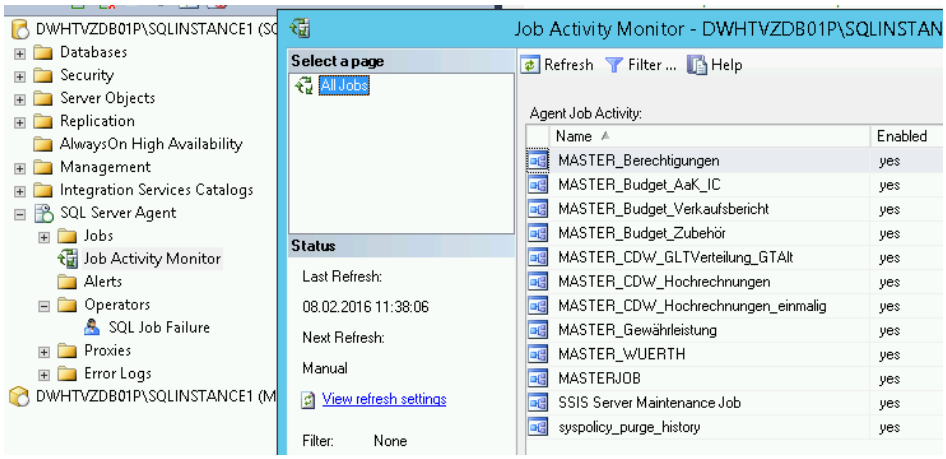


Abbildung 28: SQL Server Agent Job - Job Activity Monitor

Des Weiteren ist es möglich in den einzelnen Jobs die Historie der Ausführungen anzuzeigen. Falls ein Job fehlgeschlagen ist und dieser aus mehreren Steps besteht, kann man hier feststellen, an welchem Step der Job fehlgeschlagen ist. Über den SQL Server Agent und mit Rechtsklick auf den Job → „View History“ öffnet sich der Log Viewer mit der Historie des Jobs.

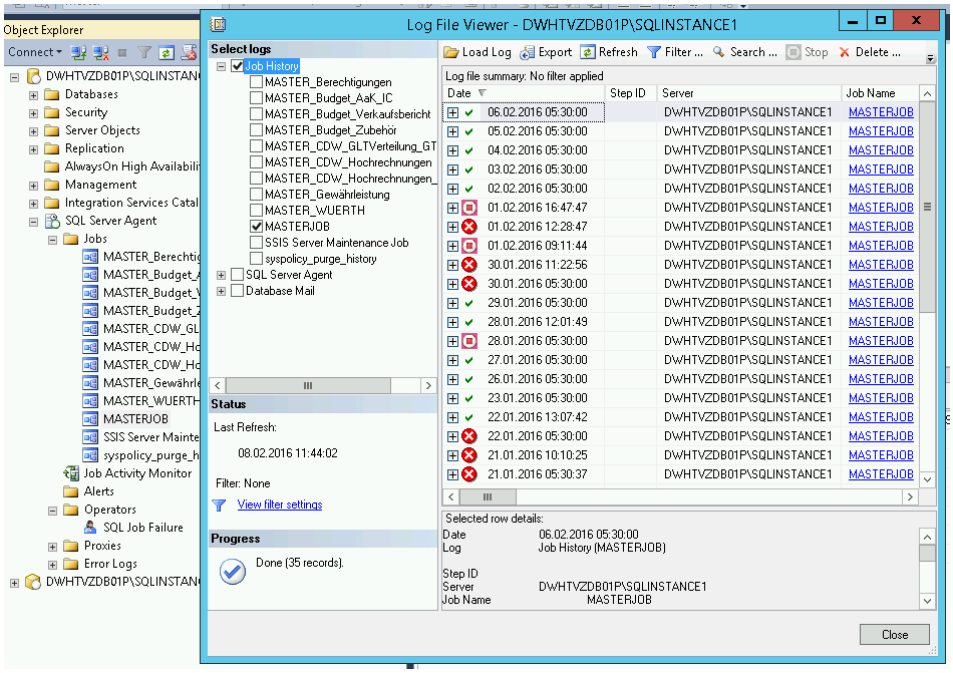


Abbildung 29: SQL Server Agent Job - Job History Log

Eine weitere Möglichkeit über das „oneLog“ wird im nächsten Unterkapitel erläutert.

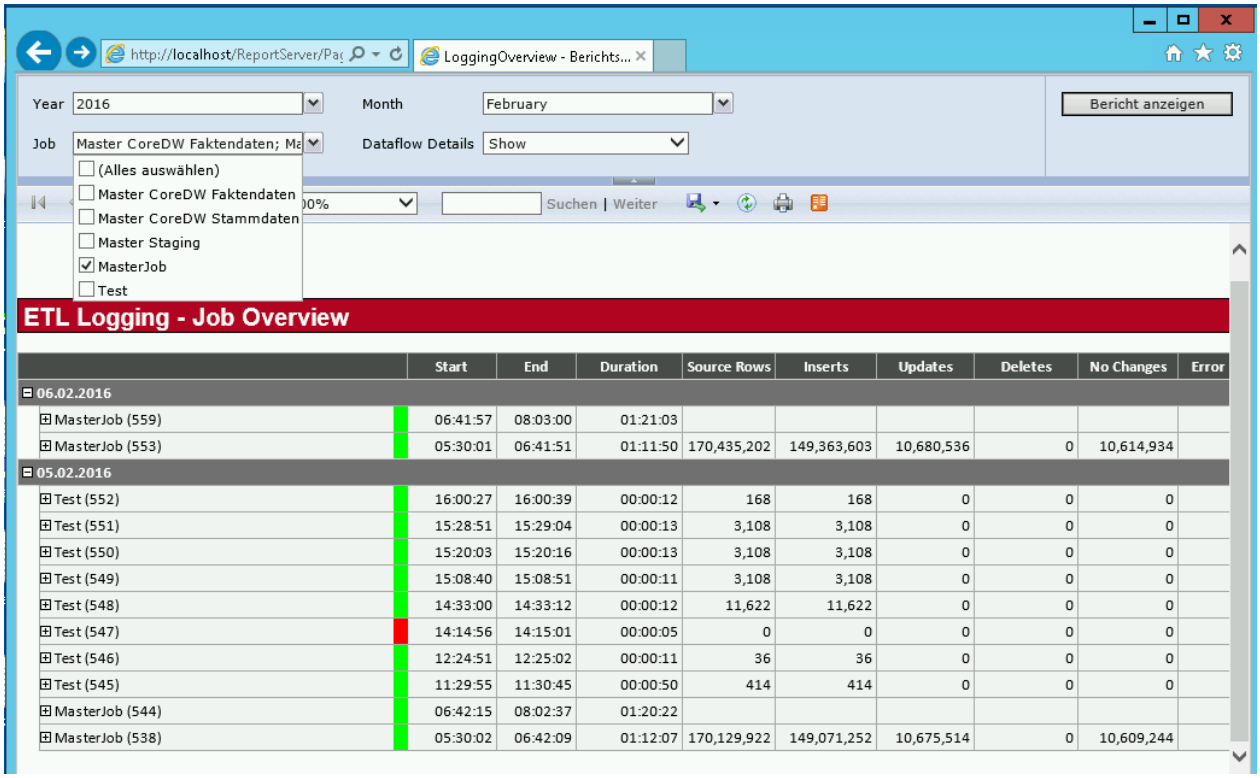
6 OneLog – Logging Framework

Der gesamte ETL Prozess wird über das oneLog mitprotokolliert. Zur oneLog gehören auch fertige Reporting Services Berichte, die eine gute Übersicht über alle Jobs und Status geben.

Standardmäßig ist der oneLog Bericht über folgende URL erreichbar:
http://localhost/ReportServer/Pages/
ReportViewer.aspx?%2fBerichte%2fLoggingOverview%rs:Command=Render

Die User benötigen **Read Rechte** auf die Datenbank **PorscheTvzLogging** und **GRANT EXECUTE**. Bevor der Bericht angezeigt wird, müssen einige Parameter ausgewählt werden. Hier eine beispielhafte Anleitung Schritt für Schritt:

1. Job auswählen – in diesem Fall der tägliche DWH Job → MasterJob



	Start	End	Duration	Source Rows	Inserts	Updates	Deletes	No Changes	Error
06.02.2016									
MasterJob (559)	06:41:57	08:03:00	01:21:03						
MasterJob (553)	05:30:01	06:41:51	01:11:50	170,435,202	149,363,603	10,680,536	0	10,614,934	
05.02.2016									
Test (552)	16:00:27	16:00:39	00:00:12	168	168	0	0	0	
Test (551)	15:28:51	15:29:04	00:00:13	3,108	3,108	0	0	0	
Test (550)	15:20:03	15:20:16	00:00:13	3,108	3,108	0	0	0	
Test (549)	15:08:40	15:08:51	00:00:11	3,108	3,108	0	0	0	
Test (548)	14:33:00	14:33:12	00:00:12	11,622	11,622	0	0	0	
Test (547)	14:14:56	14:15:01	00:00:05	0	0	0	0	0	
Test (546)	12:24:51	12:25:02	00:00:11	36	36	0	0	0	
Test (545)	11:29:55	11:30:45	00:00:50	414	414	0	0	0	
MasterJob (544)	06:42:15	08:02:37	01:20:22						
MasterJob (538)	05:30:02	06:42:09	01:12:07	170,129,922	149,071,252	10,675,514	0	10,609,244	

Abbildung 30: OneLog - Job Auswahl

2. Bericht anzeigen klicken

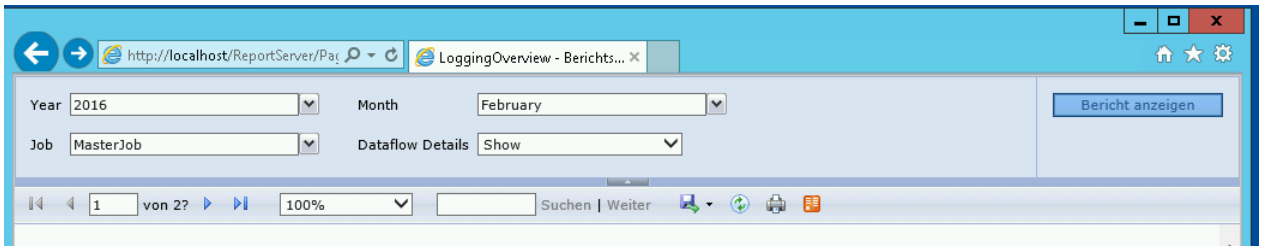
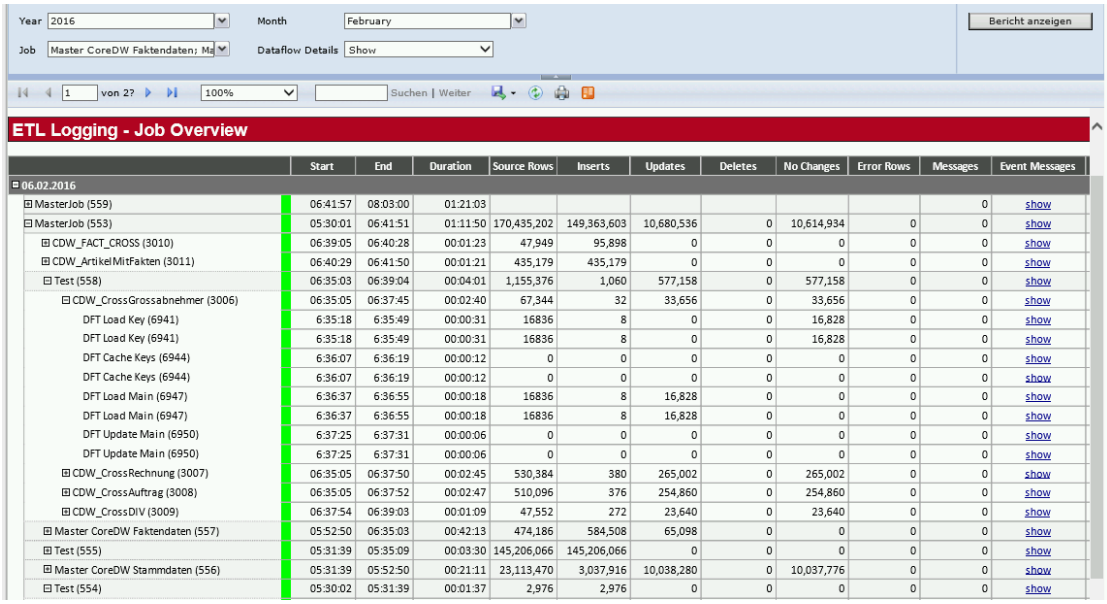


Abbildung 31: OneLog - Bericht anzeigen

3. Bericht wird generiert und angezeigt – ein Drilldown ist möglich bis auf Datenflussebene



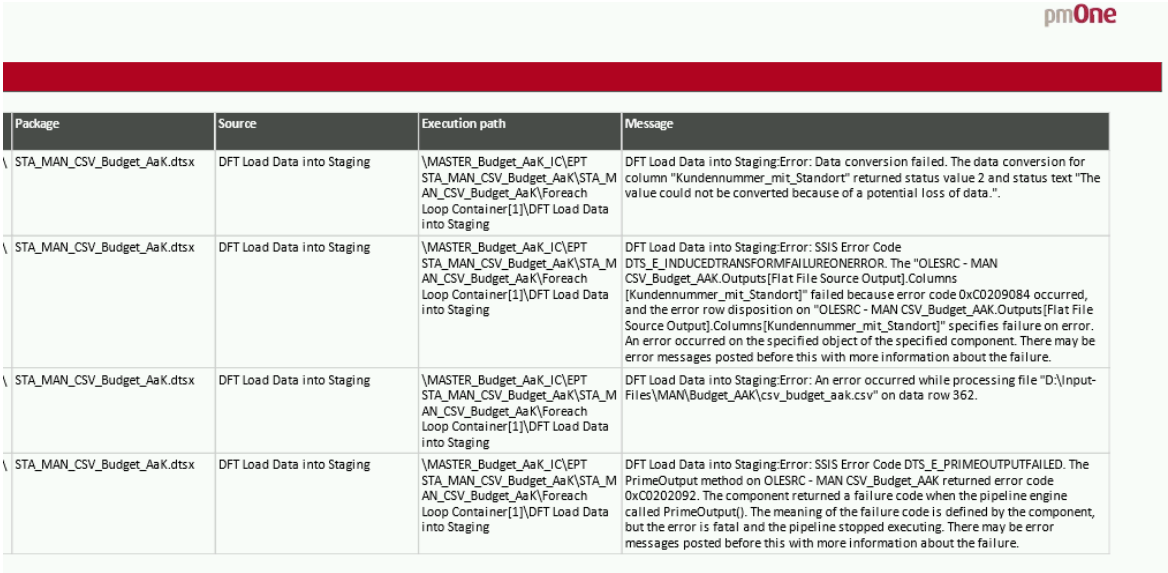
	Start	End	Duration	Source Rows	Inserts	Updates	Deletes	No Changes	Error Rows	Messages	Event Messages
06.02.2016											
MasterJob (559)	06:41:57	08:03:00	01:21:03							0	show
MasterJob (553)	05:30:01	06:41:51	01:11:50	170,435,202	149,363,603	10,680,536	0	10,614,934	0	0	show
CDW_FACT_CROSS (3010)	06:39:05	06:40:28	00:01:23	47,949	95,898	0	0	0	0	0	show
CDW_ArtikelMitFakten (3011)	06:40:29	06:41:50	00:01:21	435,179	435,179	0	0	0	0	0	show
Test (558)	06:35:03	06:39:04	00:04:01	1,155,376	1,060	577,158	0	577,158	0	0	show
CDW_CrossGrossabnehmer (3006)	06:35:05	06:37:45	00:02:40	67,344	32	33,656	0	33,656	0	0	show
DFT Load Key (6941)	6:35:18	6:35:49	00:00:31	16836	8	0	0	16,828	0	0	show
DFT Load Key (6941)	6:35:18	6:35:49	00:00:31	16836	8	0	0	16,828	0	0	show
DFT Cache Keys (6944)	6:36:07	6:36:19	00:00:12	0	0	0	0	0	0	0	show
DFT Cache Keys (6944)	6:36:07	6:36:19	00:00:12	0	0	0	0	0	0	0	show
DFT Load Main (6947)	6:36:37	6:36:55	00:00:18	16836	8	16,828	0	0	0	0	show
DFT Load Main (6947)	6:36:37	6:36:55	00:00:18	16836	8	16,828	0	0	0	0	show
DFT Update Main (6950)	6:37:25	6:37:31	00:00:06	0	0	0	0	0	0	0	show
DFT Update Main (6950)	6:37:25	6:37:31	00:00:06	0	0	0	0	0	0	0	show
CDW_CrossRechnung (3007)	06:35:05	06:37:50	00:02:45	530,384	380	265,002	0	265,002	0	0	show
CDW_CrossAuftrag (3008)	06:35:05	06:37:52	00:02:47	510,096	376	254,860	0	254,860	0	0	show
CDW_CrossDIV (3009)	06:37:54	06:39:03	00:01:09	47,552	272	23,640	0	23,640	0	0	show
Master CoreDW Faktendaten (557)	05:52:50	06:35:03	00:42:13	474,186	584,508	65,098	0	0	0	0	show
Test (555)	05:31:39	05:35:09	00:03:30	145,206,066	145,206,066	0	0	0	0	0	show
Master CoreDW Stammdaten (556)	05:31:39	05:52:50	00:21:11	23,113,470	3,037,916	10,038,280	0	10,037,776	0	0	show
Test (554)	05:30:02	05:31:39	00:01:37	2,976	2,976	0	0	0	0	0	show

Abbildung 32: OneLog - Logging Overview

Es gibt drei verschiedene Statusfarben

	Job läuft noch
	Job erfolgreich durchgelaufen
	Job fehlgeschlagen

Wenn etwas fehlschlägt lässt sich die Fehlermeldung auch direkt im OneLog auslesen mit Klick auf Event Messages „show“.



Package	Source	Execution path	Message
STA_MAN_CSV_Budget_AaK.dtsx	DFT Load Data into Staging	\MASTER_Budget_AaK_IC\EPT STA_MAN_CSV_Budget_AaK\STA_MAN_CSV_Budget_AaK\Foreach Loop Container[1]\DFT Load Data into Staging	DFT Load Data into Staging:Error: Data conversion failed. The data conversion for column "Kundennummer_mit_Standort" returned status value 2 and status text "The value could not be converted because of a potential loss of data.".
STA_MAN_CSV_Budget_AaK.dtsx	DFT Load Data into Staging	\MASTER_Budget_AaK_IC\EPT STA_MAN_CSV_Budget_AaK\STA_MAN_CSV_Budget_AaK\Foreach Loop Container[1]\DFT Load Data into Staging	DFT Load Data into Staging:Error: SSIS Error Code DTS_E_INDUCEDTRANSFORMFAILUREONERROR. The "OLESRC - MAN CSV_Budget_AAK.Outputs[Flat File Source Output].Columns [Kundennummer_mit_Standort]" failed because error code 0xC0209084 occurred, and the error row disposition on "OLESRC - MAN CSV_Budget_AAK.Outputs[Flat File Source Output].Columns[Kundennummer_mit_Standort]" specifies failure on error. An error occurred on the specified object of the specified component. There may be error messages posted before this with more information about the failure.
STA_MAN_CSV_Budget_AaK.dtsx	DFT Load Data into Staging	\MASTER_Budget_AaK_IC\EPT STA_MAN_CSV_Budget_AaK\STA_MAN_CSV_Budget_AaK\Foreach Loop Container[1]\DFT Load Data into Staging	DFT Load Data into Staging:Error: An error occurred while processing file "D:\Input-Files\MAN\Budget_AAK\csv_budget_aak.csv" on data row 362.
STA_MAN_CSV_Budget_AaK.dtsx	DFT Load Data into Staging	\MASTER_Budget_AaK_IC\EPT STA_MAN_CSV_Budget_AaK\STA_MAN_CSV_Budget_AaK\Foreach Loop Container[1]\DFT Load Data into Staging	DFT Load Data into Staging:Error: SSIS Error Code DTS_E_PRIMEOUTPUTFAILED. The PrimeOutput method on OLESRC - MAN CSV_Budget_AAK returned error code 0xC0202092. The component returned a failure code when the pipeline engine called PrimeOutput(). The meaning of the failure code is defined by the component, but the error is fatal and the pipeline stopped executing. There may be error messages posted before this with more information about the failure.

Abbildung 33: OneLog - Fehlermeldung

7 Anleitung für die Erstellung einer neuen Dimension

1. Staging Area – **Beladung MS SQL Tabelle**
    a. Template Paket **„_TEMPLATE_STA_SQL_TABLE.dtsx“** kopieren und nach folgender Namenskonvention umbenennen:
    „STA_“ + Kürzel vom Vorsystem + Tabellenname + „.dtsx“
    z.B. STA_AW_Customer.dtsx

    Namenskonvention für **STA Tabelle** (Zieltabelle):
    „[STA].[„ + Kürzel vom Vorsystem + „_“ + Tabellenname vom Vorsystem +“]“
    z.B. [STA].[AW_Customer] (AW ist der Kürzel für AdventureWorks)

    b. Folgende **Paket Variablen** setzen:

DBObject	**… Staging Tabellenname**
DBSchemaSRC	**… Schema vom Vorsystem**
DBSchemaSTA_Praefix	**… Kürzel vom Vorsystem**

    z.B. „Customer“, „Sales“, „AW“
    [Sales].[Customer] (in AW) => [STA].[AW_Customer] (in STA)

    c. Paket schließen und wieder öffnen (aufgrund der Aktualisierung der Expressions)
    d. DataFlow „DFT Load Data into Staging”
        i. Metadaten in der Quelle aktualisieren (Komponente öffnen, auf den Reiter Spalten wechseln und schließen)
        ii. CREATE Skript für STA Tabelle aus der Zielkomponente generieren lassen
        iii. Schema und Tabellenname in CREATE Skript nach obiger Namenskonvention für STA Tabelle umbenennen und auf STA DB ausführen
        iv. Mapping in der Zielkomponente setzen
    e. Paket speichern, ausführen bzw. testen und schließen
    f. Paket ins Masterpaket für STA aufnehmen
    g. **Staging View (ETL)** erstellen
    Namenskonvention für Dimension: **„[ETL].[vCDW_Dim“ + Dimensionsname + „]“**
    z.B. [ETL].[vCDW_DimKunde]

    Business Key sollte immer an der ersten Stelle sein und SystemCode am Schluss
    z.B. CONVERT(nvarchar(10),'AW') AS SystemCode
    AW … AdventureWorks

2. Staging Area – **Beladung CSV Datei**
    a. Template Paket **„_TEMPLATE_STA_CSV_File.dtsx“** kopieren und nach folgender Namenskonvention umbenennen:
    „STA_CSV“ + Tabellenname + „.dtsx“
    z.B. STA_CSV_KundenZusatzinformation.dtsx

    Namenskonvention für **STA Tabelle** (Zieltabelle):
    „[STA].[CSV_“ + Tabellename (kann von der Datei abgeleitet werden) +“]“
    z.B. [STA].[CSV_KundenZusatzinformationen]

    b. Folgende **Paket Variablen** setzen:

DBTable_STA	**… Vollständiger Name der Staging Tabelle**
FileMask	**… Dateiname** **Wildcard wird unterstützt (“*”), wenn mehrere Dateien auf einmal geladen werden sollen**

    z.B. „[STA].[CSV_KundenZusatzinformatio]“, „KundenZusatzinformation.csv“
    „KundenZusatzinformation.csv“ (im Dateisystem) => [STA].[CSV_KundenZusatzinformation] (in STA)

    c. Folgende **Projekt Parameter** setzen**:**

CSVInputPathGeneral	**… Pfad für den Ablageort**

    z.B. „//TESTSERVER/CSV/“ oder „D:/DWH/Inputs/“

    d. Paket schließen und wieder öffnen (aufgrund der Aktualisierung der Expressions)
    e. DataFlow „DFT Load Data into Staging”
        i. Connection Manager „CSV_FileConnection“ öffnen und zu beladende Datei auswählen und Spalten aktualisieren
        ii. Metadaten in der Quelle aktualisieren (Komponente öffnen, auf den Reiter Spalten wechseln und schließen)
        iii. CREATE Skript für STA Tabelle aus der Zielkomponente generieren lassen
        iv. Schema und Tabellenname in CREATE Skript nach obiger Namenskonvention für STA Tabelle umbenennen und auf STA DB ausführen
        v. Mapping in der Zielkomponente setzen
    f. Paket speichern, ausführen bzw. testen und schließen
    g. Paket ins Masterpaket für STA aufnehmen
    h. **Staging View (ETL)** erstellen
    Namenskonvention für Dimension: **„[ETL].[vCDW_Dim“ + Dimensionsname + „]“**
    z.B. [ETL].[vCDW_DimKunde]

    Business Key sollte immer an der ersten Stelle sein und SystemCode am Schluss
    z.B. CONVERT(nvarchar(10),'CSV') AS SystemCode

3. Core Datawarehouse (CDW)
    a. **Generierungsprozedur** für CDW Objekte (Tabellen und Prozeduren) ausführen

        EXEC [ETL].[GenerateCdwDim] Parameter:
            1 … Staging View mit Schema
            2 … Dimensionsname (Präfix „Dim“)
            3 … Business Key (bei Liste, durch Beistrich zu trennen)
            4 … Historisierte Felder
            5 … Flag zur Löschung von Keys
            6 … CDW Datenbank
            7 … Staging Datenbank

        Beispiel:
        EXEC [ETL].[GenerateCdwDim] 'ETL.vCDW_DimKunde', 'DimKunde', 'Kundennummer', null, 1, 'OneTemplateCoreDW', 'OneTemplateStaging'

    b. Template Paket **„_TEMPLATE_CDW_Dimensions.dtsx“** kopieren und nach folgender Namenskonvention umbenennen:
        „CDW_Dim“ + Zieltabellenname bzw. Dimensionsname + „.dtsx“
        z.B. CDW_DimKunde.dtsx

    c. Folgende **Paket Variablen** setzen:

DBObject	**… Dimensionsname**
DBBusinessKeyColumns	**… Business Keys (bei Liste, durch Beistrich zu trennen)**
- Bei Historisierung -	
DBSCD2Columns	**… Historisierte Felder (mit Business Key)**
HistoryDisabled	**… auf „False“, wenn historisiert werden soll**

        z.B. „DimKunde“, „Kundennumer“
        bei Historisierung: „Kundennummer, Kundenland“, False

    d. Paket schließen und wieder öffnen (aufgrund der Aktualisierung der Expressions)
    e. KeyCache in Connection Manager einstellen
        Business Key mit Indexposition und dem entsprechendem Datentyp
    f. DataFlow „DFT Load Keys“ + „DFT Cache Keys“ + „DFT Load Main“ + „ DFT Update Main“
        i. Metadaten in der Quelle aktualisieren
        ii. Lookup „LKP CDW Key“ für Business Key (nur bei Load Key + Load Main)
        iii. Mapping in den Zieltabellen
    g. Bei Historisierung - DataFlow „DFT Load Main History“ + „DFT Merge Main History“
        i. Metadaten in der Quelle aktualisieren
        ii. Lookup „LKP CDW Key“ für Business Key (nur bei Load Key + Load Main)
        iii. Mapping in den Zieltabellen
    h. Paket speichern, ausführen bzw. testen und schließen

i. Paket ins Masterpaket für CDW Dimensionen aufnehmen

j. **DWH View** erstellen
Alias für SID Spalte => Dimensions-/Faktenobjekt (mit Suffix) + „_SID“
Namenskonvention: „[DWH].[vDim“ + Dimensionsname + „]“

Beispiel:

```
CREATE VIEW [DWH].[vDimKunde] AS
        SELECT [SID] AS DimKunde_SID
                ,[Kundennummer],[Kundenname], [Kundenland]
        FROM [CDW].[DimProject]
```

Bei Historisierung:

```
SELECT kdh.[DimKunde_History_SID] -- Eindeutige ID für Cube Dimension Mapping
        ,kd.[SID] AS [DimKunde_SID]
        ,kd.[Kundenname]                -- Nicht Hist. von der Haupttabelle (akt. Wert)
        ,kdh.[Kundenland]               -- Historisiertes Feld von History Tabelle
FROM [CDW].[DimKunde] kd
INNER JOIN [CDW].[DimKunde_History] kdh
        ON kd.[SID] = kdh.[SID]
```

k. DWH View bzw. Dimension in den Cube hinzufügen (DataSource View)

8 Anleitung für die Erstellung von Fakten

1. Staging Area
   a. Siehe Anleitung für die Erstellung einer neuen Dimension unter Punkt 1 Staging Area für Beladung von einer MS SQL Tabelle oder Punkt 2 für die Beladung von einer CSV Datei
   b. **Staging View (ETL)** erstellen
   Namenskonvention für Fakten: **„[ETL].[vCDW_Fakt“ + Faktenname + „]“**
   z.B. [ETL].[vCDW_FaktVerkaufsauftrag]

   Aufbau: Datumsfelder, Business Keys aller angehängten Dimensionen und SystemCode
   Optional: Degenerated Dimension bzw. ID des Fakts wie z.B. Belegnummer können in der Faktenview für Testzwecke aufgenommen werden

2. Core Datawarehosue (CDW)
   a. Template Paket **„_TEMPLATE_CDW_Facts.dtsx“** kopieren und nach folgender Namenskonvention umbenennen:
   „CDW_Fakt“ + Zieltabellenname bzw. Faktenname + „.dtsx“
   z.B. CDW_FaktVerkaufsauftrag.dtsx

   b. Folgende **Paket Variablen** setzen:
   DBObject **… Faktenname**

   z.B. „FaktVerkaufsauftrag“ oder „FaktBestelleingang“

c. Paket schließen und wieder öffnen (aufgrund der Aktualisierung der Expressions)

d. DataFlow „DFT Load FACT Table“

i. Metadaten in der Quelle aktualisieren

ii. Für jede Dimension das Beispiel **Lookup “DimTable_SID”** kopieren und anpassen

Tab „Connection”

Zeitdimension

SELECT [SID], [DateValue] FROM [MAN].[DimZeit] with(nolock)

Standarddimension (Abfrage Key Tabelle)

SELECT [SID], [Business Key] FROM [CDW].[DimTable_Key] with(nolock)

Beispiel (anhand DimKunde):

SELECT [SID], [Kundennummer]

FROM [CDW].[DimKunde_Key] with(nolock)

Tab „General“

Mapping Business Key von CDW (durch “Connection” eingestellt) und Fakten

Output Alias nach SID Namenskonvention setzen

Dimensionsname + “_SID”

(bei Roleplaying) + „_“ + Roleplaying

z.B. DimKunde_SID, DimZeit_SID

bei Roleplaying: DimKunde_SID_Rechnung oder DimZeit_SID_Rechnung

i. CREATE Skript für CDW Tabelle aus der Zielkomponente generieren lassen

ii. Schema und Tabellenname in CREATE Skript nach folgender Namenskonvention für CDW Tabelle umbenennen und auf CDW DB ausführen

Namenskonvention für **CDW Tabelle** (Zieltabelle):

„[CDW].[Fakt“ + Faktenname +“]“

z.B. [CDW].[FaktVerkaufsauftrag]

iii. Mapping in der Zielkomponente setzen

e. Paket speichern, ausführen bzw. testen und schließen

f. Paket ins Masterpaket für CDW Fakten aufnehmen

g. DWH View erstellen

Namenskonvention: „[DWH].[vFakt“ + Faktenname + „]“

Beispiel:

```
SELECT
      [DimZeit_SID]
      ,[DimKunde_SID]
```

```
-- Historisierte Dimension => History SID
-- Stand zum Zeitpunkt des führenden Datums (Bestelldatum)
,COALESCE(kh.[HISTORY_SID],-1) AS DimKunde_History_SID

-- für MDX Calculation
,CONVERT(int, 1) AS DimZeittyp_SID
,CONVERT(int, 1) AS DimZeitvergleich_SID
,CONVERT(int, 1) AS DimDatenart_SID

-- Measures
,[ProduktPreis]
,[Menge]
,[Umsatz]
--
FROM [CDW].[Verkaufsauftrag] va
LEFT JOIN [CDW].[DimKunde_History] kh ON va.DimKunde_SID = kh.[SID]
        AND va.Bestelldatum BETWEEN kh.VALID_FROM AND kh.VALID_TO
```

h. DWH View bzw. Faktengruppe in den Cube hinzufügen (DataSource View)

9 Abkürzungsverzeichnis

Abkürzung	
STA	Staging
CDW	Core Datawarehouse
DWH	Datawarehouse
Expr.	Expression, d.h. der Wert setzt sich automatisch aus anderen Variablen zusammen
VS	Visual Studio bzw. Visual Data Tools
MS	Microsoft
SID	Surrogate Key, Ersatzschlüssel für das DWH
URL	Uniform Resource Locator, Internetadresse
DB	Datenbank
SCD	Slowly Changing Dimensions, sich langsam verändernde Dimensionen
SCD1	Typ 1 Dimensionsattribute wird immer mit dem aktuellen Wert überschrieben
SCD2	Typ 2 Dimensionsattribute wird bei Veränderung mit GültigVon/Bis und IsCurrent Flag historisiert

10 Tabellenverzeichnis

11 Abbildungsverzeichnis

Microsoft Partner
Gold Cloud Platform
Gold Data Analytics
Silver Collaboration and Content

SAP Partner

pmOne – wir machen #Datenversteher.

Die über 200 Mitarbeiter der 2007 gegründeten pmOne AG verbessern den Analysequotienten von Unternehmen, indem sie aus vorhandenen Daten neue Erkenntnisse für die Unternehmenssteuerung gewinnen.

Durch ihre Expertise in den Bereichen Planung, Reporting, Analyse und Prognose unterstützt die pmOne AG, zu der auch die Managementberatung verovis gehört, die Digitalisierungsstrategien ihrer über 500 Kunden.

Dabei kommen bevorzugt die technologischen Plattformen von Microsoft und SAP zum Einsatz, ergänzt um die eigenentwickelte Software cMORE.

Deutschland	pmOne AG
	Freisinger Straße 9
	D-85716 Unterschleißheim
	Tel.: +49 89 416 17 61-0
	Fax: +49 89 416 17 61-99
	www.pmone.com

Österreich	pmOne GmbH
	Media Quarter Marx 3.4
	Maria-Jacobi-Gasse 1
	A-1030 Wien
	Tel.: +43 1 890 28 52-0
	Fax: +43 1 890 28 52-15
	www.pmone.com

Schweiz	pmOne Schweiz GmbH
	Hertistrasse 27
	CH-8304 Wallisellen
	Tel.: +41 44 515 31-00
	Fax: +41 44 515 31-01
	www.pmone.com

Zentrale E-Mail-Adresse	kontakt@pmone.com