# 

MDWT2008R2 Historical Load

Integration Services

The database MDWT\_2008R2 is a simple star schema “data warehouse” database for the Adventure Works sample. The MDWT database is similar to, but simpler than, the data warehouse database provided by Microsoft on CodePlex. The MDWT version of the database is described in the book **The Microsoft Data Warehouse Toolkit for SQL Server 2008 R2**, 2011. We call this book MDWT for short.

This ReadMe describes how to use the sample Integration Services packages to run a one-time load of the MDWT\_2008R2 database. The primary purpose of these packages is to teach some principles of SSIS system design. These principles are outlined in the MDWT book (Chapter 7).

# Before You Begin

1. Install SQL Server 2008R2, including the relational database, Integration Services, SQL Studio, and BI Development Studio (BIDS). Install the sample transaction database AdventureWorks2008R2, which you can find on CodePlex. If at all possible, install the database on your local machine. If you don’t use your local machine as the database server, you’ll need to edit some package connections, as we describe below.
2. Create a new directory at c:\SSIStemp.
3. Copy the solution directories to c:\MDWT\_Projects. If you put them anywhere else, you’ll need to edit some connections inside the packages (described below).
4. You must first create the MDWT\_2008R2 relational database and tables, by running the script CrDtb\_MDWT\_2008R2.sql, which you should find in the same directory as this readme file. This script is exactly as generated by the data modeling Excel workbook, also available from the book’s website.
5. Run the script ModDtb\_MDWT\_2008R2.sql, which makes some minor tweaks to the database (defining some non-default views, adding some “dummy” rows to a few dimension tables).
6. Run the script CrDtb\_MDWT\_2008R2\_Stage.sql, which creates a staging database.

# Extracting to Stage

You will need to run two sets of packages to populate the target database. The first solution, MDWT2008\_Extract, extracts the necessary tables from the source database AdventureWorks2008R2. This is a little silly in the context of a static sample application, but illustrates the best practice of breaking your ETL system into an “extract” piece and a “transform and load” piece, as discussed in the MDWT book.

To run the extract to the newly-created staging database, navigate to the file c:\MDWT\_Projects\MDWT2008\_Extract\MDWT2008\_Extract.sln. Double-click to launch BI Studio.

You should have stored these packages in the directory c:\MDWT\_Projects. If you put the packages somewhere other than in c:\MDWT\_Projects, simply edit the connections in the master package to point to the correct location.

Run the MASTER EXTRACT PACKAGE.

Once the package completes, take a look at the results and explore the solution. In SQL Server Management Studio, look at the audit table (SELECT \* FROM DimAudit).

## Packages in the MDWT2008\_Extract solution

The solution consists of the following packages, following the principle of creating one package per target table:

|  |  |  |
| --- | --- | --- |
| MASTER EXTRACT PACKAGE  Currency  CurrencyInUse  CurrencyRate  CustomerIndividual  Demographics  Department  Employee | EmployeeTerritory  Product  ProductCategory  ProductModel  ProductSubcategory  SalesOrderDetail SalesOrderHeader  SalesReason | SalesTerritories  SpecialOffer  Store  StoreContacts  StoreDemographics |

## Characteristics of the extract solution

The packages in this solution are extremely simple. The dimension tables are all structured the same:

* A few tasks to set up the load and the audit dimension
* A data flow task with no transformations in it, to copy data to the staging database
* A few tasks to clean up and update the audit dimension

Because the packages are so simple, it’s easy to see how similar they are. There are a few important things to learn from these packages:

* Keep each package as simple as possible.
* Keep packages consistent. Use one package as the template for the next. (Don’t forget to update the package GUID in the properties pane for the package.)
* Look at how the audit dimension is set up and populated. Design this once and you never need to touch it again.
* Look at the design pattern for loading data into a target table.
* Look at the SalesOrderDetail package for a different design pattern: to stop execution of a package if the package violates a business rule. In this case, the business rule is to stop processing if we extracted fewer than X rows or Y distinct products.

# Transform and Load

There is a second solution that contains packages to transform the extracted data in the staging database, and load it into the final DW dimension and fact tables. This solution is called MDWT\_2008, and you can bring it up in BIDS by double-clicking C:\MDWT\_Projects\MDWT2008\MDWT\_2008.sln.

To run the package, execute the master package RUN THIS TO LOAD ALL.

## Packages in the MDWT\_2008 solution

The solution consists of the following packages, following the principle of creating one package per target table:

|  |  |
| --- | --- |
| Package | Purpose |
| RUN THIS TO LOAD ALL | Master package |
| Currency | Load the currency dimension – very simple. |
| Customer | Load the customer dimension. Contains two data flows, one for individual customers and one for resellers. |
| Date | Load the date dimension from a .csv created from the Excel spreadsheet. You could easily define the package to read Excel directly, but we chose a .csv to reduce problems with Excel versions. |
| Employee | Load the employee (salesperson) dimension. The source query joins two staging tables, but a third table is brought in via an SSIS lookup. Chapter 7 in MDWT discusses the pros and cons of these alternative approaches. |
| OrderInfo | Load the order info dimension – very simple. |
| Product\_SSIS | Load the product dimension, with a simple source query and using SSIS lookups to bring in decodes and additional attributes. This package includes hard-coded decodes for quite a few attributes. These really should be stripped out to lookup transforms as well, in order to make them table-driven. |
| Product\_SQL | Load the product dimension, doing most of the work in SQL. This version is not called by the master package, but is included because Chapter 7 discusses this approach versus Product\_SSIS above. |
| Orders\_Lookups | Load the orders fact table, implementing the surrogate key pipeline in lookups. |
| Orders\_SQL | Load the orders fact table, implementing the surrogate key pipeline via a complex query in the Source. This version is not called by the master package. |
| ExchangeRates | Load the exchange rates fact table – very simple. |

# Further Exploration

If you want to play around with any of the packages, be aware that we have RI defined between fact and dimension tables, and between all tables and the audit dimension. In the MDWT\_2008 solution we’ve included a SQL Script (TruncateTables.sql) that will clear out fact and dimension tables so that you can start over.

Alternatively, you may want to drop RI between dimensions and fact tables, in order to play around with SSIS more conveniently.