Managing Slowly Changing Dimension with Slow Changing Transformation in SSIS

As a data warehouse expert or as an ETL developer you will often come across scenarios in which you need to maintain and manage slowly changing dimensions. There are multiple ways to implement that in SQL Server and the easiest of those is using Slowly Changing Dimension Transformation in the data flow task of SSIS packages.

In this article I am going to provide you the steps and guidance needed to manage Slowly Changing Dimension with Slowly Changing Dimension Transformation in data flow task with an example.

**Understanding the Slowly Changing Dimension Scenario**

Dimension is a term in data management and data warehousing. It's the logical groupings of data such as geographical location, customer or product information. With Slowly Changing Dimensions (SCDs), data changes slowly rather than changing on a time-based, regular schedule.

There are different types of slowly changing dimensions:

* **SCD Type 0 (Fixed)** – This type is the least frequently used as this type does not accept changes and is fixed after first time insertion; it means once written, the value does not get overwritten.
* **SCD Type 1 (Changing)** – In this type, if the data is getting changed it gets overwritten with the new value.
* For example consider this example:

|  |  |  |
| --- | --- | --- |
| **SupplierCode** | **SupplierName** | **Address** |
| S0000001 | ABC Company | USA |
| S0000002 | XYZ Corporation | USA |

* If the name of the supplier changes over time, as you can see in the change in SupplierName below, the record will be updated. This looks pretty simple to implement, though it does not have history to keep track on.

|  |  |  |
| --- | --- | --- |
| **SupplierCode** | **SupplierName** | **Address** |
| S0000001 | ABC Company Ltd. | USA |
| S0000002 | XYZ Corporation | USA |

* **SCD Type 2 (Historical)** – In this type, if the data is changed it gets saved in a new record and the previous record with the previous value is marked as outdated.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SupplierCode** | **SupplierName** | **Address** | **EffectiveDate** | **Expiration Date** |
| S0000001 | ABC Company | USA | 3/2/2013 | 3/2/2013 |
| S0000002  - Advertisement - | XYZ Corporation | USA | 3/2/2013 |  |
| S0000001 | ABC Company Ltd. | USA | 3/3/2013 |  |

* To maintain SCD type 2, different people take different approaches. For example, one approach is to add effective and expiration dates to indicate a time period during which the record was active. If the expiration date is NULL it indicates the current active record. Another approach is to add one column to indicate the current active record. Normally people use the first approach or a combination of both.
* **SCD Type 4 (Limited history)** – This is not a frequently used SCD type as this has limited history to maintain and requires changes to the table. In this SCD type, additional columns are added in the table to keep the latest values whereas the older existing columns will have older values.

There are multiple ways to implement Slowly Changing Dimension in SQL Server and the easiest of those is using Slowly Changing Dimension Transformation in data flow task of SSIS packages, although it has its own limitation as discussed at the end of this article.

Before I can start explaining about Slowly Changing Dimension Transformation let me first explain Surrogate Key and why it’s important in data warehouse.

We often add a meaningless key known as Surrogate Key in the dimension. The surrogate is usually implemented as an integer, acts as a unique key or primary key for the dimension and gets associated to the fact table using foreign key constraint in it.

Surrogate key becomes extremely important to manage slowly changing dimension.

**Using Slowly Changing Dimension Transformation**

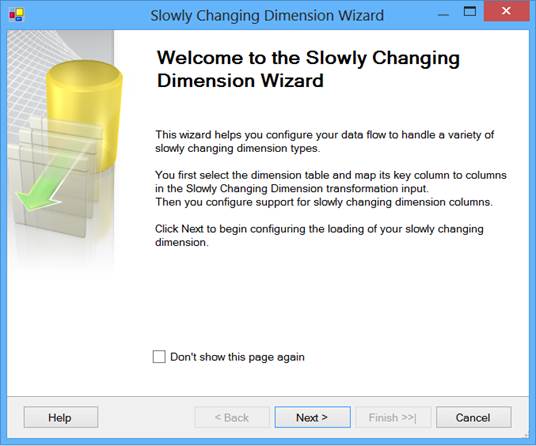
Let’s first create a supplier table, which will have source data and add some data to it; as you can notice below, I have added the SupplierCode column, which is a primary key at source and will work as a business key.

|  |
| --- |
| USE [AdventureWorks2012]  GO  CREATE TABLE [dbo].[Supplier](         [SupplierCode] CHAR(8) PRIMARY KEY,         [SupplierName] [varchar](50) NULL,         [Address] [varchar](50) NULL,  ) ON [PRIMARY]  GO  INSERT INTO [dbo].[Supplier] ([SupplierCode], [SupplierName], [Address])  VALUES  ('S0000001', 'ABC Company', 'USA'),  ('S0000002', 'XYZ Corporation', 'USA')  GO  SELECT \* FROM [dbo].[Supplier] |

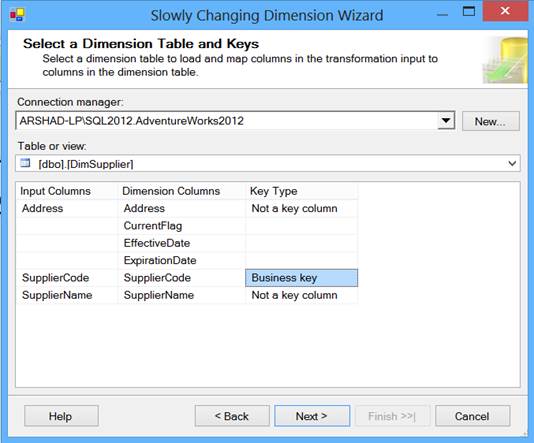
Now let’s create a dimension for storing Supplier data from the source; as you can notice below I have added a SupplierId column as a surrogate key, effective date and expiration date for keeping track of historical changes. I have also added a CurrentFlag column to indicate the current active record:

|  |
| --- |
| USE [AdventureWorks2012]  GO  CREATE TABLE [dbo].[DimSupplier](         [SupplierId] [int] IDENTITY(1,1) NOT NULL,         [SupplierCode] CHAR(8),         [SupplierName] [varchar](50) NULL,         [Address] [varchar](50) NULL,         [EffectiveDate] [date] NULL,         [ExpirationDate] [date] NULL,         [CurrentFlag] [char](1) NULL,         CONSTRAINT [PK\_DimSupplier] PRIMARY KEY CLUSTERED ([SupplierId] ASC)  ) ON [PRIMARY]  GO |

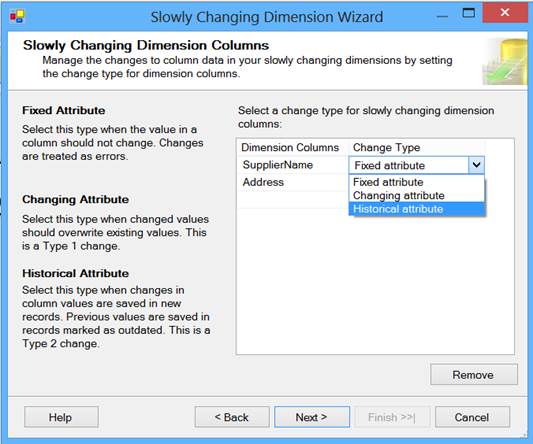
So far so good, let’s now create an SSIS package; add a data flow task to it and a source component to retrieve data from the source table. Now add a Slowly Changing Dimension Transformation component after the source component and connect the source component to the Slowly Changing Dimension Transformation component. Now double click on the Slowly Changing Dimension Transformation component to edit it and the Slowly Changing Dimension Wizard will be launched as shown below:

  
Slowly Changing Dimension Wizard

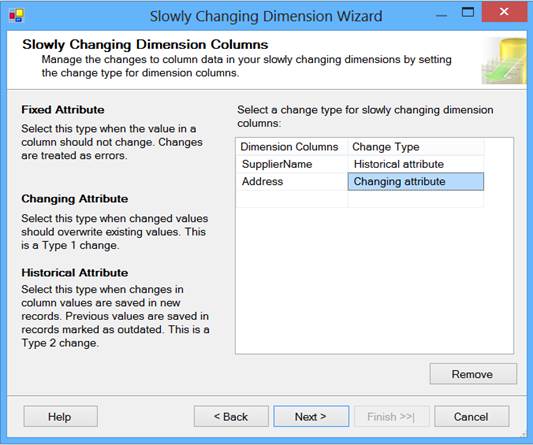
Click on the Next button to move on to the next screen of the wizard; in the next screen you need to first select the destination dimension table and map the columns. Next you need to specify the source column from the source table, which makes the business key. In my case SupplierCode is the primary key at the source table and hence I have marked it here as the business key as you can see below:

  
Business key

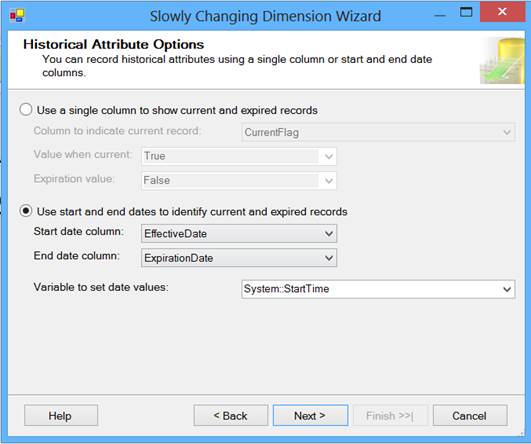
Click on the Next button to move on to the next screen of the wizard; in the next screen you need to specify each column of the dimension and whether it’s going to be treated as SCD Type 0, Type 1 or Type 2.

  
Specify each column of the dimension

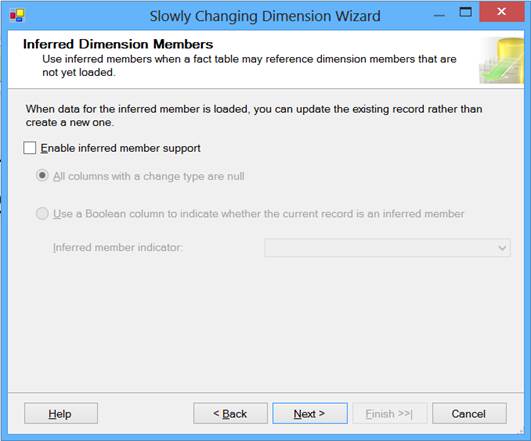
In my case I have chosen the Address to be treated as SCD Type 1 and Name to be treated as SCD Type 2 as shown below:

  
SCD Type 1 and SCD Type 2

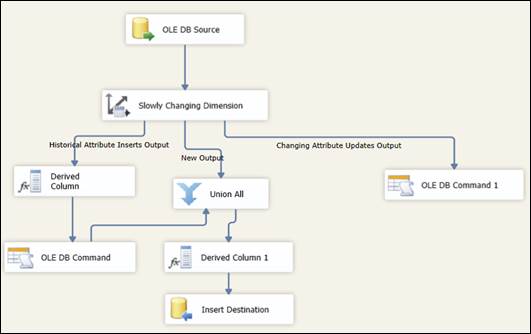
Click on Next button to move on to the next screen of the wizard; in the next screen you need to specify (as we have at least one of the column of SCD Type 2) a start date column (effective date), end date column (expiration date) and a variable to set the date value as you can see below:

  
Start and End Dates

Click on the Next button to move on to the next screen of the wizard; in the next screen you can specify the inferred dimension members setting as shown below:

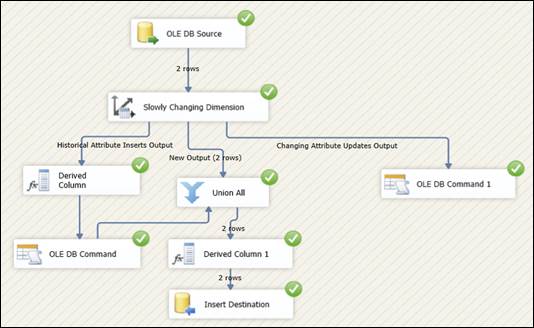
  
Inferred Dimension Members

Click on the Next button to move on the next screen of the wizard and click on the Finish button to complete the wizard; here is what you will see in the data flow task:

  
Complete the Wizard

The Slowly Changing Dimension wizard will add several tasks to manage slowly changing dimension based on your selection and configuration. In the above screen the “Changing Attribute Updates Output” path will update records for SCD Type 1 (basically overwrites). The “New Output” path will add a new entry to the dimension for maintaining historical records and the “Historical Attribute Inserts Output” path will update previous records to be outdated by updating the Expiration date column.

Now when you execute the package for the first time you will notice two records from the above created source table will be loaded in the dimension table as you can see below:

  
The Dimension Table

Now execute this query to verify data in the Supplier dimension table:

|  |
| --- |
| USE [AdventureWorks2012]  GO  SELECT \* FROM [dbo].[DimSupplier]  GO |

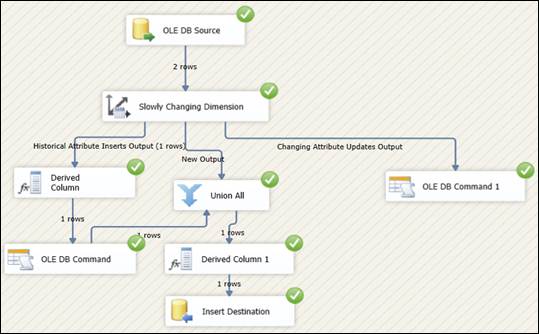
And this is what you will see after execution of the above query; it’s pretty much what we expected after first execution:

  
Results of executed query

Now let’s go back to source and update some records. With the below given script, I am updating Supplier name for SupplierCode = ‘S0000001’.

|  |
| --- |
| USE [AdventureWorks2012]  GO  UPDATE [dbo].[Supplier]  SET [SupplierName] = 'ABC Company Ltd.'  WHERE [SupplierCode] = 'S0000001'  GO  SELECT \* FROM [dbo].[Supplier]  GO |

Now execute the package again and this time you will notice one record (New) has been inserted and one record (old) has been updated or marked as outdated. This happens because the column for which the value has changed has been configured as SCD Type 2:

  
One record inserted and one record outdated

Now execute the query again and verify the data. As expected there are two records for SupplierCode = ‘S0000001’; the earlier one has been updated with an expiration date to indicate outdated record and another one as latest record with latest supplier name:

|  |
| --- |
| USE [AdventureWorks2012]  GO  SELECT \* FROM [dbo].[DimSupplier]  GO |

  
Query results

**Limitations**

The Slowly Changing Dimension transformation has been designed for ease of use and for smaller dimensions only. As we saw above, the Slowly Changing Dimension Transformation component is available out of the box in SSIS and can be quickly configured for smaller dimensions. Below are some of the reasons why Slowly Changing Dimension transformation is not fit for all scenarios especially if your dimension is large:

* The Slowly Changing Dimension transformation adds components to the data flow task based on your configuration to manage Slowly Changing Dimension. If you do any customization to any of these tasks and edit the Slowly Changing Dimension transformation again your customization gets lost.
* It can be very slow especially for larger dimensions as there is no caching of the lookup data.
* It can be used with SQL Server only.
* It uses OLEDB command for row by row updates instead of batch update.