DWNorthwindLite\_withSCD ETL Process

Author : Byungsu Jung

Date : 08/01/2018

**Introduction**

This document illustrates the extract, transform and load process works in DWNorthwindLite \_withSCD database. The incremental technique has been adapted for ETL process.

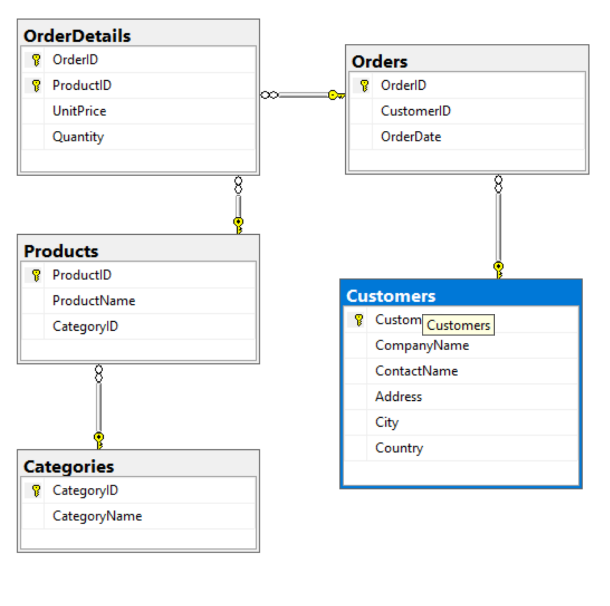
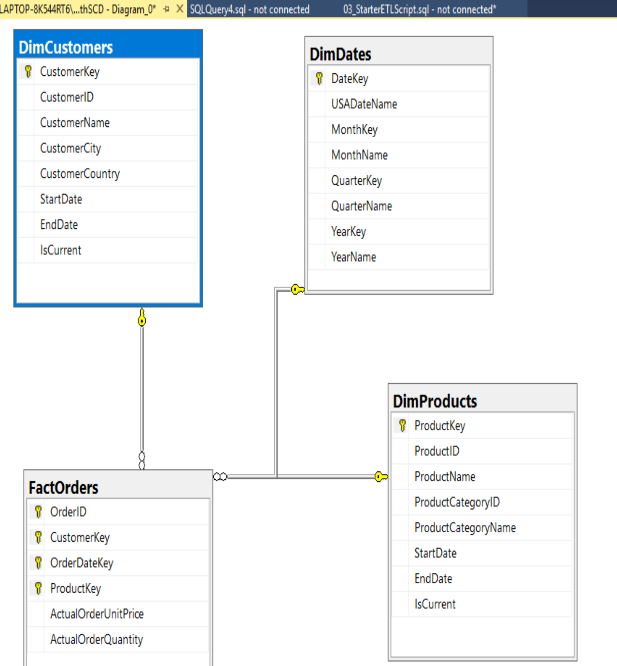
 

Figure01. ERD of NorthwindLite Figure02. ERD of DWNorthwindLite

**The following is an outline of the Incremental ETL process.**

* Unlike Flush and Fill technique, Incremental ETL process does not require dropping foreign key constraints and truncate process.
* Incrementally add in data into the tables in DWNothwindLite\_WithSCD.
  + Create views to each tables for transaction purpose:
    1. vETLDimProducts
    2. vETLDimCustomers
    3. vETLFactOrders.
  + Create transaction procedures to insert data into the tables in DWNorthwindLite\_WithSCD :
    1. pETLSyncDimProducts
    2. pETLSyncDimCustomers
    3. pETLFillDimDates
    4. pETLSyncFactOrders.
* Execute the procedures created to incrementally fill in the tables to synchronize with source tables.

1. **Fill in the tables in DWNothwindLite with data**

* **Create views to each tables for transaction purpose: vETLDimProducts, vETLDimCustomers and vETLFactOrders.**

Before filling the data into the tables, it is advised to create view to the tables. Creating the view table before filling the table allow the user to check properties of the data to be inserted before procedure. The abstract layer also gives greater ﬂexibility and lower maintenance costs. The view tables also work as an abstract layer to provide additional security protection.

During the creation of views for the tables, various casting and conversion of the data types to meet the data types of Dimensional tables is performed.

Create View vETLDimProducts

--Some Code--

SELECT

[CustomerID] = c.CustomerID,

[CustomerName] = CAST(c.CompanyName as nVarchar(100)),

[CustomerCity] = CAST(c.City as nVarchar(100)),

[CustomerCountry] = CAST(c.Country as nVarchar(100))

From NorthwindLite.dbo.Customers as c

**FIGURE03. Code to create view for DimProducts**

Create View vETLDimCustomers

--Some Code --

SELECT

[CustomerID] = c.CustomerID,

[CustomerName] = CAST(c.CompanyName as nVarchar(100)),

[CustomerCity] = CAST(c.City as nVarchar(100)),

[CustomerCountry] = CAST(c.Country as nVarchar(100))

From NorthwindLite.dbo.Customers as c

**FIGURE04. Code to create view for DimCustomers**

The view of FactOrders requires multiple joins to reach the right values for the columns because different column requires data from different tables including surrogate keys generated in dimensional tables. (FIGURE05)

In the case of [CustomerKey] column, Cast(Convert(nVarchar(50), o.OrderDate, 112) as int) code was used to convert [OrderDate], which is the datetime type, to the integer format that matches the [DateKey] in DimDates table.(FIGURE05)

Create View vETLFactOrders

--Some Code--

SELECT

[OrderID] = o.OrderID,

[CustomerKey] = dc.CustomerKey,

[OrderDateKey] = dd.DateKey,

[ProductKey] = p.ProductKey,

[ActualOrderUnitPrice] = od.UnitPrice,

[ActualOrderQuantity] = od.Quantity

From NorthwindLite.dbo.OrderDetails as od

Join NorthwindLite.dbo.Orders as o

On od.OrderID = o.OrderID

Join DWNorthwindLite.dbo.DimCustomers as dc

On o.CustomerID = dc.CustomerID

Join DWNorthwindLite.dbo.DimDates as dd

On Cast(Convert(nVarchar(50), o.OrderDate, 112) as int) = dd.DateKey

Join DWNorthwindLite.dbo.DimProducts as p

On od.ProductID = p.ProductID

**FIGURE05. Code to create view for FactOrders**

**Create transaction procedures to insert data into the tables in DWNorthwindLite : pETLSyncDimProducts, pETLSyncDimCustomers, pETLFillDimDates and pETLSyncFactOrders.**

The view tables created previously are used for inserting the data into Dimensional tables.

Two different techniques of incremental ETL process are used in this document.

One uses a Common Table Expression known as CTE. In this technique, we trasfer data to the dimension table by using a Select statement combined with EXCEPT. This technique finds difference between table from database and data warehouse to insert, update and delete the data which is not present in the data warehouse but present in the database. Both pETLSyncProducts and pETLSyncCustomers uses CTE technique to sync the DimProducts and DimCustomers tables.

Create Procedure pETLSyncDimProducts

--Some code--

With ChangedProducts

As(

Select ProductID, ProductName, ProductCategoryID, ProductCategoryName From vETLDimProducts

Except

Select ProductID, ProductName, ProductCategoryID, ProductCategoryName From DimProducts

Where IsCurrent = 1 -- Needed if the value is changed back to previous value

)UPDATE [DWNorthwindLite\_withSCD].dbo.DimProducts

SET EndDate = Cast(Convert(nvarchar(50), GetDate(), 112) as int)

,IsCurrent = 0

WHERE ProductID IN (Select ProductID From ChangedProducts)

;

-- 2)For INSERT or UPDATES: Add new rows to the table

With AddedORChangedProducts

As(

Select ProductID, ProductName, ProductCategoryID, ProductCategoryName From vETLDimProducts

Except

Select ProductID, ProductName, ProductCategoryID, ProductCategoryName From DimProducts

Where IsCurrent = 1 -- Needed if the value is changed back to previous value

)INSERT INTO [DWNorthwindLite\_withSCD].dbo.DimProducts

([ProductID],[ProductName],[ProductCategoryID],[ProductCategoryName],[StartDate],[EndDate],[IsCurrent])

SELECT

[ProductID]

,[ProductName]

,[ProductCategoryID]

,[ProductCategoryName]

,[StartDate] = Cast(Convert(nvarchar(50), GetDate(), 112) as int)

,[EndDate] = Null

,[IsCurrent] = 1

FROM vETLDimProducts

WHERE ProductID IN (Select ProductID From AddedORChangedProducts)

;

-- 3) For Delete: Change the IsCurrent status to zero

With DeletedProducts

As(

Select ProductID, ProductName, ProductCategoryID, ProductCategoryName From DimProducts

Where IsCurrent = 1 -- We do not care about row already marked zero!

Except

Select ProductID, ProductName, ProductCategoryID, ProductCategoryName From vETLDimProducts

)UPDATE [DWNorthwindLite\_withSCD].dbo.DimProducts

SET EndDate = Cast(Convert(nvarchar(50), GetDate(), 112) as int)

,IsCurrent = 0

WHERE ProductID IN (Select ProductID From DeletedProducts)

**FIGURE06. Code to create incremental ETL procedure for Dimproductss**

Create Procedure pETLSyncCustomers

With ChangedCustomers

As(

Select CustomerID, CustomerName, CustomerCity, CustomerCountry From vETLDimCustomers

Except

Select CustomerID, CustomerName, CustomerCity, CustomerCountry From DimCustomers

Where IsCurrent = 1 -- Needed if the value is changed back to previous value

)UPDATE [DWNorthwindLite\_withSCD].dbo.DimCustomers

SET EndDate = Cast(Convert(nvarchar(50), GetDate(), 112) as int)

,IsCurrent = 0

WHERE CustomerID IN (Select CustomerID From ChangedCustomers)

;

-- 2) For INSERT or UPDATES: Add new rows to the table

With AddedORChangedCustomers

As(

Select CustomerID, CustomerName, CustomerCity, CustomerCountry From vETLDimCustomers

Except

Select CustomerID, CustomerName, CustomerCity, CustomerCountry From DimCustomers

Where IsCurrent = 1 -- Needed if the value is changed back to previous value

)INSERT INTO [DWNorthwindLite\_withSCD].dbo.DimCustomers

([CustomerID],[CustomerName],[CustomerCity],[CustomerCountry],[StartDate],[EndDate],[IsCurrent])

SELECT

[CustomerID]

,[CustomerName]

,[CustomerCity]

,[CustomerCountry]

,[StartDate] = Cast(Convert(nvarchar(50), GetDate(), 112) as int)

,[EndDate] = Null

,[IsCurrent] = 1

FROM vETLDimCustomers

WHERE CustomerID IN (Select CustomerID From AddedORChangedCustomers)

;

-- 3) For Delete: Change the IsCurrent status to zero

With DeletedCustomers

As(

Select CustomerID, CustomerName, CustomerCity, CustomerCountry From DimCustomers

Where IsCurrent = 1 -- We do not care about row already marked zero!

Except

Select CustomerID, CustomerName, CustomerCity, CustomerCountry From vETLDimCustomers

)UPDATE [DWNorthwindLite\_withSCD].dbo.DimCustomers

SET EndDate = Cast(Convert(nvarchar(50), GetDate(), 112) as int)

,IsCurrent = 0

WHERE CustomerID IN (Select CustomerID From DeletedCustomers)

;

**FIGURE07. Code to create incremental ETL procedure for DimCustomers**

For DimDate ETL procedure, instead of using incremental ETL procedure, the regular Fill technique is used since data that goes into DimDates table does not change often. It stays the same most of the time. Also instead of using view to convert the data from source to meet the type and format of destination, the conversion process is directly used within the procedure. For each date between StartDate and EndDate, they are converted into different format and types to represent different measure of time such as year, date, quarter and month using while loop. By the end of while loop, Set @DateInProcess = DateAdd(d, 1, @DateInProcess) code is used to increment the date to reach the EndDate of the loop. Unlike other procedures, pETLFillDimDates require dropping foreign key constraint at the beginning and adding foreign key constraint at the end. There is no problem excuting pETLFillDimDates at first, but constraint error issue rises when one tries to run the procedure second time.

Create Procedure pETLFillDimDates --Some code—

Alter Table DwNorthwindLite\_withSCD.dbo.FactOrders

Drop Constraint fkFactOrdersToDimDates;

Declare @StartDate datetime = '01/01/1990'

Declare @EndDate datetime = '12/31/1999'

Declare @DateInProcess datetime = @StartDate

-- Loop through the dates until you reach the end date

While @DateInProcess <= @EndDate

Begin

Insert Into DimDates

( [DateKey], [USADateName], [MonthKey], [MonthName], [QuarterKey], [QuarterName],[YearKey], [YearName] )

Values (

Cast(Convert(nVarchar(50), @DateInProcess, 112) as int) -- [DateKey]

,DateName(weekday, @DateInProcess) + ', ' + Convert(nVarchar(50), @DateInProcess, 110) -- [DateName]

,Cast(Left(Convert(nVarchar(50), @DateInProcess, 112), 6) as int) -- [MonthKey]

,DateName(month, @DateInProcess) + ' - ' + DateName(YYYY,@DateInProcess) -- [MonthName]

,Cast(DateName(YYYY,@DateInProcess) + '0' + (DateName(quarter, @DateInProcess) ) as int) -- [QuarterKey]

,'Q' + DateName(quarter, @DateInProcess) + ' - ' + Cast( Year(@DateInProcess) as nVarchar(50) ) -- [QuarterName]

,Year(@DateInProcess) -- [YearKey]

,Cast(Year(@DateInProcess ) as nVarchar(50)) -- [YearName] )

-- Add a day and loop again

Set @DateInProcess = DateAdd(d, 1, @DateInProcess)

End

ALTER TABLE DWNorthwindLite\_withSCD.dbo.FactOrders

ADD CONSTRAINT fkFactOrdersToDimDates

FOREIGN KEY (OrderDateKey) REFERENCES DimDates(DateKey)

**FIGURE08. Code to create ETL fill procedure for DimDates**

In pETLSyncFactOrders, unlike the other procedures, new technique called merge is used for incremental ETL process. The procedure merge both source table and target table by primary keys. With merge, procedure provides appropriate work for ETL depending on the difference between target table and source table. If there is unmatched data, insert the data into target table. If it is matched, check to see if any one of columns in source table has been updated. If any data is updated in source table, update the same data at target table. Also, if there exist not matched data in source table that are present in target table, delete the data.

Create Procedure pETLSyncFactOrders

Merge Into FactOrders as TargetTable

Using vETLFactOrders as SourceTable

On TargetTable.OrderID = SourceTable.OrderID And

TargetTable.[CustomerKey] = SourceTable.[CustomerKey] And

TargetTable.[OrderDateKey] = SourceTable.[OrderDateKey] And

TargetTable.[ProductKey] = SourceTable.[ProductKey]

When Not Matched

Then

Insert

Values (SourceTable.OrderID, SourceTable.[CustomerKey], SourceTable.[OrderDateKey], SourceTable.[ProductKey], SourceTable.[ActualOrderUnitPrice], SourceTable.[ActualOrderQuantity])

When Matched

And SourceTable.[ActualOrderUnitPrice] <> TargetTable.[ActualOrderUnitPrice]

Or SourceTable.[ActualOrderQuantity] <> TargetTable.[ActualOrderQuantity]

Then

Update

Set TargetTable.[CustomerKey] = SourceTable.[CustomerKey],

TargetTable.[OrderDateKey] = SourceTable.[OrderDateKey],

TargetTable.[ProductKey] = SourceTable.[ProductKey],

TargetTable.[ActualOrderUnitPrice] = SourceTable.[ActualOrderUnitPrice],

TargetTable.[ActualOrderQuantity] = SourceTable.[ActualOrderQuantity]

When Not Matched By Source

Then

Delete

**FIGURE08. Code to create dimensional ETL procedure for FactOrders.**

**Summary**

For each table excluding DimDates, either CTE or Merge technique is used to create stored procedure which incrementally perform ETL process in DWNorthwindLite\_WithSCD.

The Flush&Fill ETL process that was used in previous task has number of different aspect from Incremental ETL process.

The Incremental ETL process is more efficient than Flush&Fill technique.

Incremental only update, insert and delete the data that is newly changed. However, flush&fill technique remove all the data first then fill in the whole data including both old and new data.