DWNorthwindLite ETL Process

Author : Byungsu Jung

Date : 07/28/2018

**Introduction**

This document illustrates the extract, transform and load process works in DWNorthwindLite database. The Flush&Fill technique has been adapted for ETL process.

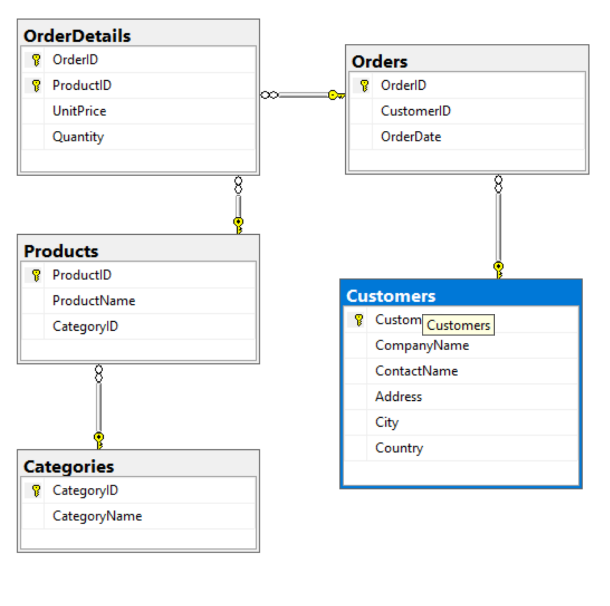
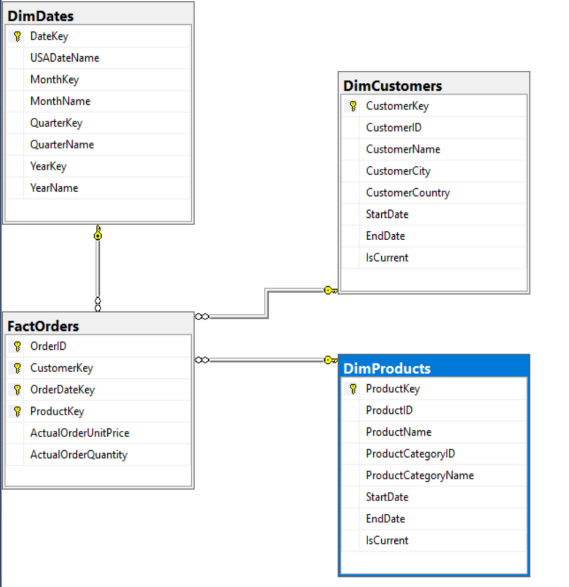
 

Figure01. ERD of NorthwindLite Figure02. ERD of DWNorthwindLite

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

* Drop Foreignkey constraints in DWNorthwindLite: pETLDropForeignKeyConstraints
* Truncate the tables in DWNorthwindLite: pETLTruncateTables
* Fill in the tables in DWNothwindLite with data
  + Create views to each tables for transaction purpose: vETLDimProducts, vETLDimCustomers and vETLFactOrders.
  + Create transaction procedures to insert data into the tables in DWNorthwindLite : pETLFillDimProducts, pETLFillDimCustomers, pETLFillDimDates and pETLFillFactOrders.
* Recreate Foreign key constraints : pETLAddForeignKeyConstraints

1. **Drop Foreignkey constraints in DWNorthwindLite**

In order to process the ETL process with flush&fill technique, foreign keys must be drop before flushing the data.

The transaction procedure called pETLDropForeignKeyConstraints, shown in figure01, drop all the foreign key constraints of the FactOrders table including fkFactOrdersToDimProducts, kFactOrdersToDimCustomers and fkFactOrdersToDimDates. Since the data in DimDates does not change often, it is optional to drop the foreign key constraint connecting FactOrders and DimDates.

Create Procedure pETLDropForeignKeyConstraints

--Some Coding --

Alter Table [DWNorthwindLite].dbo.FactOrders

Drop Constraint [fkFactOrdersToDimProducts];

Alter Table DWNorthwindLite.dbo.FactOrders

Drop Constraint fkFactOrdersToDimCustomers

-- Optional: Unlike the other tables DimDates does not change often --

Alter Table [DWNorthwindLite].dbo.FactOrders

Drop Constraint [fkFactOrdersToDimDates];

**FIGURE01. Code for dropping the foreign key constraints**

1. **Truncate the tables in DWNorthwindLite**

The truncate transaction procedure called pETLTruncateTables, as shown in the FIGURE02, flushes(erase) the data in each table including DimProducts, DimCustomers, FactOrders and DimDates.

Create Procedure pETLTruncateTables

--Some coding--

Truncate Table [DWNorthwindLite].dbo.DimProducts;

Truncate Table [dbo].[DimCustomers]

Truncate Table [dbo].[FactOrders]

-- Optional: Unlike the other tables DimDates does not change often --

Truncate Table [DWNorthwindLite].dbo.DimDates;

**FIGURE02. Code to truncate the 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 : pETLFillDimProducts, pETLFillDimCustomers, pETLFillDimDates and pETLFillFactOrders.**

The view tables created previously are used for inserting the data into Dimensional tables. For [StartDate], look up value -1 is used to indicate no data. For [EndDate], null is assigned as default. [IsCurrent] colum indicates whether the product is currently in product or not, and ‘Yes’ is assigned as a default.

Create Procedure pETLFillDimProducts

--Some code--

IF ((Select Count(\*) From DimProducts) = 0)

Begin

INSERT INTO [DWNorthwindLite].dbo.DimProducts

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

SELECT

[ProductID]

,[ProductName]

,[ProductCategoryID]

,[ProductCategoryName]

,[StartDate] = -1

,[EndDate] = Null -- Default

,[IsCurrent] = 'Yes' -- Default

FROM vETLDimProducts

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

Create Procedure pETLFillDimCustomers

--Some code--

Insert into DWNorthwindLite.dbo.DimCustomers

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

Select

[CustomerID],

[CustomerName],

[CustomerCity],

[CustomerCountry],

[StartDate] = -1,

[EndDate] = Null,

[IsCurrent] = 'Yes'

From vETLDimCustomers

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

For DimDate ETL procedure, 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.

Create Procedure pETLFillDimDates

--Some code--

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

-- Add a row into the date dimension table for this date

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)

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

Create Procedure pETLFillFactOrders

--Some Code--

Insert Into DWNorthwindLite.dbo.FactOrders

([OrderID], [CustomerKey], [OrderDateKey], [ProductKey], [ActualOrderUnitPrice], [ActualOrderQuantity])

Select

OrderID,

CustomerKey,

OrderDateKey,

ProductKey,

ActualOrderUnitPrice,

ActualOrderQuantity

From vETLFactOrders

**FIGURE09. Code to create ETL fill procedure for FactOrders**

1. **Recreate Foreign key constraints**

Once all the flush&fill procedure to insert the data is done, pETLAddForeignKeyConstraints is used to reassign foreign key constraints that were removed previously.

Create Procedure pETLAddForeignKeyConstraints

--Some code --

ALTER TABLE DWNorthwindLite.dbo.FactOrders

ADD CONSTRAINT fkFactOrdersToDimProducts

FOREIGN KEY (ProductKey) REFERENCES DimProducts(ProductKey);

ALTER TABLE DWNorthwindLite.dbo.FactOrders

ADD CONSTRAINT fkFactOrdersToDimCustomers

FOREIGN KEY (CustomerKey) REFERENCES DimCustomers(CustomerKey)

-- Optional: Unlike the other tables DimDates does not change often --

ALTER TABLE DWNorthwindLite.dbo.FactOrders

ADD CONSTRAINT fkFactOrdersToDimDates

FOREIGN KEY (OrderDateKey) REFERENCES DimDates(DateKey);

**FIGURE10. Code to re-create foreign key constraints.**

**Summary**

The whole procedure is broken down into 4 steps namely dropping foreign key constraint, flush, fill and reassigning foreign key constraints. The actual code to each procedure is included in the document.