Name: Girum Obse
Date: 01/27/2025
Course: BIDD 320 A /Data Migration Techniques (ETL Processing)
Assignment: A02_G.Obse

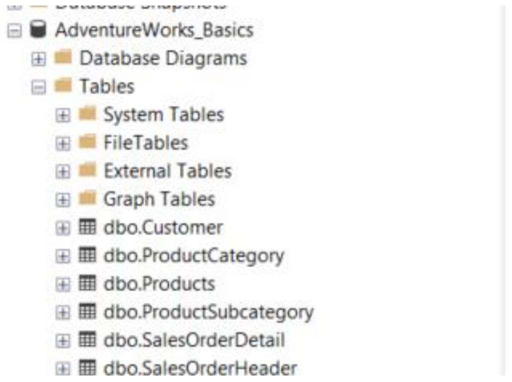
Incremental ETL Process: Technical Guide Manual

Introduction

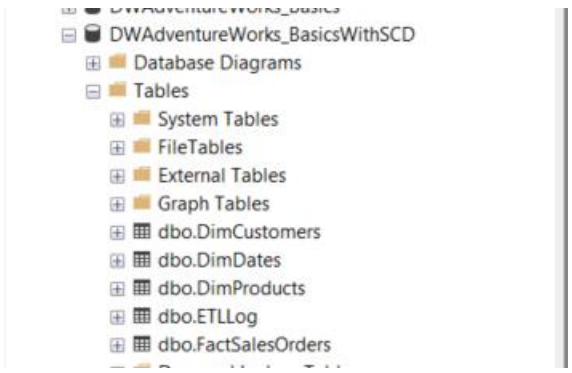
The purpose of this document is to provide an overview of our Incremental Extract, Transform, and Load (ETL) process. This guide aims to equip new hires with the knowledge necessary to understand the workflow, tools, and troubleshooting mechanisms employed in the ETL pipeline for the AdventureWorks_Basics database. ETL is an essential component in data warehousing, enabling us to extract data from source systems, transform it into a structured format, and load it into the Data Warehouse (DW). This document explains the process and outlines the techniques for effective maintenance and issue resolution.

ETL Process Overview

1. Database Setup
   - Source Database:AdventureWorks_Basics
   - Target Data Warehouse: DWAdventureWorks_BasicsWithSCD
   - The target DW includes the following tables:
     - Dimension Tables: DimCustomers, DimProducts, DimDates
     - Fact Table: FactOrders



Picture 1, source databse



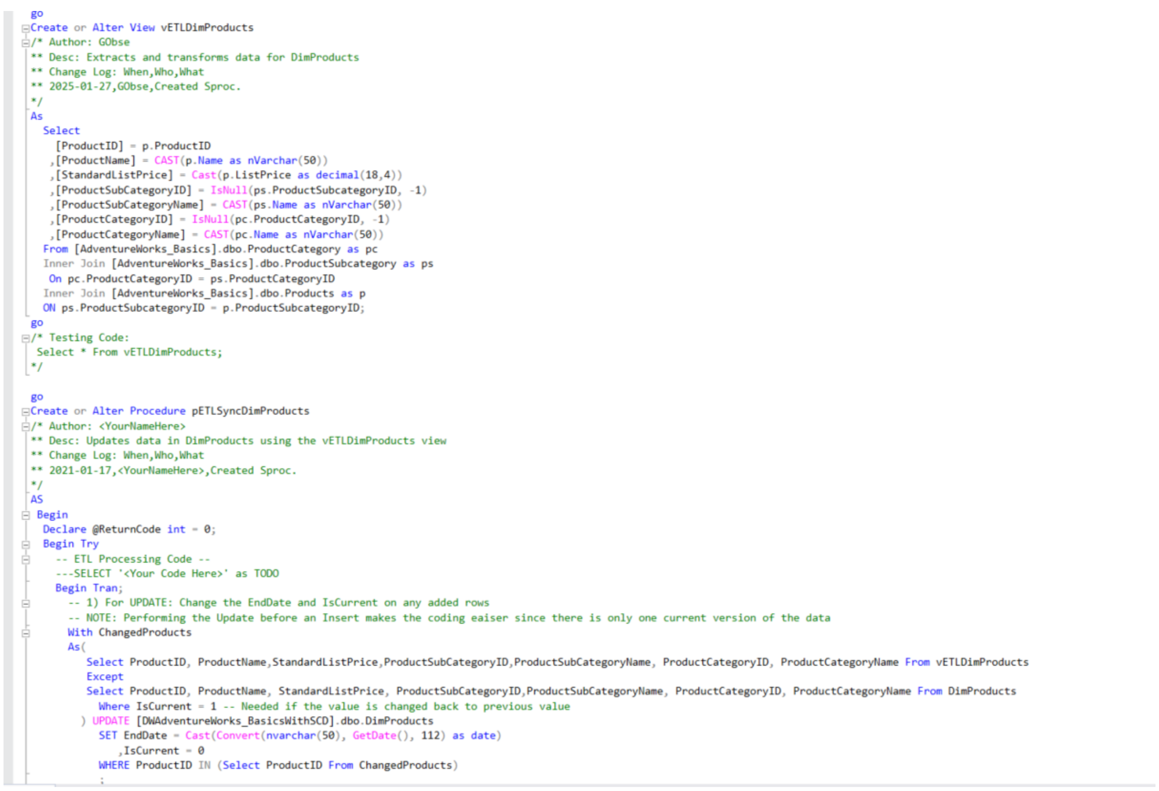
DWAdventureWorks_BasicsWithSCD
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.DimCustomers
dbo.DimDates
dbo.DimProducts
dbo.ETLLog
dbo.FactSalesOrders

Picture2, target database/datawarehouse

2. Incremental ETL Process

The incremental ETL approach minimizes resource usage by only processing data that has changed since the last load. Key features of our ETL implementation include:

- Change detection using comparison between source data and existing DW data.
- Log entries for ETL actions and errors.
- SCD Type 2 implementation for dimension tables to maintain historical data.



```sql
go
Create or Alter View vETLDimProducts
/* Author: GObse
** Desc: Extracts and transforms data for DimProducts
** Change Log: When,Who,What
** 2025-01-27,GObse,Created Sproc.
*/
As
  Select
    [ProductID] = p.ProductID
   ,[ProductName] = CAST(p.Name as nVarchar(50))
   ,[StandardListPrice] = Cast(p.ListPrice as decimal(18,4))
   ,[ProductSubCategoryID] = IsNull(ps.ProductSubcategoryID, -1)
   ,[ProductSubCategoryName] = CAST(ps.Name as nVarchar(50))
   ,[ProductCategoryID] = IsNull(pc.ProductCategoryID, -1)
   ,[ProductCategoryName] = CAST(pc.Name as nVarchar(50))
  From [AdventureWorks_Basics].dbo.ProductCategory as pc
  Inner Join [AdventureWorks_Basics].dbo.ProductSubcategory as ps
   On pc.ProductCategoryID = ps.ProductCategoryID
  Inner Join [AdventureWorks_Basics].dbo.Products as p
  ON ps.ProductSubcategoryID = p.ProductSubcategoryID;
go
/* Testing Code:
 Select * From vETLDimProducts;
*/

go
Create or Alter Procedure pETLSyncDimProducts
/* Author: <YourNameHere>
** Desc: Updates data in DimProducts using the vETLDimProducts view
** Change Log: When,Who,What
** 2021-01-17,<YourNameHere>,Created Sproc.
*/
AS
 Begin
  Declare @ReturnCode int = 0;
  Begin Try
    -- ETL Processing Code --
    ---SELECT '<Your Code Here>' as TODO
    Begin Tran;
      -- 1) For UPDATE: Change the EndDate and IsCurrent on any added rows
      -- NOTE: Performing the Update before an Insert makes the coding eaiser since there is only one current version of the data
      With ChangedProducts
      As(
         Select ProductID, ProductName,StandardListPrice,ProductSubCategoryID,ProductSubCategoryName, ProductCategoryID, ProductCategoryName From vETLDimProducts
         Except
         Select ProductID, ProductName, StandardListPrice, ProductSubCategoryID,ProductSubCategoryName, ProductCategoryID, ProductCategoryName From DimProducts
           Where IsCurrent = 1 -- Needed if the value is changed back to previous value
        ) UPDATE [DWAdventureWorks_BasicsWithSCD].dbo.DimProducts
           SET EndDate = Cast(Convert(nvarchar(50), GetDate(), 112) as date)
              ,IsCurrent = 0
           WHERE ProductID IN (Select ProductID From ChangedProducts)
           ;
```

Picture3: screenshot of the sql code showing the etl process to sync data from source databse to target datwarehouse by creating views and stored procedure

Detailed ETL Workflow

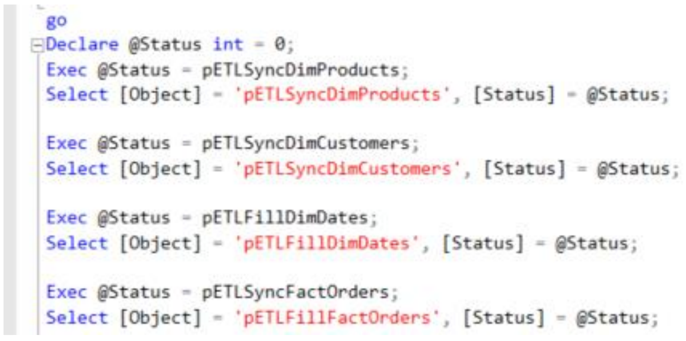
1. Logging Setup
- A dedicated logging table (`ETLLog`) is used to record metadata about ETL operations. This includes timestamps, actions performed, and error messages if any.
- Stored procedure `pInsETLLog` is employed for logging actions.

2. Dimension Tables Synchronization
- DimDates: Populated using the `pETLFillDimDates` procedure, which loops through a date range to generate date-related attributes.
- DimProducts:
  - Data is extracted and transformed using the `vETLDimProducts` view.
  - The `pETLSyncDimProducts` procedure detects changes and ensures the following:
    - Updates for modified rows (e.g., price changes).
    - Insertions for new products.
    - Soft deletion by marking old rows as inactive.
- DimCustomers:
  - Data is extracted using `vETLDimCustomers`.
  - Synchronization is handled by `pETLSyncDimCustomers` using a similar approach as `DimProducts`.

3. Fact Table Synchronization
- FactOrders:
  - Data is extracted via `vETLFactOrders`.
  - Synchronization ensures accurate aggregation of metrics and maintenance of transactional integrity.

4. ETL Error Handling
- The `Try-Catch` construct ensures that any errors encountered during ETL operations are logged in the `ETLLog` table.
- Transactions are either committed upon success or rolled back in case of failure, preserving data integrity.

Troubleshooting and Maintenance


```
go
Declare @Status int = 0;
Exec @Status = pETLSyncDimProducts;
Select [Object] = 'pETLSyncDimProducts', [Status] = @Status;

Exec @Status = pETLSyncDimCustomers;
Select [Object] = 'pETLSyncDimCustomers', [Status] = @Status;

Exec @Status = pETLFillDimDates;
Select [Object] = 'pETLFillDimDates', [Status] = @Status;

Exec @Status = pETLSyncFactOrders;
Select [Object] = 'pETLFillFactOrders', [Status] = @Status;
```

Picture4, showing the sql code that needs to be exuted to see the status of the ETL activity.

1. Common Issues and Solutions
- Missing Records in DW:
  - Verify the source data.
  - Check the synchronization views (e.g., `vETLDimProducts`).
- ETL Failures:
  - Review the `ETLLog` table for error messages.
  - Ensure that all dependent objects (tables, views, and procedures) are present and properly configured.
- Performance Bottlenecks:
  - Optimize queries within views and procedures.
  - Check indexing on key columns.

2. Monitoring Tools
- Use the `vETLLog` view to track ETL activity and identify patterns in failures or delays.
- Periodically review execution plans for stored procedures to detect potential inefficiencies.

Summary

This guide outlined the core components of our Incremental ETL process, focusing on the synchronization of dimension and fact tables within the DWAdventureWorks_BasicsWithSCD database. With robust logging and error-handling mechanisms, this process ensures data integrity and historical accuracy. By following the troubleshooting steps and maintenance tips, technicians can effectively manage and resolve issues encountered in the ETL pipeline.