Sara Black
10/19/2025
BIDD 310
Assignment 04

ETL and Stored Procedures

Intro

In this week's module, we learned about SQL based ETL and how to construct it inside of stored procedures. We had had some exposure to both of these concepts in previous modules, but this week we really did most of the heavy lifting by ourselves. One thing that was challenging about this was learning how to build complete transactions and logging in the stored procedures, as the readings were a little light on this element of the module. But when SQL Server executes and you get that little green checkmark, nothing better than that.

Clearing the DataWarehouse before a full load

One thing that was clear from the readings was that the most efficient way to clear existing tables is to truncate them, but that this can only be done without foreign key constraints. So before you can fill your tables you need to drop their foreign keys and truncate them

```
--**********************************************************************--
-- 1) Drop the Foreign Key CONSTRAINTS and Clear the tables
--**********************************************************************--
Go
Create Or Alter Proc pETLDropFks
--*************************************************************************
--
-- Desc:This Sproc drops the DW foreign keys.
-- Change Log: When,Who,What
-- 2020-01-01,RRoot,Created Sproc
-- Todo: 2025-10-24,SBlack,Added code to drop more FKs
--*************************************************************************
--
As
Begin
  Declare @RC int = 0;
  Begin Try
```

```
     Begin Tran;

      Alter Table dbo.FactEmployeeProjectHours Drop Constraint
[FK_FactEmployeeProjectHours_DimEmployees];
      Alter Table dbo.FactEmployeeProjectHours Drop Constraint
[FK_FactEmployeeProjectHours_DimProjects];
      Alter Table dbo.FactEmployeeProjectHours Drop Constraint
[FK_FactEmployeeProjectHours_DimDates];
      -- Alter Table [DWEmployeeProjects].dbo.FactEmployeeProjectHours Drop
Constraint [FK_FactEmployeeProjectHours_DimDates];
     Commit Tran;
     Exec pETLInsETLLog
            @ETLAction = 'pETLDropFks'
           ,@ETLLogMessage = 'Dropped Foreign Keys';
     Set @RC = 1;
  End Try
  Begin Catch
    If @@TRANCOUNT > 0 Rollback;
    Declare @ErrorMessage nvarchar(1000) = Error_Message()
      Exec pETLInsETLLog
          @ETLAction = 'pETLDropFks'
         ,@ETLLogMessage = @ErrorMessage;
    Set @RC = -1;
  End Catch
  Return @RC;
End
Go

Go
Create Or Alter Proc pETLTruncateTables
--*************************************************************************
--
-- Desc:This Sproc clears the data from all DW tables.
-- Change Log: When,Who,What
-- 2020-01-01,RRoot,Created Sproc
-- Todo: <Date>,<Name>,Completed code to clear all table data
--*************************************************************************
--
As
Begin
    Declare @RC int = 0;
  Begin Try
    Begin Tran;
```

```
      -- Todo: Clear FactEmployeeProjectHours;
      Truncate Table [DWEmployeeProjects].[dbo].[FactEmployeeProjectHours];
      -- Todo: Clear DimDates;
      Truncate Table [DWEmployeeProjects].[dbo].[DimDates];
      -- Todo: Clear DimEmployees;
      Truncate Table [DWEmployeeProjects].[dbo].[DimEmployees];
      -- Todo: Clear DimProjects;
      Truncate Table [DWEmployeeProjects].[dbo].[DimProjects];
    Commit Tran;
    Exec pETLInsETLLog
            @ETLAction = 'pETLTruncateTables'
           ,@ETLLogMessage = 'Truncated Tables';
    Set @RC = 1;
  End Try
  Begin Catch
    If @@TRANCOUNT > 0 Rollback;
    Declare @ErrorMessage nvarchar(1000) = Error_Message();
    Exec pETLInsETLLog
            @ETLAction = 'pETLTruncateTables'
           ,@ETLLogMessage = @ErrorMessage;
    Set @RC = -1;
  End Catch
  Return @RC;
End
Go
```

These Stored Procedures, `'pETLDropFks'` and `'pETLTruncateTables'` do just that.

Transforming and filling the DimEmployees Table

The pattern that we are using to fill these tables, is to create a view, ***extracting*** and ***transforming*** the relevant data, and then use the stored procedure to ***load*** the data from the view into our DW table. You can see that process on our **DWEmployeeProjects.dbo.DimEmployees** table here:

```
/****** [dbo].[DimEmployees] ******/
Go
Create Or Alter View vETLDimEmployees
As
```

```
    Select
      [EmployeeID] = ID
     ,[EmployeeName] = Cast((FName +  ' ' + LName) as varchar(100))
    From EmployeeProjects.dbo.Employees;
Go

Create Or Alter Proc pETLDimEmployees
--*************************************************************************
--
-- Desc:This Sproc fills the DimEmployees table.
-- Change Log: When,Who,What
-- 2020-01-01,RRoot,Created Sproc
-- Todo: 2025-10-24,SBlack,Completed code to fill the DimEmployees table.
--*************************************************************************
--
As
Begin
    Declare @RC int = 0;
    Declare @Message varchar(1000)
  Begin Try
      Begin Tran;
        -- Todo: Add Insert-Select Code
        Insert Into DimEmployees
        ( [EmployeeID], [EmployeeName] )
         Select
           EmployeeID
          ,EmployeeName
         From vETLDimEmployees;
        Set @Message = 'Filled DimEmployees (' + Cast(@@RowCount as
varchar(100)) + ' rows)';
      Commit Tran;
      Exec pETLInsETLLog
            @ETLAction = 'pETLDimEmployees'
           ,@ETLLogMessage = @Message;
    Set @RC = 1;
  End Try
  Begin Catch
    If @@TRANCOUNT > 0 Rollback;
    Declare @ErrorMessage nvarchar(1000) = Error_Message();
    Exec pETLInsETLLog
         @ETLAction = 'pETLDimEmployees'
        ,@ETLLogMessage = @ErrorMessage;
    Set @RC = -1;
  End Catch
  Return @RC;
End
Go
 Exec pETLDimEmployees; Select * From DimEmployees; Select * From ETLLog;
```

Transforming and filling the DimProjects Table

You can see the same pattern on our **DWEmployeeProjects.dbo.DimProjects** table here:

```
/****** [dbo].[DimProjects] ******/
Go
Create Or Alter View vETLDimProjects
As
    -- Todo: Complete Select Code
    Select
      [ProjectID] = ID
     ,[ProjectName] = Cast(Name as varchar(100))
    From EmployeeProjects.dbo.Projects;
Go

Create Or Alter Proc pETLDimProjects
--************************************************************************
--
-- Desc:This Sproc fills the DimProjects table.
-- Change Log: When,Who,What
-- 2020-01-01,RRoot,Created Sproc
-- Todo: 2025-10-24,SBlack,Completed code to fill the DimProjects table.
--************************************************************************
--
As
Begin
    Declare @RC int = 0;
    Declare @Message varchar(1000)
  Begin Try
      Begin Tran;
      -- Todo: Add Insert-Select Code
          Insert Into DimProjects
          ( [ProjectID], [ProjectName] )
         Select
```

```
           ProjectID
          ,ProjectName
         From vETLDimProjects;
        Set @Message = 'Filled DimProjects (' + Cast(@@RowCount as
varchar(100)) + ' rows)';
      Commit Tran;

      -- Todo: Add Logging Code
       Exec pETLInsETLLog
            @ETLAction = 'pETLDimProjects'
           ,@ETLLogMessage = @Message;
    Set @RC = 1;
  End Try
  Begin Catch
    If @@TRANCOUNT > 0 Rollback;
      -- Todo: Add Logging Code
    Declare @ErrorMessage nvarchar(1000) = Error_Message();
    Exec pETLInsETLLog
         @ETLAction = 'pETLDimProjects'
        ,@ETLLogMessage = @ErrorMessage;
   Set @RC = -1;
  End Catch
  Return @RC;
End
Go
 Exec pETLDimProjects; Select * From DimProjects; Select * From ETLLog;
```

Transforming and filling the FactEmployeeProjectHours Table

The pattern is mostly repeated on our **DWEmployeeProjects.dbo.FactEmployeeProjectHours** table as well. But there is an interesting wrinkle here where we are preserving the **Surrogate Keys.** We Inner Join in all of our Dimension tables when we create the view and extract and transform the data. This allows us to absorb changes to the source DB without having to remap all of the relationships in the DW.

```
/****** [dbo].[FactEmployeeProjectHours] ******/
Go
Create Or Alter View vETLFactEmployeeProjectHours
As
```

```
  -- Todo: Complete Select Code (Don't forget to return the Surrogate
Keys!)
    Select
       [EmployeeProjectHoursID] = EmployeeProjectHoursID
      ,[EmployeeKey] =  de.EmployeeKey
      ,[ProjectKey] = dp.ProjectKey
      ,[DateKey] =  dd.DateKey
      ,[HoursWorked] = eph.Hrs
    From EmployeeProjects.dbo.EmployeeProjectHours as eph
    Inner Join [DWEmployeeProjects].[dbo].[DimProjects] as dp
        On [eph].[ProjectID] = [dp].[ProjectID]
    Inner Join [DWEmployeeProjects].[dbo].[DimEmployees] as de
        On [eph].[EmployeeID] = [de].[EmployeeID]
    Inner Join [DWEmployeeProjects].[dbo].[DimDates] as dd
        On Cast(eph.Date as date) = dd.FullDate
Go

Create Or Alter Proc pETLFactEmployeeProjectHours
--*************************************************************************
--
-- Desc:This Sproc fills the FactEmployeeProjectHours table.
-- Change Log: When,Who,What
-- 2020-01-01,RRoot,Created Sproc
-- Todo: 2025-10-25,SBlack,Completed code to fill the
FactEmployeeProjectHours table.
--*************************************************************************
--
As
Begin
  Declare @RC int = 1;
  Declare @Message varchar(1000);
  Begin Try
      -- Todo: Add Transaction, Insert-Select, and Logging code
      Begin Tran;
      -- Todo: Add Insert-Select Code
          Insert Into FactEmployeeProjectHours
          ( [EmployeeProjectHoursID], [EmployeeKey], [ProjectKey],
[DateKey], [HoursWorked] )
         Select
           [EmployeeProjectHoursID]
           ,[EmployeeKey]
           ,[ProjectKey]
           ,[DateKey]
           ,[HoursWorked]
         From vETLFactEmployeeProjectHours;
        Set @Message = 'Filled FactEmployeeProjectHours (' +
Cast(@@RowCount as varchar(100)) + ' rows)';
      Commit Tran;
      Exec pETLInsETLLog
         @ETLAction = 'pETLFactEmployeeProjectHours'
        ,@ETLLogMessage = @Message;
    Set @RC = 1
  End Try
  Begin Catch
    -- Todo: Add Transaction and Logging code
        If @@TRANCOUNT > 0 Rollback;
      -- Todo: Add Logging Code
        Declare @ErrorMessage nvarchar(1000) = Error_Message();
        Exec pETLInsETLLog
         @ETLAction = 'pETLFactEmployeeProjectHours'
        ,@ETLLogMessage = @ErrorMessage;
    Set @RC = -1;
  End Catch
  Return @RC;
End
Go
```

Replacing the Foreign Keys

Finally, now that our tables are loaded, we need to replace the foreign keys that we dropped at the beginning of this script to maintain the relationships between the Fact and Dimension tables.

```
--**********************************************************************--
-- 3) Re-Create the Foreign Key Constraints
--**********************************************************************--
Go
Create Or Alter Proc pETLReplaceFks
--**************************************************************************
--
-- Desc:This Sproc replaces the DW foreign keys.
-- Change Log: When,Who,What
-- 2020-01-01,RRoot,Created Sproc
-- Todo: <Date>,<Name>,Added code to replace more FKs
--***************************************************************************
--
As
Begin
  -- Todo: Add FK and Logging Code

  Declare @RC int = 1;
  Begin Try
    Begin Tran;
     Alter Table FactEmployeeProjectHours
  Add Constraint FK_FactEmployeeProjectHours_DimEmployees
    Foreign Key(EmployeeKey) References DimEmployees(EmployeeKey);


Alter Table FactEmployeeProjectHours
  Add Constraint FK_FactEmployeeProjectHours_DimProjects
    Foreign Key(ProjectKey) References DimProjects(ProjectKey);


Alter Table FactEmployeeProjectHours
  Add Constraint FK_FactEmployeeProjectHours_DimDates
    Foreign Key(DateKey) References DimDates(DateKey);

    Commit Tran;
    Exec pETLInsETLLog
         @ETLAction = 'pETLReplaceFks'
        ,@ETLLogMessage = 'Replaced Foreign Keys';
    Set @RC = 1;
  End Try
  Begin Catch
      If @@TRANCOUNT > 0 Rollback;
    Declare @ErrorMessage nvarchar(1000) = Error_Message();
        Exec pETLInsETLLog
         @ETLAction = 'pETLReplaceFks'
        ,@ETLLogMessage = @ErrorMessage;
    Set @RC = -1;
  End Catch
  Return @RC;
End
Go
```

Summary

So thats our test drive of SQL ETL and Stored Procedures. It all makes sense when you know what you're doing! I am certain that in the next module we will have even more complexity added to this part of our Data Warehousing toolkit. See you then!