Creating Star Schema

An important process typically required for ETL is populating the Star Schema. The following article describes how Elixir Repertoire Composite Datasource can be used to implement the SCD1 process and create the fact table. This example uses MySQL as the RDBMS.

Database Preparation

1. Download the following - http://www.elixirtech.com/~adrian/ETL/ElixirSamples.jar. Unzip the content to your ElixirSamples Repository. In this repository, there is an "ETL" folder.
2. This sample creates a MySQL database call *foodmart-dev*, schema name called *foodmart-dev* with the username/password : *foodmart/foodmart* given full rights and access.
3. The details of the connection to the database can be found at (**repository:/ElixirSamples/ETL/Connection Pool/Foodmart.pool**). Ensure that the database connection to MySQL is successful before proceeding.
4. The table schemas need to be created initially. The SQLs to create the tables can be found at **repository:/ElixirSamples/ETL/Load/Resources**. You can use your favorite DB tools to create the schemas. (e.g MySQL Administrator etc).
   - repository:/ElixirSamples/ETL/Load/Resources/sqlCreateTableCustomer.txt
   - repository:/ElixirSamples/ETL/Load/Resources/sqlCreateTableStore.txt
   - repository:/ElixirSamples/ETL/Load/Resources/sqlCreateTableTimeByDay.txt
   - repository:/ElixirSamples/ETL/Load/Resources/sqlCreateSalesFact1997.txt
5. Once step 4 is completed, you should verify that the tables are successfully created and the necessary PK/FK created.

Dimension Table Creation

1. A star schema implementation begins with a series of Dimensional Tables. In this sample, we will be implementing a SCD1 process for Dim_Customer. We will do a direct process the rest of the Dimensions.

3. The cleansed data is found at : **repository:/ElixirSamples/ETL/Extraction/data**
   - repository:/ElixirSamples/ETL/Extraction/data/Customers-org.csv (This is the data to be initially inserted to the database) .
   - repository:/ElixirSamples/ETL/Extraction/data/Customers-insert.csv (This is the data to be used to simulate the insertion process)
   - repository:/ElixirSamples/ETL/Extraction/data/Customers-update.csv (This is the data to be used to simulate the update process)
4. We will use ”Customers-org.csv" to initially populate *Dim_Customer*. Send the repository link through the parameters and populate *Dim_Customer*. Call the store "Perform Insertion for new records" under (**repository:/ElixirSamples/ETL/Load/Dim_Customer/datasources/comDimCustomer_Builder.ds**) and 10281 records should be written to the database.

5. To update *Dim_Customer*, we will simulate by sending different CSV (Customers-insert.csv & Customers-update.csv) to *comDimCustomer_Builder.ds*. You will notice that the composite datasource handles both insert and update scenario. See detailed explanation in the annotation notes.
6. Dim_Store : **repository:/ElixirSamples/ETL/Load/Dim_Store/datasources/comDimStore_Builder.ds**
7. Dim_Time : **repository:/ElixirSamples/ETL/Load/Dim_Time/datasources/comDimTime_Builder.ds**
8. This completes the population of all the dimensional tables.

Populating the Fact Table

1. Elixir Repertoire implements row-based population of Fact Table. See (**repository:/ElixirSamples/ETL/Load/Fact_Sales_1997/datasources/comFactSales1997_Builder.ds**)
   - In this samples, you see that a series of Join Processors is used to transform the values to the surrogate keys before sending to the fact table.

Ad Hoc Cube

1. A sample of the configuration of the Ad Hoc Cube is also attached. Note that you will need the Ad Hoc License to be able to use the Ad Hoc Cube sample attached.