ETL based on TPC-DI - Part I: Initial loading

Saturday, May 1, 2021
12:00 PM

Scope: http://www.tpc.org/tpcdi/ The benchmark has its own data generators. But we have generated sample data for you. You will find the link towards the end of this document.

Problem

A necessary preparatory step for any data-analytics project is onboarding of the data to be analyzed. Usually, this data does not originate from a single source. They are usually collected from different sources under different formats. The process of onboarding is usually known as extract E) Transform T) Load L) (ETL) or more recently as data integration DI. On 2014, TPC has released TPC-DI as a benchmark to compare different offerings of ETL tools.

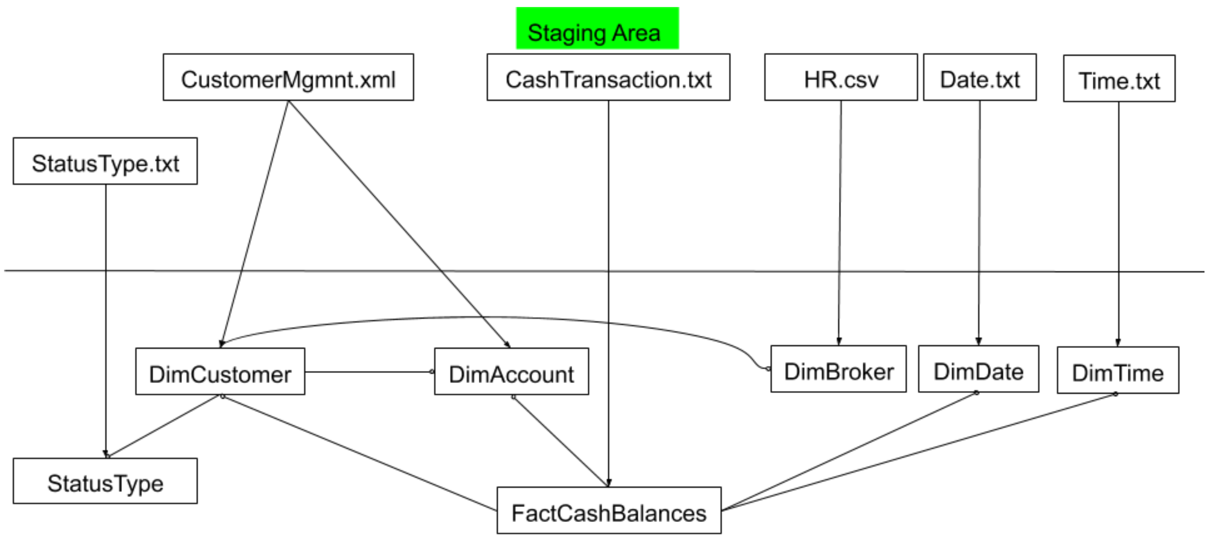
DI/ETL processes can be subdivided into either loading historical data or performing incremental updates by bringing data from change data capture. With the latter, the data should be verified to be compliant with the targeted data warehouse schema. The schema of the warehouse is usually called star-schema. Which is divided into fact and dimensions tables. Imagine a warehouse to track orders in an e-commerce system, a fact table stores the sales by day/month/year/quarter/department/store etc. Each one of those is called a dimension and the table that joins them all and stores the actual amount of orders at this level of granularity is called the sales fact table.

To populate the warehouse, the data from the original transactional processing systems (TPS) have to be frequently collected, cleansed, de-normalized and then stored in the warehouse.

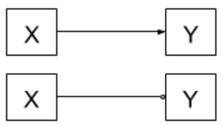
In the context of big data, current systems like Spark/Hive/Flink can serve the purpose of performing ETL operations. In this project, we need to take just once case from the TPC-DI warehouse schema which relates to Cash balance fact table.

For this we have the following dimension: Account, Customer, Date, and Transaction. Details about the schema of those dimensions can be found in the TPC-DI document along with explanation of the source tables. We shall give you the datasets of the original transactional tables. The Figure below shows the relations between data in the staging area, that came out of the generator (the data you will get) and the targeted tables.

More details about the files and transformation requirements are in the original TPC-DI document file, pointed to at the beginning of this document.



Data Warehouse Star Schema Cash Balances



X populates Y. The data in X in the staging area is used to fill Y in DW area

X depends on Y. X has a reference to the key of Y. Both in the DW area

Accounts Dimension

Accounts Source: For the historical load, the input data is found in CustomerMgmnt.xml as pointed to in the diagram above. Account data is stored as a contained element to the related Action and Customer elements. The possible ActionType values are

NEW: A new customer. A new customer is always created with 1 or more new accounts.

ADDACCT: One or more new accounts for an existing customer.

UPDACCT: Updates to the information in one or more existing accounts.

UPDCUST: One or more updates to an existing customer. When updating customer information, no account information is supplied. Note that since DimAccount contains a surrogate key to DimCustomer, a change to a customer record will require a history-tracking change to update the SK_CustomerID field of the current associated account records in DimAccount as described in clause 4.4.1 (the original specification).

CLOSEACCT: Close one or more existing accounts as described in 4.5.1.3

INACT Make an existing customer and that customer's currently active accounts inactive. No specific account information is supplied in the Source Data.

You can find the detailed transformations in the original document from page 58-71. You can ignore transformations for fact and dimension tables not mentioned in the figure above.

The source data for this dimension are given in an XML file. For this, it is better to use Spark RDDs and read the file as whole. So that each row in the RDD represents a complete XML file. Then, you will need to create a user-defined function to parse the XML file and extract the required columns for the dimension table.

NOTE: you may want to install the https://docs.python.org/2/library/xml.etree.elementtree.html library to help parse the XML file.

You will also need to impose schema on the generated frame when converting the RDD into a DataFrame.

Customer Dimension

DimCustomer data is obtained from the data file CustomerMgmt.xml. Customer data is stored as a sub-element to the related Action element. Every Action will have a related Customer, but not all actions require modifying the DimCustomer table. The possible ActionTypes are

NEW A new customer. A new customer is always created with 1 or more new accounts.

ADDACCT One or more new accounts for an existing customer. This does not require any change to DimCustomer.

UPDACCT Updates to the information in one or more existing accounts. No change to DimCustomer is required.

UPDCUST One or more updates to existing customer's information. Only the identifying data and updated property values are supplied in the Source Data.

CLOSEACCT Close one or more existing accounts. This does not require any change to DimCustomer.

INACT Make an existing customer and that customer's currently active accounts inactive. No specific account information is supplied in the Source Data.

Date Dimension

DimDate is a static table: It is loaded from the Date.txt. all rows and columns of the Date.txt file must be loaded into the corresponding columns of the DimDate table, with no modifications.

Time Dimension

DimTime is a static table: It is loaded from a file once in the Historical Load and not modified again. During the Historical Load, all rows and columns of the Time.txt file must be loaded into the corresponding columns of the DimTime table, with no modifications.

Broker Dimension

Data for DimBroker comes from the HR extract file HR.csv. Those employees from the HR file that are brokers (as indicated by the EmployeeJobCode) will have data copied to the Broker table. The Broker table is structured as a history tracking dimension table. However, nothing in the input file will provide any history of changes over time; it is simply a snapshot of the current state of the HR data.

Cash Balance Fact Table

The schema of the fact table can be found in the original pdf documentation Table 3.2.12 on page 43.

FactCashBalances data is obtained from the data file CashTransaction.txt. The net effect of all cash transactions for a given account on a given day is totaled, and only a single record is generated per account that had changes per day.

When populating fields of the FactCashBalances table:

- SK_CustomerID and SK_AccountID are obtained from DimAccount by matching CT_CA_ID

with AccountID, where IsCurrent = 1.

- SK_DateID is set to the DimDate SK_DateID field that corresponds to the **Batch Date**.
- Cash is calculated as the sum of the prior Cash amount for this account plus the sum of all
- CT_AMT values from all transactions in this account on this day. If there is no previous FactCashBalances record for the associated account, zero is used.
- Set BatchID to 1.

General Note: you have to read the original TPC-DI pdf documentation to get a better understanding of the tasks and the relationships between source and target tables.

The following tables sections are useful in the original documentation Table 4.3 (page 53) in the original TPC-DI document.

In this stage, you assume that Spark (simulating a DWH) is empty and all the fact and dimension tables may not exist. For this, you will use the first batch of the data, obtained from the data generator or from the link to the dataset mentioned below.

You have to make necessary SQL transformations on the data sources and derive the target fact and dimension tables as per the transformations and explanation in the figure above and in TPC-DI documentation.

After you have created all the data frames for the fact and dimension tables, you need to store those in an appropriate format and then in a separate job load those tables and perform queries like

- **Get the sum of transactions per customer for the first quarter grouped by year**
- **Get max transaction amount per quarter**
- **For a specific customer ID, get the total amounts of transactions per week.**

Suggested steps:

1. Use benchmarks' data generators to prepare the staged data ready for transformation and loading to the warehouse
2. If step 1 is hard for you, we have already generated some data. You can find it at this link https://nileuniversity-my.sharepoint.com/:u:/g/personal/aawad_nu_edu_eg/EaxV7FE3RqBOjJlUg2RMRZUB2YDVboUUoAKkFXJ4_7NyKw?e=rTAxPF
3. You can read the full description of the dimensions and fact tables described above in the benchmark specification
3. For this phase of the project, you will need the data in Batch 1 copy folder to initialize the data warehouse. For the next phase of the project, you can use Batch 2.
4. Study the 18 different transformations and apply them using SQL first on Spark. If not easy to do, we have to fall back to RDD transformation APIs.
5. You’re advised to use HIVE with ACID or Spark Delta Lake.