Unit 9 Notes

Designing and Developing the ETL System

Overview

Kimball proposes a 10-step plan for developing ETL systems that are either hand-coded or based on ETL tools (that is, his plan is independent of the architecture). The prerequisites for development consist of the logical design, high-level architecture plan, and source-to-target mapping (STM) for all data items. The physical design and implementation should also be under way. It is essential that the data be profiled prior to starting ETL development.

With all prerequisites in place, the ETL team draws a high-level plan as exemplified in Figure 10-1 on pp 428, and then chooses an ETL tool that best meets their needs. The team then develops a strategy for the main ETL tasks. After this step, they start working out the transformations that are needed in order to populate each individual target table. The STM and data profiling reports are incorporated in the ETL specification document described on pp 435-436. Next, we populate the dimension tables with historical data, and then we load the fact table with the historical data too. The latter process can take several days to complete. Once all tables are loaded with historic data, we start their incremental processing, first the dimension tables and then the fact tables. This processing relies on a stream of rows (ideally offered by the source system) that have been inserted, modified, or deleted since the last load. The ETL system will apply these transformations, usually once a day. The next step builds the aggregate table from the fact table. Finally, the last step sets the ETL system to operate automatically. This involves, among other things, job scheduling, error handling, and DB maintenance operations.

The members of a typical ETL team and their involvement in all steps of ETL design and development are listed in the Task List on pp 472. The next sections detail some of the steps and aspects of this process.

STM and Staging Tables

ETL systems employ staging tables to temporarily store versions of the data as they are incrementally transformed into their final form that is stored in the DW. Staging tables should be created in a separate DB or schema, which can be moved onto a separate server to support a high ETL load. Separate staging tables also keep the data model lean, and allow for simpler incremental transformations. The following table illustrates the STM from the source system to the staging table:

No	Source system	Source table	Source attribute	Transformation logic	Target table	Target attribute
1	System1	TableA	Manager_name	N/A	STG_TableA	Manager_name

We notice that loading the source data into the staging table is a relatively straightforward process, as little or no transformation occurs at this time. New data either overwrite or are appended to existing data in the staging table. By comparison, loading the data from the staging tables into the DW is a more involved process as most transformations (e.g., splitting, merging, de-duplication, string normalization, matching, formatting) occur at this time. The following table illustrates the corresponding STM:

No	Source system	Source table	Source attribute	Transformation logic	Target table	Target attribute
1	Staging	STG_TableA	Manager_name	Parse name using a comma separator	DIM_SALES	MGR_FIRST_NAME

Testing

There are various types of testing approaches to validate the data movements across DW layers. Newer versions of the ETL tools provide more meaningful execution logs to validate the data that were brought in (the input) and the data that was actually loaded into the target layer. The log information provided by the tools can be used for testing. A simple way of testing is to compare the record counts at the source layer against the record counts at the target layer. Another approach to testing while loading, for example, sales information at stores, is to compare the dollar value of sales at each store in the source layer against the summarized dollar value of sales at each store in the target layer. It is a good approach to develop a testing strategy and a supporting set of test scripts and reuse the set for multiple data load processes with minimal modifications. This approach improves the efficiency and consistency of the testing process.

Error Handling

Once all ETL processes that load the individual tables have been developed, these processes need to be organized in workflows or sequences that automatically load the various dimensions and fact tables in the preferred sequence/order. ETL tools have capabilities to design/implement conditional flows while loading various database components. When an error is encountered, the ETL process can be configured to abort the load, continue execution, or skip the data record that caused the error and move on to the next record. ETL tools also allow us to set up thresholds so that processes can continue execution until a specific number of errors are encountered.

Some loading processes can be resumed from the point of failure; others must be restarted from scratch. For instance, assume that the sales data from stores have to be loaded every day over night, for business users to evaluate store performances and further assume that it takes about 6 hours to load the dimension and fact tables. If the date of a specific sale is in the wrong format causing the load to fail, it could make sense to skip that one record and load the remaining records. The error record can be logged into a separate table for further investigation. Once the erroneous record is addressed, the record can be loaded relatively quickly.

We should analyze all failure scenarios and options to recover. Find all places where the loading process could fail, and set up an error handling procedure for every failure.