

Kimball Design Tip #9: Processing Slowly Changing Dimensions During Initial Data Load: A Pragmatic Compromise

By Lawrence Corr

Last month's tip clearly defined the type 2 slowly changing dimension technique and the proper use of surrogate keys. This month we tackle the vexing issue of handling slowly changing dimensions during the initial load of a new subject area within a data warehouse. This would occur when you have brought a new measurement (fact) source into an existing data warehouse. Dimensions like product, customer, and time are probably already defined and have a rich history reflecting many "slow changes".

First I'll describe the regular ETL (extract-transform-load) processing that would happen each night:

Dimensions are processed first, any new records from the production source are inserted into the dimension tables and are assigned the next surrogate key in sequence. Existing dimension records that have changed since the last load of the warehouse are detected and the nature of their change examined. An established slowly changing dimension policy will decide, based on which fields have changed, whether the current dimension record should be destructively updated and the old values lost (type 1) or that a new dimension record possessing the same natural ID should be created using the next surrogate key in sequence (type 2).

Following all these intricate comparisons and decisions on each dimension, it is time to face the bulk load of the facts. Here speed is of the essence. The natural IDs must be stripped off the fact records and replaced with the correct surrogate keys as rapidly as possible. Ideally we want to take advantage of the in-memory lookup processing offered by the majority of modern ETL tools. Small lookup tables for each dimension that translate natural IDs to surrogate keys are built from the dimension tables in the warehouse RDBMS using statements similar to the ones below:

```
Select customer_ID, max(customer_key) from customer  
group by customer_ID
```

or

```
Select customer_ID, customer_key from customer  
Where current = 'Y'
```

By doing so, the lookup tables will contain the surrogate keys that match the new facts and will support rapid hash lookups.

However, this simple and efficient lookup technique will not be sufficient if we are loading a significant span of historical facts in this first load. For example, imagine the situation where we are initially loading two years worth of transactions and we are 'blessed' with having the same two years of customer master file history. The two may never have met on a report before but we wish to match them precisely in this new data mart and perfectly partition history.

The dimension lookup would have to contain the historic surrogate keys for each customer detail change during the two year time period. The simple hash lookup comparable to the SQL:

```
Select customer_key from customer_lookup CL
where CL.customer_ID = factfile.customer_id
```

must be replaced with:

```
Select customer_key from customer_lookup CL
where customer_ID = factfile.customer_id and factfile.transaction_date between
CL.effective_date and CL.end_date
```

The 'between date logic' required here will slow the processing of a several hundred million row fact load with ten such dimensions to a crawl. Added to this, you now have two very different fact load applications. If you ever get this initial load to run to completion, you have to throw it away and then build, test and maintain the simpler version that will incrementally load the fact table from now on.

While being something of a compromise, the answer to this problem has an appealing simplicity. Don't build the complex bulk job, only build the incremental job! This technique provides a far truer version of the past than simply attaching all historic facts to the current dimension records and vowing to slowly change from now on, which is often the strategy adopted when faced with the initial bulk load. Here's what I suggest you do:

Take the two-year load and break it into one hundred and four jobs each representing one week's worth of transactions and run them sequentially in rapid succession. Begin each job by loading only the dimensional history relevant for that week, i.e., where the dimension detail effective_dates are less than or equal to the minimum transaction date. While this means a little additional logic in the dimension maintenance stages, the fact load can run unaltered. It can do so because the simple lookup tables contain the maximum surrogate keys applicable for the load period. You have just run 104 incremental loads. The compromise is that some fact records will be attached to dimension records that are up to a week out of date. In most cases this is a small margin of error because the dimensions are changing slowly.

The techniques for maintaining dimensions and building lookup tables are described in more detail in Ralph's June 1998 Data Warehouse Architect article "[Pipelining Your Surrogates](#)".