Design Tip #137 Creating and Managing Shrunken Dimensions

By Warren
Thornthwaite

This Design Tip continues my series on implementing common ETL design patterns. These techniques should prove valuable to all ETL system developers, and, we hope, provide some product feature guidance for ETL software companies as well.

Recall that a shrunken dimension is a subset of a dimension's attributes that apply to a higher level of summary. For example, a Month dimension would be a shrunken dimension of the Date dimension. The Month dimension could be connected to a forecast fact table whose grain is at the monthly level.

I recently ran across a good example of the need for a shrunken dimension from a Kimball Group enthusiast who works for a company that manages shopping mall properties. They capture some facts at the store level such as rent payments, and other facts at the overall property level such as shopper traffic and utility costs. Remember, a fundamental design goal is to capture data at the lowest grain possible. In this case, we would first attempt to allocate the property level data down to the store level. However, the company in question felt some of the property data could not be sensibly allocated to the store level; therefore they needed fact tables at both the store and property levels. This means they also needed dimensions at the store and property level.

Create the base dimension
There are many ways to create a shrunken dimension, depending on how the data is structured in the source system. The easiest way is to create the base dimension first. In this case, build the Store dimension by extracting store and property level natural keys and attributes from the source, assigning surrogate keys and tracking changes to important attributes with Type 2 change tracking.

The Store dimension will have several property level attributes including the property's natural key because users will want to roll up store facts by property descriptions. They will also ask questions that only involve the relationship between store and property, such as "What is the average number of stores per property?"

Create the shrunken dimension from the base dimension
Once the lowest level dimension is in place, creating the initial shrunken dimension, in this case the Property dimension, is essentially the same as creating the mini-dimension we described in Design Tip #127. Identify the attributes you want to extract from the base dimension and create a new table with a surrogate key column. Populate the table using a SELECT DISTINCT of the columns from the base dimension along with an IDENTITY field or SEQUENCE to create the surrogate key. In the property example, the following SQL would get you started:

```
INSERT INTO Dim_Property
SELECT DISTINCT Property_Name, Property_Type, Property_SqFt, MIN(Effective_Date), MAX(End_Date)
FROM Dim_Store
GROUP BY Property_Name, Property_Type, Property_SqFt;
```

The incremental processing is a bit more challenging. The easiest approach if you are working from an existing base dimension as we've describe is to use the brute force method. Create a temporary shrunken dimension by applying the same SELECT DISTINCT to the newly loaded base dimension. Then process any type 2 changes by comparing the current rows of the temporary shrunken dimension (WHERE End_Date = ‘9999-12-31’) to the current rows of the master shrunken dimension based on the shrunken dimension’s natural key. This may sound inefficient, but a test run only took 10 seconds against a base dimension with over 1.1 million rows on a virtual machine on a laptop.

Alternative: Create the base and shrunken dimensions separately
If you have separate source tables for the shrunken dimension attributes, you could use them to create the shrunken dimension directly. Again, assign surrogate keys, and track changes using Type 2 change tracking. The completed shrunken dimension can be joined back to the base dimension early in the ETL process to populate the current higher-level attributes. Then you can proceed with the base dimension change tracking. The hard part is creating the historical base and shrunken dimensions because you have to compare both the effective dates and end dates from both dimensions, inserting new rows into the base dimension when changes in the shrunken dimension occur.

Present the dimensions to the users
Once you have your historical dimensions populated and ETL processes in place, the final decision is how to represent these tables to the user. The shrunken dimension can be presented exactly as it exists, joining to higher level facts as needed. When users see the base dimension, they like to see the associated attributes from the shrunken dimension as well. This could be done in several ways: either through a view that joins the base dimension to the shrunken dimension via the shrunken dimension’s surrogate key, or by defining the joins in your BI tool’s semantic layer, or by actually joining the tables in the ETL process and physically instantiating the shrunken dimension columns in the base dimension.

All of these approaches should look exactly the same to the users. The only difference might be in performance, and you’d have to experiment with your system to see which works faster for most queries. We usually find we get the best performance and simplest semantic layer by pre-joining the tables in the ETL process and physically copying the columns, unless they are very large tables