Design Tip #51: Latest Thinking On Time Dimension Tables

By Ralph
Kimball

Virtually every fact table has one or more time related dimension foreign keys. Measurements are defined at specific points of time and most measurements are repeated over time.

The most common and useful time dimension is the calendar date dimension with the granularity of a single day. This dimension has surprisingly many attributes. Only a few of these attributes (such as month name and year) can be generated directly from an SQL date-time expression. Holidays, work days, fiscal periods, week numbers, last day of month flags, and other navigational attributes must be embedded in the calendar date dimension and all date navigation should be implemented in applications by using the dimensional attributes. The calendar date dimension has some very unusual properties. It is one of the only dimensions that is completely specified at the beginning of the data warehouse project. It also doesn't have a conventional source. The best way to generate the calendar date dimension is to spend an afternoon with a spreadsheet and build it by hand. Ten years worth of days is less than 4000 rows.

Every calendar date dimension needs a Date Type attribute and a Full Date attribute. These two fields comprise the natural key of the dimension table. The Date Type attribute almost always has the value "date" but there must be at least one record that handles the special non-applicable date situation where the recorded date is inapplicable, corrupted, or hasn't happened yet. The foreign key references in the fact table in these cases must point to a non-date date in the calendar date table! You need at least one of these special records in the calendar date table, but you may want to distinguish several of these unusual conditions. For the inapplicable date case, the value of the Date Type is "inapplicable" or "NA". The Full Date attribute is a full relational date stamp, and it takes on the legitimate value of null for the special cases described above. Remember that the foreign key in a fact table can never be null, since by definition that violates referential integrity.

The calendar date primary key ideally should be a meaningless surrogate key but many ETL teams can't resist the urge to make the key a readable quantity such as 20040718 meaning July 18, 2004. However as with all smart keys, the few special records in the calendar date dimension will make the designer play tricks with the smart key. For instance, the smart key for the inapplicable date would have to be some nonsensical value like 99999999, and applications that tried to interpret the date key directly without using the dimension table would always have to test against this value because it is not a valid date.

In some fact tables time is measured below the level of calendar day, down to minute or even second. One cannot build a time dimension with every minute second of every day represented. There are more than 31 million seconds in a year! We want to preserve the powerful calendar date dimension and simultaneously support precise querying down to the minute or second. We may also want to compute very precise time intervals by comparing the exact time of two fact table records. For these reasons we recommend a design with a calendar date dimension foreign key and a full SQL date-time stamp, both in the fact table. The calendar day component of the precise time remains as a foreign key reference to our familiar calendar day dimension. But we also embed a full SQL date-time stamp directly in the fact table for all queries requiring the extra precision. Think of this as special kind of fact, not a dimension. In this interesting case, it is not useful to make a dimension with the minutes or seconds component of the precise time stamp, because the calculation of time intervals across fact table records becomes too messy when trying to deal

with separate day and time-of-day dimensions. In previous Toolkit books, we have recommended building such a dimension with the minutes or seconds component of time as an offset from midnight of each day, but we have come to realize that the resulting end user applications became too difficult, especially when trying to compute time spans. Also, unlike the calendar day dimension, there are very few descriptive attributes for the specific minute or second within a day.

If the enterprise has well defined attributes for time slices within a day, such as shift names, or advertising time slots, an additional time-of-day dimension can be added to the design where this dimension is defined as the number of minutes (or even seconds) past midnight. Thus this time-of day dimension would either have 1440 records if the grain were minutes or 86,400 records if the grain were seconds. The presence of such a time-of-day dimension does not remove the need for the SQL date-time stamp described above.