Kimball Design Tip #5: Surrogate Keys For The Time Dimension

By Ralph Kimball

I get several data warehouse design questions each day. Because so many of them are serious and interesting, I try to answer them. But if it appears to be an assignment from a college professor, I politely decline!

Here's the question:

A consultant we had recently proposed a time dimension that looks rather different from the ones you design.

His time dimension structure was:

Key	varchar2 (8)
StartDate	date or date/time
EndDate	date or date/time

Sample data was:

Key	StartDate	EndDate
xmas99	25Nov99	06Jan00
1qtr99	01Jan99	31Mar99
newyrsdy	01Jan00	01Jan00
01Jan00	01Jan00	01Jan00

What is your view on a time dimension with this structure? For what type of scenario / business would you find this a good, viable alternative?

Here's how I responded:

I don't think I like that time dimension very much if indeed it is a dimension.

I expect a time dimension to describe the time context of a measurement expressed in a fact table. In database terms there needs to be a time-valued foreign key in every fact table record that points out to a specific record in the time dimension.

It is very important for applications simplicity that the fact table be of uniform granuarity. In other words, all the records in the fact table should represent measurements take at a daily level, a weekly level, or a monthly level, for instance.

Your proposal has time dimension records of varying grain and it appears that they overlap. If you have a measurement record that occurs on a particular date, and these "time dimension" records overlap, which one of the records do you choose for a particular fact table record?

In a uniform grain fact table you can use the associated time dimension to constrain in a simple way on many different spans of time. A time dimension table with entries for every discrete day is very flexible because in this table you can simultaneously represent all the useful time groupings you can think of.

A typical daily grain time table with an U.S. perspective could have the following structure:

> Time_Key (surrogate key; simple integers assigned from 0 to N)
> Time_Type (Normal, Not_Applicable, Hasnt_Happened_Yet, Corrupted)
> SQL_Time_Stamp (8 byte date stamp for Type=Normal, Null otherwise)
> Day_Number_in_Month (1..31)
> Day_Number_in_Year (1..366)
> Day_Number_in_Epoch (an integer, positive or negative)
> Week_Number_in_Year (1..53)
> Week_Number_in_Epoch (an integer, positive or negative)
> Month_Number_in_Year (1..12)
> Month_Number_in_Epoch (an integer, positive or negative)
> Month_Name (January, ..., December, can be derived from
> SQL_Time_Stamp) Year (can be derived from
> SQL_Time_Stamp) Quarter (1Q, ..., 4Q) Half (1H, 2H)
> Fiscal_Period (names or numbers, depending on your finance department)
> Civil_Holiday (New Years Day, July 4, ..., Thanksgiving, Christmas) Workday (Y, N)
> Weekday (Y, N)
> Selling_Season (Winter Clearance, Back to School, Christmas Season) Disaster (Hurricane Hugo, ..., Northridge Earthquake)

In this daily time table you make a record for each day of the year and you populate each field (shown above) with the relevant values for that day. All the special navigation fields like Fiscal_period and Selling_season let you define arbitrary spans of time for these special items. For example, you can constrain Selling_Season = "Back to School" and would automatically get all the days from August 15 to September 10.

In your proposed design you show the keys of the time dimension table with values like "xmas99" and "1qtr99". These are smart keys. Smart keys are dangerous in a data warehouse dimension table for several reasons. The generation of these keys is held hostage to the rules for their syntax. It is tempting to write applications and user interfaces that would make these keys visible to someone. If there is a "1qtr99" are you guaranteeing that there is a "2qtr99"? And what do you do when you need one of the Not-Applicable situations for a date stamp?

We have discussed the assignment of surrogate keys in other forums, but we really mean what we say here: the keys in the time dimension must have no applications significance. They are just integers and you can't compute with them.

Kimball Design Tip #5: Follow-up To Surrogate Keys For The Time Dimension

Posted May 21, 2000

I would like to share with you some useful comments I received about Design Tip #5, where I outlined a preferred design for a time dimension, and said that the primary key for this dimension should just be a simple integer, not a true date stamp.

Several people, who otherwise agreed with this approach, said that nevertheless it can be very useful for the surrogate keys (1,2,3, ..., N) to be assigned in the correct order corresponding to the dates in the associated time dimension records. This allows any fact table to be physically partitioned on the basis of the surrogate time key. Physically partitioning a large fact table on the basis of time is a very natural approach in any case, because it allows old data to be removed gracefully, and it allows the newest data to be re-indexed and augmented without disturbing the rest of the fact table, if you are using the table partitioning features of your database.

Also, since I have occasionally pointed out that Microsoft SQL Server is the only high end serious database that still does not support table partitioning, I was pleased to see that table partitioning is now a feature of SQL 2000.