



Design Tip #100 Keep Your Keys Simple

By Ralph Kimball

Recently a student emailed me with a design dilemma I have seen many times over my career. He said: *Last Friday I was told by my client that their method to assign surrogate keys to type 2 SCDs was the "industry standard." They are maintaining a 1 to 1 relationship between the natural keys and the surrogate keys in the dimension table. Their type 2 SCD in their EDW looks as follows:*

<i>dim_key</i>	<i>nat_key</i>	<i>attrs</i>	<i>start_dt</i>	<i>end_dt</i>
1	ABC	Blue	1/1/2008	3/31/2008
1	ABC	Red	4/1/2008	12/31/9999
2	DEF	Green	1/1/2008	2/29/2008
2	DEF	Yellow	3/1/2008	12/31/9999

I expected to see a unique surrogate key for each record, not re-using the same surrogate key for each natural key.

Here's my response:

I've seen this story a few times before! It is definitely wrong to make the SCD 2 key be a combination of a fixed key (1-1 with the natural key) plus a date range. All of your keys should be simple integers assigned by the data warehouse at ETL time, after extraction from the original source.

Big problems:

1) performance will definitely be compromised with a multi-field join from the dimension table to the fact table, especially if the fields involved are complex alphanumeric fields or date-time stamps. The best join performance is achieved by using single field integer keys between dimension tables and fact tables.

2) user/application developer understandability will definitely be compromised with the need to constrain the dates correctly. This point is expanded below.

Technical problems (show stoppers but harder to explain):

3) the date ranges in the dimension for each natural key presumably implement an unbroken overall span with no gaps. That is hard and sometimes nearly impossible to do correctly. If the end effective date of one record is exactly equal to the begin effective date of the next record (for that slowly changing member of the dimension) then you can't use BETWEEN in order to pick a specific dimension record because you could land on the exact dividing line and get two records. You must use greater-than-or-equal for the begin date and less-than for the end date which is likely to further compromise performance. You can't make the end-effective-date one "tick" less than the begin-date of the next record in order to use BETWEEN because the "tick" is machine/DBMS dependent and you can get situations in busy environments with data arriving from multiple sources where you lose transactions that fall into the gap. For example, one of my students gets 10,000 financial transactions in the last second of a fiscal period from 40 separate (and incompatible) source systems, with varying formats for the exact transaction times.

4) In situations where the begin and end effective stamps are actually date-TIMES then you have a

serious issue of how accurately does the user/application developer need to constrain the dimension. You would have to have a detailed understanding of intra-day business rules to do this correctly. Also, date-TIME stamps could be very unwieldy, depending on the DBMS.

5) Explicit date-TIME stamps are machine and DBMS dependent and therefore do not port cleanly from one environment to another.

6) In product movement situations as suggested by your sample data, the begin and end stamps MAY NOT STRADDLE the activity date of the fact record. For example, a specific product profile may be valid from Jan 1 to Feb 1, and hence those are the dates in the dimension, but the product may be sold Feb 15. Try explaining that to the users. Even worse, try building an application that requires two different date constraints. Using simple surrogate keys eliminates this objection, since the correct correspondence between dimension and fact records is resolved at ETL time.

7) The end user/developer MUST always constrain the dimension (actually EVERY dimension simultaneously) to an exact instant in time in every query for the rest of his/her life. Eight dimensions equals eight time constraints. And if one of the dimensions has DATE-TIME fields whereas the others have DATE-ONLY fields then you will get wrong answers if you constrain only to a date. Dates are dangerous because SQL is too "smart". A constraint on a DAY works on values within a day and the system doesn't warn you.

8) The style of embedding date ranges in every dimension record hints at a normalized style of modeling in which every many-to-1 relationship is encumbered with date ranges in order to handle time variance. Thus the objection raised in the previous paragraph is much worse if these normalized tables are actually exposed to the application developer or end user. Now, the data constraints would need to appear in every intermediate pair of tables, not just in the final dimension tables attached to fact tables.

9) Finally, any explicit dependence on the content of natural keys coming from a source system fails to anticipate the challenge of integrating data from multiple source systems, each with their own notion of natural keys. The separate natural keys could be of bizarrely different data types or could even overlap!