

## Design Tip #90 Slowly Changing Entities

By Ralph Kimball

From time to time we get asked whether the techniques for handling time variance in dimensions can be adapted from the dimensional world to the normalized world. In the dimensional world we call these techniques “slowly changing dimensions” or SCDs. In the normalized world we might call these “slowly changing entities” or SCEs.

We’ll show in this design tip that while it is possible to implement SCEs, in our opinion it is awkward and impractical to do so.

Consider, for example, an employee dimension containing 50 attributes. In a dimensional data warehouse, this employee dimension is a single flat table with 56 columns. Fifty of the columns are copied from the original source. We assume that these source columns include a natural key field, perhaps called Employee ID, which serves to reliably distinguish actual employees. The six additional columns are added to support SCDs in the dimensional world, and these columns include a surrogate primary key (used to join to fact tables), change date, begin-effective date-time, end-effective date-time, change reason, and most-recent-flag. The detailed use of these fields has been described many times in our design tips and Toolkit books.

Now let’s review what we do when we are handed a changed record from the source system. Suppose that an individual employee changes office location, affecting the values of five attributes in the employee record. Here are the steps for what we call Type 2 SCD processing:

1. Create a new employee dimension record with the next surrogate key (adding 1 to the highest key value previously used). Copy all the fields from the previous most-current record, and change the five office location fields to the new values.
2. Set the change date to today, the begin-effective date-time to now, the end-effective date-time to December 31, 9999 11:59 pm, the change reason to “Relocation”, and the most recent flag to True.
3. Update the end-effective date-time of the previously most-current record for that employee to now minus 1 second, and change the most recent flag to False.
4. Begin using the new surrogate key for that employee in all subsequent fact table record entries.

A slightly more complicated case arises if a hierarchical attribute affecting many employees is changed. For example if, if an entire sales office is deemed to be assigned to a new division, then every employee in that sales office needs to undergo the Type 2 steps described above, where the sales division fields are the target of the change.

Now what if we have a fully normalized employee database? Remember that normalization requires that every field in the employee record that does not depend uniquely on the employee natural key must be removed from the base employee record and placed in its own table. In a typical employee record with 50 fields, only a small number of high cardinality fields will depend uniquely on the natural key. Perhaps 40 of the fields will be low cardinality fields “normalized out” of the base employee record into their own tables. Perhaps 30 of these fields will be independent of each other and cannot be stored together in the same entity. That means the base employee record must have 30 foreign keys to these entities! There will be additional physical tables containing lower cardinality fields two or more levels away from the base employee table. Instead of maintaining one natural key and one

surrogate primary key as in the dimensional world, the DBA and application designer must maintain and be aware of more than 30 pairs of natural and surrogate keys in order to track time variance in the normalized world.

We'll assume that every entity in the normalized database contains all six SCE navigation fields, analogous to the SCD design.

To make the office location change described above to an individual employee profile, we must perform all the administrative steps listed above, for each affected entity. But even worse, if there is a change to any field removed more than one level from the base employee record, the DBA must make sure that the SCE processing steps are propagated downward from the remote entity through each intermediate entity all the way to the base employee table. The base employee table must also support the primary surrogate key for joining to the fact table, unless the normalized designer opts to omit all surrogate keys in favor of only constraining on the begin- and end-effective date-time stamps. (We think that eliminating all surrogate keys is a performance and applications mistake of the first magnitude). All of these comments are also true for the second change we described above, for the division reassignment of the sales office.

Processing time variance in a fully normalized database involves other nightmares too complicated to describe in detail in this design tip. For example, if you are committed to a correctly normalized physical representation of your data, then if you discover a business rule that changes a presumed many-to-1 relationship in your data to many-to-many, then you must undo the key administration and physical table design of the affected entities. These changes also require user queries to be reprogrammed! These steps are not needed in the dimensional world. As we have said many times, all of the actual processing to validate and administer the original input data must be performed on pre-normalized (i.e., flat) data anyway. A second serious problem for handling time variance in the normalized world is how to administer late arriving dimensional data. Instead of creating and inserting one new dimension record as in the SCD case, we must create and insert new records in every affected entity and all of the parents of these entities back to the base employee record.

As we often remark, a clever person who is a good programmer can do anything. You can make SCEs work in both the ETL back room and the BI front room if you are determined, but since the final data payload is identical to the simpler dimensional SCDs, we respectfully suggest that you win the Nobel prize on a different topic.