Design Tip #145  Time Stamping Accumulating Snapshot Fact Tables

By Joy Mundy

In Design Tip #140, I discussed the challenges of designing dimensional schemas for processes of indeterminate length such as a sales pipeline or insurance claims processing. We concluded they are best represented as accumulating snapshot fact tables characterized by one row per pipeline occurrence where each row is updated multiple times over its lifetime. However, because each row is updated, we have an imperfect recording of history. The accumulating snapshot does a great job of telling us the pipeline’s current state, but it glosses over the intermediate states. For example, a claim may move in and out of states multiple times: opened, denied, protested, re-opened, re-closed. The accumulating snapshot is hugely valuable, but there are several things that it cannot do:

- It can’t tell us the details of when and why the claim looped through states multiple times.
- We can’t recreate our “book of business” at any arbitrary date in the past.

To solve both of these problems, we’ll need two fact tables. A transaction fact table captures the details of individual state changes. Then we’ll add effective and expiration dates to the accumulating snapshot fact table to capture its history.

The transaction fact table is straightforward. As described in The Data Warehouse Toolkit, Second Edition (p. 134-135) and the cornerstone article, Fundamental Grains, we often pair the accumulating snapshot fact table with a transaction fact table that contains a row for each state change. Where the accumulating snapshot has one row per pipeline process such as a claim, the transaction fact table has one row per event. Depending on your source systems, it’s common to build the transaction fact table first, and derive the accumulating snapshot from it.

Now let’s turn our attention to the time stamped accumulating snapshot fact table. First of all, not everyone needs to bother with retaining these time stamped snapshots. For most organizations, a standard accumulating snapshot representing the current state of the pipeline, combined with the transaction fact table to show the event details, is ample. However, we’ve worked with several organizations that need to understand the evolution of a pipeline. While it’s technically possible to do that from the transaction data, it’s not child’s play.

One solution to the historical pipeline tracking requirement is to combine the accumulating snapshot with a periodic snapshot: snap a picture of the pipeline at a regular interval. This brute force method is overkill for pipelines that are relatively long in overall duration, but change infrequently. What works best in this case is to add effective and expiration change tracking to the accumulating snapshot.

Here’s how it works:

- Design a standard accumulating snapshot fact table.
- Instead of updating each row as it changes state, add a new row. Our recent designs have been at the daily grain: add a new row to the fact table any day in which something about that pipeline (e.g., claim, sales process, or drug adverse reaction) has changed.
- You need some additional metadata columns, similar to a type 2 dimension:
  - snapshot start date: the date this row became effective.
  - snapshot end date: the date this row expired, updated when a new row is added.
  - snapshot current flag: updated when we add a new row for this pipeline occurrence.

Most users are only interested in the current view, i.e., a standard accumulating snapshot. You can meet their needs by defining a view (probably an indexed or materialized view) that filters the historical snapshot rows based on snapshot current flag. Alternatively, you may choose to instantiate a physical table of current rows at the end of each day’s ETL. The minority of users and reports who need to look at the pipeline as of any arbitrary date in the past can do so easily by filtering on the snapshot start and end
dates.

The time stamped accumulating snapshot fact table is slightly more complicated to maintain than a standard accumulating snapshot, but the logic is similar. Where the accumulating snapshot will update a row, the time stamped snapshot updates the row formerly-known-as-current and inserts a new row. The big difference between the standard and time stamped accumulating snapshots is the fact table row count. If an average claim is changed on twenty days during its life, the time stamped snapshot will be twenty times bigger than the standard accumulating snapshot. Take a look at your data and your business’s requirements to see if it makes sense for you. In our recent designs, we’ve been pleasantly surprised by how efficient this design is. Although a few problematic pipeline occurrences were changed hundreds of times, the vast majority were handled and closed with a modest number of changes.