2011-03-20 - The LinkedIn Bitemporal Data Group

Two Temporal Envelopes

For now, although this may be too brief to be helpful, think of it this way: (1) you have an ordinary table; (2) You put this table in a temporal "envelope" by attaching a time period to each of its rows. This is valid time, the time when the things that the rows represent were as the rows describe them to be (or to have been, or to become in the future). This creates a uni-temporal valid-time table. (3) You put this uni-temporal table in another temporal envelope, again by attaching a time period to each of its rows. This second time period is transaction time or, as I prefer to call it, assertion time. This is the time during which we consider the valid-time rows to be true, accurate, correct, of the highest quality we can provide, etc.

When what a row represents somehow changes (customer name change, account balance change, bill of materials structure change), then that row does not become invalid. It simple becomes a row which describes a past state of the object. So we end its valid time period on the same date/timestamp that we begin the valid time period of its successor row, the one that reflects the new state of the customer/account/BOM, etc. This leaves us with the well-known kind of history table or version table that most of us are familiar with.

But when we want to change a row NOT because the thing it represents has changed, but instead because we discovered that the row contained a mistake, or because we now have better quality data about that thing during its valid time period, then we don't create a new row for that thing + valid time period. Instead, we create a new row for that thing with the same valid time period, but with a new transaction/assertion time period. We end the transaction/assertion time period of the original row on the same date/timestamp that we begin the transaction/assertion time period of the row with the corrected or better quality data.

These "two temporal envelope" tables are bitemporal tables. From them, we can select all rows current in both time periods and the result is row-level equivalent to a corresponding non-temporal table of current data. From them, we can select all rows current in transaction/assertion time, and the result is a conventional version table of our best and most accurate data. From them, we can select all rows current as of some past point in transaction/assertion time, and the result is a version table as it existed at that past point in time. And when all this is done with views, then query authors don't even need to know about the intracacies of bitemporal envelopes.

