2011-03-17 - The LinkedIn Bitemporal Data Group

What is bitemporal data, and why is it important? Part 2.

So we have a (history or) version table, and the PK is the unique id of some object (a customer, say), plus a begin time and an end time. (For why both should be used, see Chapter 4 of my book.) And code insures that new rows will not be added that have a begin date that falls between the begin and end times of some other row with the same unique id (call it an object id, or an oid).

So let's say we have a row in a Customer Version table, as follows:

{ [C123] / [1/1/09] / [9999] / Smith / 3 }

Braces delimit the row, slashes separate columns, and brackets delimit PK columns.

We use dates as our points in time, because they are shorter across the page and thus better for examples.

Usually, when we have an insert or update to carry out, we don't know for how long the resulting row will remain in effect. In that case, we use as an end date the highest date the DBMS can recognize. Let's say that it is 12/31/9999, and write it in the shortened form 9999, as shown above.

So, starting on 1/1/09, customer C123 had a name of "Smith" and a status of "3". Now let's say that it is 3/15/11, and we are updating C123's status to "4". Here's the result.

{ [C123] / [1/1/09] / [3/15/11] / Smith / 3 }
{ [C123] / [3/15/11] / [9999] / Smith / 4 }

A new row was added because something about C123 changed. But now suppose that it is 3/22/11, and we realize that we should have changed Smith's status to "5", not to "4". But for a week, we (our database) had been saying that C123's status has been "4" starting on 3/15/11. Even though that is incorrect, the fact that for a week we said that, or would have said that, on queries and reports, is important. We may be legally liable for the mistake, for example.

So how do we correct the mistake, while retaining the information that we made that mistake? How do we do this without utilizing a companion table of some kind (thus complicating the database objects that must be specified in queries)?

The fact is that, without a second time period, we can't. And if that is true, then it follows

that best practices with version tables can't correct mistakes while retaining the information that, for a certain period of time, those mistakes were presented as true and valid date.

This fact isn't widely recognized, so you should assure yourselves of this. I'll wait a couple of days before explaining how a second time period solves the problem, so you can study the problem.

