Keep Track

New features in Teradata 13.10 enhance support of temporal data.

BY RICHARD T. SNODGRASS

Information is *the* key asset of many companies. For most, this asset contains time-referenced data, which in the past was difficult to track and manage. Languages, such as standard SQL, have few facilities for such data. Fortunately, many new features introduced with Teradata 13.10 SQL provide temporal support.

These new SQL features make it easier to express data modifications (SQL's insert, delete and update statements) and greatly reduce the length and complexity of such modifications. As a result, developers can now more easily convert their applications to support time-varying data as well as more simply create new applications that exploit stored data about the past for deeper insight into the future.

Bitemporal Tables Supported

Take, for example, a mortgage company that is challenged with achieving high data quality on information stored concerning customers and their loans. A customer service representative (CSR) reports an error to the IT department; errors are also discovered by batch jobs producing quarterly reports. The more information the IT department has access to, the better it can analyze and correct these errors.

For this reason, it is important that changes to critical tables concerning customers and their loans be tracked. This implies that each such table has "transaction-time support" to maintain a history of changes. Since this table also needs to model changes in reality, it requires "valid-time support" to indicate when the data was considered valid. The result is a bitemporal table, reflecting these two aspects of underlying temporal support. Teradata 13.10 supports bitemporal tables, greatly easing the development of such applications.

With bitemporal tables, IT can first determine when the erroneous data was stored (a transaction time), roll back the table to that point and examine the valid-time history. It can then determine the correct valid-time history. With that history, IT can tell the CSR what needs to be changed. Or, if the error was in the processing of a user transaction, IT may update the database manually.

Because transaction-time support is included, these changes will be logged as well, enabling someone later to see what happened when the change itself was in error. The support for valid time and transaction time permits a sophisticated analysis of the evolution of the table, with all of the data directly at hand. The alternatives—going back through paper records to reconstruct the sequence of changes that were made, or attempting to extract that sequence from backup tapes or other secondary data sources—are simply not practical in such a dramatically changing environment.

Follow the Property

A property owner table captures both the history in reality of the owner(s) of a property over time, as well as the sequence of database states, denoting the transactions applied to this table. This bitemporal table is easy to specify in Teradata SQL (note the new keywords VALIDTIME and TRANSACTIONTIME):

```
CREATE MULTISET TABLE prop_owner (
customer_number INTEGER,
property_number INTEGER,
property_VT PERIOD(DATE) NOT NULL AS VALIDTIME,
property_TT PERIOD (TIMESTAMP(6)) WITH TIME ZONE)
NOT NULL AS TRANSACTIONTIME)
PRIMARY INDEX ( property_number );
```

The valid-time timestamp is specified as having a granularity of day, as a property cannot change hands multiple times in a single day. The transaction-time timestamp is identified at a granularity of microsecond, to differentiate rapidly executing transactions.

The property number column constitutes a primary key in valid time and transaction time. Specifically, the state of the table at any day in valid time, as stored at any instant in transaction time, should include at most one row in the table for any particular property, meaning that that property has one owner at that valid time, as recorded at that transaction time. Because the table was declared to be bitemporal (via inclusion of both valid and transaction time), this temporal integrity constraint will be checked automatically by Teradata 13.10.

Let's follow the history, over valid time and transaction time, of an apartment in Boston at 123 Main St. for the month of January 2010. On Jan. 10, this apartment was purchased by Eva Nielsen. We record this information as a current valid-time, current transaction-time insertion.

When the database management system (DBMS) starts up, the default temporal qualifier is exactly that: current in valid time and current in transaction time. The English is in italics, followed by the SQL statement, using the new Teradata functionality.

Eva Nielsen (whose customer number is 145) buys the apartment at 123 Main St. in Boston (whose property number is 7797) today (which happens to be Jan. 10, 2010).

```
INSERT INTO Prop_Owner (customer_number,
property_number)
VALUES (145, 7797)
```

This information is valid starting now and was inserted now. The DBMS encodes this information using the special valid-time and transaction-time columns, all under the covers. Note that the transaction-time extent of all modifications is from "now," in this case "2010-01-10," to "until closed," which is encoded as "9999-12-31." (See table 1.) This is also the valid-time extent, given the default temporal qualifier (which can be changed by the user).

TABLE 1 **BITEMPORAL STATE**

CUSTOMER NUMBER	NUMBER PROPERTY	PROPERTY_VT	PROPERTY_TT
145	7797	(2010-01-10, 9999-12-31)	(2010-01-10, 9999-12-31)

Let's examine a simple update:

Today (which happens to be Jan. 15) Peter Olsen (whose customer number is 827) buys this apartment, transferring ownership from Eva to him.

```
UPDATE Prop_Owner
SET customer_number = 827
WHERE property_number = 7797
```

The figure (page 3) shows the bitemporal time diagram corresponding to this update.

The horizontal axis tracks transaction time and the vertical axis tracks valid time. Information about a row, or about multiple rows associated with a primary key value, are depicted as two-dimensional polygonal regions in the diagram. Arrows extending rightward denote "until closed" in transaction time; arrows extending upward denote "forever" in valid time. Here we have two regions, one associated with Eva Nielsen and one with Peter Olsen. The left-hand region starts at time 10 (all times are relative to January 2010, so "10" corresponds to Jan. 10, 2010) in transaction time and extends to "until closed," and begins also at time 10 in valid time and extends to "forever."

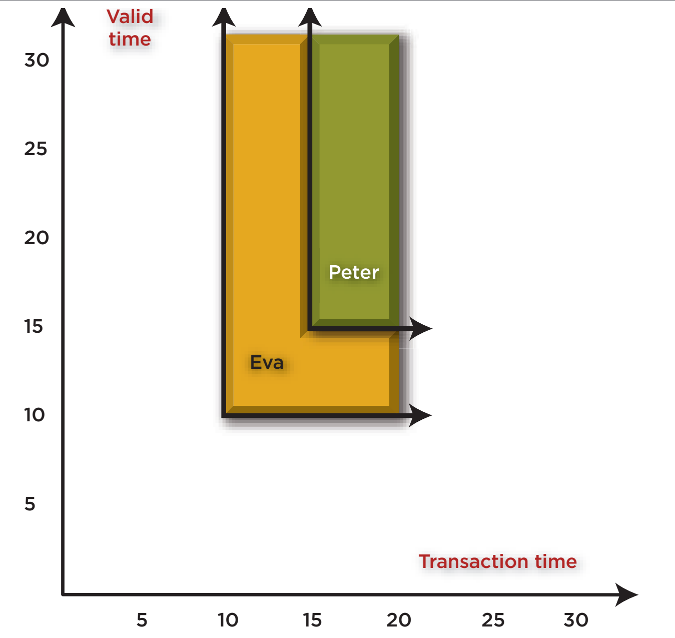
It also illustrates how this update affects the time diagram. From time 15 on, Peter owns the property; from time 10 to 15, Eva owned the property. Both regions extend to the right to "until closed." This time diagram captures two facts—Eva owning the apartment and Peter owning the apartment—each associated with a bitemporal region.

Additionally, the figure captures the evolving information content of the property owner table. Consider a transaction time-slice, which returns the valid-time history at a given transaction time. Such a time-slice can be visualized as a vertical line intersecting the x-axis at the given time.

At transaction time 5 (Jan. 5), the table has no record of the apartment being owned by anyone. At transaction time 12, the table records that the apartment was owned by Eva from Jan. 10 to "forever." If we time-traveled back to Jan. 12 and asked for the history

Many new features introduced with Teradata 13.10 SQL **provide temporal support.** These new SQL features make it **easier to express data modifications** … and **greatly reduce the length and complexity** of such modifications.

FIGURE **Bitemporal Time Diagram**



A current update: Peter buys the apartment, transacted on Jan. 15.

of the apartment, that would be the response. We *thought* then that Eva owns the apartment, and that is what the property owner table recorded then. At transaction time 17 the table records that the apartment was owned by Eva from Jan. 10 to Jan. 15, at which time ownership transferred to Peter, who now owns it to "forever." And that is the history as best known (denoted by the right-pointing arrows). It is what we think is true about the valid-time history.

Automatic Time Handling

In addition to contending with valid time, we also must ensure that the transaction-time extent of the modification is from "now" to "until closed." One important property of tables with transaction-time support is that they are append-only.

As these tables capture the state of the stored table over time, once we have recorded what the state was at a particular time, we can't go back and change it later, because we can't change the bits stored on the disk at that prior time. The changes always accumulate in the table with transaction-time support. The practical ramification is that we never physically delete a row from such a table; only physical modifications are allowed to insert rows into the table and to change the transaction-stop time of a row from "until closed" to "now," thereby logically deleting the row.

Teradata 13.10 handles this automatically. The resulting property owner table contains three rows. A careful matching of the dates in table 2 to the time diagram will aid in understanding how a bitemporal state table encodes the regions found in the time diagram.

TABLE 2 **BITEMPORAL STATE**

CUSTOMER NUMBER	NUMBER PROPERTY	PROPERTY_VT	PROPERTY_TT
145	7797	(2010-01-10, 9999-12-31)	(2010-01-10, 2010-01-15)
145	7797	(2010-01-10, 2010-01-15)	(2010-01-15, 9999-12-31)
827	7797	(2010-01-15, 9999-12-31)	(2010-01-15, 9999-12-31)

In previous versions of the Teradata Database, all of this must be done manually. You are encouraged to implement this simple update in conventional (nontemporal) SQL. It requires a surprisingly complex series of five INSERT and UPDATE statements, 31 lines in all.

Pared Down

We examined two simple variants of temporal modifications, both current in valid and transaction time: an insertion and an update. In conventional SQL these statements can be long and complex: The worst case is one in which a non-temporal update of only a few lines is expanded to some 60 lines of SQL. All are very natural to write given the temporal extensions provided in Teradata 13.10, requiring but a few lines. **T**

Richard T. Snodgrass, a professor of Computer Science at the University of Arizona, has researched temporal databases for 30 years.

ONLINE
Discover more on this topic by downloading "A Case Study of Temporal Data," by Richard T. Snodgrass, on **Teradata.com.**