Managing Temporal Data: a Comparison of Oracle 11g and the AVF. Part 2.

by Dr. Tom Johnston.

The AVF is the Asserted Versioning Framework, a software product offered by our company which manages temporal data as described in our book Managing Time in Relational Databases: How to Design, Update and Query Temporal Data (2010, Morgan-Kaufmann, ISBN 978-0-12-375041-9). For further information on this product, and on the associated services which we offer, you may contact us via links on our website, AssertedVersioning.com.

Oracle 11g, and especially its Workspace Manager component, is Oracle's current and best attempt at extending its RDBMS products to support the management of temporal data. We take our descriptions of this product from the Oracle publication Workspace Manager Developer's Guide 11g Release 2 (11.2), document E11826-02, published October 2009.

We remind our readers that a Glossary of nearly three-hundred entries is also available from our website. This Glossary explains many of the technical terms used to describe temporal data and its management, and in particular the technical vocabulary we have developed to explain the Asserted Versioning method of managing temporal data.

Here in Part 2, we begin our discussion by clearing up some terminological issues.

Terminology

A more basic disparity in terminology is between Asserted Versioning and accepted usage as established by the computer science community. That accepted terminology, which we will call "standard terminology", distinguishes between "valid time" and "transaction time".

- A valid time period indicates when the object represented by a row of data existed, and had the characteristics ascribed to it by the data in that row.

- The transaction time of a row of data begins when that row is first inserted into a table, and continues on either indefinitely, or until a new row of data is created which corrects a mistake in that row. That new row of data (i) represents the same object, and (i) has a valid time period identical to or partially overlapping that of the first row.

Asserted Versioning distinguishes between effective time and assertion time. Effective time is the same as valid time. We use the term "effective time" because we, and other IT professionals, have been using that term for a long time to mean the period of time during which what a row of data represents is in effect, from a business point of view. For
example, if today is April 13th, and we insert a row of data into a Policy table with an effective begin date of May 1st, then the policy represented by that row will not go into effect until May 1st.

Assertion time is a more difficult concept. However, if we restrict the transactions which insert, update and delete data in asserted version tables so that the assertion begin date of the rows created by those transactions is always the point in time when the rows are physically created, then our term "assertion time" is equivalent to the standard term "transaction time".

So clearly, we will sometimes need a terminology which is neutral between the standard terminology and our own. For this purpose, we borrow two terms from philosophy – ontology and epistemology. Ontology is the study of being, of what there is. Epistemology is the study of knowledge, of what we think there is. So the kind of time that tracks how things in the world change, and when, we may call ontological time. And the kind of time that tracks how our beliefs about the data which represents those things change, and when, we may call epistemological time.

In comparing 11g Workspace Manager with Asserted Versioning, there is an unfortunate disparity in terminology. Approximately, Oracle uses the term "version table" to refer to what we call an assertion table; and we use the term "version table" to refer to what Oracle calls a valid-time table. Using our neutral terms as column headings, Figure 1 displays these terminological differences.

<table>
<thead>
<tr>
<th></th>
<th>uni-temporal in ontological time</th>
<th>uni-temporal in epistemological time</th>
<th>bi-temporal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle</td>
<td>valid-time table</td>
<td>version table</td>
<td>valid-time version table</td>
</tr>
<tr>
<td>Asserted Versioning</td>
<td>version table</td>
<td>assertion table</td>
<td>asserted version table</td>
</tr>
</tbody>
</table>

**Figure 1. Terminology: Conflicting Uses of the Term "Version Table".**

We believe that our use of the term "version" is more consistent with actual use of the term by IT professionals. This is because when IT professionals create and manage version tables, they think they are creating and managing tables each of whose rows represents a version of an object, where the version time period is the period of time during which the object was in effect, i.e. existed in the real world with the characteristics ascribed to it by the data on that row. This means that versions are representations of objects which are distinguished in ontological time. But as Oracle uses the standard terminology, versions are representations of objects which are distinguished in epistemological time.

Sometimes, in this series of articles, we will need to use "version table" to refer to what Oracle calls a version table. When we do, we will add an asterisk as a prefix to that term, and, as necessary, to cognate or otherwise related terms used with the meaning Oracle
attaches to them. Lacking that prefix, when we use the term "version table", or related expressions, we will be referring to what we mean by that term. As we said, we believe that this usage is closer to the way that the term is and has been used throughout IT, for many years.

**Oracle's *Version Tables and *Version-Enabled Tables**

As we said, there are only two kinds of time recognized in bi-temporal data management: valid time and transaction time. Oracle allows you to *version-enable a table without also enabling valid time support, as the following quotation indicates.

The `validTimeRange` parameter (`WM_PERIOD DEFAULT NULL`) has been added for the `EnableVersioning` procedure. With this parameter, if you enable valid time support when you version-enable a table, you can specify an initial valid time range. (p. xviii, underline added.)

So *version tables may or may not be *version-enabled. If they are not, then rows in those tables are *versions, but are not necessarily related to valid time and therefore are not versions. So those rows in *version tables must be distinguished from one another by means of the other kind of time – transaction time.

Transaction time, as we explain in our book, is a subset of what we call assertion time. Rows of data which have specific locations in epistemological time (transaction time / assertion time) are what we call assertions. And so:

*Versions are assertions.

**Figure 2. Terminology: *Versions and Assertions.**

Tables whose rows are located in both kinds of time are bi-temporal tables. Our bi-temporal tables are what we call asserted version tables. Oracle's bi-temporal tables are what they call valid-time version tables. And this name that Oracle gives to its bi-temporal tables emphasizes the point that their term "versions" is very misleading vis-à-vis what IT professionals mean when they use the term "versions".

Next time, in Part 3, we will summarize the principal differences between Asserted Versioning and its Asserted Versioning Framework, and Oracle's support for temporal data management using its 11g DBMS and, in particular, the Workspace Manager component of 11g.