A Joint Oracle Teradata White Paper
March 2012

Configuring Oracle Business Intelligence Enterprise Edition for Teradata Temporal Tables
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Executive Overview</td>
<td>2</td>
</tr>
<tr>
<td>Introduction</td>
<td>2</td>
</tr>
<tr>
<td>“As-is” and “As-was” Example</td>
<td>3</td>
</tr>
<tr>
<td>Step 1. Define a session variable</td>
<td>4</td>
</tr>
<tr>
<td>Step 2. Define the temporal table</td>
<td>5</td>
</tr>
<tr>
<td>Step 3. Pull the table and column definition into the physical layer</td>
<td>6</td>
</tr>
<tr>
<td>Step 4. Expose the temporal columns to the end user</td>
<td>9</td>
</tr>
<tr>
<td>Step 5. Building the dashboard</td>
<td>10</td>
</tr>
<tr>
<td>Step 6. Add a new Dashboard prompt</td>
<td>11</td>
</tr>
<tr>
<td>Step 7. Add a column prompt</td>
<td>11</td>
</tr>
<tr>
<td>Step 8. Save and test the dashboard</td>
<td>13</td>
</tr>
<tr>
<td>SQL Examples</td>
<td>16</td>
</tr>
<tr>
<td>Conclusion</td>
<td>18</td>
</tr>
</tbody>
</table>
Executive Overview

Teradata temporal capabilities allow data warehouse time travel. Temporal supports the reproduction of a report that ran previously, such as six months ago, even though numerous changes have been made to the underlying data.

Oracle Business Intelligence Enterprise Edition (OBIEE) supports and can utilize the temporal capabilities of Teradata and expose them to end users in an easy-to-use fashion.

Introduction

The purpose of this document is to show how to integrate OBIEE with Teradata Temporal Tables. It details how to set up OBIEE for "as-is" and "as-was" type analysis. It is a cook book style write-up and assumes the customer has (and knows) OBIEE and that the customer knows what they want to do with temporal. It does not go into detailed analysis of different use cases for temporal tables. This write-up was done using OBIEE 11g, although it is completely applicable for OBIEE version 10.1.x. There are some variations in the setup but the concepts are the same.

Teradata Database 13.10, released in September of 2010, included a number of fully integrated, in-database data attributes, qualifiers, and predicates that are extremely useful for automating the management of time-varying data. Also included were a number of powerful functions to enable native time series analysis and comparison of periods of time.
“As-is” and “As-was” Example

Temporal tables can be configured as “regular” OBIEE source tables. However, the data returned for queries will only reflect the current transaction time or valid time. In order to return data for a specific point in time, regular OBIEE source tables cannot be used. OBIEE requires special configuration to take advantage of a table’s temporal capabilities. Basically, a normally defined OBIEE table needs to be converted to a “select” type table (also known as opaque views). A SQL select statement is defined for the temporal table and can then include temporal keywords such as “as of”. The following is an example of an OBIEE dashboard that has a temporal query.

![Figure 1. Dashboard with temporal query](image)

“Channel” is defined in Teradata as a temporal table with a valid time column. The dashboard above queries the channel table with an “as of” date based on the prompt for “calendar_date”. The temporal columns for valid start date and valid end date are exposed on the dashboard.

Follow the steps below to see how to put together a temporal dashboard.
Step 1. Define a session variable

The first step is to define a session variable that will be used to determine and store the “as of” date for the temporal query. The session variable should be set so that any user can set the value. The default Initializer should be set to “current_date” (without the quotes). The screen shot below shows the properties for the “as_of_date” variable.
Step 2. Define the temporal table

Below is the DDL for the channel example table; the vt column (in red) is the temporal column:

```sql
CREATE MULTISET TABLE SAMPLEDATA.channels_t , NO FALLBACK ,
  NO BEFORE JOURNAL,
  NO AFTER JOURNAL,
  CHECKSUM = DEFAULT,
  DEFAULT MERGEBLOCKRATIO
  (  
    CHANNEL_ID FLOAT FORMAT '+9.99999999999999E+999' NOT NULL,
    CHANNEL_DESC VARCHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC NOT NULL,
    CHANNEL_CLASS VARCHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC NOT NULL,
    CHANNEL_CLASS_ID FLOAT FORMAT '+9.99999999999999E+999' NOT NULL,
    CHANNEL_TOTAL VARCHAR(13) CHARACTER SET LATIN NOT CASESPECIFIC NOT NULL,
    CHANNEL_TOTAL_ID FLOAT FORMAT '+9.99999999999999E+999' NOT NULL,
    vt PERIOD(DATE) NOT NULL AS VALIDTIME)
PRIMARY INDEX ( CHANNEL_ID );
```
Step 3. Pull the table and column definition into the physical layer

Use the OBIEE “import metadata” wizard to pull the table and column definition into the physical layer. The wizard is used so that all of the column definitions do not need to be manually added into the physical layer. Next, the table type for the physical table properties needs to switch from “physical table” to “select” and the temporal SQL needs to be added.

![Physical Table - channels_t](image)

Change from “physical table” to “select”
The SQL for the “select” table is entered into the text box:

```
    as of coalesce(cast (trim( "VALUEOF(NO_SESSION,as_of_date) " ) as date),current_date)
    select
    channel_class,channel_class_id,channel_desc,channel_id,
    channel_total,channel_total_id,begin(vt) vt_start,end(vt) vt_end
    from channels_t
```

The SQL needs to be carefully formed in order to support any valid “as of date” the user picks as well as to support the user not picking any value for “as of date”. In the case of no “as of date”, the query will default to the current date. The SQL is reiterated below with notes for key points.
Configuring Oracle BI EE for Teradata Temporal Tables

A. "coalesce" is used to default the "as of date" when the end-user has not chosen a date
B. "cast" is used to convert the string returned from the "trim" function to the "date" data type
C. "trim" is used to get rid of the extra spaces in surrounding the "VALUEOF(…)
function
D. The string "VALUEOF(NQ_SESSION.as_of_date)" needs spaces surrounding it in order for the OBIEE to substitute the chosen date for the string. If there are no spaces around the VALUEOF() function then OBIEE will generate errors like "the repository variable as_of_date has no value definition. Substituting “sp” for space, the string needs to be formatted as such: ‘spVALUEOF(NQ_SESSION.as_of_date)sp’
E. "current_date" is used to default the “as of date” to today’s date
F. "VALUEOF(NQ_SESSION.as_of_date)" returns the chosen date stored in the variable that was defined in step 1
G. "as_of_date" is the variable that was defined in step 1
H. "Vt_start" is a derived column. Begin vt is a function that returns the beginning date of the valid time period (see Teradata temporal documentation for more details). This column is not mandatory in the physical layer. It is shown here as a way of exposing the data. The column alias name must match the column name in the physical layer.
I. "vt_end" is a derived column. End vt returns the ending date of the valid time period. This column is not mandatory in the physical layer. It is shown here as a way of exposing the data. The column alias name must match the column name in the physical layer.
Step 4. Expose the temporal columns to the end user

If you want to expose the temporal columns to the end user you will need to manually add columns to the physical definition. In the example above, two columns were manually added: vt_start and vt_end (valid time start and end). Note that the additional columns need to be defined in the “select” SQL (as in the example above) as well as actual physical columns in OBIEE:

Note that they are defined as “date” data type columns as the functions return the “date” data type.

With all of the table and column and variable definitions correctly in place, the “view data” action (right click on table name) should show data from the table:
That completes the configuration in the OBIEE Administrator. Next up is building the dashboard.

**Step 5. Building the dashboard**

In OBI Answers, configure and save your query on the temporal table. This is a standard dashboard query – there is nothing unique about it.
Step 6. Add a new Dashboard prompt

In OBI Answers, add a new Dashboard prompt:

[Image: Dashboard prompt]

Figure 8. Dashboard prompt

Step 7. Add a column prompt

On the dashboard prompt page, add a new column prompt:

[Image: Column prompt]

On the dashboard prompt page, add a new column prompt:
The column associated with the prompt is `vt_start`. Since it has a data type of “date”, OBIEE allows us to use a calendar type input. Under Options, the “Set a variable” drop-down should be set to “Request Variable”. Then the name of the variable that was defined in step 1 is entered, `as_of_date`, which is used to pick a date from a calendar.

Save the prompt.
Step 8. Save and test the dashboard

Now create and save a dashboard with the query defined in step 5 and the prompt defined in step 6.
When the dashboard is run, the data from the query defaults to the current date.

Figure 11. Run dashboard report
Clicking on the calendar brings up the prompt:

![Calendar Prompt](image)

**Figure 12. Date prompt**

Click “OK” on the prompt.
Click “Apply” on the dashboard. The data in the query now reflects the date selected on the calendar prompt.

![Figure 13. Temporal report run](image)

This completes the tutorial on temporal dashboard set-up. This technique can be applied to many different temporal tables. They can use a common date, or dates can be set up for each table.

**SQL Examples**

The SQL that was generated by OBIEE for the above query is below. The temporal bits are in red:

```sql
select distinct 0 as c1,
     D1.c1 as c2,
     D1.c2 as c3,
     D1.c3 as c4,
     D1.c4 as c5,
     D1.c5 as c6
from
    (select T2489."CHANNEL_CLASS" as c1,
         T2489."CHANNEL_ID" as c2,
         T2489."CHANNEL_TOTAL" as c3,
         T2489."vt_end" as c4,
         T2489."vt_start" as c5
    from
        (as of coalesce(cast (trim( '2011-12-09' ) as date),current_date))
```
select channel_class,channel_class_id,channel_desc,channel_id, channel_total,channel_total_id,begin(vt) vt_start,end(vt) vt_end from channels_t ) T2489 ) D1
order by 2, 3, 6, 5, 4

A query that was generated by OBIEE that includes the temporal dimension “channel” and a fact table “sales” is shown below:

select distinct 0 as c1,
    D1.c2 as c2,
    D1.c3 as c3,
    D1.c1 as c4
from (select sum(T2086."AMOUNT_SOLD") as c1,
    T2489."CHANNEL_CLASS" as c2,
    T2489."CHANNEL_TOTAL" as c3
from (as of coalesce(cast (trim( ' 2011-12-15 ') as date),current_date)
select channel_class,channel_class_id,channel_desc,channel_id, channel_total,channel_total_id,begin(vt) vt_start,end(vt) vt_end from channels_t ) T2489,
   "sales" T2086
where ( T2086."CHANNEL_ID" = T2489."CHANNEL_ID" )
group by T2489."CHANNEL_CLASS", T2489."CHANNEL_TOTAL"
) D1
order by 3, 2
Conclusion

Teradata 13.10 (and subsequent releases) allows organizations to gather, manage, and analyze “time varying” data with very little administration.

Using the above techniques allows Oracle Business Intelligence Enterprise Edition to utilize the temporal capabilities of Teradata and expose them to end users in an easy-to-use fashion.

This capability can be vital, for example, when responding to inquiries from regulators who want to know what information organizations had and when they had it.