Accessing multidimensional Data Types in Oracle 9i Release 2

Marc Bastien

ORACLE Deutschland GmbH, Notkestrasse 15, 22607 Hamburg
Marc.Bastien@oracle.com

In former Releases of the Oracle Database, multidimensional data types did not exist. In typical Warehouse situations, especially if more advanced analytics was needed, an additional database had to be chosen to store and analyze the data. These databases offered a lot functionality for advanced analytics like what-if analysis, statistical queries and so on and an excellent query performance due to their optimized physical storage concept. But, on the other hand, these databases had some issues regarding size, integration and manageability. In Oracle 9i Release 2, both technologies have been integrated in one database to achieve both optimal results for relational and multidimensional data . This document describes how the multidimensional Data types are stored in the database and could be accessed using SQL and PL/SQL.

1 About the OLAP Option in the Oracle 9iR2 database

The OLAP Option is in fact the database formerly know as “Express” built into the kernel of the relational Oracle database. Still all the features of the Express database are available; this includes a multidimensional calculation engine, multidimensional data types and multidimensional query language, called the OLAP DML. As a new feature to the OLAP Option the SQL-Access to the multidimensional data types was added. This includes the possibility to run OLAP DML commands in a SQL-Session and as well the possibility to access the multidimensional objects via SQL.

Multidimensional Data Model

A key feature of the OLAP Option is its multidimensional data model. There is both a logical data model, which could represent data physical stored in relational or multidimensional structures and the physical multidimensional data model.

Logical data model
The logical data model represents the data without the need to specify if it is stored in a relational schema as a STAR-Schema or in a multidimensional workspace. The model includes Dimensions, Cubes (Measures), Hierarchies, Levels, and Attributes. Cubes are the data in the data model, while Dimensions index the data. Levels group hierarchies and Attributes classify dimension values with non-hierarchical data like "red" or "green.

Physical data model
The multidimensional data model consists of all the multidimensional data objects like Dimensions, Relations, Composites, Formulas, Programs, Aggregation Maps, Models, Surrogates, Valuesets and Worksheets.

Like in a relational model a cube is often represented as a collection of tables or views that make up a STAR-Schema, the multidimensional objects represent the data in the multidimensional workspace. The workspace in Oracle 9iR2 is named the Analytic Workspace (AW).

The multidimensional objects that are relevant for SQL access are:

- **Dimensions**: store dimension members like the values for Month, Quarter and Year in a time dimension. The dimension values are unique across one dimension.
- **Relations**: Relations are to store attributes of dimension values. They play a central role in accessing multidimensional data via SQL since they are used in converting predicates of the SQL "WHERE"-Clause to data selections in the analytic workspace. The Relations often found in an analytic workspaces and therefore often used for SQL access are:
  - Parent R. (relate a dimension value to its parent in a hierarchy)
  - Level R. (relate a dimension value to a specific level in a hierarchy)
  - Family R. (contain all ancestors of a value in a hierarchy)
  - Attribute R. (offer the possibility to select data by non-hierarchical attributes)
- **Variables**: Variables store the actual fact data, e.g. Sales, Units etc. Variables are multidimensional arrays and are stored separately from the dimension keys. If a variable is created, the dimensionality has to be specified. A referential integrity is forced in the OLAP Option, so a value for a variable could not exists for an unknown dimension value.
- **Composites**: Composites are used to compress the actual disk space of a variable. Since the theoretical space in multidimensional Cubes could be very large, a construct is needed to reduce the actual disk space.
- **Formulas**: Formulas contain calculations in the analytic workspace. Users might probably don't notice any difference to variables, but they are very useful to calculate data at run-time. Formulas could contain very easy expressions, more advanced expression (IF-THEN-ELSE) or even calls to OLAP DML programs
- **Programs**: Programs will store procedures or functions in OLAP DML Code. It is possible to write very complex code to create, manipulate or access the data in an analytic workspace

2 Access to the multidimensional objects using SQL

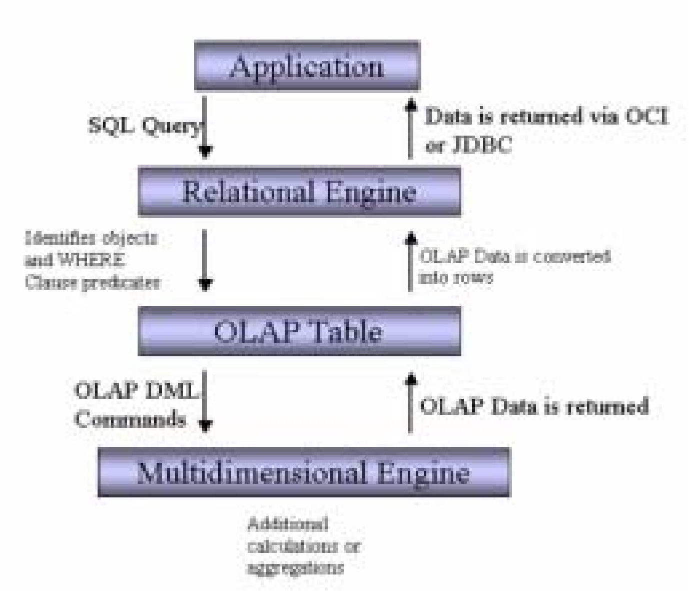
Generally spoken, there are three methods to access multidimensional objects using SQL or PL/SQL.

1) Retrieving data and access all multidimensional objects using PL/SQL
2) Retrieving data using SELECT from a Table Function
3) Retrieving data using SELECT from a relational view

This document will only discuss Method 2 and 3 as this is most often used to query data in the Analytic Workspace. Method 3 is in fact the same as method 2 since the declared view contains the same logic as a direct query using the OLAP_TABLE function.

Processing Queries in Oracle 9iR2

This is the way SQL queries will be processed:


Application
SQL Query
Data is returned via OCI or JDBC
Relational Engine
Identifies objects and WHERE Clause predicates
OLAP Data is converted into rows
OLAP Table
OLAP DML Commands
OLAP Data is returned
Multidimensional Engine
Additional calculations or aggregations

All Queries are first routed to the SQL engine of the relational database. If the SELECT statement selects data from an OLAP_TABLE, the query is processed to the OLAP Option’s Table function. This would extract all needed multidimensional data and process the predicates in the WHERE clause and translate the query into OLAP DML. The multidimensional engine limits the data to the predicates in the WHERE Clause, fetches the data – which could include some extra processing like aggregation at runtime – and returns the data to the relational engine for any additional processing (e.g. joining to other tables, views). The relational engine delivers the data through the open connection, which could be OCI or JDBC.

Creating a view on multidimensional data

The process of creating a multidimensional view is a three-step process, involving two abstract data type to describe the structure of the data and the view itself:

1) Create an abstract data type (ADT) to specify the rows
2) Create an ADT to make up a table out of the rows
3) Create the view to match the rows with the data in the AW

Example: creating a multidimensional view as a dimension lookup table for a time dimension

First: Create an ADT to specify the rows of the multidimensional view

```
create type time_type_row as object (
month varchar(10),
quarter varchar(10),
year varchar(10),
all_times varchar(10));
```

Second: Create an ADT to make up a table:

```
create type time_type_table as table of time_type_row;
```

Third: Create the view and do the mapping between the rows and the multidimensional objects.

```
CREATE OR REPLACE VIEW olap_time_view AS
SELECT *
FROM TABLE(OLAP_TABLE('DDEPOT DURATION SESSION',
'time_type_table',
'limit time KEEP levelrel_time ''MONTH''',
'DIMENSION month FROM time WITH
HIERARCHY parentrel_time
LEVELREL all_times, year, quarter
FROM familyrel_time USING leveldim_time'));
```

In this example the OLAP_TABLE function does the following:

- attaches the appropriate analytic workspace (“DDEPOT”)
- matches to structure to the Table type object (“time_type_table”)
- does some OLAP DML processing (“limit time to levelrel_time ‘MONTH’”), so only time values of the level “MONTH” are selected
- matches the row “month” to the multidimensional dimension “TIME” (and use the relation “parentrel_time” to select parents
- matches the rows “all_times”, ”year” and “quarter” to the multidimensional values coming from the relation “familyrel_time”

The view could be described like any other relational view:

```
SQL> desc olap_time_view

Name               Null? Type
----------------- ----- ---------------
MONTH                    VARCHAR2(10)
QUARTER                  VARCHAR2(10)
YEAR                     VARCHAR2(10)
ALL_TIMES                VARCHAR2(10)
```

A selection based on this view could look like this:

```
SQL> select * from olap_time_view where year = '2002';
MONTH       QUARTER     YEAR  ALL_TIMES
---------- ----------   ----- ----------
Jan02       Q1.02       2002  All Times
Feb02       Q1.02       2002  All Times
Mar02       Q1.02       2002  All Times
Apr02       Q2.02       2002  All Times
May02       Q2.02       2002  All Times
Jun02       Q2.02       2002  All Times
```

Selecting directly from OLAP_TABLE

Like a view can select on an OLAP_TABLE, a SELECT can as well. This is more dynamic than defining a view, and for example offers the possibility to run different OLAP DML programs prior to data selection.

The following example just selects the same data as the example before does

```
SQL> SELECT *
FROM TABLE(OLAP_TABLE('DDEPOT DURATION SESSION',
'time_type_table',
'limit time KEEP levelrel_time ''MONTH''',
'DIMENSION month FROM time WITH
HIERARCHY parentrel_time
LEVELREL all_times, year, quarter
FROM familyrel_time USING leveldim_time'))
where year = ‘2002’;

MONTH       QUARTER     YEAR  ALL_TIMES
---------   ----------  ----- ----------
Jan02       Q1.02       2002  All Times
Feb02       Q1.02       2002  All Times
Mar02       Q1.02       2002  All Times
Apr02       Q2.02       2002  All Times
May02       Q2.02       2002  All Times
Jun02       Q2.02       2002  All Times
```

3 The fully aggregated Cube

Concept

In multidimensional environments the key concept was always that data is fully pre-calculated in a cube. This could physically be true, so every cell need disk space, but could also mean that data has to be calculated at runtime. The multidimensional engine is designed to calculate aggregates very fast, so aggregating at runtime is mostly no performance issue. This runtime aggregation is fully transparent to any client application. To use this advantage of the multidimensional engine if the data is queried with SQL, some rules have to be followed.
The ability of the multidimensional engine to present data in a solved (calculated and aggregated) form to OLAP_TABLE allows all data in the analytic workspace (Aggregates, allocations, forecasts) and other calculations to be presented in a view or for selection directly from OLAP_TABLE. This means that:

- Views can be presented with all summary data, and;
- Complex calculations can be presented as rows and columns. Virtual dimension members are added as rows. Measure calculations appear as columns

The fact that SQL based applications can access all data - summary data and calculations - transparently dramatically reduces the complexity of SQL that must be generated by applications accessing analytic workspaces.

Best Practice in selecting data from analytic workspaces

In a classic, relational Datawarehouse, a STAR-Query would be used to query data. A typical query to get Sales Volume for all Year 2000 and 2001, Total Channel, all Products in the Product Subcategory “Trousers – Men” for all German Customers region would involve all Dimension tables and the Fact Table with at least four joins.
If this query would run against an analytic workspace, all data from the variable has to be fetched and would then be joined with the dimension data. This query would be very slow and inefficient.

Denormalized Views

As mentioned before, the star schema is a typical schema for datawarehouse purposes, allowing some overhead for dimension tables, but none for facts. With additional possibilities to optimize queries using star query optimization, the relational engine performs very well against this schema. Nevertheless, a star schema is not optimal, because of the added complexity of the joins between the dimension and fact tables. A more simple solution would be a fully denormalized schema, but in a relational world this would mean very inefficient storage of the data.

This is not the case for multidimensional storage:

- Physical storage of data within analytic workspaces is completely denormalized and is thus extremely efficient
- The multidimensional engine automatically und transparently joins dimensions and variables. This eliminates the need for expressing joins in SQL
- Dimension members and all dimension attributes are represented by Relation objects in the AW and can be efficiently mapped to a relational view (SQL WHERE will become OLAP DML LIMIT)
- The multidimensional model can be seen as fully aggregated. Therefore aggregations do not need to be done in SQL

So it is possible to create fully denormalized views on top of the multidimensional objects without any negative impact, but with all advantages of the multidimensional engine. The same example as before on the generated view would be:

```
SELECT time_id, channel_id, product_id, customer_id, sales
FROM sales_view
WHERE time_id in ('2000','2001')
AND channel_id = ‘Total Channel’
AND product_id = ‘Trousers - Men’
AND customer_id = ‘DE’
```

This query would be most efficient, because no joins would be needed.

Calculations

As discussed earlier, it is fairly easy to insert even more analytic functionality without the need of a more complex SQL Query. If the Analytic Workspace is used to calculate e.g. a value for the percentage change of sales vs. last year, a formula in OLAP DML would be used:

```
Sales_ly = lagpct(sales,1,time,nostatus)*100
```

Still, the SQL query would not turn complex, nor would any change in performance be noticed:

```
SELECT time_id, channel_id, product_id, customer_id, sales,
sales_ly
FROM sales_view
WHERE time_id in ('2000','2001')
AND channel_id = ‘Total Channel’
AND product_id = ‘Trousers - Men’
AND customer_id = ‘DE’
```

This allows adding analytic capabilities to any SQL Application without the need of complex SQL Queries.

4 Summary

The new OLAP features of the Oracle 9iR2 database offer a lot functionality to create and build analytic applications. Its possibility to access the multidimensional data with SQL adds a lot power to any SQL-aware application or tool.

The key to successful use of the SQL interface to multidimensional data is to build views that allow the translation of predicates in the WHERE clause to OLAP DML limit commands. This is accomplished by mapping columns to dimensions and relations, and using supported operators in SELECT statements.
Other key factors include the use of edge and manager queries or denormalized views and proper Management of sparse data.

Sources

[1]: Oracle 9i OLAP Users Guide Release 2 (9.2)
[2]: Oracle 9i OLAP Developer’s Guide to the OLAP DML Release 2 (9.2)
[3]: Oracle 9i OLAP Developer’s Guide to the OLAP API Release 2 (9.2)
[4]: SQL Access to Multidimensional Data Types