

Costantinos Bourboulas, Oracle Software (Schweiz) GmbH

# Oracle Database 11g OLAP Option

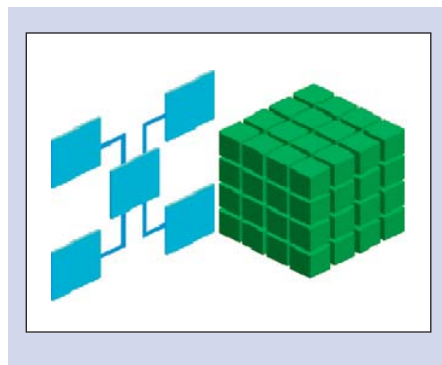
**Remark: For the reader experienced with Oracle Database OLAP , we'll discuss new features introduced with Oracle Database 11g in part 1. For readers not familiar with the technology, I suggest to start with [Part 2: Short Overview of the OLAP Option](#).**

## Part 1: Spotlight on Oracle Database 11g OLAP Option

The OLAP Option to Oracle Database 11g continues the development trends of Oracle9i and Oracle Database 10g, especially in deepening integration with the database and enhancing SQL access to cubes, tightening cube security and providing metadata integration. Oracle Database 11g also introduces the cube as a summary management solution for relational OLAP (ROLAP) implementations.

### Automatic Maintenance of Cube and Dimension Views

Oracle Database 11g automatically creates and maintains relational views for every cube, dimension, and hierarchy in the database (*before, OLAP views had to be created and maintained manually*). If you modify a dimensional object, such as adding a calculated measure to a cube, the view is immediately re-created to reflect the change. The cube is represented as a star schema – there are relational views for each of the cube's dimensions and a single view to represent the cube's measures. The Oracle database defines these views using the new CUBE\_TABLE function, which extracts data from a cube or dimension



and returns it in the two-dimensional format of a relational table (plus enabling the OLAP cubes to be a part of the SQL Optimizer).

Example: CUBE\_TABLE can be demonstrated nicely by using the function in a common select statement, in this case retrieving the `channel` dimension.

```
SELECT * FROM TABLE(CUBE_TABLE('global.channel'));
```

DIM_KEY	LEVEL_NAME	LONG_DESCRIP	SHORT_DESCRIP	TOTAL_CHANNEL_ID	CHANNEL_ID
1	TOTAL_CHANNEL	All Channels	All Channels	1	
2	CHANNEL	Direct Sales	Direct Sales	1	2
3	CHANNEL	Catalog	Catalog	1	3
4	CHANNEL	Internet	Internet	1	4

### Cube-Organized Materialized Views (MVs)

*About Materialized Views: Introduced with Oracle 8i, table-based MVs are used to implement `summary tables` in a data warehouse and can vastly enhance query performance. One of the biggest benefits is that the end-users do not have to be `summary aware`. Instead, they can write their query against the detail tables or views in the database. Then the database's query rewrite mechanism will automatically rewrite the SQL query to use the MVs.*

11g Cube-Organized MVs play the same role as table-based MVs. How-

ever, in the case of cube-organized MVs, the data is managed in the cube rather than in a table, thus automatically inheriting all of the query and refresh performance benefits of the cube, including:

- Management of all possible summaries in as single database object
- Compressed cube aggregation technology
- Cost-based aggregation
- Array based storage and fast cell (row) access
- Fast, incremental refresh and aggregation

A cube-organized MV is only a metadata object. The data is managed and stored in the cube, the MV only contains the metadata that is needed by the database for query rewrite and refresh of the cube via the MV refresh

system. Data is not replicated from the cube to the cube-organized MV. Notably, because the cube-organized MV represents all possible summaries,

a) it can replace the tens, hundreds or even thousands of table-based MVs we witness in typical DWH environments today, thus lowering administrative efforts and improving the efficiency of the query rewrite feature by dramatically reducing the number of MVs that the optimizer needs to consider before executing the query

b) the rate at which the database rewrites queries to the MV is typically very high (as testing with leading BI tools has shown).

Like table-based MVs, cube-organized MVs can be refreshed ON COMMIT, ON DEMAND or on a schedule, with the option to refresh COMPLETE, FAST or FORCE. For example, the following command calls for a fast refresh of the <UNITS\_CUBE>:

```
dbms_mview.refresh('CB$UNITS_CUBE','F');
```

A FAST refresh reads materialized view logs on the source tables for inserts, updates and deletes. These changes are applied incrementally to the cube. The cube is then incrementally pre-aggregated, with only the ancestors of new or changed members being recalculated. A COMPLETE refresh will truncate fact data in the cube and load all data from the source table and aggregate the cube. A FORCE refresh attempts to do a FAST refresh if possible – otherwise it performs a COMPLETE refresh.

## OLAP Metadata Integration

All metadata for cubes and dimensions is stored in the Oracle database and revealed in the data dictionary views, so that you can query the entire business model in SQL. Use of the data dictionary to store the metadata officially codifies the dimensional model in the database, provides significant improvements for metadata queries and supports other new features such as SQL object security for cubes and dimensions.

## Object and Data Security

Oracle Database 11g introduces both object security and data security to OLAP cubes and dimensions. Both types of security are granted to database users and roles. Object security controls access to analytic workspaces, cubes and dimensions using standard SQL GRANT and REVOKE syntax. Data security controls access to the data in a cube or a dimension, you can grant SELECT, INSERT, UPDATE and DELETE privileges to dimen-

sion members (keys) either globally or in the context of a particular cube to control access to the data in a cube.

## Cube Scripts

A cube script is an ordered list of commands that prepare a cube for querying, such as Clear Data, Load Data, Aggregate, Execute PL/SQL, and Execute OLAP DML. For many applications, cube scripts will eliminate the need to use procedural programs for processing cubes.

## Cost-Based Aggregation

Fast updates and uniform querying performance are two hallmarks of the OLAP option. Cost-based aggregation enhances performance in both areas by executing a fine-grained pre-aggregation strategy and storing sparse data sets very efficiently.

## Calculation Expression Syntax

OLAP calculation expressions extend the syntax of the SQL analytic functions. This syntax (for instance LAG, LEAD, RANK, etc.) is already familiar to SQL developers and DBAs, so that it is easier for them to adopt than proprietary OLAP languages and APIs. This syntax is used to define calculations that are embedded in the cube, such as dynamically calculated facts or measures.

*This concludes Part 1 «Spotlight on Oracle Database 11g OLAP Option»*

## Part 2: Short overview of the Oracle Database OLAP Option

The OLAP Option to the Oracle database is a full-featured on-line analytical processing (OLAP) server embedded within the Oracle database. The OLAP Option can be used to broaden the capabilities of SQL-based business intelligence tools and appli-

cations by improving query performance and enriching them with analytic content. As an OLAP solution that is deeply embedded in the Oracle database, the OLAP Option allows centralized management of data and business rules in a secure, scalable and enterprise-ready platform.

## Why OLAP?

Users' queries are often unpredictable. On different days, the same users will perform trend analysis, drill down on specific product lines, and compare a week's sales against those of the same week last year. With standard relational systems, it is difficult to optimize data structures that provide consistently good query performance for such an unpredictable query pattern.

To address this need, DBAs and designers frequently create a system of summary tables or materialized views. OLAP cubes, which provide consistently fast query performance across an entire data model, often provide a better alternative to summary management. Sophisticated calculations can be easily embedded within the cube to enhance the analytic content of applications.

These calculations often rely on data from many rows and interrow calculations. For example, an OLAP cube might include a calculation that compares the current year's sales for each region and product line with those from the same period last year and two years ago. The cube structure is optimized to handle this kind of querying and analysis.

## Why Oracle OLAP?

Oracle OLAP uses an analytic workspace in the database to perform OLAP analysis. Oracle OLAP stores data in the database as multidimensional cubes, which are designed for fast incremental update and query. Cubes are organized by dimensions, which act as keys to the fact data and define the basic structure of the cube.

In many ways, a cube is similar to a star schema. The cube plays the role of the fact table, and an OLAP dimension plays the role of a dimension table. Dimensions can be simple lists of members, or they can be organized into levels and hierarchies. Hierarchical dimensions enable data to be aggregated from lower levels to higher levels of summarization. They support navigation such as drill-down and certain types of calculations such as Share to Parent, Share within Ancestor, and Rank within Parent. They also support many time-series calculations such as Year to Date. These types of calculations are easy to define within the analytic workspace manager (the administrative tool of Oracle OLAP) and are efficiently computed within the cube at runtime.

Oracle OLAP can significantly shorten query processing times for users of SQL-based business intelligence (BI) tools such as Oracle Business Intelligence Suite Enterprise Edition and other third-party tools. As an embedded component of the Oracle database, the OLAP Option benefits from the scalability, high availability and security features that make the Oracle database the market leading enterprise-ready database:

- The OLAP option is embedded into the Oracle database kernel and runs in the same service as the relational database. There is no separate database service to manage
- OLAP cubes and dimensions are stored safely and securely in Oracle files. There are no separate data files to manage
- As a feature of the Oracle database, the OLAP option fully leverages large scale computer hardware. It is fully supported by scalability and availability features like Real Applications Clusters and Grid Computing
- OLAP cubes are secured by Oracle database security features
- OLAP cubes are easily accessible using SQL. Both dimensional and relational applications can easily query the same cube

### Improving the business intelligence solutions you already own

The vast majority of BI applications query relational databases using SQL. This is only natural given that most data is already stored in the relational database and that organizations find it cost effective to leverage the relational database towards BI.

The OLAP Option has been designed to be compatible with SQL-based BI applications. The SQL interface to OLAP cubes allows SQL-based applications to efficiently query cubes and gain access to their rich analytic content. End users benefit from excellent query performance and enhanced content in BI applications.

From the IT or application developer perspective, the OLAP Option is designed to allow organizations to easily substitute OLAP cubes and their views for relational tables. This allows the IT organization to switch to cubes and cube views with minimal costs and no disruption to the end user community. The organizations investment in BI tools is preserved.

### Direct SQL query of the cube

The full content of the cube – including summary data and calculations – can be queried by SQL-based applications that select directly from cube and dimension relational views. The analytic content embedded in the cube and revealed through views allows even the most basic SQL-based reporting tool to deliver high-end OLAP content. Simply think of it as relational objects that happen to offer improved performance and a lot of analytic content.

The SQL used to query cubes is very easy to write because calculation rules are embedded into the cube and as a result, do not need to be expressed in the query. For example:

- Measure calculations are simply revealed as additional fact columns in the cube fact view. The application can query sophisticated calculations by simply selecting the column of the cube fact view
- Aggregation rules are simply revealed as additional rows in the cube fact view. The application just needs to select data at the correct level of summarization; it does not need to include aggregation functions and GROUP BY in the query
- Partitioned outer joins are automatically and efficiently executed within the cube as needed by calculations embedded into the cube. For example, time series functions such as leads, lags and parallel periods are automatically ‘densified’ in the cube, thus eliminating the need for complicated outer join syntax in SQL queries

An example of accessing cube views using simple SQL:

```
SELECT t.long_description time,
       p.long_description product,
       cu.long_description customer,
       f.profit profit,
       f.profit_ytd,
       f.profit_ytd_yr_ago,
       f.profit_chg_ytd_yr_ago,
       f.profit_pctchg_ytd_yr_ago
FROM time_calendar_view t,
     product_primary_view p,
     customer_segment_view cu,
     channel_primary_view ch,
     units_cube_view f
WHERE t.dim_key = f.time
     AND p.dim_key = f.product
     AND cu.dim_key = f.customer
     AND ch.dim_key = f.channel
     AND t.level_name = 'CALENDAR_YEAR'
     AND p.level_name = 'CLASS'
     AND cu.level_name = 'TOTAL'
     AND ch.level_name = 'TOTAL';
```

This query accessing the OLAP cube views follows the style of a star query, but differs by the absence of aggregation operators (like SUM) and a GROUP\_BY clause (the query already returns summary data). Also note the filters on the \*.level\_name columns defining the correct level of summarization.

SQL query of the cube provides some important opportunities that are not usually available to BI tools querying relational tables. For instance, because joins are very efficiently execu-

ted in the cube it is very reasonable to issue multi-way joins, including queries that join to multiple fact views. Also, it is very easy to drill from aggregate level data in the cube to detail data in relational tables.

For example, here we used Oracle Application Express – which is not specifically aware of BI or cubes – to quickly build a drill enabled and responsive dashboard:

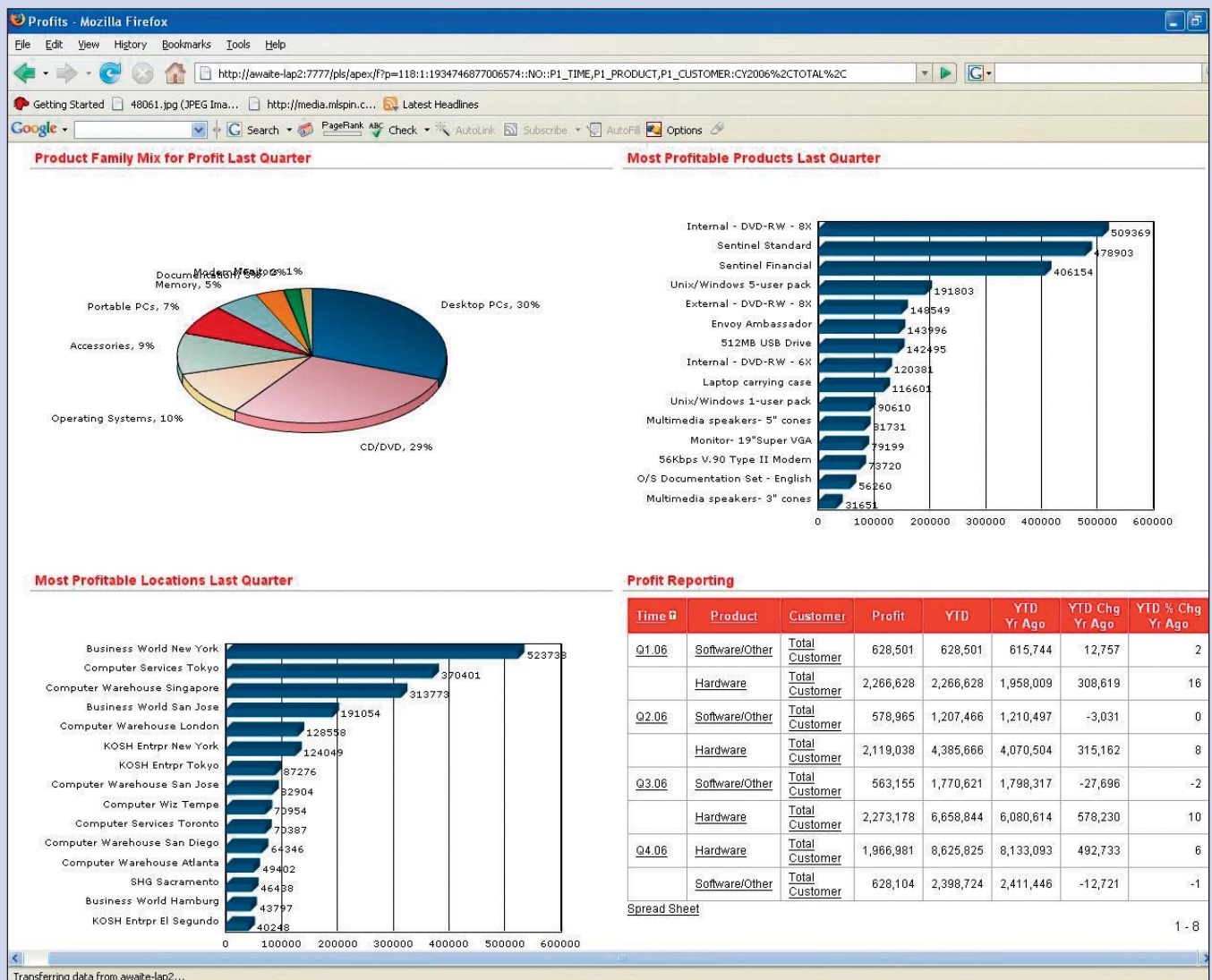


Figure 1: Oracle Application Express querying an OLAP cube using SQL

### Dimensional query of the cube

The OLAP Option supports dimensionally aware applications with the OLAP query API (Java) which presents

cubes and dimensions to applications in the context of the dimensional model (dimensions, hierarchies, cube and measures rather than columns and rows) and allows those applications to query in the cube with dimensional

style queries – with drill and pivot, multidimensional query filters and multidimensional calculations.

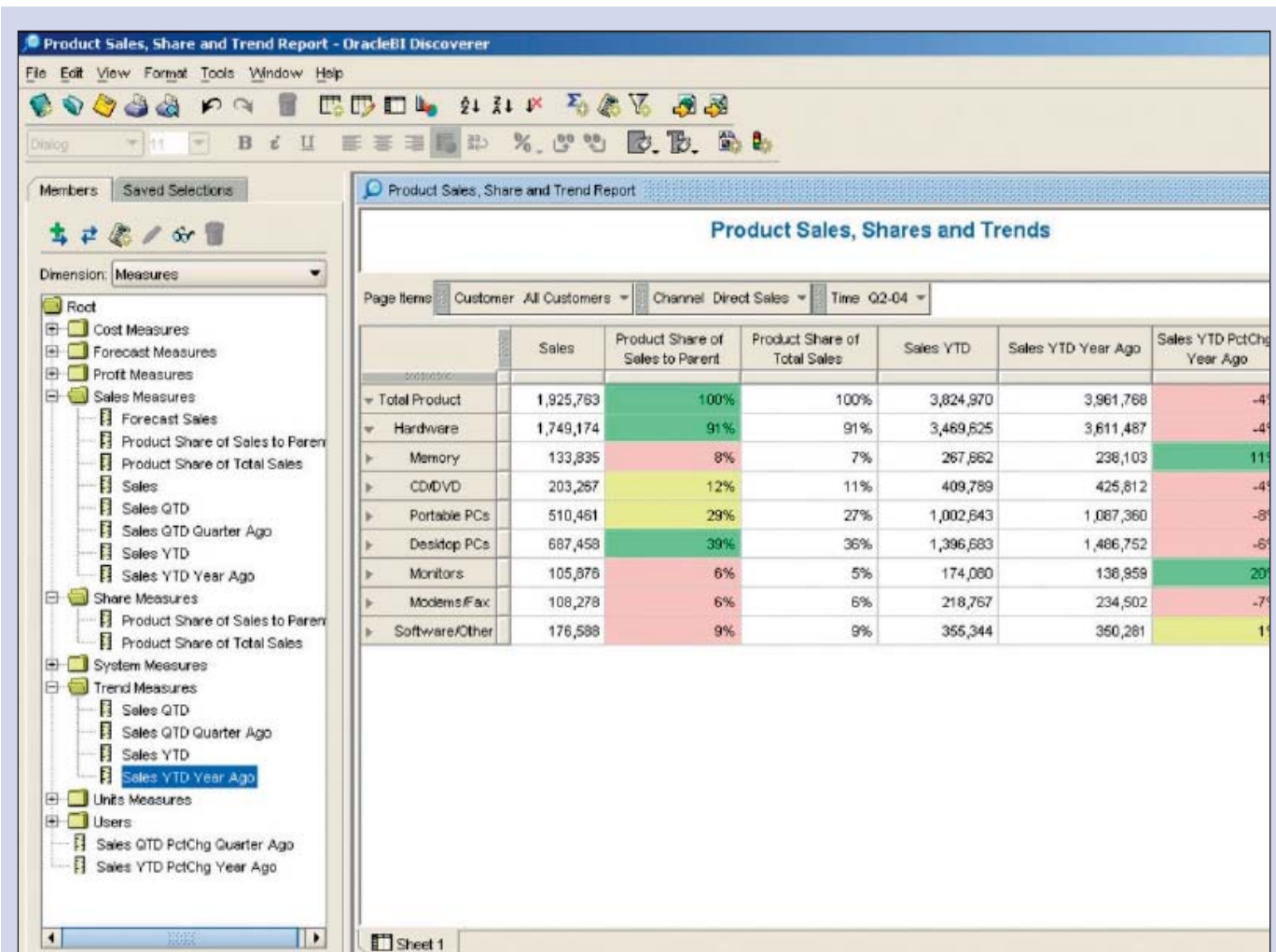


Figure 2: Oracle Discoverer Plus, an example of a dimensionally aware BI Tool querying OLAP

### Administer OLAP objects with the Analytic Workspace Manager

The AWM is an administrative tool that is used to design and manage OLAP cubes and dimensions. Using AWM, the DBA or application developer designs dimensions hierarchies, cubes, stored measures, calculated measures, aggregation rules, forecasting rules, allocations and security policies for cubes and dimensions. As part of this process, dimensions and cubes are often mapped to source relational tables.

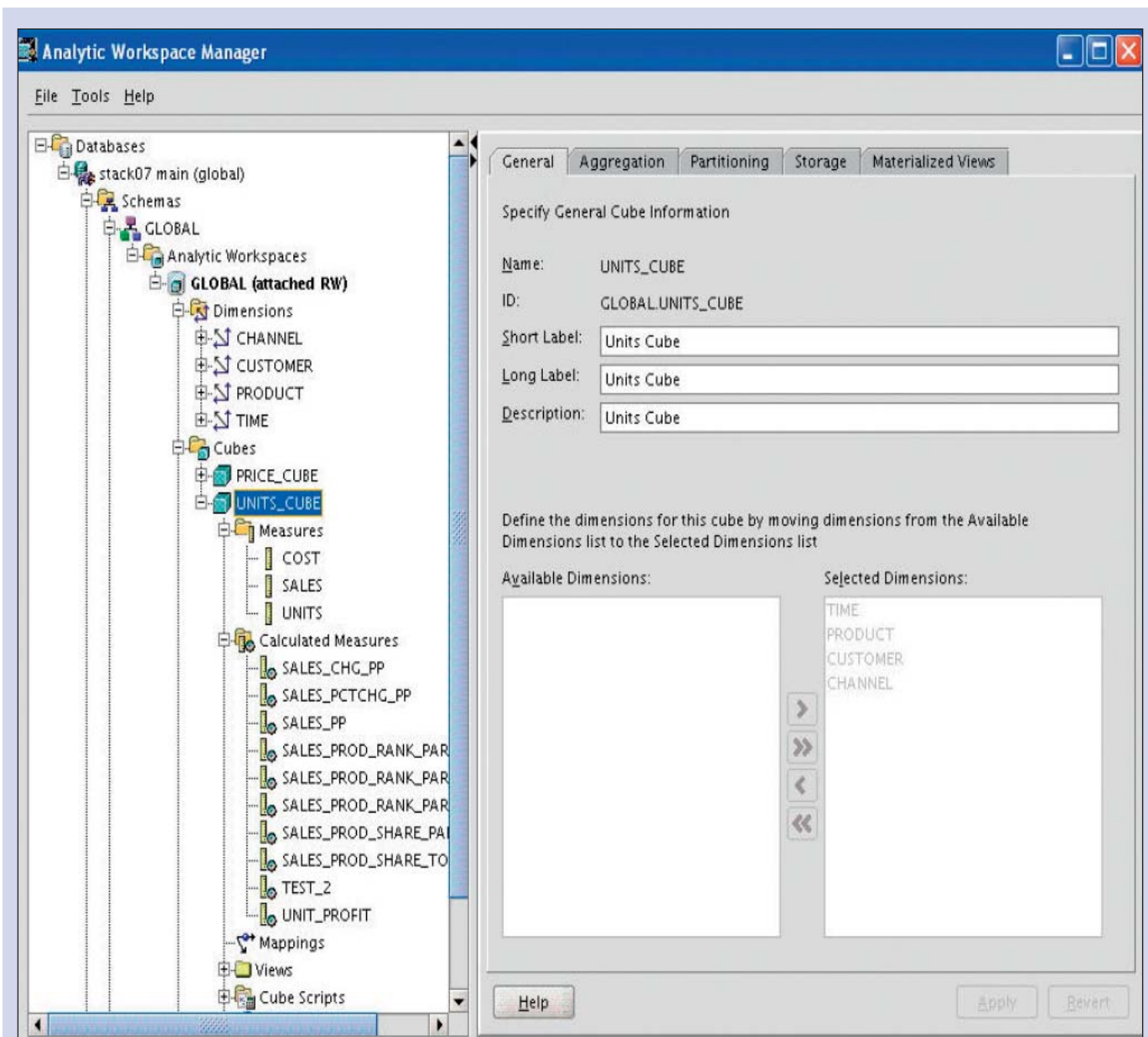


Figure 3: The Analytic Workspace Manager

The user may also use AWM to manage periodic refreshes of dimensions and cubes, to view data within the cube, to view relational views of the cube and to manage cube-organized materialized views.

## Conclusion

Benefiting from the architecture and features of the Oracle database, the OLAP Option provides organizations with the means to empower the BI tools and applications they already use with improved query performance and analytic content. Simple SQL access to the analytic content of the cube greatly improves developer productivity and simplifies application development.

## Recommended WhitePaper

«Improve SQL based Business Intelligence tools with Oracle OLAP 11g»

[http://www.oracle.com/technology/products/bi/olap/Oracle\\_OLAP\\_11g\\_TWP.pdf](http://www.oracle.com/technology/products/bi/olap/Oracle_OLAP_11g_TWP.pdf)

Or follow

<http://otn.oracle.com>

>Products / Database > 'BI & Data Warehousing' > 'Oracle OLAP' ■

## Contact

Oracle Software (Schweiz) GmbH

C. Bourboulas

E-Mail:

costantinos.bourboulas@oracle.com