

New Features and Performance Advances in the OLAP Option to the Oracle Database 10g

New features in Oracle Database 10g enhance support for large multidimensional data sets and SQL access to multidimensional data types

EXECUTIVE SUMMARY

Oracle9i Database Release 2 was the first, and is still the only, relational-multidimensional database. As a relational-multidimensional database, it combined a relational engine and relational data types with a full-featured multidimensional engine and multidimensional data types. The concept was simple – retain the advantages of a multidimensional database, solve the problems associated with stand-alone multidimensional databases and leverage the scalable, secure and resilient platform of the Oracle Database.

Advantages of multidimensional databases are compelling. They include a dimensional data model that simplifies the task of defining calculations and expressing queries, support for advanced multidimensional calculations and planning functions and a transaction model suitable for what-if analysis and modeling. Multidimensional databases are also known for excellent query response times.

Although the advantages of multidimensional databases are convincing, there are also numerous problems associated with stand-alone multidimensional databases. As compared with mature relational databases such as Oracle, they lack robust disaster recovery and high availability capabilities. Their security apparatus is immature. They typically replicate data already in the data warehouse. They require specialized query and reporting tools.

As a result of their deficiencies stand-alone multidimensional databases are typically an adjunct to the data warehouse rather than being part of the core data warehouse. After all, stand-alone multidimensional databases replicate small subsets of the data warehouse and they can't support the SQL based tools used to query the data warehouse. In order to be considered part of the data warehouse, multidimensional databases would need to provide support for very large data sets and query by SQL based tools.

The OLAP option to the Oracle9i Database Release 2 completely changed the multidimensional database market. It is not a stand-alone multidimensional database. Instead, a multidimensional engine and multidimensional data types were introduced into the kernel of the Oracle Database. The multidimensional technology shares the same platform as the relational technology. It is highly secure. It has mature high availability and disaster recovery features. It supports SQL as an interface to multidimensional data types.

Oracle OLAP multidimensional data types are first class data types in the Oracle Database. It is managed by the Oracle database management system, stored in Oracle data files and is accessible by SQL. As a result of this integration of the multidimensional data type, it is no longer necessary to host dimensionally modeled data in a stand-alone multidimensional database and OLAP can be considered an integral part of the data warehouse.

Developments to the OLAP option to the Oracle Database 10g focused primarily on those issues affecting the status of multidimensional data types within the data warehouse, scalability and the ability to support SQL as a query language.

Oracle Database 10g Release 1 first introduced support for features such as compressed cube technology, partitioned multidimensional data types, and parallelism within the cube building and aggregation process. The SQL interface was significantly enhanced to optimize support for a broader use of the SQL language when querying multidimensional data types.

Oracle Database 10g Release 2 extended support for compressed cube technology, further enhancing the scalability and applicability of multidimensional data types, and offers significant advances in query performance. The result is the ability to efficiently manage very large multidimensional data sets and to support a wide range of SQL based tools and applications with the OLAP Option to the Oracle Database 10g Release 2.

A BRIEF OVERVIEW OF THE OLAP OPTION

This section will provide a brief overview of the capabilities and architecture of the OLAP Option to the Oracle Database 10g Release 2.

Components of the OLAP Option

The OLAP Option is a component of the Oracle Database Enterprise Edition. When an instance of the Oracle Database is started all of the capabilities of the OLAP Option are available; there is no separate process for the OLAP Option. Likewise, all data managed by the OLAP Option is stored in Oracle data files; there are no separate data files to store OLAP data.

The main components of the OLAP Option are:

- The multidimensional ‘engine’ that manages the storage of multidimensional data types and provides the calculation capabilities of the OLAP Option.
- The Java API for Analytic Workspaces provides an API for building and managing dimensions and cubes.
- The Java OLAP API provides a dimensionally aware query interface.
- The SQL interface to multidimensional data types presents OLAP data to the SQL language for query.

Multidimensional Engine and Data Types

The multidimensional engine and data types provide support for complex, multidimensional calculations and planning functions. The multidimensional engine provides support for a wide range of functions such as non-additive aggregation methods, time series calculations, indices and statistical function. It also supports user defined analytic functions.

The multidimensional engine offers exceptional support for planning applications that require features such as forecasting and allocations. Also in support of planning applications, the multidimensional engine provides a 'read-repeatable' transaction model. This transaction model allows multiple users to simultaneously engage in what-if analysis sessions where they can make session level changes to both the data and the data dictionary.

The multidimensional engine utilizes array based data structures known as *variables* for data storage. These variables are true multidimensional data types stored in Oracle data files. They are very efficient in terms of data storage and query performance. Patented compressed cube technology, first introduced in Release 1 and extended in Release 2, is designed to offer excellent scalability and performance characteristics with the sparse data sets that are typical to a data warehouse.

The multidimensional engine provides a dimensionally aware calculation language known as the *OLAP DML*. This is a procedural programming language that can be used to express various types of calculations, design custom analytic functions, and control the data loading and calculation processes related to multidimensional data types. The OLAP DML is accessible through SQL and PL/SQL, as well as the OLAP Worksheet tool.

A collection of multidimensional data types (commonly referred to as cubes and dimensions) and OLAP DML programming code is stored in an *analytic workspace* within the database. An analytic workspace has two basic purposes. First, it is a container for a collection of multidimensional data types within a schema. Second, it plays a role in defining the scope of a read-repeatable transaction (or, in other words, the boundaries of a what-if session).

Java API for Analytic Workspaces

The Java API for Analytic Workspaces, new to Oracle Database 10g Release 1, is used to define the dimensions, cubes, measures and calculations of the dimensional model and, optionally, map them to source objects such as tables and views. The Java API for Analytic Workspaces also provides the data loading and calculation solving services used to manage the analytic workspace throughout its lifecycle.

The Java API for Analytic Workspaces is used by Analytic Workspace Manager and Oracle Warehouse Builder in the design and management of analytic workspaces.

OLAP API

The Java OLAP API is an object-oriented Java application programming interface used to query the dimensional. It provides a multidimensional object model and a broad range of classes and methods that allow applications to select, navigate and calculate multidimensional data. The Java OLAP API can be used to query both multidimensional (MOLAP) and relational (ROLAP) implementations of the dimensional model. The OLAP API is primarily targeted to the professional software development community.

SQL Interface to Multidimensional Data Types

The SQL interface to multidimensional data types allows an application to query cubes and dimensions with SQL. The SQL interface to multidimensional data types works by presenting certain elements of a SQL query, for example the select list and the where clause, to the multidimensional engine. The multidimensional engine accesses and calculates data according to the rules of the dimensional model and the returns the results of the query as rows.

At the core of the SQL interface to multidimensional data types is OLAP_TABLE, a table function. The SQL interface to multidimensional data types can be made transparent to the SQL application by creating a view that selects from OLAP_TABLE. Views may be designed in any number of styles ranging from views in the shape of a star schema to fully denormalized fact views that include dimensional structure and fact data in a single view.

OLAP_TABLE can also be selected from without using views, thus providing applications with the opportunity to interact with the multidimensional engine from within a select statement.

The following examples illustrates how a simple fact view might be created and queried.

```
create or replace view sales_view as
  select *
  from table(OLAP_TABLE('global DURATION session',
    '',
    '',
    'DIMENSION time_id AS varchar2(10)FROM time
    DIMENSION channel_id AS varchar2(10)FROM channel
    DIMENSION product_id AS varchar2(10)FROM product
    DIMENSION customer_id AS varchar2(10)FROM customer
    MEASURE sales AS varchar2(10)FROM sales
    MEASURE units AS varchar2(10)FROM units
    MEASURE extended_cost AS varchar2(10)FROM
      extended_cost
    MEASURE forecast_sales AS varchar2(10)FROM
      fcast_sales
    ROW2CELL olap_calc'));
```

The main features of a SELECT from OLAP_TABLE are the arguments that identify the analytic workspace ('GLOBAL', in this example), that specify the column names and data types and the mappings to the source object in the analytic workspace.

This sample view could then be queried with a SELECT statement such as:

```
select  time_id,
        channel_id,
        product_id,
        customer_id,
        sales
from    sales_view
where   time_id = '2003'
       and channel_id = 'CATALOG'
       and product_id in ('SALSA','CHIPS')
       and customer_id = 'TEXAS';
```

There are several methods that applications can use to interact with the multidimensional engine to define calculations and perform other tasks. One method is the inclusion of an OLAP DML expression in a select statement. The following example includes time series and a market share calculations.

```
select  product_id,
        time_id,
        sales,
        olap_expression
        (olap_calc, 'lagdif(sales,1,time,status)')
        as SALES_CHG_PRIOR_PRIOD,
        olap_expression
        (olap_calc, 'sales/sales(product '1') * 100')
        as PRODUCT_SHARE
from    sales_view
where   time_id = '2003'
       and channel_id = 'CATALOG'
       and product_id in ('GUNS','LIPSTICK')
       and customer_id = 'TEXAS';
```

In this example, the actual OLAP DML expression is in underlined text. Note the relative simplicity of the code – the dimensional data model makes this possible. The remainder of the bold text is the wrapper for the OLAP DML code that allows its use within the SQL select statement.

NEW FEATURES AND ENHANCEMENTS IN ORACLE10G

This document describes features new to Oracle Database 10g Release 1 and Oracle Database 10g Release 2 that enhance support for large multidimensional data sets and SQL access to multidimensional data types.

Enhanced Support for Very Large Multidimensional Data Sets

Oracle Database 10g brings cube compression technology and the time-tested techniques of partitioning and parallelism to multidimensional data sets. Cube compression technology optimizes aggregation and storage of multidimensional data types. Partitioning and parallelism allow for more efficient utilization of hardware resources and more efficient management of warehouses with large multidimensional data sets.

Compressed Cube Technology

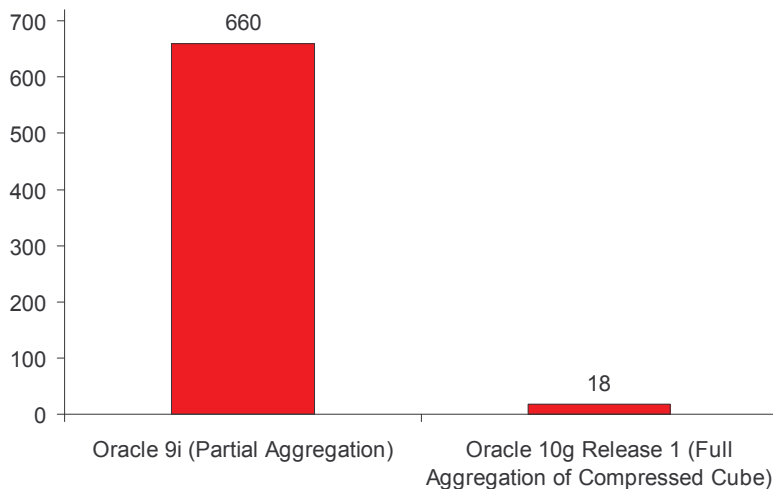
Oracle Database 10g Release 1 introduced patented compressed cube technology that optimized the aggregation, storage and query of the sparse data sets that are typical of OLAP and data warehouse data sets. A sparse data set is one where relatively few data points in the dimensional model have non-null values as compared to the model as a whole.

Consider, for example a cube that has 5 distribution channels, 120 months, 500 product items and 10,000 customers. Just at these lowest levels the model represents 250 million cells or data points ($5 * 120 * 500 * 10,000$). Since it is likely that customers purchased only certain products on only some days through perhaps one or two distribution channels, the data is likely to be extremely sparse at lower regions of the hierarchies. Data will tend to become more dense in upper regions of hierarchies as it is aggregated because the likelihood of a parent member having at least one non-null child increases.

The compressed cube technology takes advantage the sparsity patterns in hierarchical data sets by mapping more than one cell within a node of a hierarchy to a single stored value rather than replicating that same value up the hierarchy. The result is faster aggregation, more efficient storage on disk and faster query.

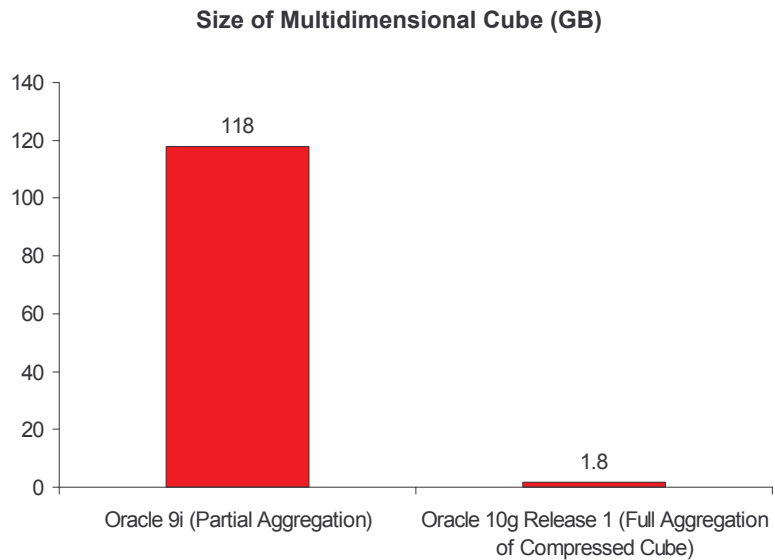
Results of cube compression can be dramatic. For one customer data set, which is fairly typical of a data set with sales and marketing data, both the amount of time required to prepare the data set for query and the size of the cube were significantly reduced. The following chart compares the amount of time needed to load and aggregate the data set in an Oracle9i Database Release 2 cube without compression with a Oracle Database 10g Release 1 compressed cube.

Time to Aggregate Multidimensional Cube (Minutes)



Minutes required to aggregate a customer data set in a multidimensional cube over all time periods. Note that the Oracle9i Database Release 2 case is partially aggregated while the Oracle Database 10g Release 1 case is fully aggregated. The fully aggregated cube will yield better query performance..

As might be expected, the size of the cube is also dramatically reduced by cube compression.



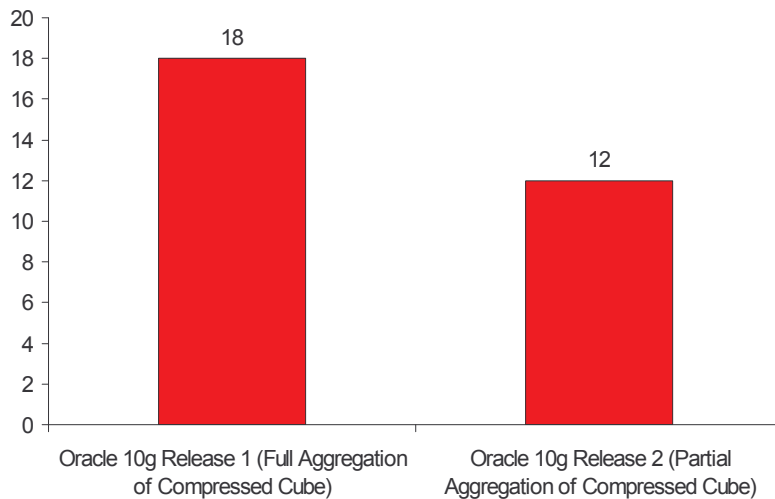
Sizes of non-compressed Oracle9i Database Release 2 cube and compressed Oracle Database 10g Release 1 cube

Compressed cubes in Oracle Database 10g Release 1 needed to be fully aggregated. That is, all summary level data was precalculated and stored in the compressed format. Uncompressed cubes could be partially preaggregated, with the remaining data being aggregated as needed when queried. Even so, the compressed cube can be fully aggregated in just a fraction of the time it takes for an uncompressed cube to be partially aggregated.

In Oracle Database 10g Release 2 compressed cubes are enhanced with dynamic aggregation and support for additional aggregation methods. The dynamic aggregation feature allows only certain regions of the compressed cube to be preaggregated as a query performance optimization; regions that are not preaggregated are automatically aggregated at runtime when queried.

Continuing with the previous customer example, the time to aggregate the compressed cube was reduced by 33 percent while retaining excellent query performance.

Time to Aggregate Multidimensional Cube (Minutes)



Partial aggregation of compressed cube supports further reductions in the time needed to aggregate the cube

Partitioning Cubes

Automatic partitioning and parallel processing of data loads and aggregation within the analytic workspace were introduced in Oracle Database 10g Release 1. With Oracle Database 10g Release 1 both compressed and uncompressed cubes can be partitioned and processed in parallel using the Java API for Analytic Workspaces (and administrative tools such as Analytic Workspace Manager which are based on this API). Furthermore, beginning with Oracle Database 10g Release 1, the AW\$ table which is used to store the analytic workspace can be physically partitioned to avoid I/O bottlenecks.

Partitioning Cubes in the Analytic Workspace

The Java API for Analytic Workspaces implements partitioning of uncompressed cubes using *partition templates*, a new feature added to the multidimensional engine in Oracle Database 10g Release 1. The partition template is used to define the partitioning strategy within the multidimensional engine. The most commonly used partitioning strategies are list and range partitioning.

Range partitioning allows data to be partitioned based on a range of dimension members. For example, one partition might contain time dimension members that are less than '13', another that are less than '25', and so on.

List partitioning allows data to be partitioned based on a list of specific dimension members. For example, a partition might contain dimension members <'JAN05','FEB05','MAR05'> and another partition might contain the members <'APR05','MAY05','JUN05'>.

When a partition template is used, the multidimensional engine automatically creates and deletes partitions as needed. When an application writes back to the

measure, the application writes to the logical measures and the multidimensional engine assigns data to partitions as appropriate. This makes it convenient for the application to write back to partitioned measures. The multidimensional engine also optimizes the write back process for better performance.

In Oracle Database 10g Release 1 the Java API for Analytic Workspaces implemented partitioning of compressed cubes as a series of separate multidimensional variables and conveniently presents these variables for query through a single formula which binds these variables together.

The Java API for Analytic Workspaces automatically maintains each of the separate variables in a manner similar to how the multidimensional engine manages the partitions of an uncompressed cube: it loads data into them, aggregates data within them and it automatically creates and drops partitions as needed. The main difference, from the application developers point of view, is that the application must write data to the correct variable representing the compressed cube partition.

In Oracle Database 10g Release 2 the multidimensional engine partitions compressed cubes using partition templates. As a result, the multidimensional engine automatically assigns data to partitions when the application writes back to the measure. This is much more convenient for the application developer who will be writing back to measures in partitioned compressed cubes.

Partitioning the AW\$ Table

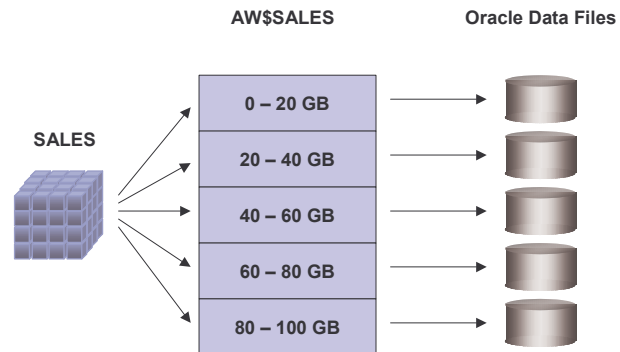
Partitioning multidimensional data types within the analytic workspace allows multiple processes to maintain dimensions and cubes. I/O bottlenecks can be avoided by partitioning data on disk by partitioning the AW\$ table and distributing the partitions across the disk system.

Analytic workspaces are stored in the Oracle database in an AW\$ table. This table uses BLOB data types to store the data of the multidimensional data types in an analytic workspace. The AW\$ table can only be written to and read from by the multidimensional engine of the OLAP Option to the Oracle Database. There is one AW\$ table for each analytic workspace.

When an Oracle9i Database Release 2 analytic workspace exceeds a certain size, additional data is stored in a new row in the AW\$ table. The Oracle9i Database Release 2 analytic workspace can be partitioned across multiple rows in the AW\$ table by specifying a maximize segment size (the maximize amount of data for any particular row). This feature was required to support large analytic workspaces since there is a size limit for each row of a BLOB data type. The AW\$ table itself could be partitioned using relational table partitioning features to reduce I/O bottlenecks.

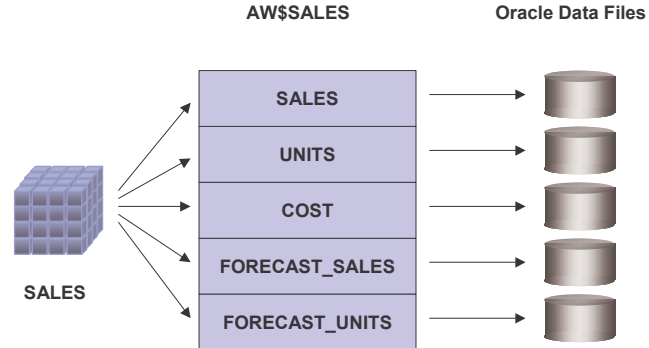
In Oracle9i Database Release 2, the extent of the database administrator's control over this form of partitioning is limited to specifying the maximum segment size; the multidimensional engine automatically distributed data across the rows of the AW\$ table.

For example, if there is an analytic workspace named SALES with a segment size of 20GB, each row in the AW\$ table will contain a maximum of 20GB of data. The AW\$SALES table could be partitioned using table partitioning as shown in the following illustration.



AW\$ table partitioning in Oracle9i Database Release 2

In Oracle Database 10g Release 1 the storage model was enhanced such that each object in an analytic workspace is stored in a separate row in the AW\$ table. If a measure belongs to a partitioned cube, each partition is stored in a separate row in the AW\$ table. Objects will be further partitioned by segment size if the size of the multidimensional object grows too large for a single row. This all occurs automatically by the multidimensional engine. Like Oracle9i, the AW\$ table can then be partitioned across multiple data files.



AW\$ table partitioning in Oracle Database 10g

With the enhanced storage model database administrators have complete control over how data is distributed across data files and can optimize I/O for data update and data access patterns. For example, data from two different cubes might be stored on separate disks to eliminate contention for disk access when these cubes are being updated by parallel processes.

Separation of objects into different rows of the AW\$ table is also forms the basis for the multi-write access mode which is new to Oracle Database 10g Release 1

since the multidimensional engine can use row level locking as the basis for locking objects in the analytic workspace.

Parallel Processing

Parallel processing of cubes is available both within the multidimensional engine and the Java API for Analytic Workspaces.

The Java API for Analytic Workspaces supports parallelization of data loading and aggregation when a maintenance procedure involves more than one cube and when a cube is partitioned. If there is more than one cube being processed in a maintenance procedure, each cube can be processed separately and in parallel. When a cube is partitioned, each partition can be processed in parallel.

Analytic Workspace Manager allows for the specification of the number of parallel processes that may be used within a maintenance procedure. Dimensions are always processed by the Java API for Analytic Workspaces before cubes to avoid locking conflicts when cubes are processed in parallel. The multidimensional engine can automatically parallelize certain phases of aggregation within a cube or a partition.

When data is written to an analytic workspace during a session the changed data is stored in a temporary LOB in the database. When the UPDATE command is used to permanently save the changes made to the analytic workspace the changed data is moved from the temporary LOB storage to the AW\$ table. Since there can be large volumes of data involved with an UPDATE it is useful to be able to process the update in parallel. In Oracle Database 10g Release 1 the UPDATE is run in parallel on multiple objects in the analytic workspace whenever possible. On a system with adequate I/O performance parallel UPDATE offers significant performance benefits.

Incremental Updates

The Java API for Analytic Workspaces and the multidimensional engine each provide support for incremental updates of cubes and measures. All support for incremental update is automatic.

With both uncompressed and compressed cubes it has always been possible to incrementally load fresh data for both new and historical time periods. In Oracle Database 10g Release 1 the behavior of aggregation depends on whether the cube is compressed or uncompressed. If the cube is uncompressed, the only data that are aggregated are the parents and ancestors of the newly loaded data regardless of whether the cube is partitioned or not.

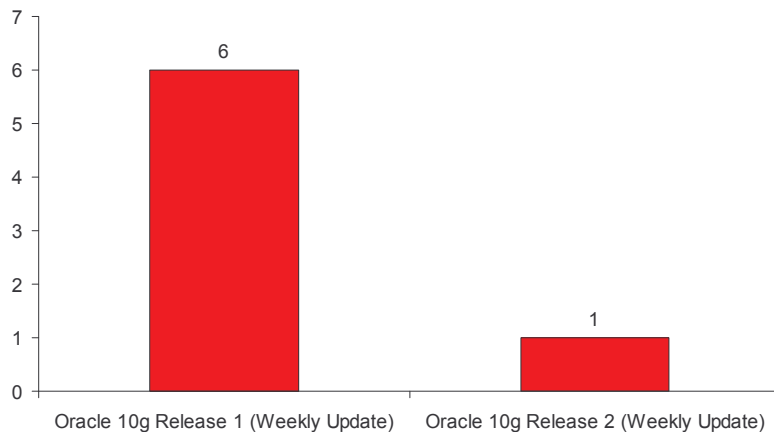
If the Oracle 10g Release 1 cube is compressed and is not partitioned, the entire cube will be aggregated whenever data is loaded into the cube. If the Oracle 10g Release 1 cube is partitioned, only the partitioned that received new data will be aggregated. As a result, the practice of partitioning Oracle 10g Release 1 cube can yield significantly shorter aggregation times after a cube has been updated.

In Oracle Database 10g Release 2 data are incremented aggregated only to the parents and ancestors of the newly loaded data in both compressed and uncompressed cubes regardless of whether the cube is partitioned or not. This can yield significantly shorter aggregation times in the following common cases:

- When a cube is compressed and is not partitioned.
- When a cube is compressed and partitioned, but only a relatively small amount of data in the cube has been updated.
- When a cube is compressed and partitioned, but the changed data runs against the grain of partitioning (for example, is the cube was partitioned by geography and was updated with new time periods).

Returning to our customer example, update performance of Oracle Database 10g Release 1 and Oracle Database 10g Release 2 can be compared by examining an update in which one week of new data are added to the analytic workspace. The cube is compressed and partitioned by three years. When the Oracle Database 10g Release 1 is analytic workspace updated the entire partition containing that week is processed. When the Oracle Database 10g Release 2 analytic workspace is updated only the parents and ancestors of the newly loaded data is processed. This results in significantly lower processing times.

Time to Incrementally Aggregate Multidimensional Cube (Minutes)



Time required for weekly update of cube in Oracle Database 10g Release 1 and Oracle Database 10g Release 2. In the Oracle Database 10g Release 1 case the entire partition is processed. In the Oracle Database 10g Release 2 case only the new data and its ancestors are processed.

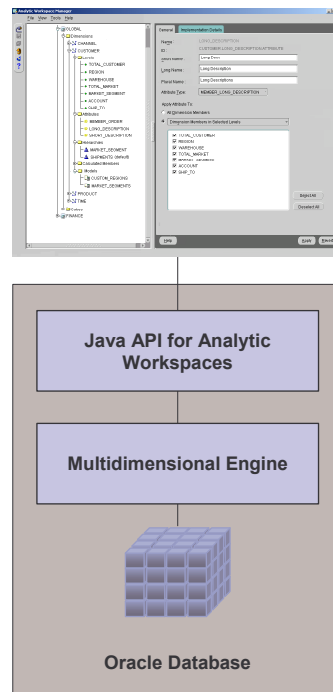
There are several benefits to highly efficient processing of incremental updates. Data sets can be larger. Analytic workspaces can be updated more frequently. What-if analysis where lower level data is updated and summary level data is queried can occur more quickly and more frequently. In many situations the OLAP Option to the Oracle Database 10g Release 2 supports near real-time analysis.

It should be noted that this is just one example of performance characteristics of the compressed cube technology. The purpose of these examples is to highlight the advances in storage and query technology of the OLAP option from Oracle 9i Database Release 2 through Oracle Database 10g Release 2. Actual results are dependant on the particular data model and data.

APIs and Tools That Make Power Accessible

Partitioning, parallelism and cube compression provide the technology to support very large multidimensional data sets. It would be fair to ask how easy it is to leverage this technology in a real business intelligence solution. That is, would the average database administrator or line of business user who needs to design and implement analytic workspaces be able to take advantage of these scalability features? The answer to this question is 'yes'.

For the application developer, the Java API for Analytic Workspaces takes the responsibility of creating the optimal physical implementation of an analytic workspace and maintaining it throughout its lifecycle. Tools such as Analytic Workspace Manager and Oracle Warehouse Builder reveal the power of the Java API for Analytic Workspaces and the multidimensional engine in graphical administrative tools.



Layers of administrative tools and API for the design and management of Oracle Database 10g analytic workspaces.

Java API for Analytic Workspaces

The Java API for Analytic Workspaces, introduced in Oracle Database 10g Release 1, provides services that simplify the process of defining the structure and calculation rules of the dimensional model. Using the Java API for Analytic Workspaces the application developer describes only the logical dimensional model. The Java API for Analytic Workspaces automatically implements the dimensional model and calculation rules in the analytic workspace according to best practices and the most current features of the database. For example, by providing a few simple hints the application developer can indicate that cube compression should be used and that the cube should be partitioned by months in the time dimension.

As an example of how the Java API for Analytic Workspaces automatically optimizes the physical design of the analytic workspace for each new version of the Oracle Database, the implementation of a partitioned compressed cube in Oracle Database 10g Release 2 shifts to the newer, more efficient partition template. No changes to application level code is required to take advantage of this new Oracle Database 10g Release 2 feature. This is but one example of how the Java API for Analytic Workspaces enhances the scalability and power of the analytic workspace by making high end features and state of the art analytic workspace implementation readily accessible to the application developer.

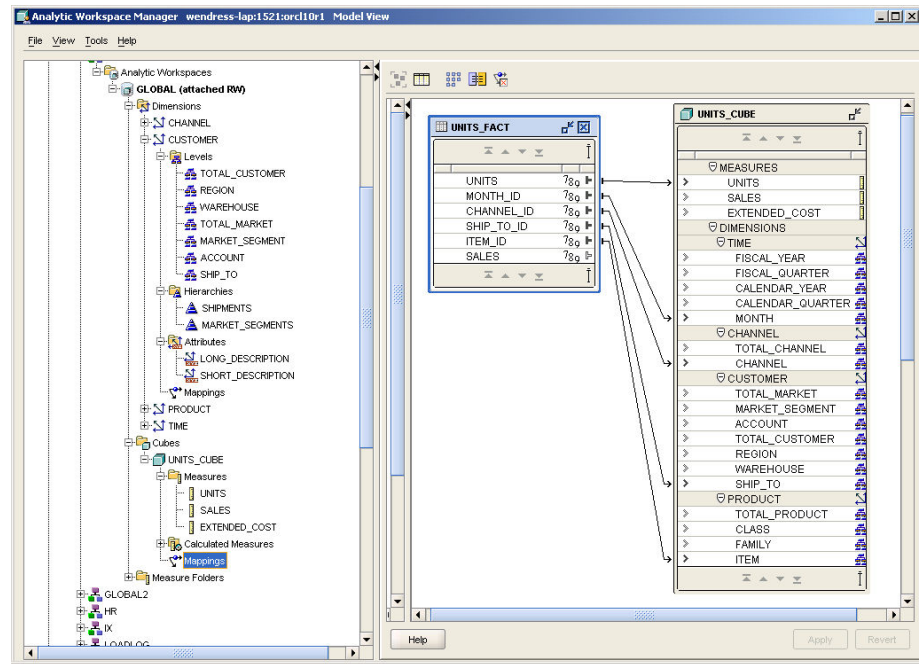
Analytic Workspace Manager and Oracle Warehouse Builder

Oracle offers two administrative tools for the design and implementation of analytic workspaces, Oracle Warehouse Builder and Analytic Workspace Manager. For Oracle Database 10g Release 2 both have been upgraded to use the Java API for Analytic Workspace for each new version of the Oracle Database. The adoption of the Java API for Analytic Workspaces allows Oracle Warehouse Builder and Analytic Workspace Manager to produce highly scalable and efficient analytic workspaces.

Oracle Warehouse Builder allows the user to design the dimensional model and deploy it directly to an analytic workspace through a sophisticated extraction, transformation and loading (ETL) process. With Oracle Warehouse Builder the analytic workspace is a deployment of a data warehouse. Oracle Warehouse Builder is typically used by IT organizations that want to embed dimensional modeling and multidimensional data types in the process of defining and managing a data warehouse.

Analytic Workspace Manager is an administrative tool that is focused entirely on the definition and implementation of the dimensional model in an analytic workspace. It has an interface dedicated to dimensional modeling and does not offer ETL services. It tends to be used by line of business users who want to create cubes from data that has been produced by an ETL process. Analytic Workspace Manager produces the same highly efficient analytic workspace implementation that Oracle Warehouse Builder does.

Analytic Workspace Manager received a major upgrade in Oracle Database 10g Release 1 with the addition of the Model View. Line of business users who can describe their dimensional model from the perspective of their end user reporting requirements will find that the new Model View puts the design of analytic workspaces within easy reach.



The new Model View of Analytic Workspace Manager displaying the outline of the dimensional model and the mapping of a cube to a fact table.

SQL INTERFACE TO MULTIDIMENSIONAL DATA TYPES

The ability to manage large multidimensional data sets meets is critical requirement for OLAP in the data warehouse. Another important requirement is the ability to query multidimensional data types using SQL.

The OLAP Option to the Oracle Database 10g Release 2 provides both an OLAP API and a SQL interface for query of multidimensional data types. The purpose of the OLAP API is to provide support for query in the context of the dimensional model. It is the API of choice for those applications that are based on the dimensional model and that require services for dimensional query, data navigation and calculation definition.

The SQL interface to multidimensional data types allows a SQL based application query data stored and calculated in an analytic workspace. The SQL based application might have no awareness of the dimensional model or it might understand dimensional concepts such as dimensions, hierarchies and cubes. The SQL application might choose to interact with the multidimensional engine during

a SQL query or it might not. The OLAP Option to the Oracle Database 10g provides support for each of these styles of query.

The SQL interface to multidimensional data types received a major upgrade in Oracle Database 10g Release 1 with optimization of WHERE clauses, SQL MODEL optimizations and automatic generation of abstract data types. In addition, the query equivalency feature allows for query rewrite to analytic workspaces.

WHERE Clause Optimizations

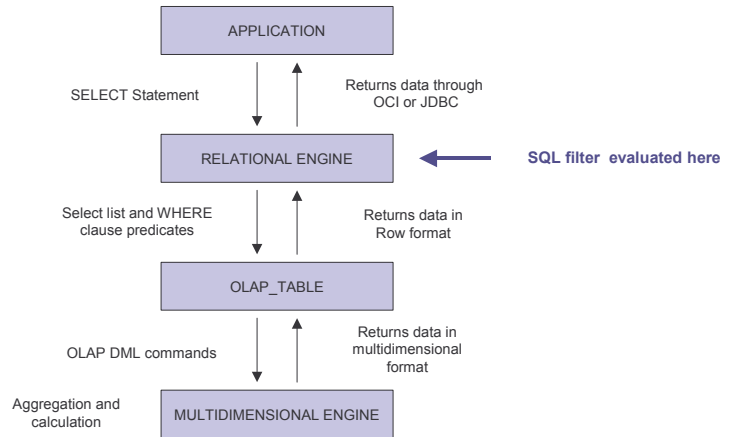
The first requirement of SQL query of multidimensional data types is fidelity to the relational model. That is, a SQL query to a multidimensional data type should yield the same result as the same SQL query to a relational data type if both data types have the same data. The second requirement is excellent query performance.

The architecture of the SQL interface ensures that any select statement will run against multidimensional data types and that the results are the same as with the same select statement against a relational table. Key to this architecture is the layering of the relational engine over both the object technology of the Oracle 10g Database and the multidimensional engine.

In this architecture, the object technology of the Oracle Database is used to redirect processing of a SQL query to the multidimensional engine when selecting from multidimensional data types. The multidimensional engine applies predicates from the WHERE clause, performs any necessary calculations and returns data to the OLAP_TABLE table function. OLAP_TABLE converts the data to a row set and passes it to the relational engine for any additional processing that might be needed.

Because not all functions and predicates can be transformed to and executed in the multidimensional engine, it is critical that the relational engine apply functions and filters in order to ensure 100 percent accurate results. The following illustration shows the processing steps and notes how SQL filters are evaluated in the relational engine in Oracle9i Release 2.

The query process starts with the issuance of a SELECT statement against the database and is completed with data beginning returned to the client.



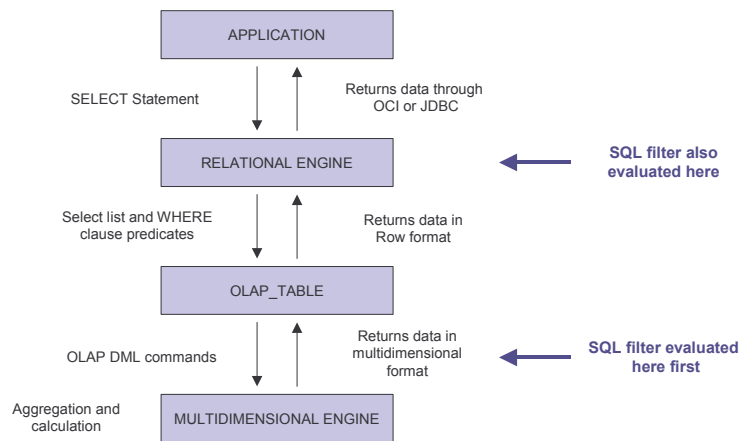
Processing SQL against multidimensional data types in Oracle9i Database Release 2

Since the relational engine always applies functions and filters on the data returned from multidimensional data types it can be guaranteed that results of the select statement will be the same as if data was selected from a table or view. If, for example, OLAP_TABLE could not process a certain filter to within the multidimensional engine the filter will be processed by the relational engine.

While the Oracle9i Database will always return the correct results when using SQL to access multidimensional data types, applications needed to be aware of the styles of SQL that would be processed efficiently.

Application of SQL Filter by Multidimensional Engine

In Oracle Database 10g Release 1 the SQL interface is enhanced to optimize a wider range of SQL predicates when selecting from multidimensional data types. This is accomplished by applying SQL filters to data before the data are converted to a row set by the OLAP_TABLE table function. The risk of pushing large volumes of data through OLAP_TABLE is minimized and query performance improves. As a result, a wider variety of SQL applications can be used with the OLAP option without special considerations.



Processing SQL against multidimensional data types in Oracle Database 10g

Note that like Oracle9i Database Release 2, the SQL filter is always evaluated in the relational engine to ensure that any SQL can be executed against multidimensional data types. The overall query process is optimized by reducing the amount of data transported through OLAP_TABLE and by reducing the number of rows that must be filtered in the relational engine.

Optimization of WHERE Clause in Hierarchical Queries

Dimensional queries typically include predicates that select data based on elements of dimension hierarchies. For example, it is very common for dimensional queries to filter based in a members parent or ancestors or based on whether a member belongs to a hierarchy. Oracle Database 10g Release 2 provides performance optimizations to support hierarchical queries on very large dimensions.

Query Rewrite to Views Over Multidimensional Data Types

Query rewrite and materialized views revolutionized how SQL based applications approached querying summary level data in the data warehouse. Before query rewrite, SQL based applications needed to either navigate summary tables themselves or summarize data at runtime using GROUP BY. Mapping to summary tables might be very tedious for the database administrator. Navigating summary tables can be very expensive for the application when the data model was complex.

Query rewrite, first introduced in Oracle8i, allows applications to defer navigation of summary tables to the Oracle Database. Instead of mapping the application to the summary tables, the application can map to only the detail level fact table. When the application issues SQL with a GROUP BY, the database will automatically rewrite the query to the materialized view. This dramatically simplified the maintenance of application metadata and often improved query performance.

In Oracle8i and Oracle9i query rewrite is limited to materialized views. A materialized view is a table that is registered in the data dictionary for use with

query rewrite. It was not possible to use query rewrite with other objects such as views.

In Oracle Database 10g Release 1 a new feature, *query equivalence*, allows query rewrite to be used with views. With query equivalence, the DBA indicates to the database what SQL *could have* been used to create the view even if the view was created in some other way. For example, if the application typically emits SQL with SUM ... GROUP BY, but the view is created with entirely different SQL, the DBA can indicate that the view is equivalent to SUM ... GROUP BY and will match queries to the view based on that equivalence.

This feature of the database is extremely useful with the OLAP option since SQL access to multidimensional data types is often through views. The DBA can create a view over an analytic workspace with syntax such as:

```
SELECT TIME, PRODUCT, CUSTOMER, SALES  
FROM OLAP_TABLE ...
```

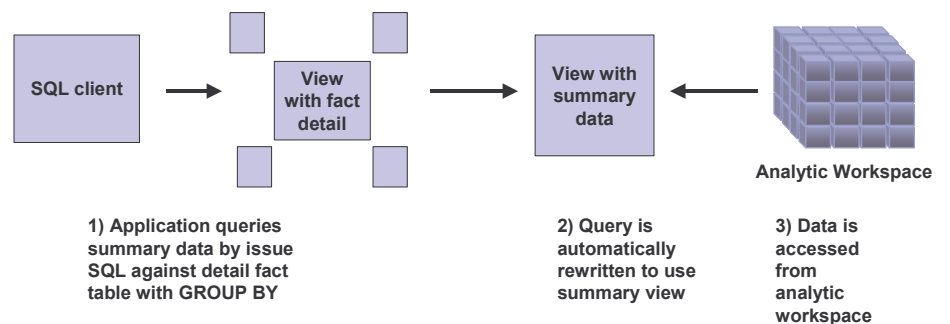
And indicate to the database that the view is equivalent to:

```
SELECT TIME.TIME, PRODUCT.PRODUCT, CUSTOMER.CUSTOMER,  
SUM(FACT.SALES) ...  
GROUP BY ...
```

If the application issues a query that is consistent with the equivalence of the view, such as the example below, the query will be automatically rewritten to the view over the analytic workspace.

```
SELECT TIME.TIME, PRODUCT.PRODUCT, CUSTOMER.CUSTOMER,  
SUM(FACT.SALES) ...  
GROUP BY ...
```

This provides the DBA and application with the benefits of simplified maintenance and improved query performance similar applications that rely on query rewrite into materialized views. The process of querying an analytic workspace using views and query equivalence is illustrated below.



Query processing using query rewrite to a view over an analytic workspace

Automatic Runtime Generation of Abstract Data Types

Abstract data types are used by object technology of the Oracle Database to define the relational columns for data that is returned from a non-relational data source. In the case of the OLAP option, abstract data types describe data being selected from analytic workspaces in terms of relational columns.

In Oracle9i Database Release 2, it is a requirement that abstract data types be created as part of the administrative process of enabling analytic workspaces for query by SQL. To provide applications and database administrators with additional flexibility in the administration of SQL access to analytic workspaces, Oracle10g Database supports automatic runtime generation of abstract data types as part of the query process.

As a default, the data are returned using the analytic workspace object name and data type as the relational column name and data type. Extensions to the limit map argument of OLAP_TABLE provide control over column names and data types the data. The use of automatically generated abstract data types with OLAP_TABLE is optional; predefined abstract data types can still be used.

With the addition of this new feature, it is now possible to query analytic workspaces without requiring the database administrator to predefine either abstract data types or views. This reduces the workload of the database administrator and adds considerable flexibility to the SQL interface to multidimensional data types.

CONCLUSION

The OLAP Option to the Oracle10g Database Release 2 represents a truly unique offering to the OLAP market. It offers an industrial-strength calculation engine and performance unmatched by any stand-alone multidimensional database, yet it does so in the context of the more reliable and more secure platform of the Oracle Database. If simply playing the role of a stand-alone OLAP server is all that is required the OLAP option, with all its calculation power and the OLAP API, would be more than sufficient. Oracle's objective, however, is for OLAP to be a central component to the data warehouse rather than as an add-on to the data warehouse.

To this end, the OLAP Option to the Oracle Database 10g Release 2 is designed in such a way that it allows multidimensional data types to be implemented as part of the data warehouse. This means that all of the calculation power of the OLAP option can be obtained without the requirement that data always be replicated from relational tables to separate cubes.

There are several core requirements that have been met in order to make this vision a reality. The system is highly secure and extremely reliable, multidimensional data types are managed side by side with other data types in the database, it supports large multidimensional data sets and can be queried using SQL. Much of this vision was seen in Oracle9i Database Release 2 with the integration of the

multidimensional engine into the Oracle kernel. In Oracle10g Database the role of multidimensional data types in the warehouse is solidified by support for very large dimensional data sets and additional support for SQL access to multidimensional data types.

Key to the support of large multidimensional data types are compressed cube technology, partitioning, parallel processing and enhancements to incremental processing and the physical storage model of multidimensional data types. Combining these features with other features such as Real Application Clusters and Oracle Grid Computing additional opportunities to support very large multidimensional data sets.

The other half of the equation – data access – is enhanced through the optimization of SQL access to multidimensional data types. Oracle10g Database Release 2 builds on the Oracle9i implementation by moving the processing of SQL filters to a point closer to the multidimensional engine. Management of summary level data is dramatically simplified through the use of query equivalence and query rewrite to views over multidimensional data types. As a result, a wider variety of SQL based applications will be able to work with the OLAP option without optimizing SQL to multidimensional data types.