Climbing to the OLAP Summit with Oracle Warehouse Builder 10gR2

An Oracle White Paper
January 2006

ORACLE

Note:

This document is for informational purposes. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described in this document remains at the sole discretion of Oracle.

Climbing to the OLAP Summit with Oracle Warehouse Build 10g

Climbing to the OLAP Summit with Oracle Warehouse Builder 10gR2

EXECUTIVE OVERVIEW

The aim of this whitepaper is to review how Oracle Warehouse Builder 10gR2 can help companies successfully implement new projects and extend existing projects to include OLAP (multi-dimensional) data models.

Although many companies recognize the benefits of providing multi-dimensional models within a data warehouse, to date the design, implementation and life-cycle management of these models has typically been handled outside the domain of normal ETL tools. This has resulted in a huge disconnect between the multi-dimensional model and its supporting detailed relational schemas. It also has resulted in increased the costs of building and maintaining OLAP based reporting environments since they have required additional hardware, skilled sets. This severely limits the number of organizations that can be run an efficient enterprise scalable data warehouse environments that also incorporates a multi-dimensional model.

With the introduction of 9i of the database Oracle provided a unified relational and multi-dimensional server. Oracle Warehouse builder supported this unified environment with its dimensional and cube based modeling features. However, with the availability of the next generation of Warehouse Builder the life cycle management of multi-dimensional models has moved to the next level.

Oracle Warehouse Builder is the first ETL product to provide a single integrated and complete environment for managing enterprise data warehouse solutions that also incorporate multi-dimensional schemas. The extensions to the dimensional modeling capabilities have been built on established relational concepts, with the option to seamlessly move from a relational deployment model to a multi-dimensional model at the click of a button.

This now means that ETL designers can logically model a complete data warehouse solution using one single tool and control the physical implementation of a logical model at deployment time. As a result data warehouse projects that need to provide a multi-dimensional model as part of the overall solution, can be designed and implemented faster and more efficiently.

INTRODUCTION

Who needs OLAP? Multi-dimensional, data that can provide answers to an organization's specific business questions is valuable to both IT and business users. These multi-dimensional schemas are typically user to extend and enhance an organization's ability to answer a broad range of business related questions:

- How do sales for our five most profitable products across the US, for this quarter compare with sales a year ago?
- What are the differences in the product-sales mix between the regions relative to the global sales mix?
- In what ways does the mix vary by salesperson, and what is the relative performance of our salespeople?
- What are our forecast units, unit price per service, unit costs per product, sales, costs and profits for the next 12 months?

These types of questions involve adding additional analytical calculations to the base data and enhancing the results by examining inter-time period relationships. The answers from the these questions usually result in additional drill and pivot operations as users mine their cubes following ad-hoc query paths. This type of environment is perfect for OLAP solutions where query performance has to be maintained despite the ad-hoc nature of queries. In many cases a multi-dimensional model is able to manage the business calculations in a more performant manner due to an extremely rich metadata model and because the business rules are embedded directly within the metadata.

From a business user perspective, multi-dimensional models are the perfect solution, enabling them to design and manager their own queries and calculations with little or no additional input from IT users.

From an IT perspective, OLAP implementations are problematic because they require additional hardware, DBAs who are skilled at using the specialized administrative tools, and ETL experts familiar with the bespoke multidimensional design languages. This severely limits the number of organizations that can be run an efficient enterprise scalable data warehouse environments that also incorporates a multi-dimensional model.

However, multidimensional technology has been available within the Oracle database for a number of years. Organizations no longer need to choose between a multidimensional OLAP versus a relational database. By integrating multidimensional objects directly within the relational database, Oracle provides the power of multidimensional analysis with the manageability, scalability, and reliability of the Oracle Database.

Along with an integrated database, Oracle now provides the first ETL product to provide a single integrated and complete environment for managing enterprise data warehouse solutions that also incorporate multi-dimensional schemas. Warehouse Builder 10gR2 now provides extensions to the dimensional modeling capabilities.

These additional features of the multi-dimensional model have been built on established relational concepts, with the option to seamlessly move from one a relational deployment model to a multi-dimensional model.

As a result ETL Designers do not need to learn any new APIs or additional proprietary multi-dimensional commands. Warehouse Builder can directly load data into a multi-dimensional model using the full power of its transformation library. The load process can be integrated into the normal ETL process flows. Full lineage and impact analysis for life-cycle management has also been incorporated.

The storage and deployment options for cubes and dimension contain intelligent defaults to minimize the time spent configuring the physical model. However, as with relational implementations, Warehouse Builder exposes all the multi-dimensional tuning and configuration options, which are based on traditional relational concepts of data partitioning and data compression. The following sections of this document will explain the features of the new multi-dimensional model and how to extend the basic model to include additional calculations such as % growth, year to date, share, rank and forecast calculations.

CREATING A MULTI-DIMENSIONAL MODEL

Warehouse Builder 10gR2 makes it very easy to build a multi-dimensional model. ETL designers that are familiar with current relational dimensional modeling techniques will quickly be able to design and deploy a multi-dimensional model.

Two methods are provided for creating dimensions and cubes: a wizard and/or manual configuration using the Data Object Editor. The aim of the wizard to is to quickly define either a cube or a dimension; most settings and options are intelligently defaulted by Warehouse Builder. The Data Object Editor provides full control over the design and implementation options, allowing the designer to override any of the default settings.

Creating a Dimension via the Wizard

The dimension wizard requires just six simple steps to define a dimension and is the basically the same for both multi-dimensional and relational dimensions. A new step has been added to this wizard for 10g, which is used to determine the actual type of implementation: ROLAP or MOLAP. This can be changed later and is only used to control how subsequent questions presented in the wizard train.

In a MOLAP, multi-dimensional, implementation, the dimensional object data is stored in an analytic workspace as an object of type *Dimension*. Each dimension is implemented according to the guidelines of "OLAP Standard Form", which defines how the dimension should be implemented in terms of the additional objects required to support the dimension. For example objects may be required created to store the parent relationship, level relationship, long and short descriptions, attribute descriptions and hierarchy dimensions etc. All these additional objects are transparently managed by the database.

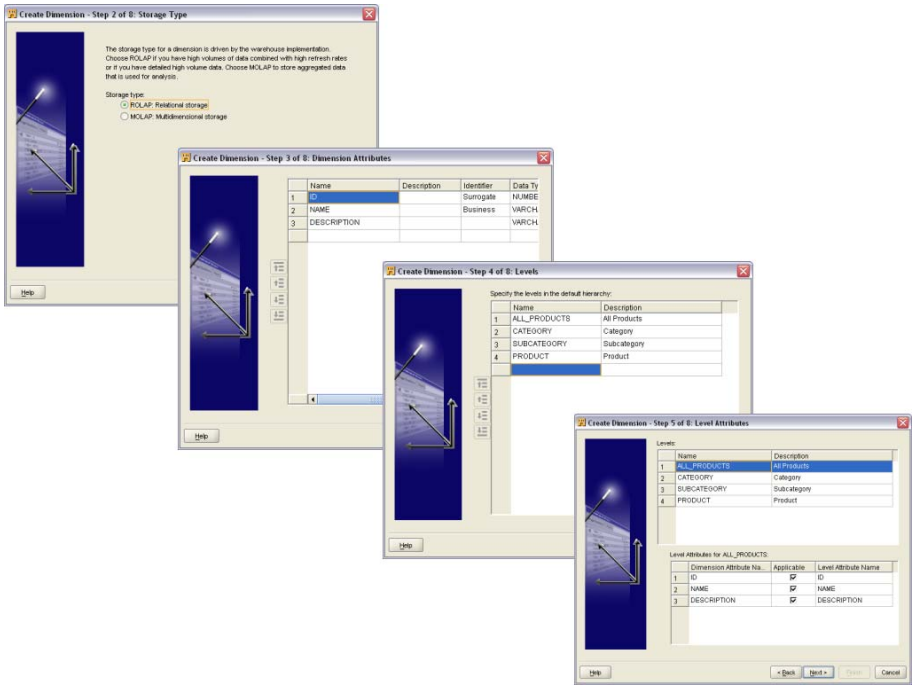

Figure 1 'Create New Dimension' Wizard

Extending the concept of attributes?

A dimension attribute is a descriptive characteristic of a dimension member. It has a name and a data type. A dimension attribute is applicable to one or more levels in the dimension. The list of dimension attributes must include all the attributes that are needed for any of the levels in the dimension. For example, a dimension called Product must have an attribute called *Description*. This attribute is applicable to all levels: Total, Categories, Subcategories, and Products and stores the description for each of the members within each level. An attribute can also be used to provide additional information about a dimension member. Some attributes are used for display purposes only. For example, a Product dimension could implement two description-based attributes: the first might contain Stock Keeping Units (SKUs) reference for use by brand managers familiar with the SKU codes, the other attribute could store a more descriptive *English-like* text for those unfamiliar with the SKU codes. These attributes, short and long descriptions are automatically defined when using the dimension wizard.

Other attributes, non-descriptive, can also be defined. Such as colors, flavors, or sizes. These types of attribute can be used for data selection and answering business questions such as: Which colors were the most popular in women's dresses in the summer of 2002? How does this compare with the previous summer?

The attribute definition stage has been enhanced to reflect these additional metadata requirements when creating multi-dimensional models. There are now six categories of attributes:

- Surrogate
- Business
- Descriptions (Short and Long)
- Parent
- Time based (End date and Time span)
- User-defined

The first four of these attributes are required metadata for each dimension. The two descriptions are used within the Oracle Business Intelligence 10g products to manage display labels within reports. Traditionally, OLAP products do not expose the keys to business users, these are hidden and only the description attributes (short and long descriptions) are made visible.

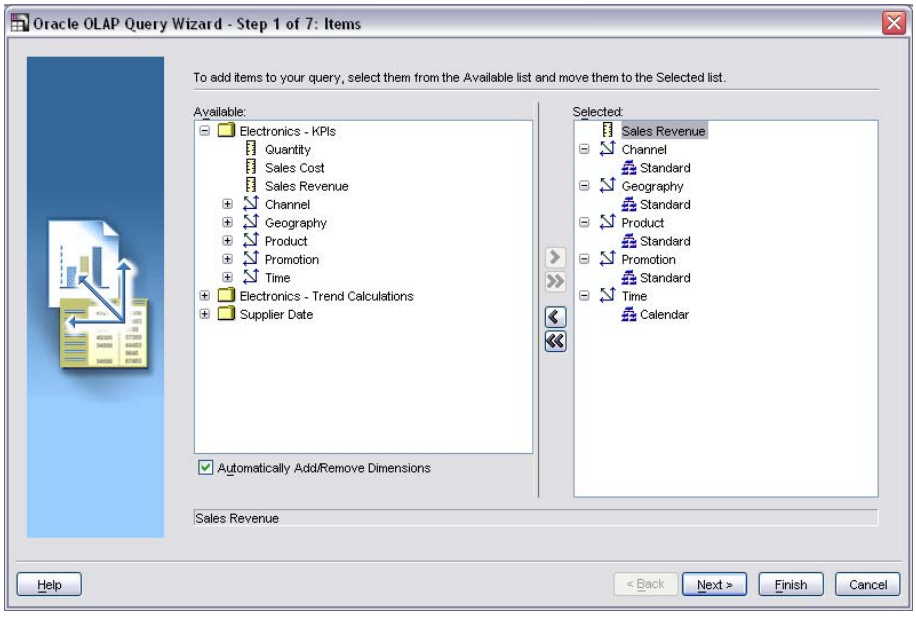

Figure 2 Oracle OLAP Query Builder Showing Object Business Display Labels

The "*parent*" attribute is used when defining a value-based hierarchy to identify the parent key for each member - subsequent sections will explain the various types of hierarchies supported by Warehouse Builder 10gR2.

In a multi-dimensional model time is an important dimension. Many calculations are based on time, such as % growth based on prior year or % growth based on prior period. To help make these types of calculations simple to define and implement, additional metadata is required to help explain the relationship between different time periods. Two attributes are required when defining a dimension of type Time: End date and time span. These attributes are used to identifying the last day or the number of days in each time period.

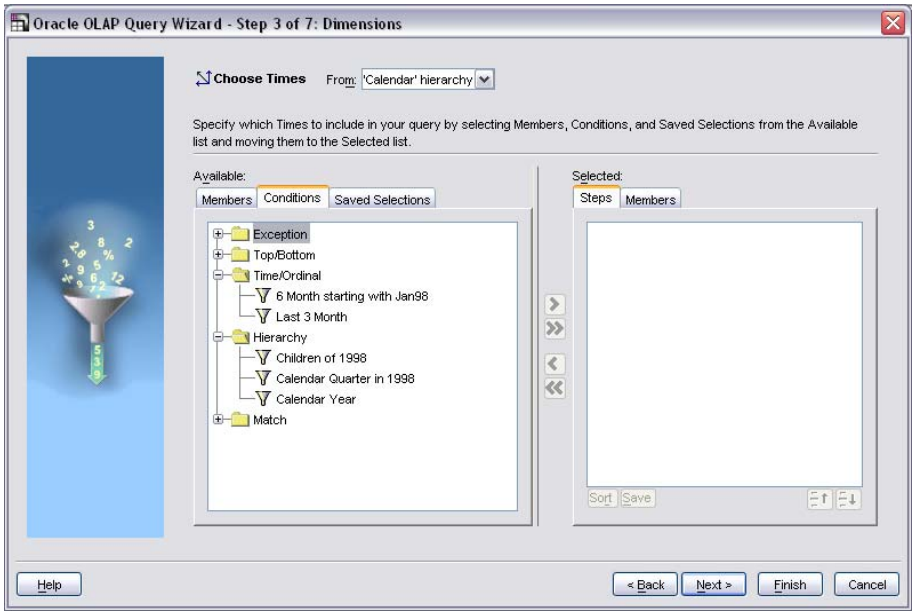

Figure 3 Oracle OLAP Query Wizard for Time Dimension Member Selection

Creating a Cube via the Wizard

The process of creating a cube is a simple four-step wizard. The wizard manages all the implementation options using intelligent defaults.

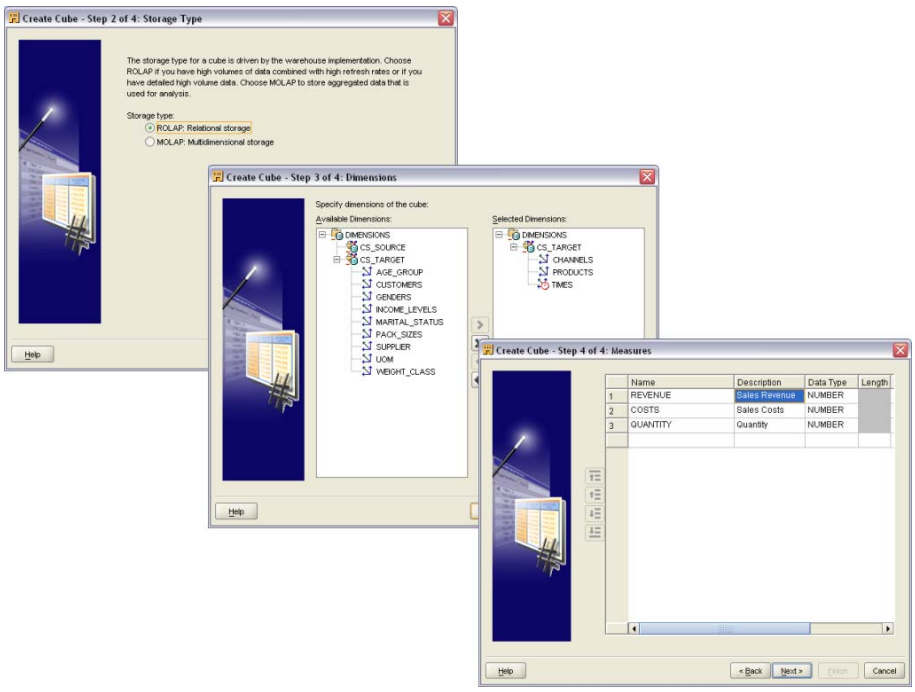

Figure 4 'Create New Cube' Wizard

Deploying Dimensions and Cubes

A multi-dimensional model can be mapped and deployed in exactly the same way as a relational dimension. All the usual transformations are available from the palette for use within the mapping. However, some additional features have been added to help manage the multi-dimensional loading process.

A '*Truncate before Load*' property is provided to control the deletion of all dimension members before loading the new members. Unlike relational implementations, if dimension members are deleted, all the data in related cubes is also automatically removed as well.

When loading data into a cube it is not necessary to completely recalculate all measures, since many values will not have been updated. The '*Incremental Aggregate*' property allows fine-grained control over the aggregation process. Setting this property to 'yes' will cause aggregation to occur on just the values that have been loaded. This dramatically improves load times and is similar to the process of a fast refresh on a materialized view.

The process of aggregating, or solving, data can be split into different stages and separated out from the loading of data. The '*Solve the Cube'* property allows a map to just load data. A separate process flow can then be designed to manage the aggregation process. It can be extremely useful to separate the load and solve processes especially when the availability of source data is constrained by other operational activities and data needs to be loaded as quickly as possible. The solve process can then be deferred to a later point in time.

The code generated for a multi-dimensional mapping is a mixture of PL/SQL and AW XML. All of this is totally transparent to the ETL designer and so there is no real need to understand the following section: it is included merely for completeness. Oracle OLAP supports the design of a multi-dimensional model as an XML document. The XML document is passed to the database where it is converted into multi-dimensional schema objects. The example below shows the XML used to load the dimension values for a dimension called *Product*.

```
<?xml version = ''1.0'' encoding = ''UTF-8'' ?>
<!-- <!DOCTYPE XMI SYSTEM ''Model.dtd'' > -->
<AWXML version = ''1.0'' timestamp = ''Mon Feb 11 13:29:11 2002'' >
<AWXML.content>
<Attach  Id="Action2" AWName="CS_AW">
<ActiveObject >
<AW  Name="CS_AW" LongName="CS_AW" ShortName="CS_AW" PluralName="CS_AW"
Id="CS_AW.AW">
</AW>
</ActiveObject>
</Attach>'
<BuildDatabase  Id="Action3" AWName="CS_AW" BuildType="EXECUTE"
RunSolve="false" CleanMeasures="false" CleanAttrs="' ||
TRIM(CLEAN_OBJ_LOCAL) || '" CleanDim="' || TRIM(CLEAN_OBJ_LOCAL) || '"
TrackStatus="false" MaxJobQueues="0">
<BuildList XMLIDref="PODUCTS.DIMENSION" />
</BuildDatabase>
<Detach  Id="Action4" AWName="CS_AW"/>
</AWXML.content>
</AWXML>
```

Note the XML does not contain the information relating to the mapping. When the map to load the dimension is deployed the mapping information is not stored in a separate package or program it is merged into the definition of the dimension or cube that has been deployed to the target analytic workspace. Hence, the XML just needs to reference the object that needs to be built; information relating to the mapping is extracted from the deployed definition of the dimension. Again, this is transparent to the designer and is mentioned only for completeness. The important point is the process for mapping a dimension and executing that mapping is the same for both relational and multi-dimensional objects.

What happens during deployment?

Analytic Workspace

An analytic workspace is a container within the Oracle database that stores data in a multidimensional format. An analytic workspace can contain a variety of objects such as dimensions, variables and formulas. It is stored in a relational database table, which can be partitioned across multiple disk drives like any other table and is owned by a particular user. Other users can be granted access to it. The analytic workspace acts as the container for all deployed multi-dimensional objects. For more information about analytic workspaces, refer to Chapter 6, Understanding Data Storage, of the Oracle OLAP Application Developer's Guide 10g Release 2(10.2). The deployment process will automatically, and transparently, creates the analytic workspace when the first multi-dimensional object is deployed.

OLAP Catalog

The OLAP catalog is the metadata repository for the OLAP option in the Oracle database. This metadata describes the data stored in both relational tables and in analytic workspaces. OLAP metadata is dynamically projected through a series of views called the active catalog views (views whose names begin with ALL_OLAP2_AW). All OLAP products use the OLAP catalog metadata to access multi-dimensional objects. For example, applications such as BI Beans and Discoverer use the OLAP catalog to query multidimensional metadata when accessing their query and calculation wizards. As these catalog views are active views, as soon as a dimension or cube has been deployed it is immediately visible and can be queried via the OLAP catalog metadata views.

Refining the Multi-Dimensional Model

Object Naming Conventions

In most cases object naming is largely unimportant, however, as Warehouse Builder is increasingly able to create complete ready-to-go business intelligence environments, the creation and management display labels for objects becomes more important. For example when a schema is made available to business users the expectation will be that names displayed within their BI product of choice will be correctly formatted, in the proper case and with spaces rather than '_' used to separate words. Business users will expect to see "Customer Income Level" and not "CUSTOMER_INCOME_LEVEL".

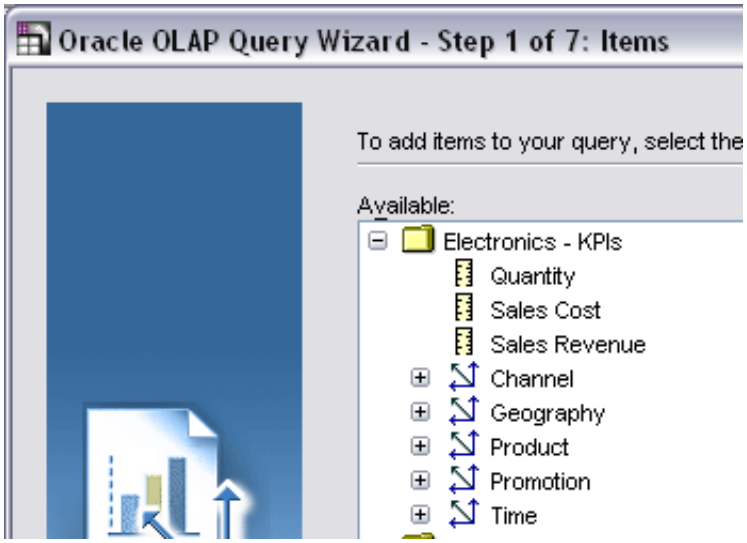

Figure 5 Object Display Labels in Query Builder

The format and case of names within Warehouse Builder is determined by the mode of operation. By default, Warehouse Builder operates in "Physical Names" mode. This means that all object names entered will be converted into upper case and cannot contain blank spaces. This is fine from an modeling and object deployment perspective. The end-user friendly mode is referred to as 'Business Names". Using the Preferences dialog the naming convention can be swapped to the desired mode. When the GUI is operating in physical names mode, the properties panel for each dimension can be used to amend the registered business label for each object. The same dialog can be used to provide the multi-language translations of object names as well.

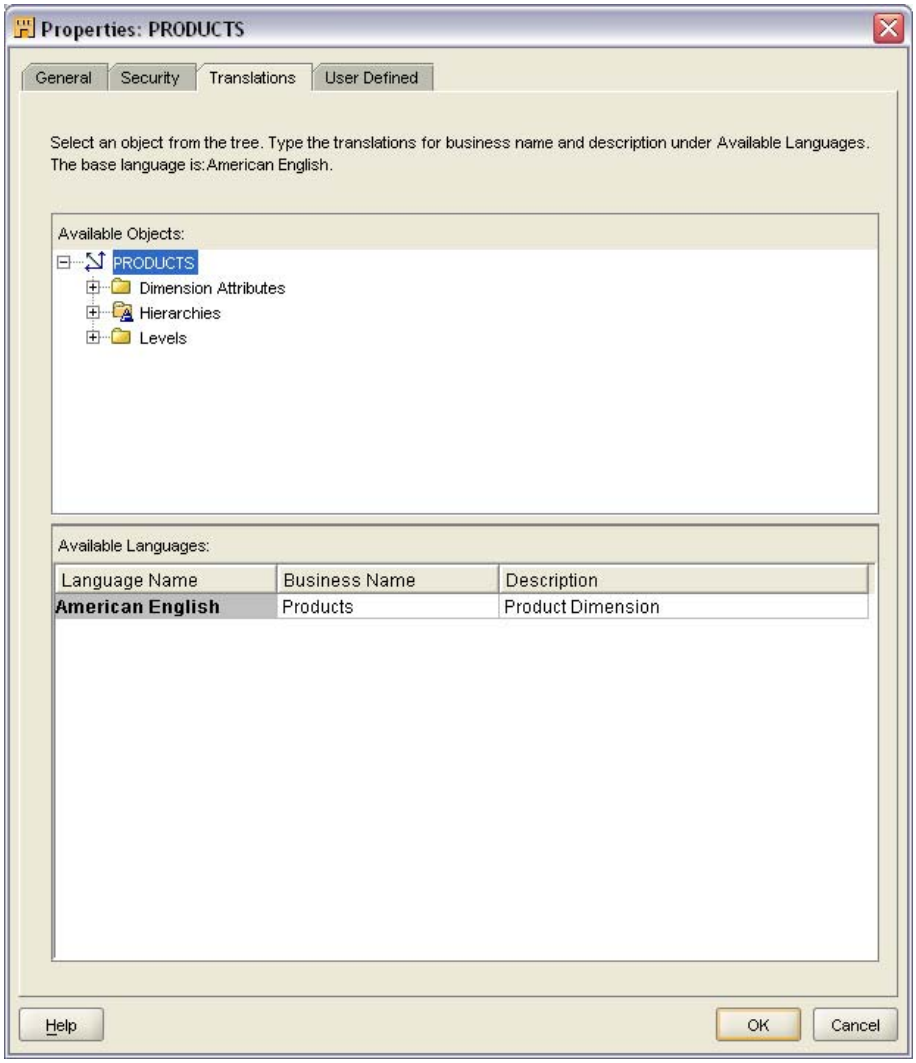

Figure 6 Setting Display Labels

Refining Dimensions

Managing Storage Options

The wizard defaults the analytic workspace name to the same name as the target module. However, this can be changed using the Data Object Editor. It is also possible to override the default the tablespace for the AW.

Surrogate vs. Natural keys

The use of surrogate keys is defaulted by the wizard but can be overridden in the Data Object Editor. Most source relational systems do enforce unique keys across all levels but simply rely on unique values existing within each column. If all these values were transposed into a single column of values the uniqueness would be lost and many values would simply disappear. A multi-dimensional object is

implemented as a single column of keys. Therefore, to ensure uniqueness it is advisable to enable the surrogate key generation.

Creating Multiple Hierarchies

The wizard creates a dimension with a single hierarchy. Using the Data Object Editor additional hierarchies can be added. There is no limit the number of hierarchies that can be implemented. Four basic types of hierarchies are supported that extend the simple relationships that are possible with relational queries:

- Level Based
  - Normal
  - Ragged
  - Skip
- Value Based

Level based hierarchies consist of one or more levels of aggregation. Members roll up into the next higher level in a many-to-one relationship, and these members roll up into the next higher level, and so forth to the top level. Ragged hierarchies are an extension of the basic level-based hierarchy and contain leaf nodes at more than one level. This type of hierarchy is typical in retail product dimensions.

Skip-level hierarchies contain at least one member whose parents are more than one level above in the hierarchical structure, creating a hole in the hierarchy. An example of a skip-level hierarchy is City-State-Country, where at least one city has a country as its parent (for example, Washington D.C. in the United States). In relational source tables, a skip-level hierarchy may contain nulls in the level columns and can only skip from one node to one of two other nodes (the immediate parent level or another pre-determined level). Within a multi-dimensional hierarchy there are no constraints on the number of levels that can be defined as skip-to targets.

A dimension may also contain a value-based or simple parent-child relation that does not support levels. For example, an employee dimension might have a parent-child relation that identifies each employee's supervisor. However, levels that group together first-, second-, and third-level supervisors and so forth may not be meaningful for analysis. In this situation, the parent-child relationship is based on dimension member values instead of levels. Only natural keys can be used to create a value-based hierarchy, because surrogate keys are formed with the names of the levels.

Data Viewer

The Data Object Editor provides a live data viewer that can be used to preview the members of a dimension. This assumes the dimension has been deployed and a mapping executed to load the dimension members.

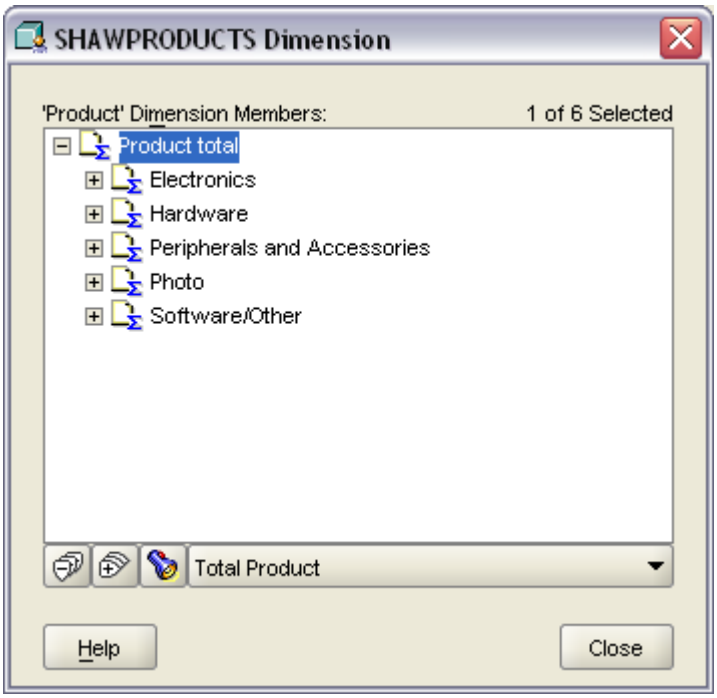

Figure 7 Dimension Data Viewer

Refining Cubes

Managing Sparsity

Sparsity refers to the extent to which cells with a cube contain null (NA) values instead of data. For example, if a cube is 25 percent sparse, then 25 percent of that cube's cells contain NA values and 75 percent contain data. If a cube's detail-level data is more than 80 percent sparse, then sparsity must be managed by identifying the sparse dimensions in order to promote efficient storage, good aggregation and query performance.

When trying to evaluate and manage data sparsity there are two types of sparsity that need to be considered:

- Controlled sparsity means that a range of values of one or more dimensions has no data. This is often a result of the way you design your analytic workspace. For example, a Time dimension might contain future time periods that will be populated by a forecast after the data is loaded into the analytic workspace. This type of sparsity is temporary and should be disregarded when evaluating the sparsity of the cube. Other types of controlled sparsity may remain sparse and should be considered as a factor in evaluating sparsity.
- Random sparsity means that NA values are scattered throughout a measure, usually because some combinations of dimension values never have any data. This is often a result of the nature of business transactions. Random sparsity tends to be more common than controlled sparsity.

Schemas can exhibit one or both types of sparsity. When designing cubes sparsity becomes important because all measures within the cube need to share the same profile:

- They have exactly the same dimensions
- They have many of the same empty cells
- They have roughly the same number of empty cells

Measures that share these characteristics should be created in the same cube. However, if the sparsity patterns are very different, then different cubes should be defined, even if the measures share the same dimensionality. The Oracle OLAP Application Developer's Guide 10g Release 2(10.2) has more information on this subject and provides examples of SQL queries that can be used to help determine sparsity.

In general most cubes tend to be sparse and measures that are mapped to the same source tables tend to have the same sparsity pattern. By default Warehouse Builder defines all dimensions, except Time, as sparse. Experienced data modelers can experiment with these settings to further refine the sparsity management.

When sparsity management is used additional structures are created at the physical level, these are known as *composite dimensions*. Composite dimensions are managed automatically by the OLAP engine and are not exposed directly within Warehouse Builder. Using the sparsity management options greatly reduces the amount of required storage space and as a result improves query performance for sparse data sets.

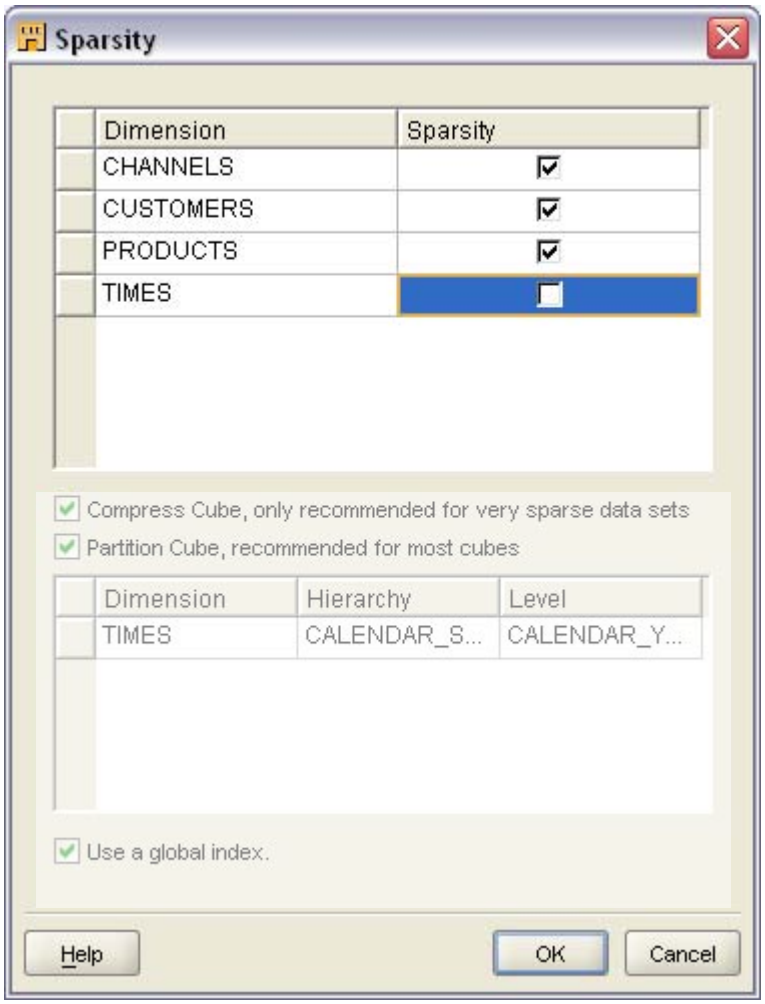

Figure 8 Setting Dimension Sparsity

Compressed storage is for measures that are extremely sparse. Extreme sparsity often results from these factors:

- A cube has a large number of dimensions (seven or more).
- One dimension has more than 300,000 members.

- Two dimensions have more than 100,000 members each.
- Dimension hierarchies have numerous levels, with little change to the number of dimension members from one level to the next, so that many parents have only one descendant for several contiguous levels.

Compressed storage for this type of sparsity uses less space and results in faster aggregation than normal sparse storage.

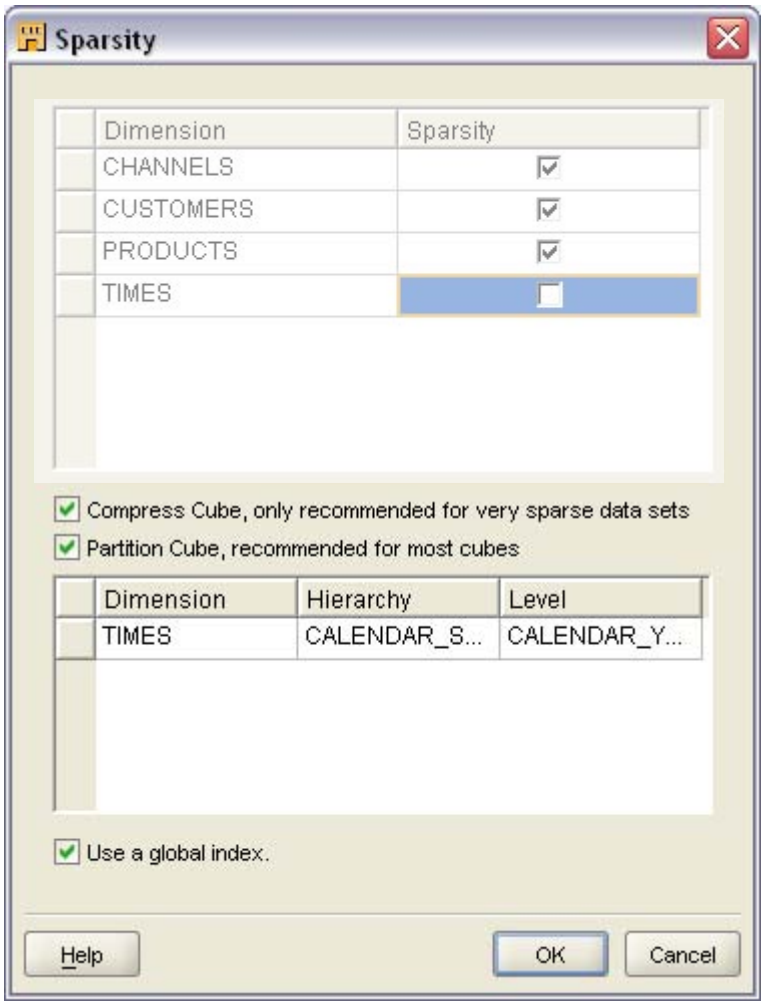

Figure 9 Enabling Cube Compression

Partitioning Cubes

As with relational schemas, multi-dimensional schemas support partitioning of cubes. For multi-dimensional schemas, partitioning improves the performance of large measures in the following ways:

- Improves scalability by keeping data structures small. Each partition functions like a smaller measure.
- Keeps the working set of data smaller both for queries and maintenance, since the relevant data is stored together.
- Enables parallel aggregation during data maintenance. A separate process can aggregate each partition.
- Allows different client sessions to have write access to different partitions of the same object at the same time.
- Simplifies removal of old data from storage. Old partitions can be dropped as a unit, and new partitions can be added.
- Stores each partition of a compressed cube in a separate analytic workspace object. If a compressed cube is not partitioned, then all measures of the cube are stored in one object.

Time typically tends to be a good candidate for use as a partitioning dimension. This is because it provides support for life-cycle maintenance. Old time periods can be dropped as a unit in a partition, and new time periods can be added in a new partition. Most importantly, the partitions will be approximately the same size. However, if life-cycle maintenance is not a major consideration, then the most-dense dimension should be used for partitioning. The *most-dense* dimension is frequently the one with the fewest members.

Controlling Aggregation

The multi-dimensional model provides an extremely rich aggregation environment. Each dimension can implement a different aggregation method. This provides support for measures that cannot be summarized using a simple sum operation. For example, stock measures cannot not be summed over time. Each quarter is the result from the last month in that quarter and each year is the result from the last quarter. This implies using an aggregation method of '*Last'* for Time and 'Sum' for all other dimensions

By default Warehouse Builder assumes each dimension requires a sum aggregation, however, this can easily be controlled via the Aggregation tab that is part of the Data Object Editor.

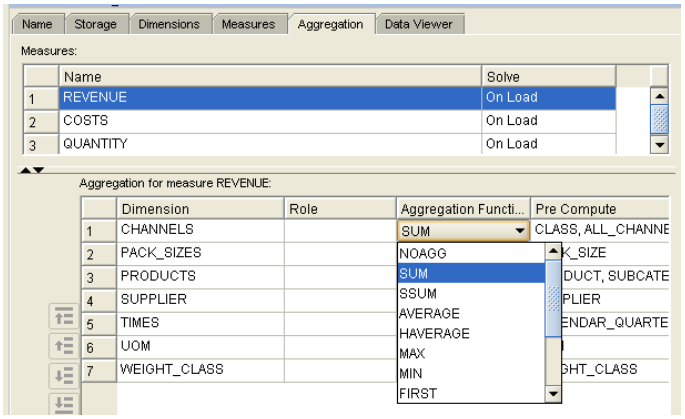

Figure 10 Managing Aggregation Options

To view summary data, the pre-defined aggregation methods have to be executed. This can affect query performance as the aggregation step is executed when the user creates a new report or opens an existing report. Warehouse Builder allows the ETL designer to pre-compute aggregations across each dimension. The Data Object Editor provides a dialog to select the levels in the dimension along which the measure data is pre-computed. It is possible to pre-compute all levels for each dimension. By default, Warehouse Builder starting with leaf-nodes flags alternate levels for each dimension as pre-computed. This is an intelligent compromise based the need for query performance versus the amount of disk space consumed.

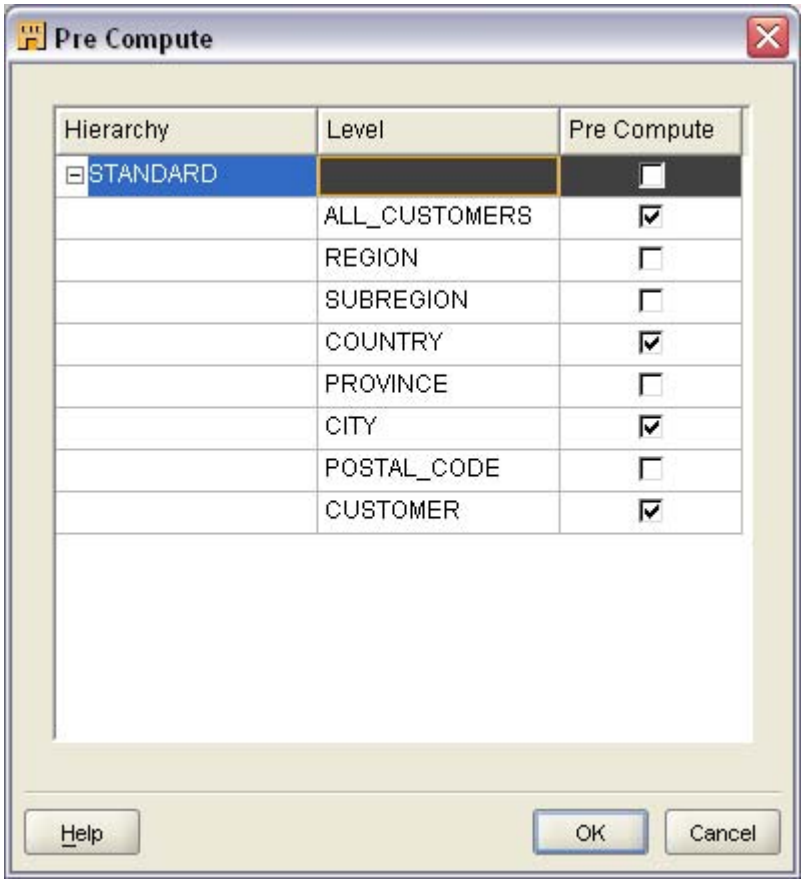

Figure 11 Default Aggregation Settings

Extending the Multi-Dimensional Model

Adding Value With Calculated Measures

Creating measures within a multi-dimensional model is a very simple process. However, it is these calculations that provide the real power and business benefit, since they allow business users to quickly and easily create extremely complex queries quickly and easily. Business questions such as

- How do sales for our five most profitable products across the US., for this quarter compare with sales a year ago?
- What are the differences in the product-sales mix between the regions relative to the global sales mix?
- In what ways does the mix vary by salesperson, and what is the relative performance of our salespeople?
- What are our forecast units, unit price per service, unit costs per product, sales, costs and profits for the next 12 months?

require more than just simple measures such as revenue and costs that are normally part of every model.

From a design perspective there are other advantages of creating calculations directly within the model. Business rules for a calculation can be defined and maintained once and then re-used by all business users, rather than each user creating their own version of the same calculation. By incorporating these business calculations into the basic model, the ETL designer can leverage the lineage and impact analysis features of Warehouse Builder when making changes to the model.

New measures can be added either using a template driven wizard or alternatively can be defined manually using the custom express editor.

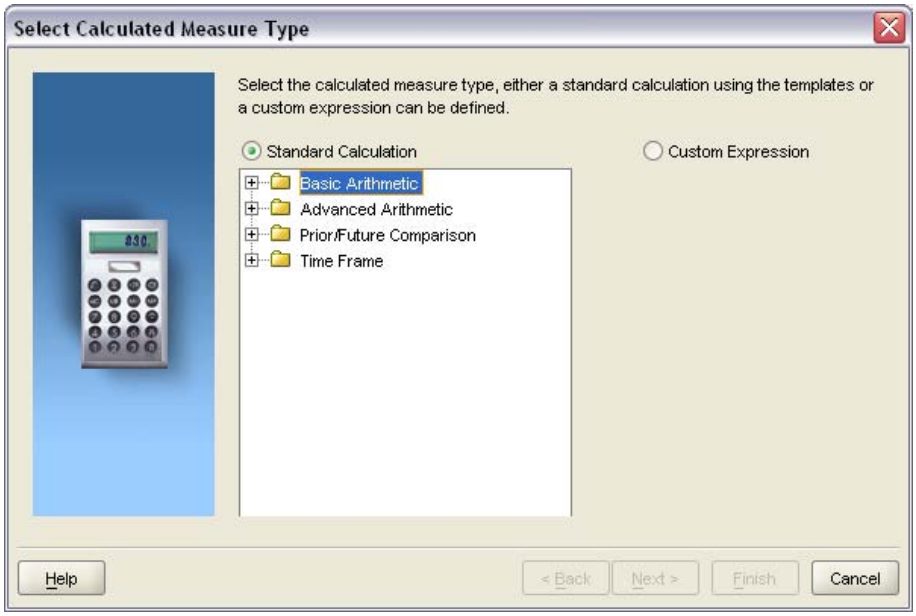

Figure 12 Measure Calculation Builder

The template wizard has a number of pre-built templates:

- Basic Arithmetic
  - Addition
  - Subtraction
  - Multiplication
  - Division
- Advanced Arithmetic
  - Cumulative Total
  - Index
  - Percent Markup
  - Percent Variance

    - Rank
    - Share
    - Variance
- Prior/Future Comparison
    - Prior Value
    - Difference from Prior Period
    - % Difference from Prior Period
    - Future Value
- Time Frame
    - Moving Average
    - Moving Maximum
    - Moving Minimum
    - Moving Total
    - Year to Date

These calculations are key to adding value to the multi-dimensional model and help simplify the process of creating complex business focused queries.

As each cube will typically require many of the above measures, Warehouse Builder provides a quick and easy way to add many of these measures via an alternative checkbox based interface.

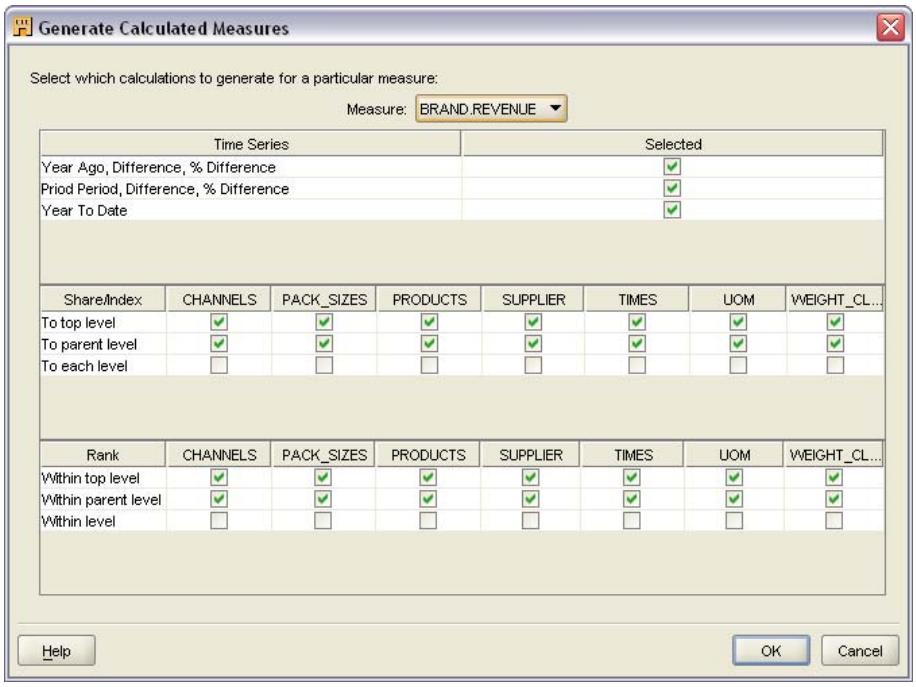

Figure 13 UI for Automatically Generating Calculated Measures for a Cube

Adding Value with Predictive Analytics

Many business-reporting dashboards and applications seek to compare actual against predicted performance. This usually requires loading data into another database engine to perform forecast or predict future values for sales and costs and then loading that data back into the reporting server.

In addition to supporting many different types of calculations, Oracle OLAP 10g also provides support for many predictive analytic functions, such as forecasting. Oracle OLAP provides a number of forecasting options. However, many companies have their own tried and tested data models for forecasting. So can Warehouse Builder be extended to provide access to these additional data models?

Warehouse Builder provides an extensible environment that can be customized and enhanced using *Experts*. As part of the release of Warehouse Builder 10gR2 an OLAP Forecast Wizard has been designed to showcase the concept of wizards and demonstrate how companies could incorporate their own forecast models. However, the OLAP forecast model is extremely powerful and should be investigated if general predictions are required.

The *OLAP Forecast* Expert is currently based on four simple screens that determine the following information:

- Name of AW
- Name of Cube
- Time Span information
    - Last time period for which there is data
    - Number of historical periods to analyze
    - Periodicity of the data
    - How many periods to forecast forwards
- Forecast method
    - Best fit
    - Exponential
    - Liner
    - Non-Linear
    - Seasonal
    - Trend

- Dimensionality of the forecast

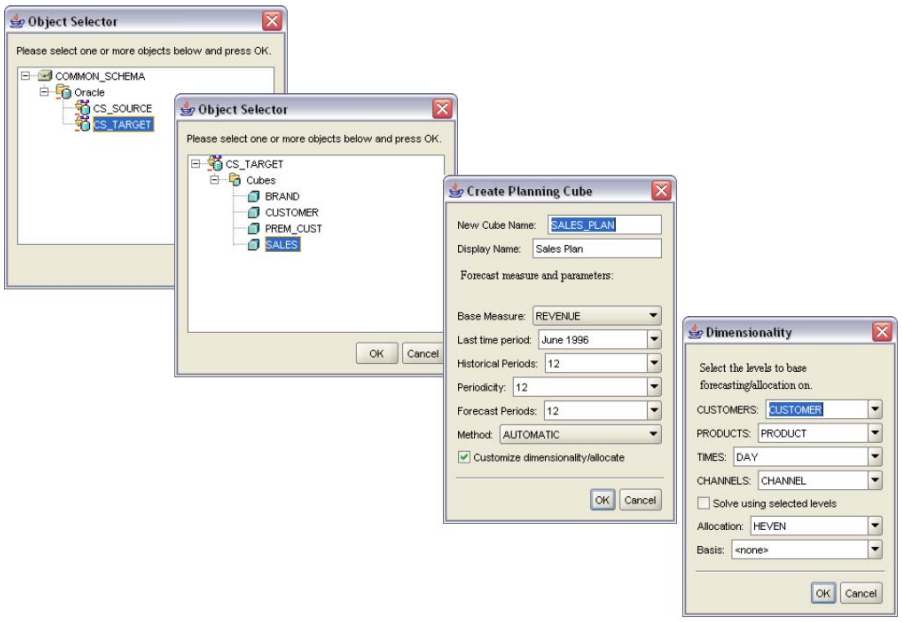

Figure 14 – Creating a Forecast

The OLAP forecast model has many other properties that can be used to fine tune the model, however, for reasons of simplicity these additional properties have not been exposed in this particular expert.

The expert automatically creates a number of objects required to manage a forecast process:

- Creating a Planning Cube – a new cube is created to manage the forecast measures. As forecasting is typically computed and stored for data points above the base level different aggregation plans need to be defined. This is controlled by the last dialog in the wizard
- Solve Procedure – this is the controlling procedure that can be used to execute the forecast, allocate the forecast values across the lower levels of the hierarchy and then finally solve the cube and rollup the data. The allocation step is optional.
    - Forecast Procedure – this implements the forecast model based on the options derived from the wizard
    - Aggregation Procedure – Computes the higher level aggregates, i.e. rolls-up the data points across the various hierarchies.

- Allocation Procedure – Allocation rules can be used, if required, to allocated aggregate forecast data points across lower levels in he hierarchy.

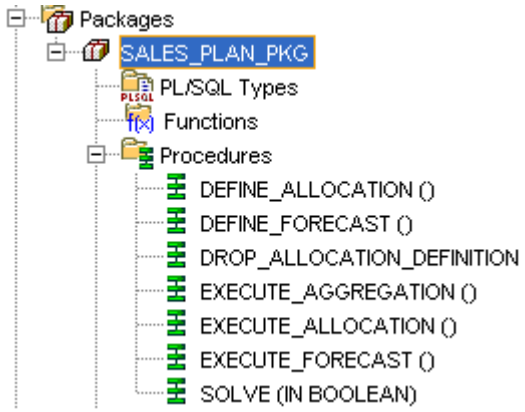

Figure 15 Forecast Procedures

All of the above steps are typical of any forecast process and Warehouse Builder 10gR2 organizations can either integrate their own data models or leverage the power of Oracle's own predictive analytic functions that are part of the OLAP option.

The procedures are all defined as PL/SQL and can be managed and executed like any other PL/SQL procedure. Again the implementation method is transparent to the designer. However, as before, because these new processes are defined within Warehouse Builder they are directly integrated into all the normal features such as lineage and impact analysis.

Once deployed these forecast, or planning, measures appear as normal measures to business users. However, these types of cubes provide tremendous opportunities to both measure current business performance and also predict with confidence likely future performance.

GETTING MORE INFORMATION

Oracle Business Intelligence OTN Home Page

This provides an overview of the complete Oracle BI solution with links to the various products that comprise the Oracle BI10g suite

http://www.oracle.com/technology/products/bi/index.htm

Oracle Warehouse Builder

This is the OTN home page for Warehouse Builder. It provides links to documentation, whitepapers, viewlets and customer success stories

http://www.oracle.com/technology/products/warehouse/index.html

As well as the OTN Forum, where you can post questions and also share your expertise with other forum users by answering other proples questions as well.

http://www.oracle.com/forums/forum.jsp?forum=57

Utilities, Tips and Tricks Exchange

The purpose of the Oracle Warehouse Builder Utility Exchange is to provide the user community with a place where Warehouse Builder utilities, code samples, tips, tricks etc. can be exchanged on a non-profit, non-support basis

http://otn.oracle.com/products/warehouse/htdocs/OWBexchange.html

Software Development Kit

he Oracle Warehouse Builder Software Development Kit (SDK) is a robust framework to extend the capabilities of Warehouse Builder. On this page, you will find technical information and sample code to help add functionality to the already extensive design, ETL and runtime capabilities of Warehouse Builder. With more than twenty different APIs covering the entire product you will be able to extent the product allowing you to solve these specific requirements for your project in a timely and scalable manner

http://www.oracle.com/technology/products/warehouse/SDK/SDK_Home.htm

Oracle OLAP

This is the OTN home page for the Oracle OLAP option. It provides links to documentation, whitepapers, and viewlets.

http://www.oracle.com/technology/products/bi/olap/index.html

CONCLUSION

Who needs OLAP? Multi-dimensional, data that can provide answers to an organization's specific business questions is valuable to both IT and business users. These multi-dimensional schemas are typically user to extend and enhance an organization's ability to answer a broad range of business related questions:

From an IT perspective, OLAP implementations are problematic because in the past they have required additional hardware, DBAs who are skilled at using specialized administrative tools, and ETL experts familiar with the bespoke multidimensional design languages. This has severely limited the number of organizations that can be run an efficient enterprise scalable data warehouse environments that also incorporates a multi-dimensional model.

However, multidimensional technology has been available within the Oracle database for a number of years. By integrating multidimensional objects directly into the relational database, Oracle provides the power of multidimensional analysis with the manageability, scalability, and reliability of the Oracle Database.

Now, for the first time ETL designers can also use one single tool for managing their data warehouse projects – Oracle Warehouse Builder 10gR2. This is the first ETL product to provide a single integrated and complete environment for managing enterprise data warehouse solutions that also incorporate multi-dimensional schemas. The extensions to the dimensional modeling capabilities have been built on established relational concepts. As a result data warehouse projects that need to provide a multi-dimensional model as part of the overall solution, can be designed and implemented faster and more efficiently.

Climbing to the OLAP Summit with Oracle Warehouse Builder 10gR2
January 2006
Keith Laker, Principal Product Manager Business Intelligence & Data Warehousing

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
www.oracle.com