Oracle® OLAP

Application Developer's Guide,

10*g* Release 2 (10.2)

B14349-05

February 2012

Oracle OLAP Application Developer's Guide, 10*g* Release 2 (10.2)

B14349-05

Contents

Preface

The *Oracle OLAP Application Developer's Guide* explains how SQL and Java applications can extend their analytic processing capabilities by using the OLAP option in the Enterprise edition of the Oracle Database. It also provides information for Oracle DBAs about managing resources for OLAP.

The preface contains these topics:

- Audience
- Documentation Accessibility
- Related Documents
- Conventions

Audience

This manual is intended for applications developers and DBAs who perform these tasks:

- Develop business intelligence applications
- Design and develop dimensional data stores (analytic workspaces)
- Administer Oracle Database with the OLAP option

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Related Documents

For more information, see the following manuals in the Oracle Database 10*g* documentation set:

- *Oracle OLAP Reference*

Explains the syntax of PL/SQL packages and types and the column structure of views related to Oracle OLAP.

- *Oracle OLAP DML Reference*

  Contains a complete description of the OLAP Data Manipulation Language (OLAP DML) used to define and manipulate analytic workspace objects.

- *Oracle OLAP Developer's Guide to the OLAP API*

  Introduces the Oracle OLAP API, a Java application programming interface for Oracle OLAP, which is used to perform online analytical processing of the data stored in an Oracle database. Describes the API and how to discover metadata, create queries, and retrieve data.

- *Oracle OLAP Java API Reference*

  Describes the classes and methods in the Oracle OLAP Java API for querying analytic workspaces and relational data warehouses.

- *Oracle OLAP Analytic Workspace Java API Reference*

  Describes the classes and methods in the Oracle OLAP Java API for building and maintaining analytic workspaces.

For more information about Oracle data warehouse and business intelligence technology, view the web page at `http://www.oracle.com/technetwork/database/focus-areas/bi-datawarehousing/index.html`.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

What's New in Oracle OLAP Applications Development?

The following identifies some of the major changes from prior releases.

Oracle Database 10g Release 10.2.0.3 Oracle OLAP

Analytic Workspace Manager in Release 10.2.0.3 provides a Sparsity Advisor, which examines the source data and makes recommendations for defining OLAP cubes that provide the best performance. The functionality of calculation plans has been enhanced to provide post-load forecasting, allocation, and aggregation. Analytic Workspace Manager also supports Java add-ins, so that any Java developer can extend and customize the tools in Analytic Workspace Manager.

See Also:

- Chapter 3 for information about the Sparsity Advisor and plug-ins
- Chapter 7 for information about forecasting
- Chapter 8 for information about aggregation
- Chapter 9 for information about allocation

Oracle Database 10*g* Release 10.2 Oracle OLAP

Oracle OLAP in Oracle Database 10*g* Release 2 (10.2) provides numerous performance enhancements and extensions to the dimensional data model.

Enhanced Data Model in Analytic Workspace Manager

Analytic Workspace Manager 10.2 supports Calculation Plans and multiple languages. Compressed composites provide support for partial computation and non-additive operators.

See Also:

- Chapter 1 for upgrade instructions
- Chapter 3 for new features in Analytic Workspace Manager

Support for Transportable Tablespaces

Analytic workspaces are included with other database objects in transportable tablespaces.

See Also: Chapter 12 for information about backing up analytic workspaces

Oracle Database 10*g* Release 10.1.0.4 Oracle OLAP

Oracle OLAP 10.1.0.4 provides a simpler approach to building and enabling analytic workspaces while introducing the more powerful analytic tools of the OLAP engine into the build process.

New Storage Format for Analytic Workspaces

Analytic workspaces are still stored in LOB tables in Oracle Database 10*g*, but in a different format that supports partitioning and multiple writers.

See Also:

- Chapter 1 for upgrade instructions
- Chapter 12 for a description of the storage format

New Model View in Analytic Workspace Manager

The Model View in Analytic Workspace Manager 10*g* enables you to define the logical model of your analytic workspace directly in database standard form. You no longer create logical models in the OLAP Catalog for building analytic workspaces. Analytic Workspace Manager supports a wider range of schema designs than the OLAP Catalog.

See Also: Chapter 3 for instructions on using the Model View

Database Standard Form 10*g*

Analytic Workspace Manager and the PL/SQL `DBMS_AWM` package generate a new version of standard form metadata that supports the new features of Oracle Database 10*g*.

Dynamic Enabling for the OLAP API and OracleBI Beans

The `SELECT` statements for the views of an analytic workspace are stored in the analytic workspace itself. Enablement no longer requires the creation of database objects.

Direct Metadata Access

The OLAP API and OracleBI Beans query the Active Catalog views, which display the database standard form metadata stored in analytic workspaces. Enablement no longer requires the creation of OLAP Catalog `CWM2` metadata.

1

Overview

This chapter introduces the powerful analytic resources available in Oracle Database 10*g* installed with the OLAP option. It consists of the following topics:

OLAP Technology in the Oracle Database

Oracle Database offers the industry's first and only embedded OLAP server. Oracle OLAP provides native multidimensional storage and speed-of-thought response times when analyzing data across multiple dimensions. The database provides rich support for analytics such as time series calculations, forecasting, advanced aggregation with additive and nonadditive operators, and allocation operators. These capabilities make the Oracle database a complete analytical platform, capable of supporting the entire spectrum of business intelligence and advanced analytical applications.

Full Integration of Multidimensional Technology

By integrating multidimensional objects and analytics into the database, Oracle provides the best of both worlds: the power of multidimensional analysis along with the reliability, availability, security, and scalability of the Oracle database.

Oracle OLAP is fully integrated into Oracle Database. At a technical level, this means:

- The OLAP engine runs within the kernel of Oracle Database.
- Dimensional objects are stored in Oracle Database in their native multidimensional format.
- Applications can query dimensional objects using either SQL or Java.

The benefits to your organization are significant. Oracle OLAP offers the power of simplicity: One database, standard administration, standard interfaces and development tools.

Ease of Administration

Because Oracle OLAP is completely embedded in the Oracle database, there is no administration learning curve as is typically associated with standalone OLAP servers. You can leverage your existing DBA staff, rather than invest in specialized administration skills.

Security

With Oracle OLAP, standard Oracle Database security features are used to secure your multidimensional data.

In contrast, with a standalone OLAP server, administrators must manage security twice: once on the relational source system and again on the OLAP server system. Additionally, they must manage the security of data in transit from the relational system to the standalone OLAP system.

Unmatched Performance and Scalability

Business intelligence and analytical applications are dominated by actions such as drilling up and down hierarchies and comparing aggregate values such as period-over-period, share of parent, projections onto future time periods, and a myriad of similar calculations. Often these actions are essentially random across the entire space of potential hierarchical aggregations. Because Oracle OLAP precomputes or efficiently computes on the fly all aggregates in the defined multidimensional space, it delivers unmatched performance for typical business intelligence applications.

When Oracle Database is installed with Oracle Real Application Clusters (Oracle RAC), OLAP applications receive the same benefits in performance, scalability, fail over, and load balancing as any other application.

Reduced Costs

These features add up to reduced costs. Administrative costs are reduced because existing personnel skills can be leveraged. Standard security reduces administration costs as well. Hardware costs are reduced by Oracle OLAP's efficient management of aggregations and Oracle RAC, which enables highly scalable systems to be built from low-cost commodity components.

About Multidimensional Data Stores

Multidimensional data is stored in **analytic workspaces**, where it can be manipulated by the OLAP engine in Oracle Database. Individual analytic workspaces are stored in tables in a relational schema, and they can be managed like other relational tables. An analytic workspace is owned by a particular user ID, and other users can be granted access to it. Within a single database, many analytic workspaces can be created and shared among users.

Analytic workspaces have been designed explicitly to handle multidimensionality in their physical data storage and manipulation of data. The multidimensional technology that underlies analytic workspaces is based on an indexed multidimensional array model, which provides direct cell access. This intrinsic multidimensionality affords analytic workspaces much of their speed and power in performing multidimensional analysis.

Creating Analytic Workspaces

Creating an analytic workspace involves a physical transformation of the data. The first step in that transformation is defining dimensional objects such as measures, dimensions, levels, hierarchies, and attributes. Afterward, you can map the dimensional objects to the data sources. The analytic workspace instantiates the logical objects as physical objects, and the data loading process transforms the data from a relational format into a dimensional format.

The analytic workspaces that are created by Oracle Warehouse Manager and Analytic Workspace Manager are in database standard form (typically called simply "standard form"). Standard form specifies the types of physical objects that are used to instantiate logical objects (such as dimensions and measures), and the type, form, and storage location of the metadata that describes these logical objects.

This metadata is exposed to SQL in the Active Catalog. The Active Catalog is composed of views of standard form metadata that is stored in analytic workspaces. These views are maintained automatically, so that a change to a standard form analytic workspace is reflected immediately by a change to the Active Catalog. Discoverer Plus OLAP and Spreadsheet Add-In use the Active Catalog to query data in analytic workspaces.

Structured Data Stores

The dimensional data model is highly structured. Structure implies rules that govern the relationships among the data and control how the data can be queried. Analytic workspaces are the physical implementation of the dimensional model, and thus are highly optimized for dimensional queries. The OLAP engine leverages the model in performing highly efficient cross-cube joins (for inter-row calculations), outer joins (for time series analysis), and indexing. Dimensions are pre-joined to the measures.

Processing Analytic Queries

For data stored in analytic workspaces, the OLAP calculation engine performs analytic operations and supports sophisticated analysis, such as modeling and what-if analysis. If you require these types of analysis, then you need analytic workspaces. The OLAP engine also provides the fastest run-time response to analytic queries, which is important if you anticipate user sessions that are heavily analytical.

Creating Summary Data

A basic characteristic of business analysis is hierarchically structured data; detail data is summarized at various levels, which allows trends and patterns to emerge. An analyst who has detected a pattern can drill down to lower levels to identify the factors that contributed to this pattern.

The creation and maintenance of summary data is a serious issue for DBAs. If no summary data is stored, then all summarizations must be performed in response to individual queries. This can easily result in unacceptably slow response time. At the other extreme, if all summary data is stored, then the database can quickly multiply in size.

Analytic workspaces store aggregate data in the same objects as the base level data. Aggregates can be stored permanently in the analytic workspace, or only for the duration of an individual session, or only for a single query. Aggregation rules identify which aggregates are stored for each measure. When an application queries the analytic workspace, either the aggregate values have already been calculated and can simply be retrieved, or they can be calculated on the fly from a small number of stored aggregates. The data is always presented to the application as fully solved; that is, both detail and summary values are provided, without requiring that calculations be specified in the query. Analytic workspaces are optimized for multidimensional calculations, making run-time summarizations extremely fast.

Analytic workspaces provide an extensive list of aggregation methods, including weighted, hierarchical, and weighted hierarchical methods.

Using SQL Tools to Query Dimensional Objects

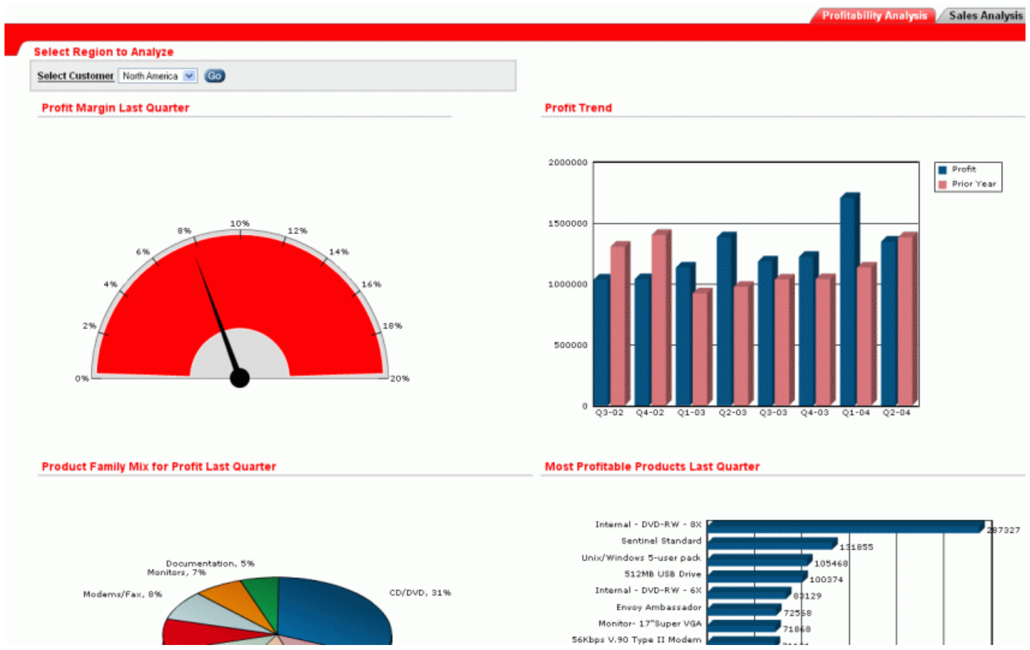
Analysts can choose any SQL query and analysis tool for selecting, viewing, and analyzing the data. You can use your favorite tool or application, or use one of the tools supplied with Oracle Database.

Figure 1–1 displays a portion of a dashboard created in Oracle Application Express, which is distributed with Oracle Database. Application Express generates HTML reports that display the results of SQL queries. It only understands SQL; it has no special knowledge of dimensional objects.

This dashboard demonstrates information-rich calculations such as ratio, share, prior period, and cumulative total. Separate tabs on the dashboard present Profitability Analysis and Sales Analysis. Each tab presents the data in dials, bar charts, horizontal bar charts, pie charts, and cross-tabular reports. A drop-down list in the upper left corner provides a choice of Customers.

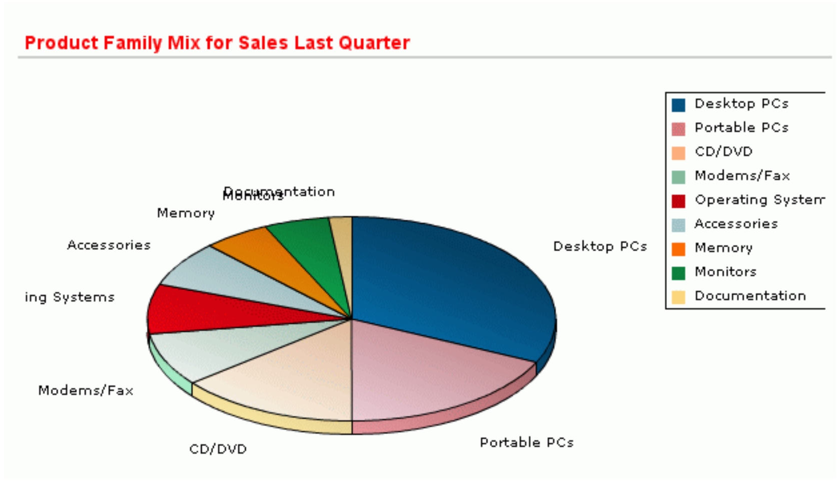
The dial displays the quarterly profit margin. To the right is a bar chart that compares current profits with year-ago profits.

Figure 1–1 Dashboard Created in Oracle Application Express



The pie chart in Figure 1–2 displays the percent share that each product family contributed to the total profits in the last quarter.

Figure 1–2 Contributions of Product Families to Total Profits



The horizontal bar chart in Figure 1–3 displays ranked results for locations with the largest gains in profitability from a year ago. Decision makers can see at a glance how each location improved by the last quarter.

Figure 1–3 Geographic Locations Ranked by Profit

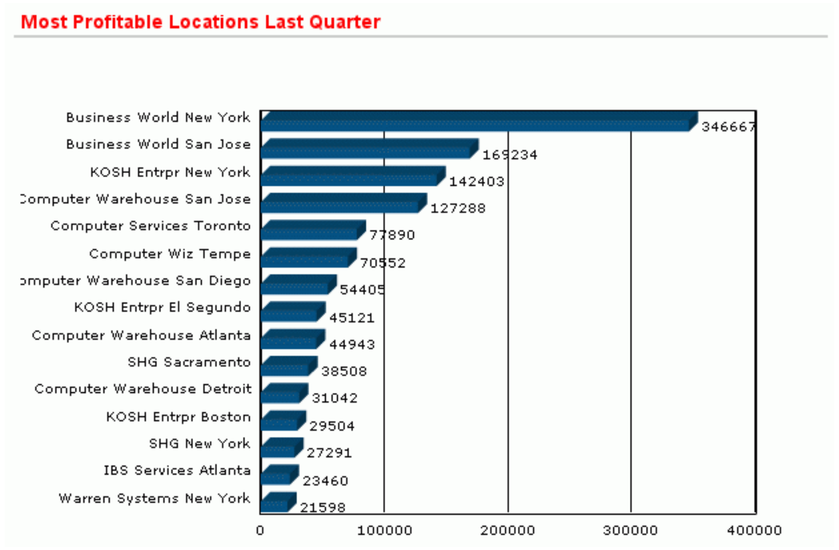
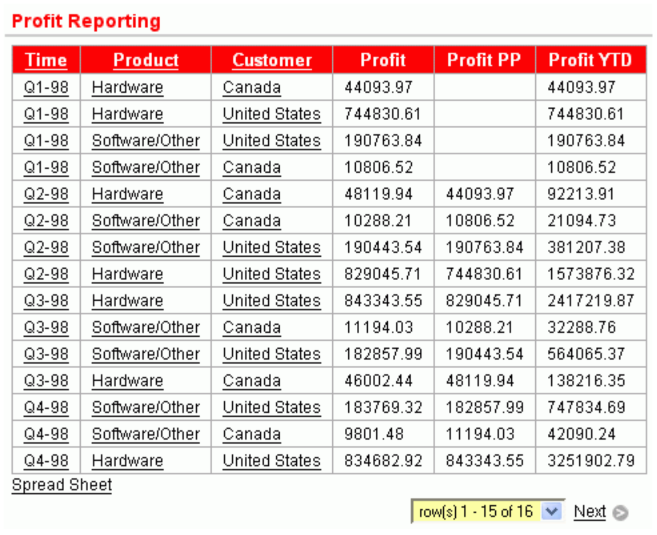


Figure 1–4 compares current profits with prior period and year-to-date profits. The cross-tabular report features interactive drilling, so that decision makers can easily see the detailed data that contributed to a parent value of interest.

Figure 1–4 Comparison of Current Profits With Other Time Periods


Profit Reporting

Time	Product	Customer	Profit	Profit PP	Profit YTD
Q1-98	Hardware	Canada	44093.97		44093.97
Q1-98	Hardware	United States	744830.61		744830.61
Q1-98	Software/Other	United States	190763.84		190763.84
Q1-98	Software/Other	Canada	10806.52		10806.52
Q2-98	Hardware	Canada	48119.94	44093.97	92213.91
Q2-98	Software/Other	Canada	10288.21	10806.52	21094.73
Q2-98	Software/Other	United States	190443.54	190763.84	381207.38
Q2-98	Hardware	United States	829045.71	744830.61	1573876.32
Q3-98	Hardware	United States	843343.55	829045.71	2417219.87
Q3-98	Software/Other	Canada	11194.03	10288.21	32288.76
Q3-98	Software/Other	United States	182857.99	190443.54	564065.37
Q3-98	Hardware	Canada	46002.44	48119.94	138216.35
Q4-98	Software/Other	United States	183769.32	182857.99	747834.69
Q4-98	Software/Other	Canada	9801.48	11194.03	42090.24
Q4-98	Hardware	United States	834682.92	843343.55	3251902.79

Using OLAP Tools to Query Dimensional Objects

Analysts can choose between two query and analysis tools developed specifically for selecting, viewing, and analyzing dimensional data:

- OracleBI Spreadsheet Add-In
- OracleBI Discoverer Plus OLAP

In addition, OracleBI Beans is available for developing custom applications, as described in Chapter 11.

Formulating Queries

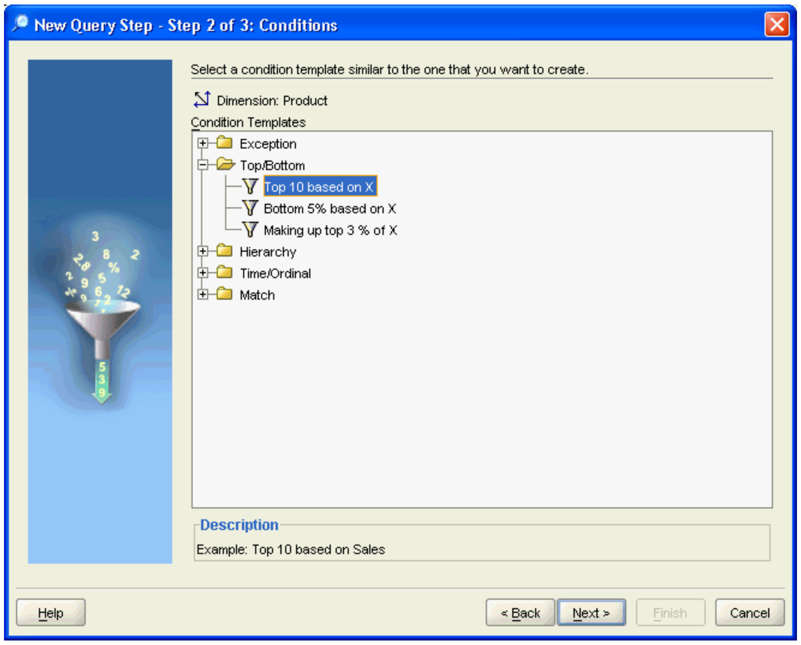
Both Discoverer Plus OLAP and Spreadsheet Add-In use a dimensional data model so that analysts can formulate their queries in the language of business. Dimensions provide the context for the data. Consider the following request for information:

For **fiscal years** 2003 and 2004, show the percent change in **sales** for the top 10 **products** for each of the top 10 **customers** based on sales.

The sales measure is dimensioned by time periods, products, and customers. This request is articulated in business terms, but easily translates into a query in the language of dimensional analysis: dimensions, levels, hierarchies, and attributes.

Figure 1–5 shows a step in the Query Wizard in Discoverer Plus OLAP for selecting the top 10 products. The Query Wizard assists users in selecting by criteria, by value, and by saved selections. All OLAP tools provide a Query Wizard to assist users in formulating these queries.

Figure 1–5 Selecting Dimension Values By Criteria



Select a condition template similar to the one that you want to create.

Dimension: Product

Condition Templates

- Exception
- Top/Bottom
  - Top 10 based on X
  - Bottom 5% based on X
  - Making up top 3 % of X
- Hierarchy
- Time/Ordinal
- Match

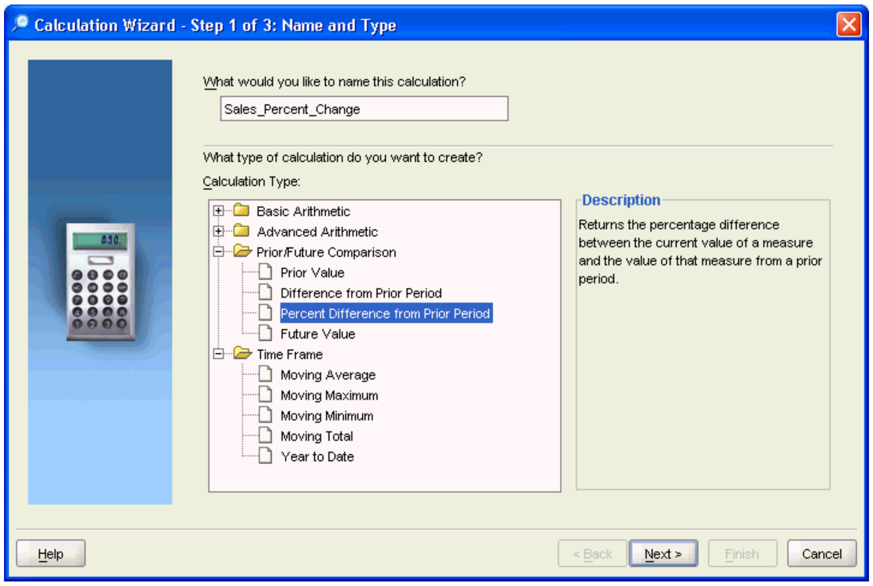
Description

Example: Top 10 based on Sales

Creating Calculations

Multidimensional data types facilitate the creation of calculations. From the measures stored in your data warehouse, you can use numerous operators and functions to generate a wealth of information. In addition to the calculated measures created by the DBA as part of an analytic workspace, users can create their own calculations. Figure 1–6 shows a step in the Calculation Wizard of Discoverer Plus OLAP for calculating percent change in sales. Spreadsheet Add-In has the same Calculation Wizard.

Figure 1–6 Choosing a Calculation Method for a Custom Measure



What would you like to name this calculation?

Sales_Percent_Change

What type of calculation do you want to create?

Calculation Type:

- Basic Arithmetic
- Advanced Arithmetic
- Prior/Future Comparison
  - Prior Value
  - Difference from Prior Period
  - Percent Difference from Prior Period
  - Future Value
- Time Frame
  - Moving Average
  - Moving Maximum
  - Moving Minimum
  - Moving Total
  - Year to Date

Description

Returns the percentage difference between the current value of a measure and the value of that measure from a prior period.

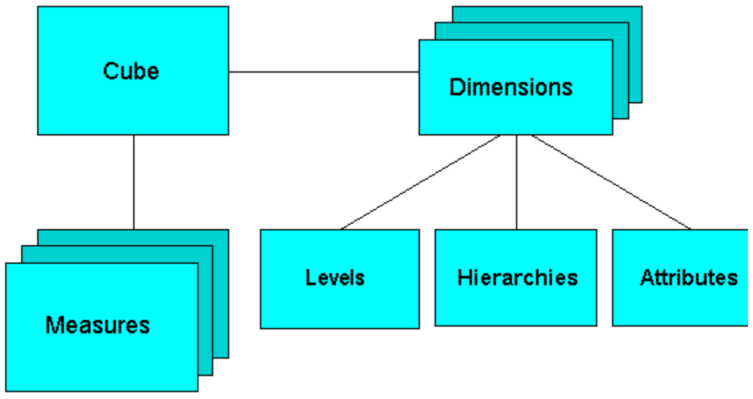
Overview of the Dimensional Data Model

The dimensional data model is an integral part of On-Line Analytical Processing, or OLAP. Because OLAP is on-line, it must provide answers quickly; analysts pose iterative queries during interactive sessions, not in batch jobs that run overnight. And because OLAP is also analytic, the queries are complex.

The dimensional data model is composed of cubes, measures, dimensions, hierarchies, levels, and attributes. The simplicity of the model is inherent because it defines objects that represent real-world business entities. Analysts know which business measures they are interested in examining, which dimensions and attributes make the data meaningful, and how the dimensions of their business are organized into levels and hierarchies.

Figure 1–7 shows the general relationships among objects.

Figure 1–7 Diagram of the OLAP Dimensional Model



Cubes

Cubes provide a means of organizing measures that have the same shape, that is, they have the exact same dimensions. Measures in the same cube have the same relationships to other objects and can easily be analyzed and displayed together.

Measures

Measures populate the cells of a cube with the facts collected about business operations. Measures are organized by dimensions, which typically include a Time dimension.

An analytic database contains snapshots of historical data, derived from data in a transactional database, legacy system, syndicated sources, or other data sources. Three years of historical data is generally considered to be appropriate for analytic applications.

Measures are static and consistent while analysts are using them to inform their decisions. They are updated in a batch window at regular intervals: weekly, daily, or periodically throughout the day. Some administrators refresh their data by adding periods to the time dimension of a measure, and may also roll off an equal number of the oldest time periods. Each update provides a fixed historical record of a particular business activity for that interval. Other administrators do a full rebuild of their data rather than performing incremental updates.

A critical decision in defining a measure is the lowest level of detail. Users may never view this **base level data**, but it determines the types of analysis that can be performed. For example, market analysts (unlike order entry personnel) do not need to know that Beth Miller in Ann Arbor, Michigan, placed an order for a size 10 blue polka-dot dress on July 6, 2005, at 2:34 p.m. But they might want to find out which color of dress was most popular in the summer of 2005 in the Midwestern United States.

The base level determines whether analysts can get an answer to this question. For this particular question, Time could be rolled up into months, Customer could be rolled up into regions, and Product could be rolled up into items (such as dresses) with an attribute of color. However, this level of aggregate data could not answer the question: At what time of day are women most likely to place an order? An important decision is the extent to which the data has been aggregated before being loaded into a data warehouse.

Dimensions

Dimensions contain a set of unique values that identify and categorize data. They form the edges of a cube, and thus of the measures within the cube. Because measures are typically multidimensional, a single value in a measure must be qualified by a member of each dimension to be meaningful. For example, the Sales measure has four dimensions: Time, Customer, Product, and Channel. A particular Sales value (43,613.50) only has meaning when it is qualified by a specific time period (Feb-01), a customer (Warren Systems), a product (Portable PCs), and a channel (Catalog).

Hierarchies and Levels

A **hierarchy** is a way to organize data at different levels of aggregation. In viewing data, analysts use dimension hierarchies to recognize trends at one level, drill down to lower levels to identify reasons for these trends, and roll up to higher levels to see what affect these trends have on a larger sector of the business.

Level-Based Hierarchies

Each **level** represents a position in the hierarchy. Each level above the base (or most detailed) level contains aggregate values for the levels below it. The members at different levels have a one-to-many **parent-child relation**. For example, `Q1-05` and `Q2-05` are the children of `2005`, thus `2005` is the parent of `Q1-05` and `Q2-05`.

Suppose a data warehouse contains snapshots of data taken three times a day, that is, every 8 hours. Analysts might normally prefer to view the data that has been aggregated into days, weeks, quarters, or years. Thus, the Time dimension needs a hierarchy with at least five levels.

Similarly, a sales manager with a particular target for the upcoming year might want to allocate that target amount among the sales representatives in his territory; the allocation requires a dimension hierarchy in which individual sales representatives are the child values of a particular territory.

Hierarchies and levels have a many-to-many relationship. A hierarchy typically contains several levels, and a single level can be included in multiple hierarchies.

Value-Based Hierarchies

Although hierarchies are typically composed of levels, they do not have to be. The parent-child relations among dimension members may not define meaningful levels. For example, in an employee dimension, each manager has one or more reports, which forms a parent-child relation. Creating levels based on these relations (such as individual contributors, first-level managers, second-level managers, and so forth) may not be meaningful for analysis. Likewise, the line item dimension of financial data does not have levels. This type of hierarchy is called a **value-based hierarchy.**

Attributes

An **attribute** provides additional information about the data. Some attributes are used for display. For example, you might have a product dimension that uses Stock Keeping Units (SKUs) for dimension members. The SKUs are an excellent way of uniquely identifying thousands of products, but are meaningless to most people if they are used to label the data in a report or a graph. You would define attributes for the descriptive labels.

You might also have attributes like colors, flavors, or sizes. This type of attribute can be used for data selection and answering questions such as: Which colors were the most popular in women's dresses in the summer of 2005? How does this compare with the previous summer?

Time attributes can provide information about the Time dimension that may be useful in some types of analysis, such as identifying the last day or the number of days in each time period.

Upgrading Oracle Database 10*g* Release 1 Analytic Workspaces

If you created an analytic workspace in Oracle 10*g* Release 1, you can upgrade it to Release 2 using the following procedure. Upgrading is optional. However, upgrading enables you to use the new features of Analytic Workspace Manager 10.2, such as additional aggregation operators for compressed composites, support for multiple languages, and performance improvements.

To upgrade an analytic workspace, take these steps:

1. Open Analytic Workspace Manager in the Model View.
2. In the navigation tree, select the name of the Oracle Database instance where your analytic workspace is stored.
3. On the Basic tab of the Database property sheet, verify that the database is running in 10.2 compatibility mode.
4. Right-click the analytic workspace, and select **Upgrade Analytic Workspace to 10.2**.
5. Complete the Analytic Workspace Upgrade to Version 10.2 dialog box.

   Click **Help** for additional information.

Upgrading Oracle9*i* Analytic Workspaces

If you have analytic workspaces that were created in Oracle9*i*, then you should upgrade them to take advantage of new features such as partitioning and compressed composites.

Upgrading may break custom OLAP DML programs. For this reason, you can choose to upgrade at a time that is convenient for you. You can continue to manage your older analytic workspaces by using an older version of Analytic Workspace Manager (such as Oracle9*i* Release 9.2.0.4.1).

Any new analytic workspaces that you create using the new Oracle Database 10*g* version of Analytic Workspace Manager is automatically in 10*g* standard form, as long as Oracle Database is running in 10*g* compatibility mode.

If Oracle Database is running in 9*i* compatibility mode, then you continue to work the same way as before without upgrading the analytic workspaces.

To upgrade an analytic workspace, take these steps:

1. Set the `COMPATIBLE` parameter to 10.0.0.0 or later in the database initialization file.
2. Upgrade the physical storage format.
3. Upgrade the standard form metadata.

You can upgrade the physical storage format without upgrading the standard form metadata, if you wish. This change improves performance and supports partitioning. However, the analytic workspace is not enabled dynamically for OracleBI Beans until you upgrade the metadata.

You can perform the upgrade steps either in the Object View of Analytic Workspace Manager or in PL/SQL.

Upgrading the Physical Storage Format

Convert the physical storage format by using either of these methods:

- Recreate the analytic workspace by following these steps:
  1. Export the contents to an EIF file.
  2. Delete the old analytic workspace.
  3. Create a new, empty analytic workspace.
  4. Import the contents from the EIF file.

  You can export and import in Analytic Workspace Manager. For more information, see these topics in Help: "Exporting Workspace Objects" and "Importing Workspace Objects"
- Use the PL/SQL conversion program:

  ```
  EXECUTE dbms_aw.convert('aw_name');
  ```

  Tip: Use a program such as SQL*Plus to execute this procedure. For the full syntax, refer to the *Oracle OLAP Reference*.

Upgrading the Standard Form Metadata

To upgrade the standard form metadata, follow these steps:

1. In Analytic Workspace Manager, open the Object View.
2. Expand the navigation tree until you see the name of the analytic workspace.
3. Right-click the analytic workspace and choose **Upgrade Analytic Workspace From 9i to 10g Standard Form** from the popup menu.
4. Upgrade to Release 2 by following the instructions in "Upgrading Oracle Database 10g Release 1 Analytic Workspaces" on page 1-10.

2

Getting Started with Oracle OLAP

This chapter describes the preliminary steps you must take to use Oracle OLAP. It assumes that you have already installed Oracle Database 10*g* Enterprise Edition. The OLAP option is installed automatically as part of a Basic installation of Oracle Database.

Note: To start querying dimensional objects immediately, install the Global analytic workspace, as described in "Installing the Sample Schema". Then follow the instructions in one of these chapters:

- Chapter 4, "Querying Dimensional Objects Using SQL"
- Chapter 5, "Querying Dimensional Objects Using OLAP Tools"

This chapter includes the following topics:

- Installing the Sample Schema
- Database Management Tasks
- Granting Privileges to DBAs and Application Developers
- Getting Started with Analytic Workspace Manager

Installing the Sample Schema

You can download and install the sample Global schema from the Oracle Web site and use it to try the examples shown throughout this guide:

`http://www.oracle.com/technetwork/database/options/olap/olap-downloads-098860.html`

Instructions for installing the schema are provided in the `README` file.

Sales History (`SH`) is a sample star schema that is delivered with Oracle Database. Although Global is used for most of the examples in this manual, Sales History has a very different set of data characteristics and demonstrates a correspondingly different set of build choices.

You can download a template for a Sales History analytic workspace from:

`http://www.oracle.com/technetwork/database/options/olap/prevsampschemasfordoc-085509.html`

Then you can simply examine the definitions of various objects instead of creating them manually. You still must run the Maintenance wizard to load the data.

Database Management Tasks

You should create undo, permanent, and temporary tablespaces that are appropriate for use by dimensional objects. Follow the recommendations in "Storage Management" on page 12-3.

Granting Privileges to DBAs and Application Developers

Anyone who needs to create or manage dimensional objects in Oracle Database must have the necessary privileges. These privileges are different from those needed just to query the data stored in dimensional objects, which are described in "Security of Multidimensional Data in Oracle Database" on page 12-4.

DBAs and application developers need the following roles and privileges.

To create dimensional objects in the user's own schema:

- `OLAP_USER` role
- `CREATE SESSION` privilege

To create dimensional objects in different schemas:

- `OLAP_DBA` role
- `CREATE SESSION` privilege

Users also need an unlimited quota on the tablespace in which the dimensional objects are stored. The tablespaces should be defined specifically for OLAP use, as described in Chapter 12.

If the source tables are in a different schema, then the owner of the dimensional objects needs `SELECT` object privileges on those tables.

Example 2–1 shows the SQL statements for creating the `GLOBAL` user.

Example 2–1 SQL Statements for Creating the GLOBAL User

```
CREATE USER "GLOBAL" IDENTIFIED BY password
   DEFAULT TABLESPACE glo
   TEMPORARY TABLESPACE glotmp
   QUOTA UNLIMITED ON glo
   PASSWORD EXPIRE;

GRANT OLAP_USER TO GLOBAL;
GRANT CREATE SESSION TO GLOBAL;
```

Getting Started with Analytic Workspace Manager

In this section, you learn how to install Analytic Workspace Manager software and make a connection to Oracle Database.

Installing Analytic Workspace Manager

You can install Analytic Workspace Manager from the Oracle Database installation media or from the Oracle Technology Network.

Analytic Workspace Manager is distributed on the Oracle Database Client installation media.

If you are installing on the same system as the database, then choose a **Custom** installation and install into the same Oracle home directory as the database. Select **OLAP Analytic Workspace Manager and Worksheet** from the list of components.

If you are installing on a remote system, then choose either an **Administrator** or a **Custom** installation. The Administrator choice automatically installs Analytic Workspace Manager on the client.

> **See Also:** An installation guide for your client platform, such as the *Oracle Database Client Quick Installation Guide for 32-Bit Windows*.

A more recent version of Analytic Workspace Manager may be available for download from the Oracle Technology Network. Check the available downloads at

`http://www.oracle.com/technetwork/database/options/olap/olap-downloads-098860.html`

Follow the installation instructions provided in the README file.

Opening Analytic Workspace Manager

On Windows, open Analytic Workspace Manager from the Start menu. Choose **Oracle - *Oracle_home***, then **Integrated Management Tools**, and then **OLAP Analytic Workspace Manager and Worksheet**.

On Linux, open Analytic Workspace Manager from the shell command line:

```
$ORACLE_HOME/olap/awm/awm.sh
```

Defining a Database Connection

You can define a connection to each database that you use for OLAP. After you define a connection, the database instance is listed in the navigation tree for you to access at any time.

To define a database connection:

1. Right-click the top Databases folder in the navigation tree, then choose **Add Database to Tree** from the shortcut menu.
2. Complete the Add Database to Tree dialog box.

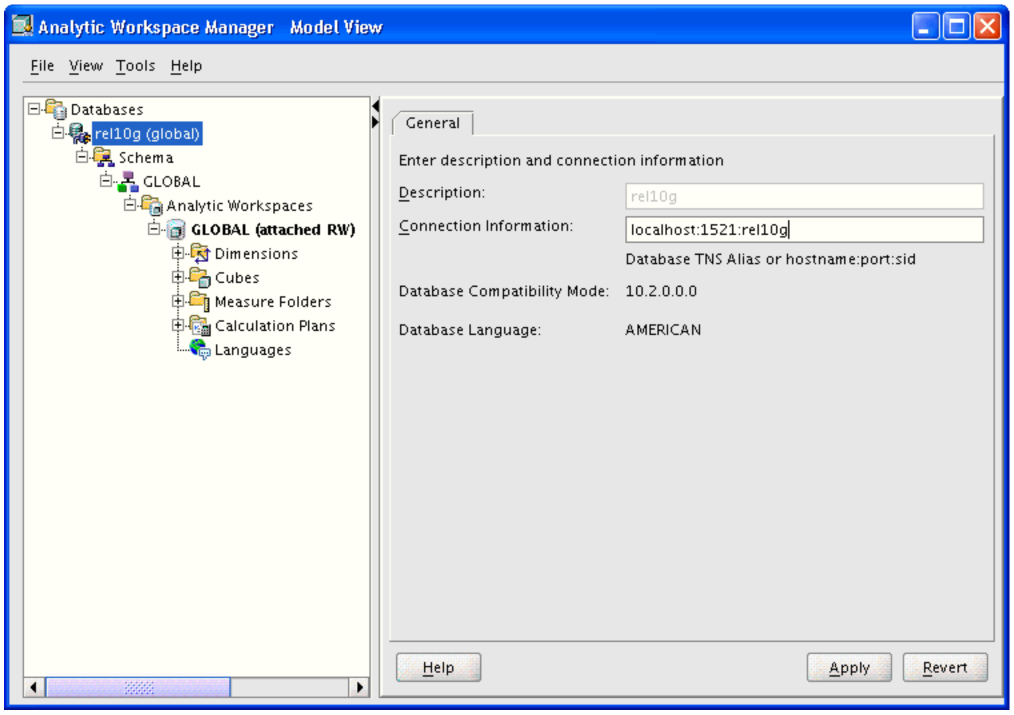
Opening a Database Connection

To connect to a database:

1. Click the plus icon (+) next to a database in the navigation tree.
2. Complete the Connect to Database dialog box.

Figure 2–1 shows Analytic Workspace Manager displaying the properties of the database connection by the Global user.

Figure 2–1 Analytic Workspace Manager Connection to Oracle Database



Enter description and connection information

Description: rel10g

Connection Information: localhost:1521:rel10g

Database TNS Alias or hostname:port:sid

Database Compatibility Mode: 10.2.0.0.0

Database Language: AMERICAN

Installing Plug-ins

Plug-ins extend the functionality of Analytic Workspace Manager. Any Java developer can create a plug-in. Plug-ins are distributed as JAR files. The developer should provide information about what the plug-in does and how to use it.

If you have one or more plug-ins, then you only need to identify their location to Analytic Workspace Manager.

To Use Plug-ins:

1. Create a local directory for storing plug-ins for Analytic Workspace Manager.
2. Copy the JAR files to that directory.
3. Open Analytic Workspace Manager.
4. Choose **Configuration** from the Tools menu.

   The Configuration dialog box opens.
5. Select **Enable Plug-ins** and identify the plug-in directory. Click **OK**.
6. Close and reopen Analytic Workspace Manager.

   The new functionality provided by the plug-ins is available in the navigator.

> **See Also:** *Developing Analytic Workspace Manager Plug-ins*, which you can download from the Oracle Technology Network at `http://www.oracle.com/technetwork/database/options/olap/index.html`.

3

Creating Dimensions and Cubes

This chapter explains how to design a dimensional data model and create dimensions and cubes using Analytic Workspace Manager.

This chapter contains the following topics:

Designing a Dimensional Model for Your Data

Chapter 1 introduced the dimensional objects: Cubes, measures, dimensions, levels, hierarchies, and attributes. In this chapter, you learn how to define them in Oracle Database, but first you must decide upon the dimensional model you want to create. What are your measures? What are your dimensions? How can you distinguish between a dimension and an attribute in your data? You can design a dimensional model using pencil and paper, a database design software package, or any other method that suits you.

If your source data is already in a star or snowflake schema, then you already have the elements of a dimensional model:

- Fact tables correspond to cubes.
- Data columns in the fact tables correspond to measures.
- Foreign key constraints in the fact tables identify the dimension tables.
- Dimension tables identify the dimensions.
- Primary keys in the dimension tables identify the base-level dimension members.
- Parent columns in the dimension tables identify the higher level dimension members.
- Columns in the dimension tables containing descriptions and characteristics of the dimension members identify the attributes.

You can also get insights into the dimensional model by looking at the reports currently being generated from the source data. The reports identify the levels of aggregation that interest the report consumers and the attributes used to qualify the data.

While investigating your source data, you may decide to create relational views that more closely match the dimensional model that you plan to create.

See Also:

"Overview of the Dimensional Data Model" on page 1-8 for an introduction to dimensional objects

Appendix A, "Designing a Dimensional Model," for a case study of developing a dimensional model for the Global analytic workspace

Introduction to Analytic Workspace Manager

Your goal in using Analytic Workspace Manager is to create a multidimensional data store that supports business analysis. Analytic Workspace Manager is the primary tool for creating, developing, and managing analytic workspaces. The main window provides two views: the Model View and the Object View. You can switch between views using the View menu. In addition, there are menus, a toolbar, a navigation tree, and property sheets. When you select an object in the navigation tree, the property sheet to the right provides detailed information about that object. When you right-click an object, you get a choice of menu items with appropriate actions for that object.

Analytic Workspace Manager has a full online Help system, which includes context-sensitive Help.

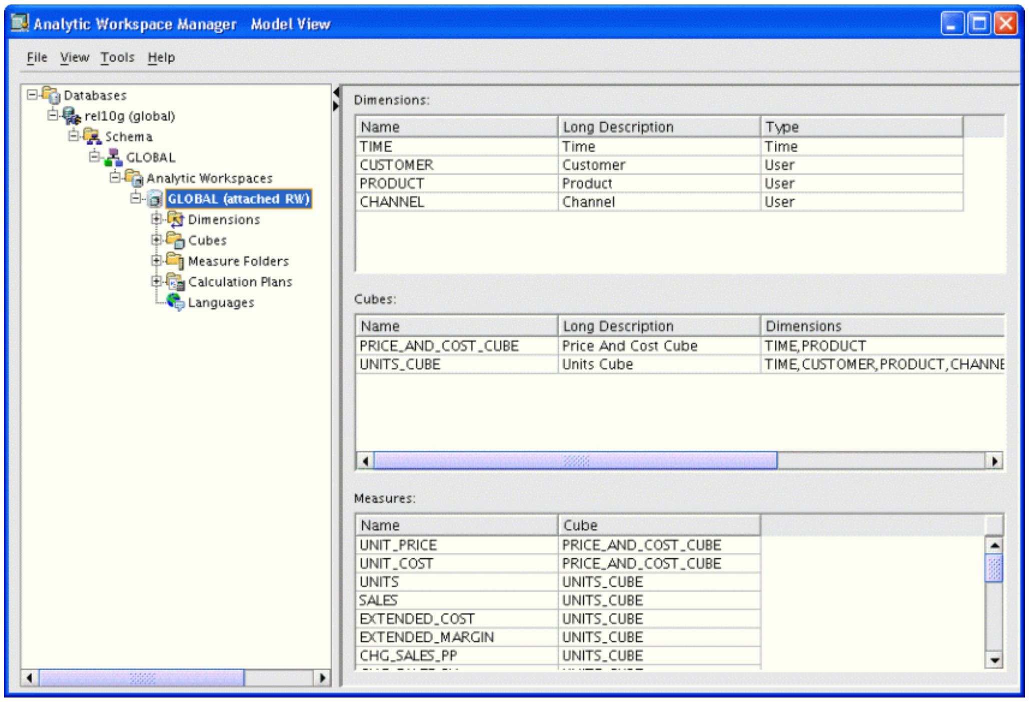
Model View

The Model View enables you to define a multidimensional model composed of dimensions, levels, hierarchies, attributes, measures, calculated measures, and measure folders. The model is stored in the analytic workspace as **database standard form** metadata.

A drag-and-drop user interface facilitates mapping of the objects to columns in relational tables, views, and synonyms in Oracle Database. The source columns can be star, snowflake, or any other schema design that supports the multidimensional model.

Figure 3–1 shows the dimensions, cubes, and measures created in the `GLOBAL` analytic workspace.

Figure 3–1 Model View in Analytic Workspace Manager



Dimensions:

Name	Long Description	Type
TIME	Time	Time
CUSTOMER	Customer	User
PRODUCT	Product	User
CHANNEL	Channel	User

Cubes:

Name	Long Description	Dimensions
PRICE_AND_COST_CUBE	Price And Cost Cube	TIME,PRODUCT
UNITS_CUBE	Units Cube	TIME,CUSTOMER,PRODUCT,CHANNE

Measures:

Name	Cube
UNIT_PRICE	PRICE_AND_COST_CUBE
UNIT_COST	PRICE_AND_COST_CUBE
UNITS	UNITS_CUBE
SALES	UNITS_CUBE
EXTENDED_COST	UNITS_CUBE
EXTENDED_MARGIN	UNITS_CUBE
CHG_SALES_PP	UNITS_CUBE

Object View

The Object View provides a graphical user interface to the OLAP DML. You can create, modify, and delete individual workspace objects. This view is provided for users who are familiar with the OLAP DML and want to upgrade from Express databases or modify custom applications. You should not use this view to manually change a standard form analytic workspace, because you may create inconsistencies in the metadata.

Creating a Dimensional Data Store Using Analytic Workspace Manager

An analytic workspace is a container for storing related cubes. You create dimensions, cubes, and other dimensional objects within the context of an analytic workspace.

Basic Steps for Creating an Analytic Workspace

To create an analytic workspace:

1. Open Analytic Workspace Manager and connect to your database instance as the user defined for this purpose.
2. Create a new analytic workspace container in your database:
   a. In the Model View navigation tree, expand the folders until you see the schema where you want to create the analytic workspace.
   b. Right-click the schema name, then choose **Create Analytic Workspace** from the shortcut menu.
   c. Complete the Create Analytic Workspace dialog box, then choose **Create**.

      The new analytic workspace appears in the Analytic Workspaces folder for the schema.

3. Define the dimensions for the data.

   See "Creating Dimensions" on page 3-4.

4. Define the cubes for the data.

   See "Creating Cubes" on page 3-12.

When you have finished, you have an analytic workspace populated with the detail data fetched from relational tables or views. You may also have summarized data and calculated measures.

Adding Functionality to Dimensional Objects

In addition to the basic steps, you can add functionality to an analytic workspace in these ways:

- Define measure folders to simplify access for end users.

  See "Defining Measure Folders" on page 3-19.

- Support multiple languages by adding translations of metadata and dimension attributes.

  See "Supporting Multiple Languages" on page 3-20.

How Analytic Workspace Manager Saves Changes

Analytic Workspace Manager saves changes automatically that you make to the analytic workspace. You do not explicitly save your changes.

Saves occur when you take an action such as these:

- Click **OK** or the equivalent button in a dialog box.

  For example, when you click **Import** in the Import From EIF File dialog box, the contents are imported, and the revised analytic workspace is committed to the database. Likewise, when you click **Create** in the Create Dimension dialog box, the new dimension is committed to the database.

- Click **Apply** in a property sheet.

  For example, when you change the labels on the General property page for an object, the change takes effect when you click **Apply**.

Creating Dimensions

Dimensions are lists of unique values that identify and categorize data. They form the edges of a logical cube, and thus of the measures within the cube.

Dimensions are the parents of levels, hierarchies, and attributes in the logical model. You define these supporting objects, in addition to the dimension itself, in order to have a fully functional dimension.

You can define dimensions that have any of these common forms:

- List or flat dimensions that have no levels or hierarchies.
- Level-based dimensions that use parent-child relationships to group members into levels. Most dimensions are level-based.
- Value-based dimensions that have parent-child relationships among their members, but these relationships do not form meaningful levels.

Dimension Members Must Be Unique

Every dimension member must be a unique value. Depending on your data, you can create a dimension that uses either natural keys or surrogate keys from the relational sources for its members.

- **Natural keys** are read from the relational sources without modification. To use natural keys, the values must be unique across levels. Because each level may be mapped to a different relational column, this uniqueness may not be enforced in the source data.

  For example, a Geography source table might have a value of `NEW_YORK` in the `CITY` column and a value of `NEW_YORK` in the `STATE` column. Unless you take steps to assure uniqueness, the second value for `NEW_YORK` overwrites the first.

  If a dimension is flat or value-based, then it must use natural keys because no levels are defined as metadata. You must take whatever steps you need to assure that the dimension members are unique.

- **Surrogate keys** ensure uniqueness by adding a level prefix to the members while loading them into the analytic workspace. For the previous example, surrogate keys create two dimension members named `CITY_NEW_YORK` and `STATE_NEW_YORK`, instead of a single member named `NEW_YORK`. A dimension that has surrogate keys must be defined with at least one level-based hierarchy.

Time Dimensions Have Special Requirements

You can define dimensions as either User or Time dimensions. Business analysis is performed on historical data, so fully defined time periods are vital. A time dimension table must have columns for period end dates and time span. These required attributes support time-series analysis, such as comparisons with earlier time periods. If this information is not available, then you can define Time as a User dimension, but it will not support time-based analysis.

You must define a Time dimension with at least one level to support time-based analysis, such as a custom measure that calculates the difference from the prior period.

To create a dimension:

1. Expand the folder for the analytic workspace.
2. Right-click **Dimensions**, then choose **Create Dimension** from the shortcut menu.

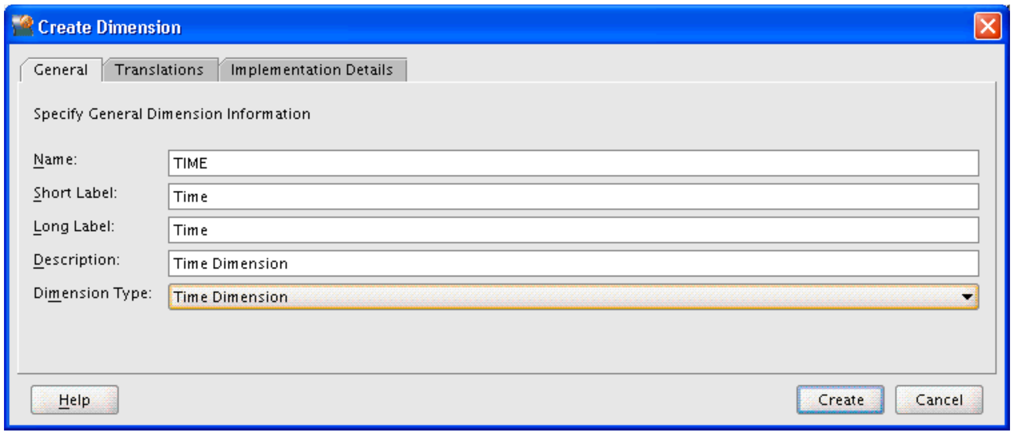
   The Create Dimension dialog box is displayed.
3. Complete all tabs.

   Click **Help** for specific information about your choices.
4. Click **Create**.

   The new dimension appears as a subfolder under Dimensions.

Figure 3–2 shows the creation of the Time dimension.

Figure 3–2 Creation of the Time Dimension



Specify General Dimension Information

Name:	TIME
Short Label:	Time
Long Label:	Time
Description:	Time Dimension
Dimension Type:	Time Dimension

Creating Levels

For business analysis, data is typically summarized by level. For example, your database may contain daily snapshots of a transactional database. Days are thus the base level. You might summarize this data at the weekly, quarterly, and yearly levels.

Levels have parent-child or one-to-many relationships, which form a **level-based hierarchy**. For example, each week summarizes seven days, each quarter summarizes 13 weeks, and each year summarizes four quarters. This hierarchical structure enables analysts to detect trends at the higher levels, then drill down to the lower levels to identify factors that contributed to a trend.

For each level that you define, you must identify a data source for dimension members at that level. Members at all levels are stored in the same dimension. In the previous example, the Time dimension contains members for weeks, quarters, and years.

To create a level:

1. Expand the folder for the dimension.
2. Right-click **Levels**, then choose **Create Level**.

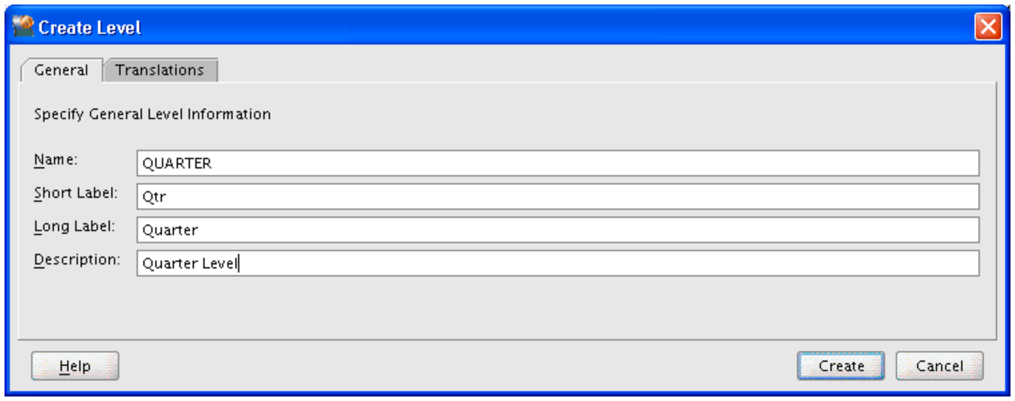
   The Create Level dialog box is displayed.
3. Complete all tabs of the Create Level dialog box.

   Click **Help** for specific information about these choices.
4. Click **Create**.

   The new level appears as an item in the Levels folder.

Figure 3–3 shows the creation of the Quarter level for the Time dimension.

Figure 3–3 Creation of the Quarter Level



Specify General Level Information

Name:	QUARTER
Short Label:	Qtr
Long Label:	Quarter
Description:	Quarter Level

Creating Hierarchies

Dimensions can have one or more hierarchies. Most hierarchies are level-based. Analytic Workspace Manager supports these common types of level-based hierarchies:

- **Normal 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** contain at least one member with a different base, creating a "ragged" base level for the hierarchy.
- **Skip-level hierarchies** contain at least one member whose parents are multiple levels above it, 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.

You may also have dimensions with parent-child relations that do 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. Similarly, you might have a line-item dimension with members that cannot be grouped into meaningful levels. In this situation, you can create a **value-based hierarchy** defined by the parent-child relations, which does not have named levels. You can create value-based hierarchies only for dimensions that use natural keys, because surrogate keys are formed with the names of the levels.

To create a hierarchy:

1. Expand the folder for the dimension.
2. Right-click **Hierarchies**, then choose **Create Hierarchy**.

   The Create Hierarchy dialog box is displayed.
3. Complete all tabs of the Create Hierarchy dialog box.

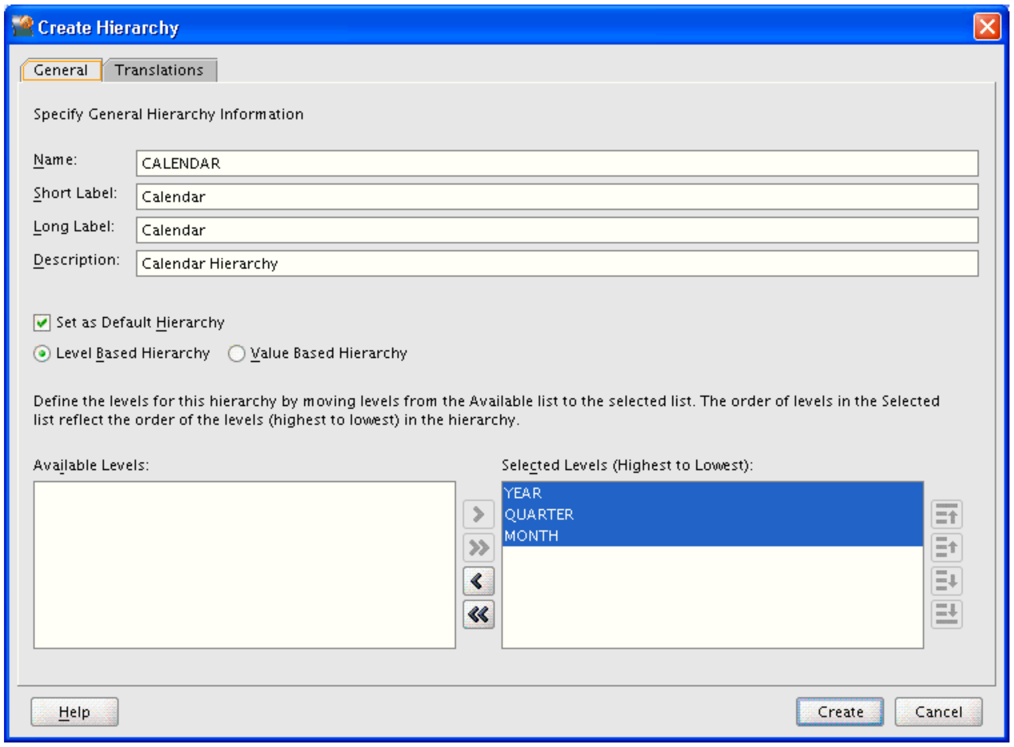
   If you define multiple hierarchies, be sure to define one of them as the default hierarchy.

   Click **Help** for specific information about these choices.
4. Click **Create**.

The new hierarchy appears as an item in the Hierarchies folder.

Figure 3–4 shows creation of the Calendar hierarchy for the Time dimension. Time has only one hierarchy, so Calendar is the default hierarchy.

Figure 3–4 Creation of the Calendar Hierarchy



Specify General Hierarchy Information

Name:	CALENDAR
Short Label:	Calendar
Long Label:	Calendar
Description:	Calendar Hierarchy

✓ Set as Default Hierarchy

[✓ Level Based Hierarchy] Value Based Hierarchy

Define the levels for this hierarchy by moving levels from the Available list to the selected list. The order of levels in the Selected list reflect the order of the levels (highest to lowest) in the hierarchy.

Available Levels:

Selected Levels (Highest to Lowest):
YEAR
QUARTER
MONTH

Creating Attributes

Attributes provide information about the individual members of a dimension. They are used for labeling crosstabular and graphical data displays, selecting data, organizing dimension members, and so forth.

Automatically Defined Attributes

Analytic Workspace Manager creates some attributes automatically when creating a dimension. These attributes have a unique type, such as "Member Long Description," which OLAP client applications expect to find.

All dimensions are created with long and short description attributes. If your source tables include long and short descriptions, then you can map the attributes to the appropriate columns. However, if your source tables include only one set of labels, then you should always map the long description attributes. You can decide whether to map the short description attributes to the same column. If you do, the data is loaded twice.

Discoverer Plus OLAP, Spreadsheet Add-In, and OracleBI Beans use long description attributes in selection lists and for labeling crosstabs and graphs. The Add-In initially makes limited use of short description attributes, but users can switch to long descriptions. If the appropriate descriptions are not available, then these tools use dimension members. For example, if the Product dimension has short descriptions but no long descriptions, then the tools display Product dimension members.

Time dimensions are created with time-span and end-date attributes. This information must be provided for all Time dimension members.

Be sure to examine all of these attribute definitions, because you may wish to change the default settings. In particular, expand the hierarchy tree on the Basic tab to verify that the correct levels are selected. These choices affect the number of columns that you can map to the dimension.

User-Defined Attributes

You can create additional "User" attributes that provide supplementary information about the dimension members.

To create an attribute:

1. Expand the folder for the dimension.
2. Right-click **Attributes**, then choose **Create Attribute**.

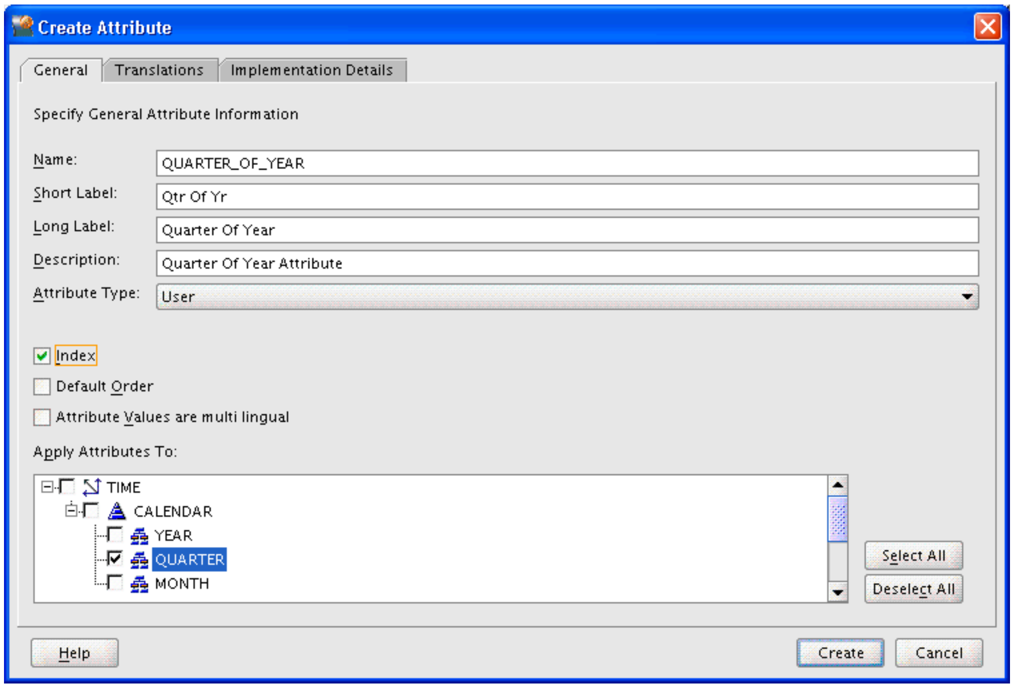
   The Create Attribute dialog box is displayed.
3. Complete all tabs of the Create Attribute dialog box.

   Click **Help** for specific information about these choices.
4. Click **Create**.

   The new attribute appears as an item in the Attributes folder.

Figure 3–5 shows the creation of an attribute that pertains only to dimension members at the Quarter level of the Time dimension.

Figure 3–5 Creation of the Time Quarter of Year Attribute



Specify General Attribute Information

Name:	QUARTER_OF_YEAR
Short Label:	Qtr Of Yr
Long Label:	Quarter Of Year
Description:	Quarter Of Year Attribute
Attribute Type:	User

- [✓] Index
- [ ] Default Order
- [ ] Attribute Values are multi lingual

Apply Attributes To:

- [ ] TIME
  - [ ] CALENDAR
    - [ ] YEAR
    - [✓] QUARTER
    - [ ] MONTH

Mapping Dimensions

Mapping identifies the relational data source for each dimensional object. After mapping a dimension to a column of a relational table or view, you can load the data.

You can create, map, and load each dimension individually, or perform each step for all dimensions before proceeding to the next step.

Mapping Window

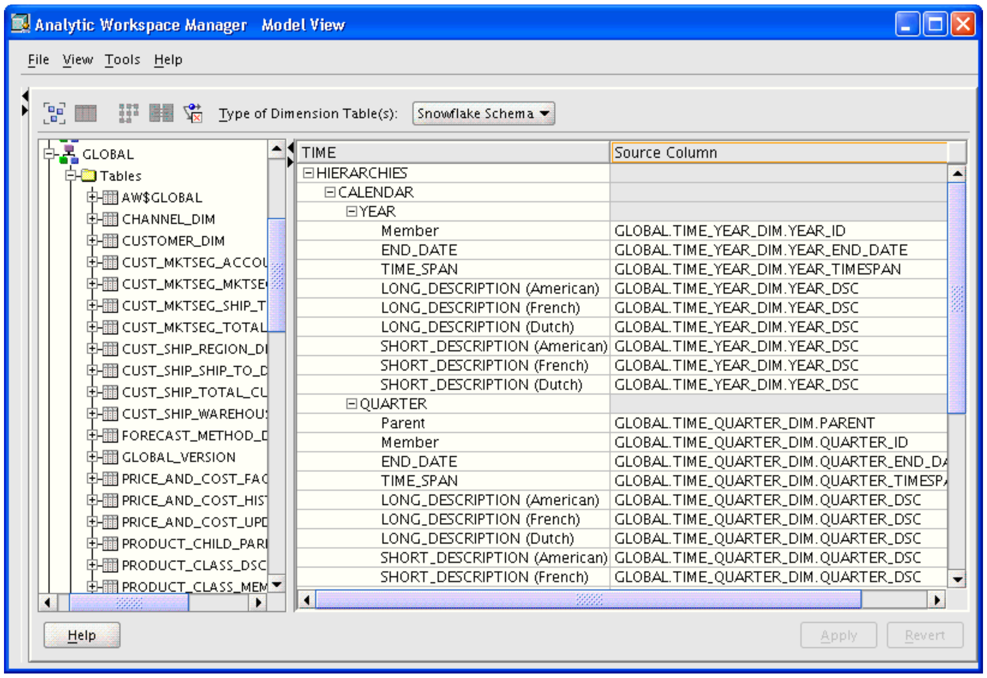
The mapping window has a tabular view and a graphical view.

- **Tabular view**. Drag-and-drop the names of individual columns from the schema navigation tree to the rows for the logical objects.
- **Graphical view**. Drag-and-drop icons, which represent tables and views, from the schema navigation tree onto the mapping canvas. Then you draw lines from the columns to the logical objects.

Click Help on the Mapping page for more information. When you are done mapping the dimension, click **Apply**.

Figure 3–6 shows the `TIME` dimension mapped in the tabular view. The toolbar appears across the top and the schema navigation tree is on the left.

Figure 3–6 Time Dimension Mapped in a Tabular View



Type of Dimension Table(s): Snowflake Schema

```
GLOBAL
  Tables
    AW$GLOBAL
    CHANNEL_DIM
    CUSTOMER_DIM
    CUST_MKTSEG_ACCOU
    CUST_MKTSEG_MKTSE
    CUST_MKTSEG_SHIP_T
    CUST_MKTSEG_TOTAL
    CUST_SHIP_REGION_DI
    CUST_SHIP_SHIP_TO_D
    CUST_SHIP_TOTAL_CU
    CUST_SHIP_WAREHOU
    FORECAST_METHOD_D
    GLOBAL_VERSION
    PRICE_AND_COST_FAC
    PRICE_AND_COST_HIS
    PRICE_AND_COST_UPD
    PRODUCT_CHILD_PARI
    PRODUCT_CLASS_DSC
    PRODUCT_CLASS_MEM
```

TIME	Source Column
⊟HIERARCHIES	
⊟CALENDAR	
⊟YEAR	
Member	GLOBAL.TIME_YEAR_DIM.YEAR_ID
END_DATE	GLOBAL.TIME_YEAR_DIM.YEAR_END_DATE
TIME_SPAN	GLOBAL.TIME_YEAR_DIM.YEAR_TIMESPAN
LONG_DESCRIPTION (American)	GLOBAL.TIME_YEAR_DIM.YEAR_DSC
LONG_DESCRIPTION (French)	GLOBAL.TIME_YEAR_DIM.YEAR_DSC
LONG_DESCRIPTION (Dutch)	GLOBAL.TIME_YEAR_DIM.YEAR_DSC
SHORT_DESCRIPTION (American)	GLOBAL.TIME_YEAR_DIM.YEAR_DSC
SHORT_DESCRIPTION (French)	GLOBAL.TIME_YEAR_DIM.YEAR_DSC
SHORT_DESCRIPTION (Dutch)	GLOBAL.TIME_YEAR_DIM.YEAR_DSC
⊟QUARTER	
Parent	GLOBAL.TIME_QUARTER_DIM.PARENT
Member	GLOBAL.TIME_QUARTER_DIM.QUARTER_ID
END_DATE	GLOBAL.TIME_QUARTER_DIM.QUARTER_END_DA
TIME_SPAN	GLOBAL.TIME_QUARTER_DIM.QUARTER_TIMESPA
LONG_DESCRIPTION (American)	GLOBAL.TIME_QUARTER_DIM.QUARTER_DSC
LONG_DESCRIPTION (French)	GLOBAL.TIME_QUARTER_DIM.QUARTER_DSC
LONG_DESCRIPTION (Dutch)	GLOBAL.TIME_QUARTER_DIM.QUARTER_DSC
SHORT_DESCRIPTION (American)	GLOBAL.TIME_QUARTER_DIM.QUARTER_DSC
SHORT_DESCRIPTION (French)	GLOBAL.TIME_QUARTER_DIM.QUARTER_DSC

Source Data Query

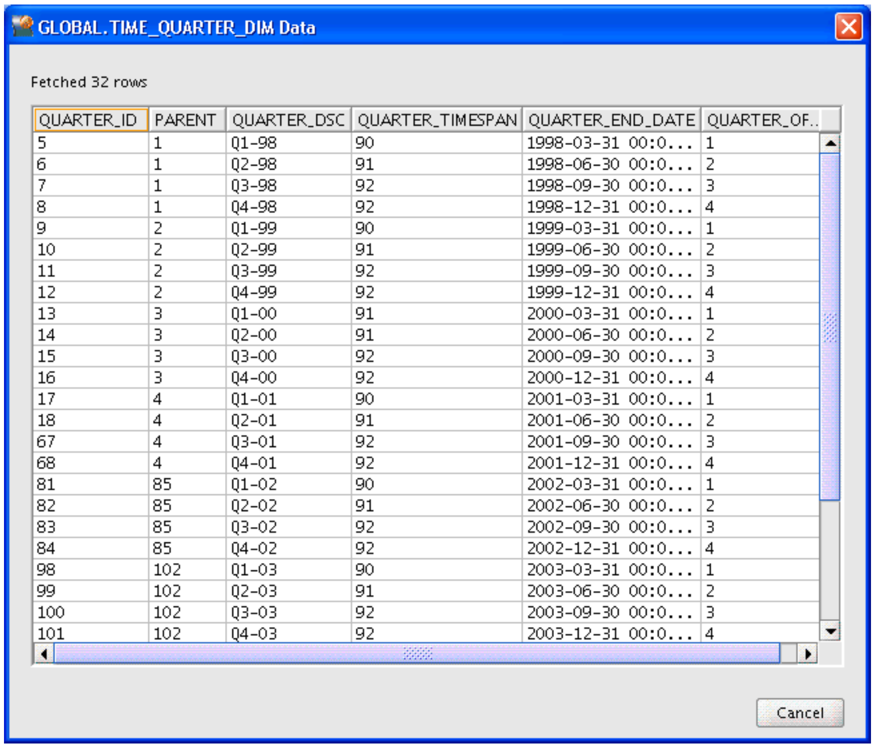
You can view the contents of a particular source column without leaving the mapping window. The information is readily available, eliminating the guesswork when the names are not adequately descriptive.

To see the values in a particular source table or view:

1. Right-click the source object in either the schema tree or the graphical view of the mapping canvas.
2. Choose **View Data** from the shortcut menu.

Figure 3–7 shows the data stored in the `TIME_QUARTER_DIM` table.

Figure 3–7 Data in a Source Dimension Table



Fetched 32 rows

QUARTER_ID	PARENT	QUARTER_DSC	QUARTER_TIMESPAN	QUARTER_END_DATE	QUARTER_OF..
5	1	Q1-98	90	1998-03-31 00:0...	1
6	1	Q2-98	91	1998-06-30 00:0...	2
7	1	Q3-98	92	1998-09-30 00:0...	3
8	1	Q4-98	92	1998-12-31 00:0...	4
9	2	Q1-99	90	1999-03-31 00:0...	1
10	2	Q2-99	91	1999-06-30 00:0...	2
11	2	Q3-99	92	1999-09-30 00:0...	3
12	2	Q4-99	92	1999-12-31 00:0...	4
13	3	Q1-00	91	2000-03-31 00:0...	1
14	3	Q2-00	91	2000-06-30 00:0...	2
15	3	Q3-00	92	2000-09-30 00:0...	3
16	3	Q4-00	92	2000-12-31 00:0...	4
17	4	Q1-01	90	2001-03-31 00:0...	1
18	4	Q2-01	91	2001-06-30 00:0...	2
67	4	Q3-01	92	2001-09-30 00:0...	3
68	4	Q4-01	92	2001-12-31 00:0...	4
81	85	Q1-02	90	2002-03-31 00:0...	1
82	85	Q2-02	91	2002-06-30 00:0...	2
83	85	Q3-02	92	2002-09-30 00:0...	3
84	85	Q4-02	92	2002-12-31 00:0...	4
98	102	Q1-03	90	2003-03-31 00:0...	1
99	102	Q2-03	91	2003-06-30 00:0...	2
100	102	Q3-03	92	2003-09-30 00:0...	3
101	102	Q4-03	92	2003-12-31 00:0...	4

Loading Data Into Dimensions

Analytic Workspace Manager provides several ways to load data into dimensional objects. The quickest way when developing a data model is using the default choices of the Maintenance Wizard. Other methods may be more appropriate in a production environment than the one used here.

To load data into the dimensions:

1. In the navigation tree, right-click the Dimensions folder or the folder for a particular dimension, then choose **Maintain Dimension**.

   The Maintenance Wizard opens on the Select Objects page.

2. Select one or more dimensions from Available Target Objects and use the shuttle buttons to move them to Selected Target Objects.

3. Click **Finish** to load the dimension values immediately.

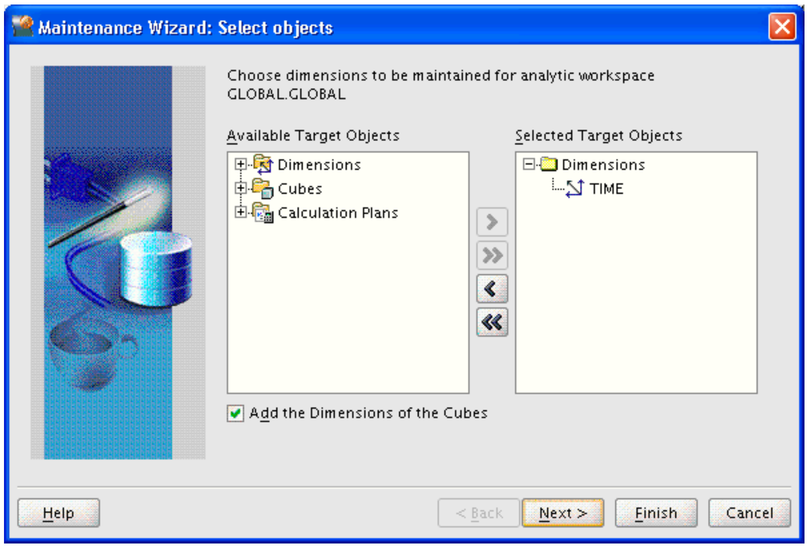
   The additional pages of the wizard enable you to create a SQL script or submit the load to the Oracle job queue. To use these options, click **Next** instead.

4. Review the build log, which appears when the build is complete. If the log shows that errors occurred, then fix them and run the Maintenance Wizard again.

   Errors are typically caused by problems in the mapping. Check for incomplete mappings or changes to the source objects.

Figure 3–8 shows the first page of the Maintenance Wizard. Only the Time dimension has been selected for maintenance. All the Time dimension members and attributes are fetched from the mapped relational sources.

Figure 3–8 Loading Dimension Values into the Time Dimension


Maintenance Wizard: Select objects
Choose dimensions to be maintained for analytic workspace GLOBAL.GLOBAL
Available Target Objects
Dimensions
Cubes
Calculation Plans
Selected Target Objects
Dimensions
TIME
Add the Dimensions of the Cubes
Help
< Back
Next >
Finish
Cancel

Displaying the Dimension Members

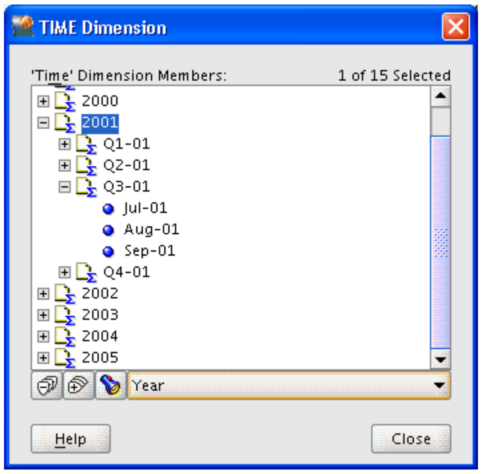
After loading a dimension, you can see its members by using the dimension viewer.

To display dimension members:

1. In the navigation tree, right-click the name of a dimension.
2. Choose **View Data**.

Figure 3–9 shows the Time dimension in the dimension viewer.

Figure 3–9 Displaying the Time Dimension



Creating Cubes

Cubes are informational objects that identify measures with the exact same dimensions and thus are candidates for being processed together at all stages: data loading, aggregation, storage, and querying.

Cubes define the shape of your business measures. They are defined by a set of ordered dimensions. The dimensions form the edges of a cube, and the measures are the cells in the body of the cube.

To create a cube:

1. Expand the folder for the analytic workspace.
2. Right-click **Cubes**, then choose **Create Cube**.

   The Create Cube dialog box is displayed.
3. Complete all tabs except Implementation Details of the Create Cube dialog box.

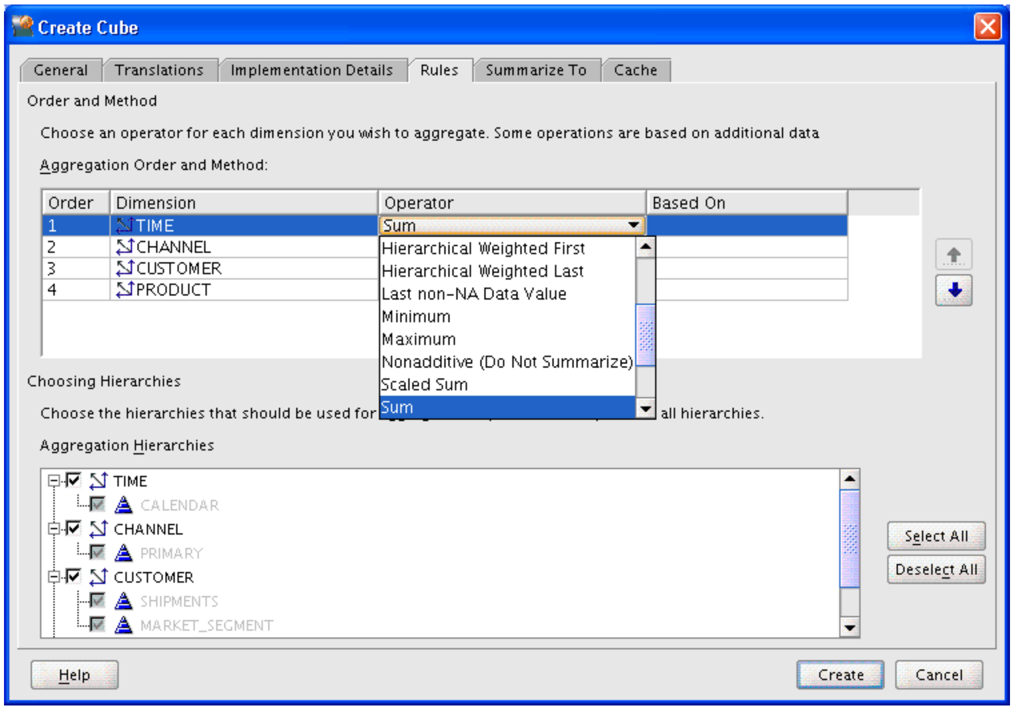
   Important: After mapping the cube, run the Sparsity Advisor to see the recommended settings for the Implementation Details tab. For more information about the Summary To tab, refer to Chapter 8.
4. Click **Create**.

   The new cube appears as a subfolder under **Cubes**.

Figure 3–10 shows the Rules tab for the Units cube, with the list of aggregation operators displayed.

> **See Also:** For descriptions of the aggregation operators:
>
> - "Aggregation Operators" on page 8-3
> - Click **Help** on the Rules tab in Analytic Workspace Manager

Figure 3–10 Selecting an Aggregation Operator



Order and Method

Choose an operator for each dimension you wish to aggregate. Some operations are based on additional data

Aggregation Order and Method:

Order	Dimension	Operator	Based On
1	TIME	Sum	
2	CHANNEL		
3	CUSTOMER		
4	PRODUCT		

Hierarchical Weighted First
Hierarchical Weighted Last
Last non-NA Data Value
Minimum
Maximum
Nonadditive (Do Not Summarize)
Scaled Sum
Sum

Choosing Hierarchies

Choose the hierarchies that should be used for ... all hierarchies.

Aggregation Hierarchies

- ☑ TIME
  - ☑ CALENDAR
- ☑ CHANNEL
  - ☑ PRIMARY
- ☑ CUSTOMER
  - ☑ SHIPMENTS
  - ☑ MARKET_SEGMENT

Creating Measures

Measures store the facts collected about your business. Each measure belongs to a particular cube, and thus shares particular characteristics with other measures in the cube, such as the same dimensions. The default attributes of a measure are inherited from the cube.

To create a measure:

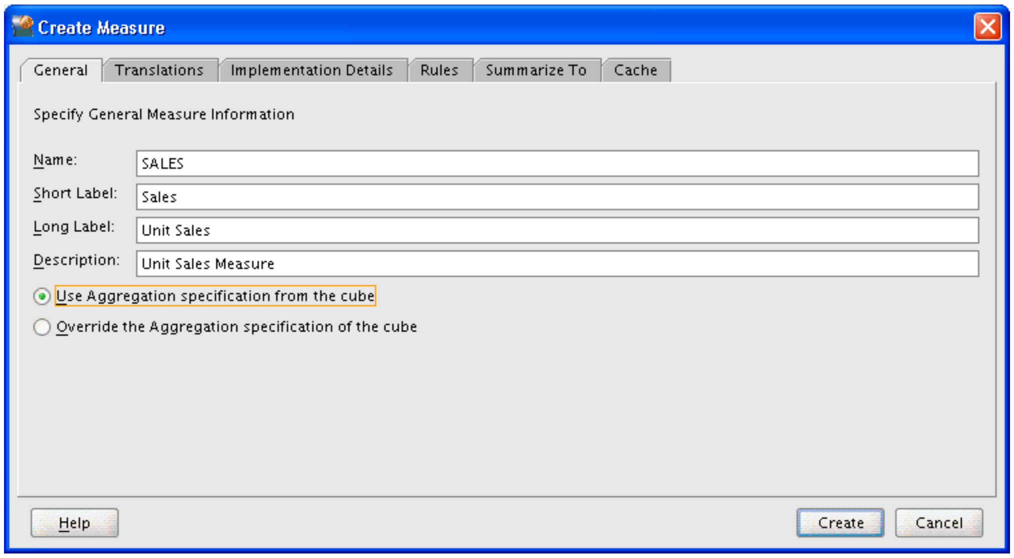
1. Expand the folder for the cube that has the dimensions of the new measure.
2. Right-click **Measures**, then choose **Create Measure**.

   The Create Measure dialog box is displayed.
3. Complete the General tab of the Create Measure dialog box. Complete the other tabs to override the cube settings.
4. Click **Create**.

   The new measure appears as an item in the Measures folder.

Figure 3–11 shows the General tab of the Create Measure dialog box.

Figure 3–11 Creating the Sales Measure



Specify General Measure Information

Name:	SALES
Short Label:	Sales
Long Label:	Unit Sales
Description:	Unit Sales Measure

[✓ Use Aggregation specification from the cube]
Override the Aggregation specification of the cube

Mapping Cubes

You use the same interface to map cubes as you did to map dimensions, as described in "Mapping Dimensions" on page 3-9.

To map a cube in the graphical view:

1. Define the cube and its measures.
2. In the navigation tree, expand the **Cubes** folder and click **Mappings**.

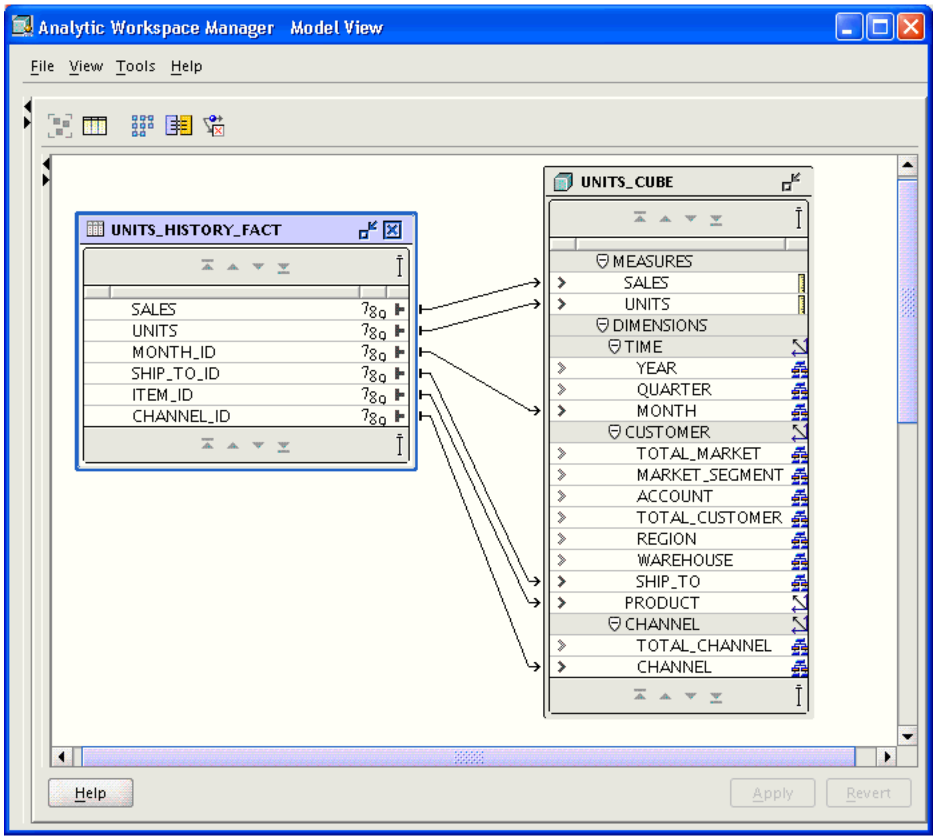
   The Mapping Window is displayed in the right pane. You see a schema navigation tree and a table with rows for the measures, dimensions, and levels.
3. Enlarge the mapping window by dragging the divider to the left.
4. In the schema navigation tree, locate the tables with the measures. Drag-and-drop them onto the mapping canvas.
5. Draw lines from the source columns to the target objects.

To draw a line, click the output connector of the source column and drag it to the input connector of the target object. You must map both the measures and the related dimension keys.

6. To uncross the lines, click the Auto Arrange Mappings tool.
7. When you have mapped all objects for the dimension, drag the divider to the right to restore access to the navigation tree.

Figure 3–12 shows the mapping canvas with the Price and Cost cube mapped to columns in the `PRICE_AND_COST_HIST_FACT` table. The mapping toolbar is at the top, and the schema navigation tree is on the left.

Figure 3–12 Units Cube Mapped in Graphical View



Choosing a Data Storage Strategy

The creation of a cube requires several decisions about data storage that affect the performance of the analytic workspace. These choices are on the Implementation Details tab for the cube. The Sparsity Advisor in Analytic Workspace Manager evaluates the data in the relational tables and recommends the appropriate settings. You can accept the recommendations or modify them before implementing them in the cube.

Partitioning a Cube

Partitioning is a method of physically storing the measures in a cube. It 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. Each partition can be aggregated by a separate process.
- Simplifies removal of old data from storage. Old partitions can be dropped, and new partitions can be added.

The number of partitions affects the database resources that can be allocated to loading and aggregating the data in a cube. Partitions can be aggregated simultaneously when sufficient resources have been allocated.

The Sparsity Advisor analyzes the source tables and develops a partitioning strategy. You can accept the recommendations of the Sparsity Advisor, or you can make your own decisions about partitioning. Analytic Workspace Manager does not partition cubes by default.

Choosing a Dimension for Partitioning

If your partitioning strategy is driven primarily by life-cycle management considerations, then you should partition the cube on the Time dimension. Old time periods can then be dropped as a unit, and new time periods added as a new partition. In Figure 3–14, for instance, the Quarter level of the Time dimension is used as the partitioning key.

If life-cycle management is not a primary consideration, then partition on whatever dimension the Sparsity Advisor recommends. If the source fact table is partitioned, then the cube is partitioned on the same dimension. Otherwise, the Sparsity Advisor develops a strategy designed to achieve optimal build and query performance.

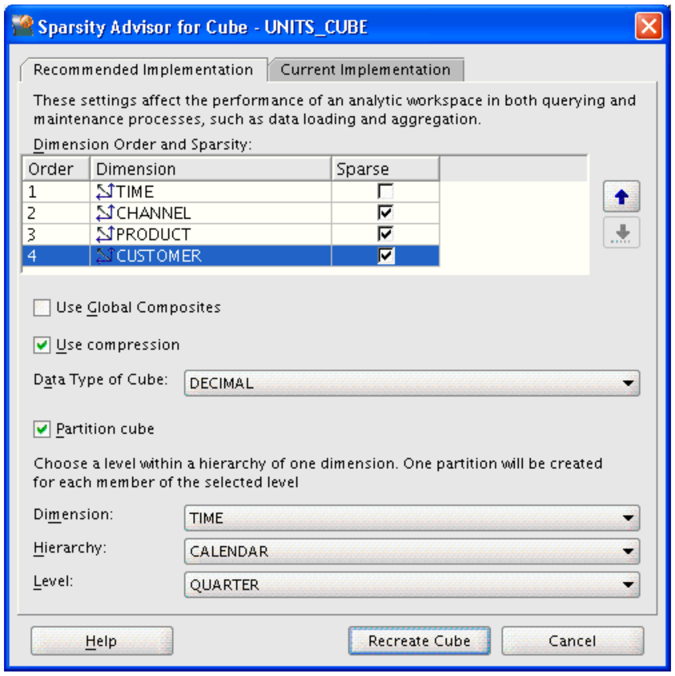
To run the Sparsity Advisor:

1. Create a cube and map it to a relational data source.
2. In the navigation tree, right-click the cube and choose **Sparsity Advisor**.

   Wait while the Sparsity Advisor analyzes the cube. When it is done, the Sparsity Advisor for Cube dialog box displays the recommendations.
3. For compressed cubes, be sure to select a data type for the cube.
4. Look over the recommendations and make any changes.
5. Click **Recreate Cube** to implement the recommendations.

Figure 3–13 shows the Sparsity Advisor dialog box with settings that are typical for a real-world cube.

Figure 3–13 Storage Strategy for a Cube


Sparsity Advisor for Cube - UNITS_CUBE

Recommended Implementation | Current Implementation

These settings affect the performance of an analytic workspace in both querying and maintenance processes, such as data loading and aggregation.

Dimension Order and Sparsity:

Order	Dimension	Sparse
1	TIME	☐
2	CHANNEL	☑
3	PRODUCT	☑
4	CUSTOMER	☑

☐ Use Global Composites

☑ Use compression

Data Type of Cube: DECIMAL

☑ Partition cube

Choose a level within a hierarchy of one dimension. One partition will be created for each member of the selected level

Dimension: TIME

Hierarchy: CALENDAR

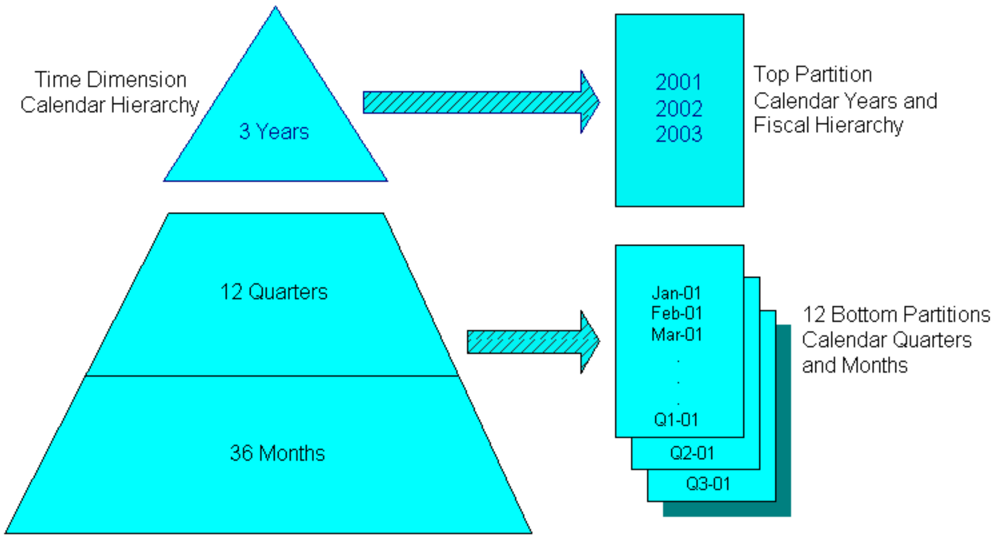
Level: QUARTER

Example of a Partitioned Dimension

The Partitioning Advisor might recommend partitioning at the Quarter level of the Calendar hierarchy of the Time dimension. Each Quarter and its descendants are stored in a separate partition. If there are three years of data in the analytic workspace, then partitioning on Quarter produces 12 bottom partitions, in addition to the default top partition. The top partition contains all remaining levels, that is, those above Quarter (such as Year) and those in other hierarchies (such as Fiscal Year or Year-to-Date).

Figure 3–14 illustrates a Time dimension partitioned by Quarter.

Figure 3–14 Partitioning Time by Quarter



Loading Data Into a Cube

You load data into cubes using the same methods as dimensions. However, loading and aggregating the data for your business measures typically takes more time to complete. Unless you are developing a dimensional model using a small sample of data, you may prefer to run the build in one or more background processes.

1. In the navigation tree, right-click the Cubes folder or the name of a particular cube.
2. Choose **Maintain Cube**.

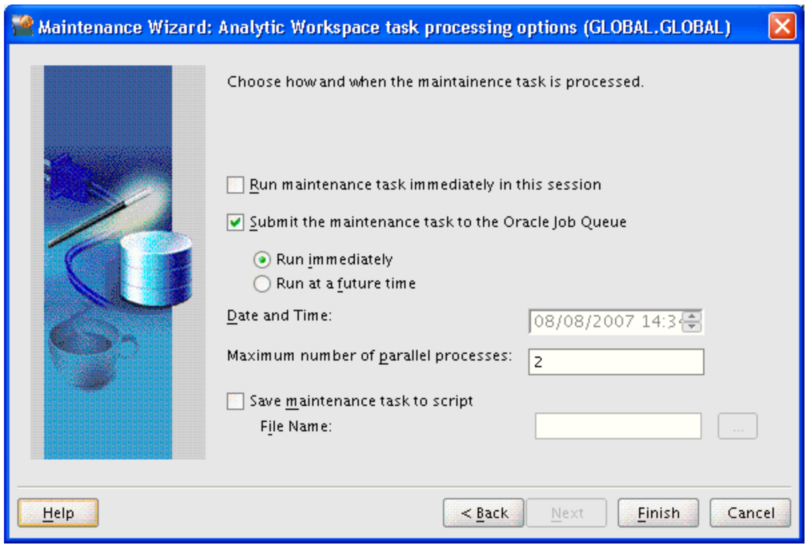
   The Maintenance Wizard opens on the Select Objects page.
3. Select one or more cubes from Available Target Objects and use the shuttle buttons to move them to Selected Target Objects. If the dimensions are already loaded, you can omit them from Selected Target Objects.
4. On the Dimension Data Processing Options page, you can choose to do a complete or a partial refresh of the cube.
5. On the Task Processing Options page, you can submit the build to the Oracle job queue or create a SQL script that you can run outside of Analytic Workspace Manager.

   You can also select the number of processes to dedicate to this build. The number of parallel processes is limited by the smallest of these numbers: The number of partitions in the cube, the number of processes dedicated to the build, and the setting of the `JOB_QUEUE_PROCESSES` initialization parameter.

   Click **Help** for information about these choices.
6. Click **Finish**.

Figure 3–15 shows the build submitted immediately to the Oracle job queue.

Figure 3–15 Selecting the Task Processing Options


Maintenance Wizard: Analytic Workspace task processing options (GLOBAL.GLOBAL)

Choose how and when the maintainence task is processed.

☐ Run maintenance task immediately in this session

☑ Submit the maintenance task to the Oracle Job Queue

● Run immediately

○ Run at a future time

Date and Time: 08/08/2007 14:3

Maximum number of parallel processes: 2

☐ Save maintenance task to script

File Name:

Displaying the Data in a Cube

After loading a cube, you can display the data for your business measures in Analytic Workspace Manager.

To display the data in a cube:

1. In the navigation tree, right-click the cube.
2. Choose **View Data** from the shortcut menu.

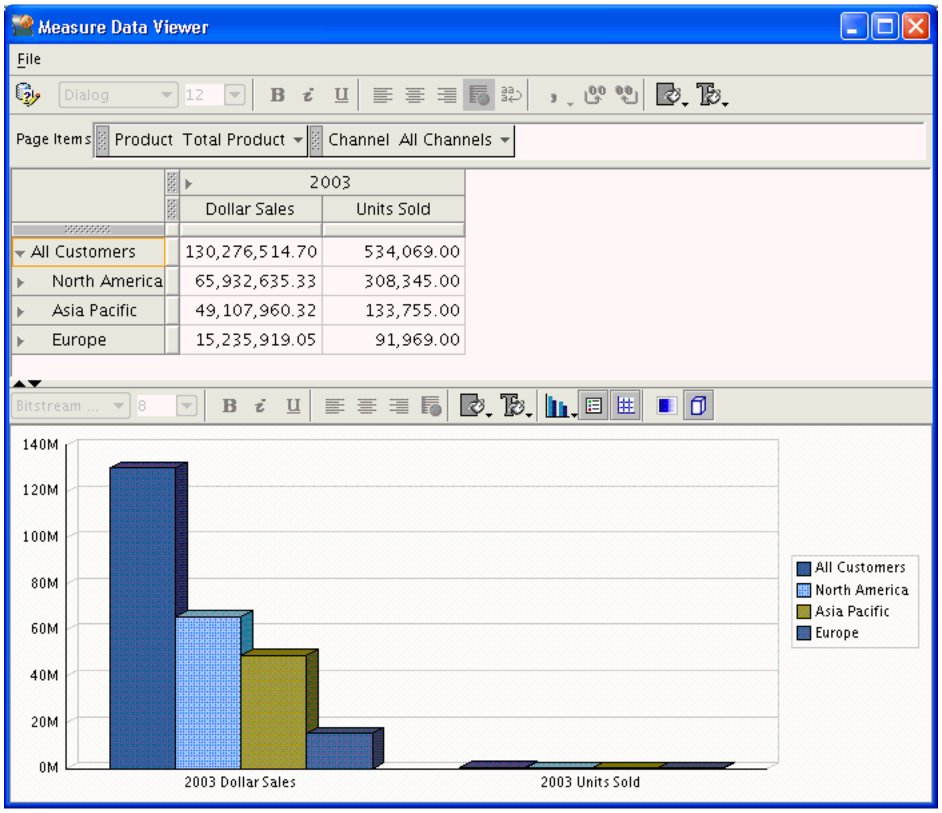
The Measure Data Viewer displays the selected measure in a crosstab at the top of the page and a graph at the bottom of the page. On the crosstab, you can expand and collapse the dimension hierarchies that label the rows and columns. You can also change the location of a dimension by pivoting or swapping it. If you wish, you can use multiple dimensions to label the columns and rows, by nesting one dimension under another.

- To pivot, drag a dimension from one location and drop it at another location, usually above or below another dimension.
- To swap dimensions, drag and drop one dimension directly over another dimension, so they exchange locations.

To make extensive changes to the selection of data, choose **Query Builder** from the File menu.

Figure 3–16 shows the Units cube in the Measure Viewer.

Figure 3–16 Displaying the Units Cube



	2003	
	Dollar Sales	Units Sold
All Customers	130,276,514.70	534,069.00
North America	65,932,635.33	308,345.00
Asia Pacific	49,107,960.32	133,755.00
Europe	15,235,919.05	91,969.00

Defining Measure Folders

You can define a measure folder for use by OLAP tools, so that the measures can be located and identified quickly by users. They may have access to several analytic workspaces or relational schemas with measures named Sales or Costs, and they have no means of differentiating them outside of a measure folder. The Cube Viewer in Analytic Workspace Manager also uses measure folders.

To create a measure folder:

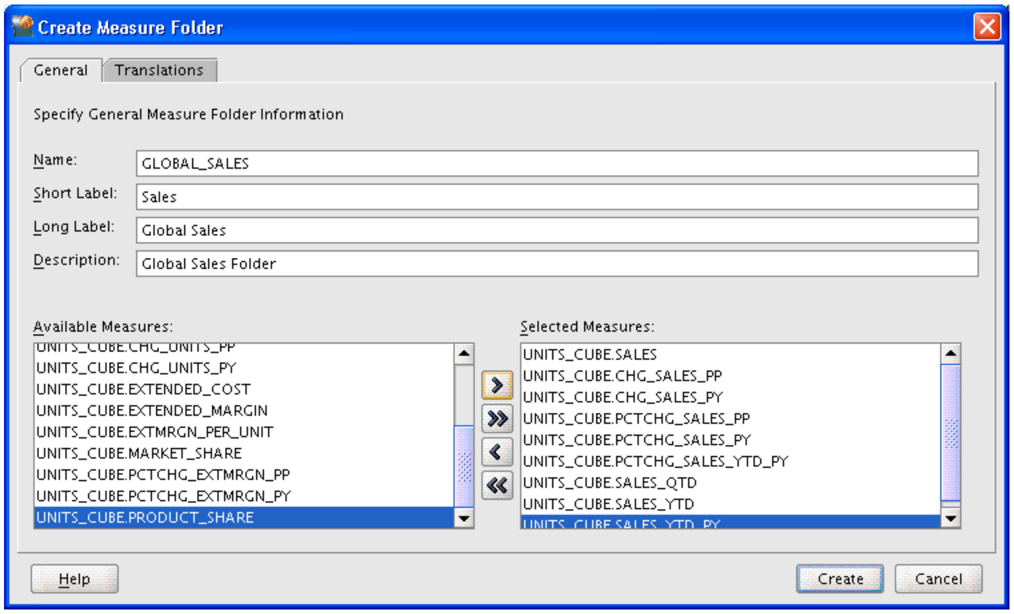
1. Expand the folder for the analytic workspace.
2. Right-click Measure Folders, then choose **Create Measure Folder**.
3. Complete the General tab of the Create Measure Folder dialog box.

   Click **Help** for specific information about these choices.

The new measure folder appears in the navigation tree under Measure Folders. You can also create subfolders.

Figure 3–17 shows creation of a measure folder.

Figure 3–17 Creating a Measure Folder



Specify General Measure Folder Information

Name: GLOBAL_SALES
Short Label: Sales
Long Label: Global Sales
Description: Global Sales Folder

Available Measures:
UNITS_CUBE.CHG_UNITS_PP
UNITS_CUBE.CHG_UNITS_PY
UNITS_CUBE.EXTENDED_COST
UNITS_CUBE.EXTENDED_MARGIN
UNITS_CUBE.EXTMRGN_PER_UNIT
UNITS_CUBE.MARKET_SHARE
UNITS_CUBE.PCTCHG_EXTMRGN_PP
UNITS_CUBE.PCTCHG_EXTMRGN_PY
UNITS_CUBE.PRODUCT_SHARE

Selected Measures:
UNITS_CUBE.SALES
UNITS_CUBE.CHG_SALES_PP
UNITS_CUBE.CHG_SALES_PY
UNITS_CUBE.PCTCHG_SALES_PP
UNITS_CUBE.PCTCHG_SALES_PY
UNITS_CUBE.PCTCHG_SALES_YTD_PY
UNITS_CUBE.SALES_QTD
UNITS_CUBE.SALES_YTD
UNITS_CUBE.SALES_YTD_PY

Supporting Multiple Languages

A single analytic workspace can support multiple languages. This support enables users of OLAP applications and tools to view the metadata in their native languages. For example, you can provide translations for the display names of measures, cubes, and dimensions. You can also map attributes to multiple columns, one for each language.

The number and choice of languages is restricted only by the database character set and your ability to provide translated text. Languages can be added or removed at any time.

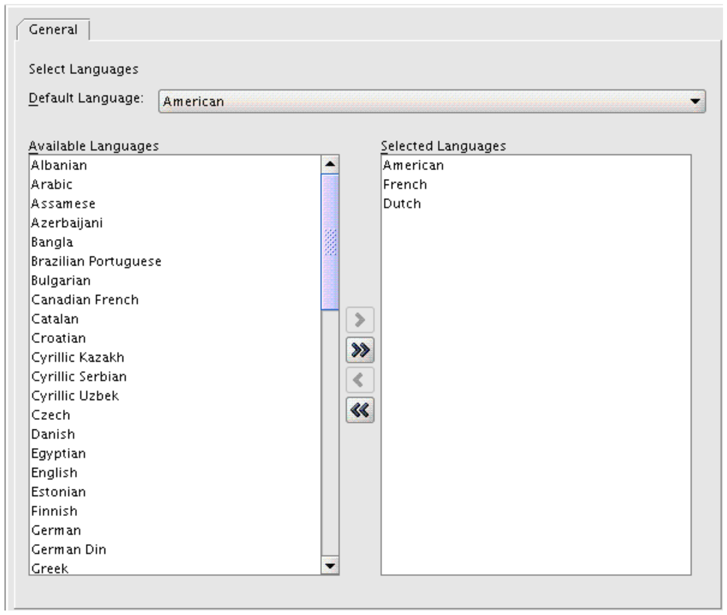
To add support for multiple languages:

1. In the navigation tree, expand the folder for the analytic workspace.
2. Click **Languages** and select the languages for the analytic workspace on the Basic tab.

3. For each dimension, level, hierarchy, attribute, cube, measure, calculated measure, and measure folder, select the Translations tab of the property sheet. Enter the object labels and descriptions in each language.
4. For each dimension, open the Mappings window. Map the attributes to columns for each language.

Figure 3–18 shows the selection of languages.

Figure 3–18 Adding Language Support



Default Language: American

Available Languages	Selected Languages
Albanian	American
Arabic	French
Assamese	Dutch
Azerbaijani	
Bangla	
Brazilian Portuguese	
Bulgarian	
Canadian French	
Catalan	
Croatian	
Cyrillic Kazakh	
Cyrillic Serbian	
Cyrillic Uzbek	
Czech	
Danish	
Egyptian	
English	
Estonian	
Finnish	
German	
German Din	
Greek	

Using Templates to Re-Create Dimensional Objects

Analytic Workspace Manager enables you to save all or part of the multidimensional model as a text file. This text file contains the XML definitions of the dimensional objects, such as dimensions, levels, hierarchies, attributes, and measures. Only the metadata is saved, not the data or any customizations. Templates are small files, so you can easily distribute them by e-mail or on a Web site, just as the templates for Global and Sales History are distributed on the Oracle Web site. To re-create the logical objects, you simply identify the templates in Analytic Workspace Manager.

You can save the following types of objects as XML templates:

- **Analytic workspace**: Saves all logical objects. You can save measure folders and calculation plans only by saving the complete analytic workspace.
- **Cube**: Saves the cube and its measures, calculated measures, and mappings.
- **Calculated measure**: Saves just the calculated measure.
- **Dimension**: Saves the dimension and its levels, hierarchies, attributes, and mappings.

To create a template:

In the navigation tree, right-click the object and choose **Save *object* to Template**.

To create logical objects from a template:

In the navigation tree, right-click the object type and choose **Create *object* From Template**.

4

Querying Dimensional Objects Using SQL

You can query the rich data stored in dimensional objects using SQL. This chapter explains the basics of querying relational views of cubes and dimensions.

This chapter includes the following topics:

See Also:

- Chapter 5, "Querying Dimensional Objects Using OLAP Tools"
- Chapter 10, "Developing Reports and Dashboards"

Querying Dimensional Data in SQL

Oracle OLAP adds power to your SQL applications by providing extensive analytic content and fast query response times. A SQL query interface enables any application to query cubes and dimensions without any knowledge of OLAP.

You can generate relational views of cubes and dimensions. SQL applications query these views to display the information-rich contents of dimensional objects to analysts and decision makers.

The SQL `OLAP_TABLE` function provides the basic technology for querying OLAP dimensional objects in SQL. You can use it for querying the objects directly or for creating views that can be queried with standard SQL `SELECT` statements. There are also some example programs for generating views available on the Oracle Technology Network.

See Also:

- *Oracle OLAP Reference* for more information about `OLAP_TABLE`
- Oracle Technology Network for a sample view generator at

  `http://www.oracle.com/technetwork/database/options/olap/i`
  `ndex.html`

Exploring the Shape of OLAP Views

You can create views of OLAP cubes and dimensions. For querying the data, you only need cube views. Cube views are denormalized views of all the measures, dimensions, levels, and attributes. They contain all of the data found in the tables of a star schema.

If you are developing an application, you can use dimension views as a convenient way to populate choice lists. Dimension views are equivalent to the dimension tables of a star schema.

Cube Views

Like a fact table, a cube view contains a column for each measure, calculated measure, and dimension of the cube. A cube view also contains columns for the parents, levels, and attributes of all the dimensions, so that the view is fully denormalized. These are the types of columns that should be included in a cube view:

- **Dimensions**: The dimension columns contain all the dimension keys at all levels of the dimension. Example 4–1 describes the columns of a view of the Global Units Cube. There are columns for TIME, CUSTOMER, PRODUCT, and CHANNEL. The TIME column contains dimension keys for months, quarters, and years.
- **Parents**: The parent columns define the parent-child relationships in a particular hierarchy. Example 4–1 shows a column name TIME_CALENDAR_YEA_PRNT that identifies the parent in the Calendar Year hierarchy of the dimension key in the TIME column. In this hierarchy, every month has a quarter for a parent, and every quarter has a year.
- **Hierarchy**: The hierarchy columns provide the ancestor at each level of a particular dimension, using the description instead of the dimension key. Example 4–1 shows hierarchy columns named TIME_YEAR_LVLDSC, TIME_QUARTER_LVLDSC, and TIME_MONTH_LVLDSC. A TIME value at the quarter level has values at the year and quarter levels, but not at the month level.
- **Levels**: The level columns identify the level of the dimension key. The TIME_LEVEL column in Example 4–1 has values of MONTH, QUARTER, and YEAR.
- **Attributes**: The attribute columns contain the attribute values for the dimensions. Example 4–1 has attribute columns named TIME_END_DATE, TIME_TIME_SPAN, TIME_LDSC, TIME_SDSC, TIME_QUARTER_OF_YEAR, TIME_MONTH_OF_QUARTER, and TIME_MONTH_OF_YEAR.
- **Measures**: The measure columns contain the facts for all combinations of dimension keys at all levels of the hierarchy. Thus, a cube view returns data at all levels of aggregation, from the detail level to the topmost level of consolidation. Example 4–1 has columns for the UNITS and SALES measures.
- **Calculated Measures**: These columns contain the calculated measures that have been defined for the cube. Like the measure columns, they contain business facts at all levels of aggregation. Example 4–1 shows that the Units Cube has calculated measures named SALES_PCT_CHG_PP, SALES_CHG_PP, PROD_SALES_SHARE_WITHIN_TOTAL, PROD_SALES_SHARE_WITHIN_PARENT, AND THREE_PERIOD_MOVING_AVG_SALES.

The DSO and OLAP_CALC columns shown in Example 4–1 are not queried by SQL.

Example 4–1 Cube View of the Global Units Cube

```
SQL> DESCRIBE units_cube_cubeview
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
```

```
TIME                                              VARCHAR2(4000)
CUSTOMER                                          VARCHAR2(4000)
PRODUCT                                           VARCHAR2(4000)
CHANNEL                                           VARCHAR2(4000)
TIME_CALENDAR_YEA_PRNT                            VARCHAR2(4000)
TIME_YEAR_LVLDSC                                  VARCHAR2(4000)
TIME_QUARTER_LVLDSC                               VARCHAR2(4000)
TIME_MONTH_LVLDSC                                 VARCHAR2(4000)
TIME_END_DATE                                     DATE
TIME_TIME_SPAN                                    NUMBER
TIME_LDSC                                         VARCHAR2(4000)
TIME_SDSC                                         VARCHAR2(4000)
TIME_QUARTER_OF_YEAR                              VARCHAR2(4000)
TIME_MONTH_OF_QUARTER                             VARCHAR2(4000)
TIME_MONTH_OF_YEAR                                VARCHAR2(4000)
TIME_TIME_DSO_1                                   NUMBER
TIME_LEVEL                                        VARCHAR2(4000)
         .
         .
         .
UNITS                                             NUMBER
SALES                                             NUMBER
SALES_PCT_CHG_PP                                  NUMBER
SALES_CHG_PP                                      NUMBER
PROD_SALES_SHARE_WITHIN_TOTAL                     NUMBER
PROD_SALES_SHARE_WITHIN_PARENT                    NUMBER
THREE_PERIOD_MOVING_AVG_SALES                     NUMBER
OLAP_CALC                                         RAW(16)
```

You can display the facts in a cube view quickly with a query like the one shown in Example 4–2.

Example 4–2 Querying the Facts in a Cube View

```
SQL> SELECT time, customer, product, channel, units, sales
     FROM units_cube_cubeview WHERE rownum < 15;

TIME             CUSTOMER         PRODUCT          CHANNEL               UNITS      SALES
---------------- ---------------- ---------------- ---------------- ---------- ----------
YEAR_145         TOTAL_CUSTOMER_1 TOTAL_PRODUCT_1  TOTAL_CHANNEL_1
YEAR_4           TOTAL_CUSTOMER_1 TOTAL_PRODUCT_1  TOTAL_CHANNEL_1      415392  116931479
YEAR_2           TOTAL_CUSTOMER_1 TOTAL_PRODUCT_1  TOTAL_CHANNEL_1      330425  134109248
YEAR_3           TOTAL_CUSTOMER_1 TOTAL_PRODUCT_1  TOTAL_CHANNEL_1      364233  124173522
YEAR_1           TOTAL_CUSTOMER_1 TOTAL_PRODUCT_1  TOTAL_CHANNEL_1      253816  100870877
YEAR_85          TOTAL_CUSTOMER_1 TOTAL_PRODUCT_1  TOTAL_CHANNEL_1      364965   92515295
YEAR_119         TOTAL_CUSTOMER_1 TOTAL_PRODUCT_1  TOTAL_CHANNEL_1      339831 80846147.8
YEAR_102         TOTAL_CUSTOMER_1 TOTAL_PRODUCT_1  TOTAL_CHANNEL_1      534069  130276515
QUARTER_12       TOTAL_CUSTOMER_1 TOTAL_PRODUCT_1  TOTAL_CHANNEL_1       87521 33761936.8
QUARTER_13       TOTAL_CUSTOMER_1 TOTAL_PRODUCT_1  TOTAL_CHANNEL_1       88484 31522409.5
QUARTER_81       TOTAL_CUSTOMER_1 TOTAL_PRODUCT_1  TOTAL_CHANNEL_1       84100 21499269.6
QUARTER_141      TOTAL_CUSTOMER_1 TOTAL_PRODUCT_1  TOTAL_CHANNEL_1
QUARTER_143      TOTAL_CUSTOMER_1 TOTAL_PRODUCT_1  TOTAL_CHANNEL_1
QUARTER_6        TOTAL_CUSTOMER_1 TOTAL_PRODUCT_1  TOTAL_CHANNEL_1       61320 24993273.3

14 rows selected.
```

A query like the one in Example 4–3 displays the level, ancestry, and attributes of each dimension key.

Example 4–3 Querying a Dimension in a Cube View

```
SQL> SELECT time, time_level, time_calendar_yea_prnt parent, time_year_lvldsc year,
        time_quarter_lvldsc quarter, time_month_lvldsc month, time_ldsc description,
        time_end_date end_date, time_time_span time_span
     FROM units_cube_cubeview WHERE rownum < 15;

TIME         TIME_LEVEL PARENT     YEAR     QUARTER  MONTH    DESCRIPTION  END_DATE   TIME_SPAN
------------ ---------- ---------- -------- -------- -------- ------------ --------- ----------
YEAR_145     YEAR                  2005                       2005         31-DEC-05        365
YEAR_4       YEAR                  2001                       2001         31-DEC-01        365
YEAR_2       YEAR                  1999                       1999         31-DEC-99        365
YEAR_3       YEAR                  2000                       2000         31-DEC-00        366
YEAR_1       YEAR                  1998                       1998         31-DEC-98        365
YEAR_85      YEAR                  2002                       2002         31-DEC-02        365
YEAR_119     YEAR                  2004                       2004         31-DEC-04        366
YEAR_102     YEAR                  2003                       2003         31-DEC-03        365
QUARTER_12   QUARTER    YEAR_2     1999     Q4-99             Q4-99        31-DEC-99         92
QUARTER_13   QUARTER    YEAR_3     2000     Q1-00             Q1-00        31-MAR-00         91
QUARTER_81   QUARTER    YEAR_85    2002     Q1-02             Q1-02        31-MAR-02         90
QUARTER_141  QUARTER    YEAR_145   2005     Q1-05             Q1-05        31-MAR-05         90
QUARTER_143  QUARTER    YEAR_145   2005     Q3-05             Q3-05        30-SEP-05         92
QUARTER_6    QUARTER    YEAR_1     1998     Q2-98             Q2-98        30-JUN-98         91

14 rows selected.
```

Dimension Views

A dimension view contains all the information typically found in a dimension table of a star schema. The view contains a column for the dimension keys, parents, levels, hierarchies, and attributes. All of these columns are also found in the cube views, so there is no need to join a cube view to the dimension views. However, you may want to create dimension views to support user interface components such as choice lists.

Example 4–4 shows the columns of a dimension view for the Global Time dimension. For descriptions of these columns, refer to "Cube Views" on page 4-2.

Example 4–4 Dimension View for the Global Time Dimension

```
SQL> DESCRIBE time_dimview

 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 TIME                                               VARCHAR2(4000)
 TIME_LEVEL                                         VARCHAR2(4000)
 TIME_TIME_DSO_1                                    NUMBER
 TIME_MONTH_OF_YEAR                                 VARCHAR2(4000)
 TIME_MONTH_OF_QUARTER                              VARCHAR2(4000)
 TIME_QUARTER_OF_YEAR                               VARCHAR2(4000)
 TIME_SDSC                                          VARCHAR2(4000)
 TIME_LDSC                                          VARCHAR2(4000)
 TIME_TIME_SPAN                                     NUMBER
 TIME_END_DATE                                      DATE
 TIME_MONTH_LVLDSC                                  VARCHAR2(4000)
 TIME_QUARTER_LVLDSC                                VARCHAR2(4000)
 TIME_YEAR_LVLDSC                                   VARCHAR2(4000)
 TIME_CALENDAR_YEA_PRNT                             VARCHAR2(4000)
```

You can display the hierarchical information provided in a dimension view with a query like the one shown in Example 4–5.

Example 4–5 Querying the Global Time Calendar Hierarchy

```
SQL> SELECT time, time_level, time_calendar_yea_prnt parent,
        time_year_lvldsc year,time_quarter_lvldsc quarter, time_month_lvldsc month
     FROM time_dimview WHERE rownum < 15;

TIME         TIME_LEVEL PARENT     YEAR     QUARTER  MONTH
------------ ---------- ---------- -------- -------- --------
YEAR_145     YEAR                  2005
YEAR_4       YEAR                  2001
YEAR_2       YEAR                  1999
YEAR_3       YEAR                  2000
YEAR_1       YEAR                  1998
YEAR_85      YEAR                  2002
YEAR_119     YEAR                  2004
YEAR_102     YEAR                  2003
QUARTER_12   QUARTER    YEAR_2     1999     Q4-99
QUARTER_13   QUARTER    YEAR_3     2000     Q1-00
QUARTER_81   QUARTER    YEAR_85    2002     Q1-02
QUARTER_141  QUARTER    YEAR_145   2005     Q1-05
QUARTER_143  QUARTER    YEAR_145   2005     Q3-05
QUARTER_6    QUARTER    YEAR_1     1998     Q2-98

14 rows selected.
```

A query like the one shown in Example 4–6 displays the attributes of the dimension.

Example 4–6 Querying the Global Time Attributes

```
SQL> SELECT time, time_ldsc description, time_month_of_year mo_of_yr,
        time_month_of_quarter mo_of_qtr, time_end_date end_date,
        time_time_span time_span
     FROM time_dimview
     WHERE time_month_of_year IS NOT NULL AND rownum < 15
     ORDER BY end_date;

TIME         DESCRIPTION  MO_OF_YR   MO_OF_QTR  END_DATE   TIME_SPAN
------------ ------------ ---------- ---------- --------- ----------
MONTH_29     Nov-98       11         2          30-NOV-98         30
MONTH_35     May-99       5          2          31-MAY-99         31
MONTH_36     Jun-99       6          3          30-JUN-99         30
MONTH_40     Oct-99       10         1          31-OCT-99         31
MONTH_41     Nov-99       11         2          30-NOV-99         30
MONTH_52     Oct-00       10         1          31-OCT-00         31
MONTH_55     Jan-01       1          1          31-JAN-01         31
MONTH_66     Dec-01       12         3          31-DEC-01         31
MONTH_75     Jul-02       7          1          31-JUL-02         31
MONTH_120    Jun-04       6          3          30-JUN-04         30
MONTH_123    Sep-04       9          3          30-SEP-04         30
MONTH_126    Dec-04       12         3          31-DEC-04         31
MONTH_127    Jan-05       1          1          31-JAN-05         31
MONTH_135    Sep-05       9          3          30-SEP-05         30

14 rows selected.
```

Creating Basic Queries

When querying a cube, remember these guidelines:

- Apply a filter to every dimension.

The cube contains both detail level and aggregated data. A query with an unfiltered dimension typically returns more data than users need, which negatively impacts performance.

- Let the cube aggregate the data.

  Because the aggregations are already calculated in the cube, a typical query does not need a `GROUP BY` clause. Simply select the aggregations you want by using the appropriate filters on the dimension keys or attributes.

Applying a Filter to Every Dimension

To create a level filter, you must know the names of the dimension levels. You can easily acquire them by querying the cube or dimension views:

```
SQL> SELECT DISTINCT time_level FROM units_cube_cubeview;

TIME_LEVEL
----------
QUARTER
MONTH
YEAR
```

To see the importance of applying a filter to every dimension, consider the query in Example 4–7, which has no filter on the time dimension.

Example 4–7 Displaying Aggregates at All Levels of Time

```
/* Select key descriptions and facts */
SELECT time_ldsc time,
     ROUND(sales) sales
/* From cube view */
   FROM units_cube_cubeview
/* No filter on Time */
   WHERE product_level = 'TOTAL_PRODUCT'
      AND customer_level = 'TOTAL_CUSTOMER'
      AND channel_level = 'TOTAL_CHANNEL'
   ORDER BY time_end_date;
```

Without a filter on the Time dimension, the query returns values for every level of time. This is more data than users typically want to see, and the volume of data returned can degrade performance.

```
TIME              SALES
------------ ----------
Jan-98          8338545
Feb-98          7972132
Q1-98          24538588
Mar-98          8227911
Apr-98          8470315
May-98          8160573
Q2-98          24993273
Jun-98          8362386
Jul-98          8296226
Aug-98          8377587
Sep-98          8406728
Q3-98          25080541
Oct-98          8316169
Nov-98          8984156
Q4-98          26258474
1998          100870877
```

```
          .
          .
          .
```

Now consider the results when a filter restricts Time to yearly data.

Example 4–8 shows a basic query. It selects the long description attributes of Time and the Sales measure from `UNITS_CUBE_VIEW`, and joins the keys from the cube view to the hierarchy views to get descriptions of the keys.

Example 4–8 Basic Cube View Query

```
/* Select key descriptions and facts */
SELECT time_ldsc time,
     ROUND(sales) sales
/* From cube view */
   FROM units_cube_cubeview
/* Filters on all dimensions */
   WHERE time_level = 'YEAR'
     AND product_level = 'TOTAL_PRODUCT'
     AND customer_level = 'TOTAL_CUSTOMER'
     AND channel_level = 'TOTAL_CHANNEL'
   ORDER BY time_end_date;
```

Example 4–8 selects the following rows. For `CUSTOMER`, `PRODUCT`, and `CHANNEL`, only one value is at the top level. `TIME` has a value for each calendar year.

```
TIME              SALES
------------ ----------
1998          100870877
1999          134109248
2000          124173522
2001          116931479
2002           92515295
2003          130276515
2004           80846148
2005

8 rows selected.
```

Dimension attributes also provide a useful way to select the data for a query. The `WHERE` clause in Example 4–9 uses attributes values to filter all of the dimensions.

Example 4–9 Selecting Data With Attribute Filters

```
/* Select key descriptions and facts */
SELECT time_ldsc time,
     product_ldsc product,
     customer_ldsc customer,
     ROUND(sales) sales
/* From dimension views and cube view */
FROM units_cube_cubeview
/* Create attribute filters */
WHERE time_ldsc IN ('2001', '2002')
    AND product_package = 'Laptop Value Pack'
    AND customer_ldsc LIKE '%Boston%'
    AND channel_ldsc = 'Internet'
ORDER BY time, customer;
```

The query selects two calendar years, the products in the Laptop Value Pack, the customers in Boston, and the Internet channel.

```
TIME       PRODUCT                        CUSTOMER                  SALES
---------- ------------------------------ --------------------- ----------
2001       Laptop carrying case           KOSH Entrpr Boston          4995
2001       56Kbps V.92 Type II Fax/Modem  KOSH Entrpr Boston          9683
2001       Internal 48X CD-ROM            KOSH Entrpr Boston          2122
2001       Envoy Standard                 KOSH Entrpr Boston         24335
2001       Standard Mouse                 KOSH Entrpr Boston           419
2001       Laptop carrying case           Warren Systems Boston        747
2001       Standard Mouse                 Warren Systems Boston        107
2001       56Kbps V.92 Type II Fax/Modem  Warren Systems Boston       1743
2001       Envoy Standard                 Warren Systems Boston      14438
2001       Internal 48X CD-ROM            Warren Systems Boston        129
2002       Internal 48X CD-ROM            KOSH Entrpr Boston          2161
2002       56Kbps V.92 Type II Fax/Modem  KOSH Entrpr Boston         17573
2002       Envoy Standard                 KOSH Entrpr Boston
2002       Standard Mouse                 KOSH Entrpr Boston           487
2002       Laptop carrying case           KOSH Entrpr Boston          5584
2002       Laptop carrying case           Warren Systems Boston       3357
2002       Envoy Standard                 Warren Systems Boston      24511
2002       56Kbps V.92 Type II Fax/Modem  Warren Systems Boston       1249
2002       Standard Mouse                 Warren Systems Boston        142
2002       Internal 48X CD-ROM            Warren Systems Boston

20 rows selected.
```

Allowing the Cube to Aggregate the Data

A cube contains all of the aggregate data. As shown in this chapter, a query against a cube just needs to select the aggregate data, not calculate the values.

The following is a basic query against a fact table:

```
/* Querying a fact table */
SELECT t.year_dsc time,
     SUM(sales) sales
  FROM time_dim t, units_history_fact f
  WHERE t.year_dsc IN ('2001', '2002')
     AND t.month_id = f.month_id
  GROUP BY t.year_dsc;
```

The next query fetches the exact same results from a cube using filters:

```
/* Querying a cube */
SELECT time_ldsc time, sales
  FROM units_cube_cubeview
/* Apply filters to every dimension */
  WHERE time_ldsc IN ('2001', '2002')
     AND product_level = 'TOTAL_PRODUCT'
     AND customer_level = 'TOTAL_CUSTOMER'
     AND channel_level = 'TOTAL_CHANNEL'
  ORDER BY time;
```

Both queries return these results:

```
TIME            SALES
---------- ----------
2001        116931479
2002         92515295
```

The query against the cube does not compute the aggregate values with a `SUM` operator and `GROUP BY` clause. Because the aggregates exist already in the cube, this would re-aggregate previously aggregated data. Instead, the query selects the aggregates directly from the cube and specifies the desired aggregates by applying the appropriate filter to each dimension.

Query Processing

The most efficient queries allow the OLAP engine to filter the data, so that the minimum number of rows required by the query are returned to SQL.

The following are among the `WHERE` clause operations that are pushed into the OLAP engine for processing:

- `=`
- `!=`
- `>`
- `!>`
- `<`
- `!<`
- `IN`
- `NOT IN`
- `IS NULL`
- `LIKE`
- `NOT LIKE`

The OLAP engine also processes nested character functions, including `INSTR`, `LENGTH`, `NVL`, `LOWER`, `UPPER`, `LTRIM`, `RTRIM`, `TRIM`, `LPAD`, `RPAD`, and `SUBSTR`.

SQL processes other operations and functions in the `WHERE` clause, and all operations in other parts of the `SELECT` syntax.

Creating Hierarchical Queries

Drilling is an important capability in business analysis. In a dashboard or an application, users click a dimension key to change the selection of data. Decision makers frequently want to drill down to see the contributors to a data value, or drill up to see how a particular data value contributes to the whole. For example, the Boston regional sales manager might start at total Boston sales, drill down to see the contributions of each sales representative, then drill up to see how the Boston region contributes to the New England sales total.

The views include a parent column that identifies the parent of every dimension key. This column encapsulates all of the hierarchical information of the dimension. If you know the parent of every key, then you can derive the ancestors, the children, and the descendants.

For level-based hierarchies, the level column supplements this information by providing a convenient way to identify all the keys at the same depth in the hierarchy, from the top to the base. For value-based hierarchies, the parent column provides all the information about the hierarchy.

See Also: Chapter 10, "Developing Reports and Dashboards," about using bind variables to support drilling

Drilling Down to Children

You can use the parent column of a view to select only the children of a particular value. The following WHERE clause selects the children of calendar year 2001.

```
/* Select children of calendar year 2001 */
WHERE time_calendar_yea_prnt = 'YEAR_4'
     AND product = 'TOTAL_PRODUCT_1'
     AND customer = 'TOTAL_CUSTOMER_1'
     AND channel = 'TOTAL_CHANNEL_1'
```

The query drills down from Year to Quarter. The four quarters Q1-05 to Q4-05 are the children of year CY2005 in the Calendar hierarchy.

```
TIME            SALES
---------- ----------
Q1-01        27595330
Q2-01        27798427
Q3-01        29691668
Q4-01        3184605
```

Drilling Up to Parents

The parent column of a hierarchy view identifies the parent of each dimension key. The following WHERE clause selects the parent of a Time key based on its long description attribute.

```
WHERE time =
         (SELECT DISTINCT time_calendar_yea_prnt
          FROM units_cube_cubeview
          WHERE time_ldsc='Sep-01')
     AND product  = 'TOTAL_PRODUCT_1'
     AND customer = 'TOTAL_CUSTOMER_1'
     AND channel  = 'TOTAL_CHANNEL_1'
```

The query drills up from Month to Quarter. The parent of month Sep-01 is the quarter Q3-01 in the Calendar Year hierarchy.

```
TIME            SALES
---------- ----------
Q3-01        29691668
```

Drilling Down to Descendants

The following WHERE clause selects the descendants of calendar year 2001 by selecting the rows with a Time level of MONTH and a year of 2001.

```
/* Select Time level and ancestor */
WHERE time_level = 'MONTH'
     AND time_year_lvldsc = '2001'
     AND product  = 'TOTAL_PRODUCT_1'
     AND customer = 'TOTAL_CUSTOMER_1'
     AND channel  = 'TOTAL_CHANNEL_1'
```

The query drills down two levels, from year to quarter to month. The 12 months Jan-01 to Dec-01 are the descendants of year 2001 in the Calendar Year hierarchy.

```
TIME            SALES
---------- ----------
Jan-01        9377798
Feb-01        9080969
Mar-01        9136563
```

```
Apr-01        9145284
May-01        9028805
Jun-01        9624338
Jul-01        9789531
Aug-01        9581753
Sep-01       10320384
Oct-01       10117410
Nov-01       10866341
Dec-01       10862303

12 rows selected.
```

Drilling Up to Ancestors

The hierarchy views provide the full ancestry of each dimension key. The following `WHERE` clause uses the year level key column to identify the ancestor of a `MONTH` dimension key.

```
/* Select the ancestor of a Time key based on its Long Description attribute */
WHERE time_ldsc =
         (SELECT distinct time_year_lvldsc
          FROM units_cube_cubeview
          WHERE time_ldsc = 'Sep-01')
     AND product  = 'TOTAL_PRODUCT_1'
     AND customer = 'TOTAL_CUSTOMER_1'
     AND channel  = 'TOTAL_CHANNEL_1'
```

The query drills up two levels from month to quarter to year. The ancestor of month `Sep-01` is the year `2001` in the Calendar hierarchy.

```
TIME            SALES
---------- ----------
2001        116931479
```

Using Calculations in Queries

A DBA can create calculated measures in Analytic Workspace Manager, so they are available to all applications. This not only simplifies application development, but ensures that all applications use the same name for the same calculation.

Nonetheless, you may want to develop queries that include your own calculations. In this case, you can use an inner query to select aggregate data from the cube, then perform calculations in an outer query. You can select data from cubes that use any type of aggregation operators, and you can use any functions or operators in the query. You only need to make sure that you select the data from the cube at the appropriate levels for the calculation, and that the combination of operators in the cube and in the query create the calculation you want.

Example 4–10 shows a query that answers the question, What was the average sales of Sentinel Standard computers to Government customers for the third quarter of calendar year 2001. `UNITS_CUBE` is summed over all dimensions, so that `QUARTER_67` (Q3-01) is a total for July, August, and September. The inner query extracts the data for these months, and the outer query uses the `MIN`, `MAX`, and `AVG` operators and a `GROUP BY` clause to calculate the averages.

Example 4–10 Calculating Average Sales Across Customers

```
SELECT customer, ROUND(MIN(sales)) minimum, ROUND(MAX(sales)) maximum,
   ROUND(AVG(sales)) average
```

```
FROM
    (SELECT customer_ldsc customer, time_ldsc time, sales
       FROM units_cube_cubeview
       WHERE time_calendar_yea_prnt = 'QUARTER_67'
       AND product_ldsc = 'Sentinel Standard'
       AND customer_market_segme_prnt = 'MARKET_SEGMENT_4'
       AND channel_level  = 'TOTAL_CHANNEL'
    )
GROUP BY customer
ORDER BY customer;
```

This is the data extracted from the cube by the inner query:

```
CUSTOMER                       TIME            SALES
------------------------------ ---------- ----------
Dept. of Communication         Aug-01        1752.06
Dept. of Communication         Jul-01         5344.2
Dept. of Communication         Sep-01        3507.06
Dept. of Labor                 Aug-01
Dept. of Labor                 Sep-01        1753.53
Dept. of Labor                 Jul-01         3562.8
Ministry of Finance            Jul-01         1781.4
Ministry of Finance            Aug-01        3504.12
Ministry of Finance            Sep-01        7014.12
Ministry of Intl Trade         Jul-01         5344.2
Ministry of Intl Trade         Sep-01        5260.59
Ministry of Intl Trade         Aug-01        5256.18
Royal Air Force                Jul-01         3562.8
Royal Air Force                Sep-01        3507.06
Royal Air Force                Aug-01         8760.3
UK Environmental Department    Aug-01
UK Environmental Department    Sep-01
UK Environmental Department    Jul-01         3562.8
US Dept. of Research           Jul-01         1781.4
US Dept. of Research           Aug-01        1752.06
US Dept. of Research           Sep-01        1753.53
US Marine Services             Sep-01
US Marine Services             Aug-01
US Marine Services             Jul-01
```

The outer query calculates the minimum, maximum, and average sales for each customer:

```
CUSTOMER                          MINIMUM    MAXIMUM    AVERAGE
------------------------------ ---------- ---------- ----------
Dept. of Communication               1752       5344       3534
Dept. of Labor                       1754       3563       2658
Ministry of Finance                  1781       7014       4100
Ministry of Intl Trade               5256       5344       5287
Royal Air Force                      3507       8760       5277
UK Environmental Department          3563       3563       3563
US Dept. of Research                 1752       1781       1762
US Marine Services
```

Using Attributes for Aggregation

An OLAP cube aggregates the data within its hierarchies, using the parent-child relationships revealed in the hierarchy views. The OLAP engine does not calculate aggregates over dimension attribute values.

Nonetheless, you may want to aggregate products over color or size, or customers by age, zip code, or population density. This is the situation when you can use a GROUP BY clause when querying a cube. Your query can extract data from the cube, then use SQL to aggregate by attribute value.

The cube must use the same aggregation operator for all dimensions, and the aggregation operator in the SELECT list of the query must match the aggregation operator of the cube. You can use a GROUP BY clause to query cubes that use these operators:

- First Non-NA Value
- Last Non-NA Value
- Maximum
- Minimum
- Sum

Aggregating Measures Over Attributes

Example 4–11 shows a query that aggregates over an attribute named Package. It returns these results:

```
TIME       PACKAGE                   SALES
---------- -------------------- ----------
2001       All                   2176753.8
2001       Executive            25793371.5
2001       Laptop Value Pack    16118203.4
2001       Multimedia           19887248.8
```

Units Cube uses the SUM operator for all dimensions, and the query uses the SUM operator to aggregate over Sales. The Package attribute applies only to the Item level of the Product dimension, so the query selects the Item level of Product. It also eliminates nulls for Package, so that only products that belong to a package are included in the calculation. The GROUP BY clause breaks out Total Sales by Time and Package

Example 4–11 Aggregating Over an Attribute

```
SELECT time_ldsc time,
     product_package package,
     SUM(sales) sales
/* From cube view */
   FROM units_cube_cubeview
/* Filters on all dimensions */
   WHERE time_ldsc = '2001'
     AND product_level = 'ITEM'
     AND product_package IS NOT NULL
     AND customer_level = 'TOTAL_CUSTOMER'
     AND channel_level = 'TOTAL_CHANNEL'
   GROUP BY time_ldsc, product_package
   ORDER BY product_package;
```

Aggregating Calculated Measures Over Attributes

Before using the technique described in "Aggregating Measures Over Attributes" on page 4-13, be sure that the calculation is meaningful. For example, the common calculation Percent Change might be defined as a calculated measure in a cube.

Summing over Percent Change would produce unexpected results, because the calculation for Percent Change (`(a-b)/b`,) is not additive.

Consider the following rows of data. The correct Total Percent Change is `.33`, whereas the sum of the percent change for the first two rows is `.75`.

Row	Sales	Sales Prior Period	Percent Change
1	15	10	.50
2	25	20	.25
Total	**40**	**30**	**.33**

Example 4–12 shows a query that aggregates over the Package attribute and calculates Percent Change From Prior Period. The inner query aggregates Sales and Sales Prior Period over the attributes, and the outer query uses the results to compute the percent change. These are the results of the query, which show the expected results for `PCT_CHG`:

```
TIME       PACKAGE                   SALES PRIOR_PERIOD    PCT_CHG
---------- -------------------- ---------- ------------ ----------
2001       All                   2176753.8   2048166.74        .06
2002       All                   1840963.8    2176753.8       -.15
2001       Executive            25793371.5   26391852.4       -.02
2002       Executive            18717348.1   25793371.5       -.27
2001       Laptop Value Pack    16118203.4     18884919       -.15
2002       Laptop Value Pack    11085266.8   16118203.4       -.31
2001       Multimedia           19887248.8   21262926.7       -.06
2002       Multimedia           16218667.2   19887248.8       -.18

8 rows selected.
```

Example 4–12 Querying Over Attributes Using Calculated Measures

```
/* Calculate Percent Change */
SELECT time, package, sales, prior_period,
      round((sales - prior_period) / prior_period, 2) pct_chg
   FROM
       (SELECT time_ldsc time, product_package package,
          sum(sales) sales, sum(sales_pp) prior_period
          FROM units_cube_cubeview
          WHERE product_level = 'ITEM'
             AND product_package IS NOT NULL
             AND time_ldsc IN ('2001', '2002')
             AND customer_level = 'TOTAL_CUSTOMER'
             AND channel_level  = 'TOTAL_CHANNEL'
        GROUP BY time_ldsc, product_package
        ORDER BY package);
```

Querying the Active Catalog

If you are developing a generic application -- that is, one where the names of the dimensional objects are not known -- then your application can retrieve this information from the Active Catalog.

The Active Catalog is a set of views that display metadata for dimensional objects. These views always reflect the current state of the analytic workspace. You can query the views using standard SQL.

Active Catalog views provide information about dimensional objects in all analytic workspaces accessible to the current user. The public synonyms for these views are named with the `ALL_OLAP2_AW` prefix.

Table 4–1 provides brief descriptions of the Active Catalog views.

Table 4–1 Active Catalog Views

PUBLIC Synonym	Description
`ALL_OLAP2_AW_ATTRIBUTES`	List of dimension attributes in analytic workspaces
`ALL_OLAP2_AW_CATALOG_MEASURES`	Lists the measures in the measure folders
`ALL_OLAP2_AW_CATALOGS`	Lists the measure folders in analytic workspaces
`ALL_OLAP2_AW_CUBE_AGG_LVL`	List of levels in aggregation plans in analytic workspaces
`ALL_OLAP2_AW_CUBE_AGG_MEAS`	List of measures in aggregation plans in analytic workspaces
`ALL_OLAP2_AW_CUBE_AGG_OP`	List of aggregation operators in aggregation plans in analytic workspaces
`ALL_OLAP2_AW_CUBE_AGG_SPECS`	List of aggregation plans in analytic workspaces
`ALL_OLAP2_AW_CUBE_DIM_USES`	List of cubes with their associated dimensions in analytic workspaces
`ALL_OLAP2_AW_CUBE_MEASURES`	List of cubes with their associated measures in analytic workspaces
`ALL_OLAP2_AW_CUBES`	List of cubes in analytic workspaces
`ALL_OLAP2_AW_DIM_HIER_LVL_ORD`	List of hierarchical levels in analytic workspaces
`ALL_OLAP2_AW_DIM_LEVELS`	List of levels in analytic workspaces
`ALL_OLAP2_AW_DIMENSIONS`	List of dimensions in analytic workspaces
`ALL_OLAP2_AW_OBJ_PROP`	List of properties associated with standard form objects in analytic workspaces
`ALL_OLAP2_AW_PHYS_OBJ`	List of standard form objects in analytic workspaces
`ALL_OLAP2_AWS`	Lists the analytic workspaces

See Also: *Oracle OLAP Reference* for more information about the Active Catalog

5 Querying Dimensional Objects Using OLAP Tools

OLAP tools are designed specifically to locate all cubes and dimensions that are accessible to the current user. They automatically use the implicit relationships among cubes, dimensions, hierarchies, levels, and attributes. For example, drilling is automatically supported, children are clearly identified under their parent values, and description attributes are used as labels instead of dimension keys.

This chapter introduces two OLAP querying tools, the Oracle Business Intelligence Spreadsheet Add-In and Discoverer Plus OLAP.

This chapter includes the following topics:

- Analyzing Dimensional Data in a Spreadsheet
- Creating Reports in Discoverer Plus OLAP

See Also:

- Chapter 4, "Querying Dimensional Objects Using SQL"
- Chapter 11, "Developing Java Applications for OLAP"

Analyzing Dimensional Data in a Spreadsheet

OracleBI Spreadsheet Add-In enables analysts to work with live dimensional data in the familiar spreadsheet environment of Microsoft Excel. The Add-In fetches data using an active connection to an OLAP data store, and displays the data in a spreadsheet. Users can use the Add-In to perform OLAP operations such as drilling, rotation, and data selection.

You can obtain the software, tutorials, and documentation from the Oracle Technology Network at

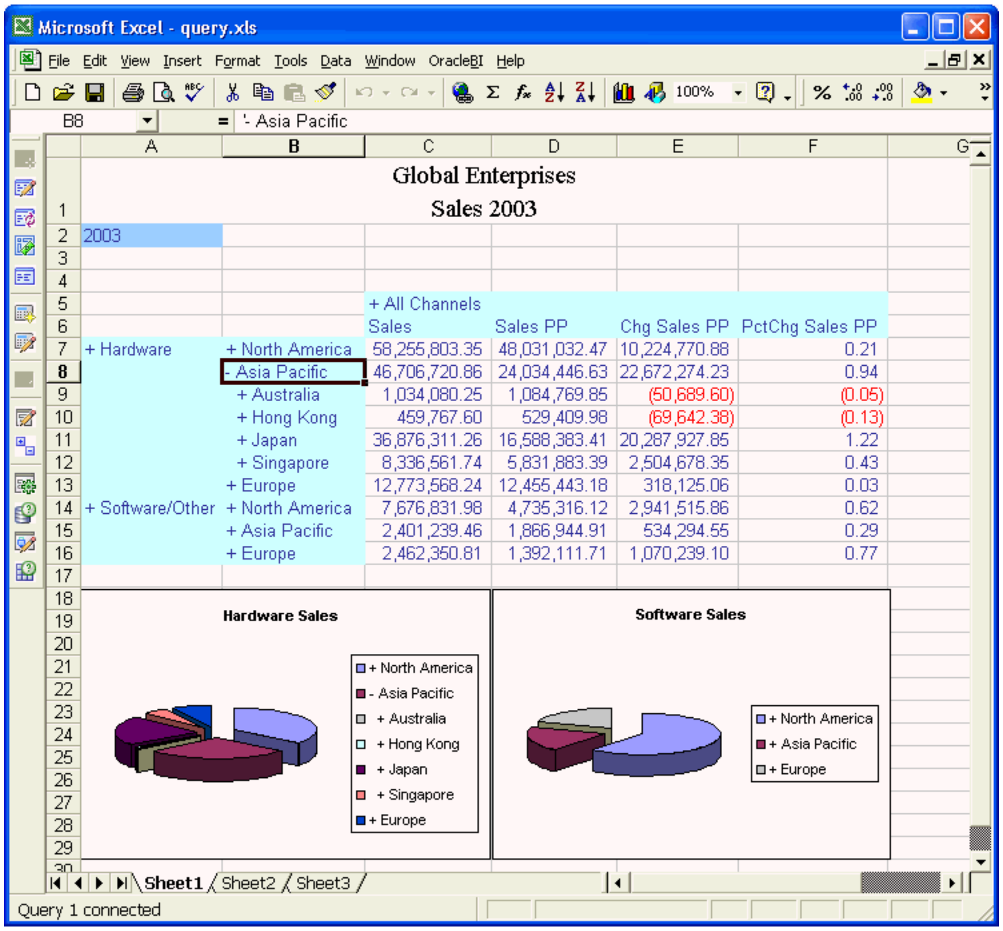
`http://www.oracle.com/technetwork/middleware/bi-foundation/downloads/download-088181.html`

Figure 5–1 shows data from the Global Units Cube displayed in a spreadsheet. The title, data formatting, and pie charts were implemented in Excel. The Add-In maintains a live connection with Oracle Database, which can be reactivated in later sessions. This figure shows the results from a single query, but you can insert numerous queries into a single worksheet.

By clicking a plus (+) next to a dimension member in the crosstab, you can drill down to the contributing members. The charts change dynamically to show the same selection of data as the crosstab. In Figure 5–1, Asia Pacific is expanded under

Hardware but not under Software. The difference in data selection appears in the charts also.

Figure 5–1 Displaying Oracle Dimensional Data in a Spreadsheet


Microsoft Excel - query.xls
File Edit View Insert Format Tools Data Window OracleBI Help
100%
B8
= - Asia Pacific
Sheet1 Sheet2 Sheet3
Query 1 connected

Global Enterprises

Sales 2003

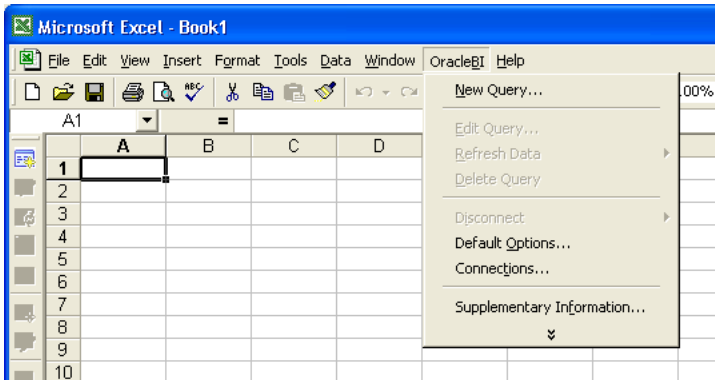
	A	B	C	D	E	F
2	2003					
5			+ All Channels			
6			Sales	Sales PP	Chg Sales PP	PctChg Sales PP
7	+ Hardware	+ North America	58,255,803.35	48,031,032.47	10,224,770.88	0.21
8		- Asia Pacific	46,706,720.86	24,034,446.63	22,672,274.23	0.94
9		+ Australia	1,034,080.25	1,084,769.85	(50,689.60)	(0.05)
10		+ Hong Kong	459,767.60	529,409.98	(69,642.38)	(0.13)
11		+ Japan	36,876,311.26	16,588,383.41	20,287,927.85	1.22
12		+ Singapore	8,336,561.74	5,831,883.39	2,504,678.35	0.43
13		+ Europe	12,773,568.24	12,455,443.18	318,125.06	0.03
14	+ Software/Other	+ North America	7,676,831.98	4,735,316.12	2,941,515.86	0.62
15		+ Asia Pacific	2,401,239.46	1,866,944.91	534,294.55	0.29
16		+ Europe	2,462,350.81	1,392,111.71	1,070,239.10	0.77

Getting Started With the OracleBI Spreadsheet Add-In

Installation of the Add-In creates a new OracleBI menu on the Excel menu bar, as shown in Figure 5–2. You can add queries to any spreadsheet. The currently selected cell in Excel identifies the upper left cell of the crosstab that will be downloaded from Oracle Database. A warning message appears if the Oracle data will overwrite any data already entered in the spreadsheet.

On the OracleBI menu, click **Supplementary Information** for links to the tutorials and demonstrations. The Online Tutorial provides detailed instructions for the tasks shown briefly in this chapter.

Figure 5–2 Spreadsheet Add-In Menu



Creating a Query Using the Add-In

The Spreadsheet Add-In uses the same query wizard as the Measure Viewer in Analytic Workspace Manager. You select the measures, the layout, and the dimension members. The first page of the wizard lists all the measures that you have privileges to query.

To create a query:

1. On the OracleBI menu, choose **New Query**.

   The Connect Query to Oracle OLAP Data Source dialog box opens.

2. On the Connection Editor tab, define a connection to Oracle Database.

   Click **Help** for information about your choices.

3. On the OLAP Connection tab, provide your credentials to log in to Oracle Database.

   The OracleBI Query Wizard opens.

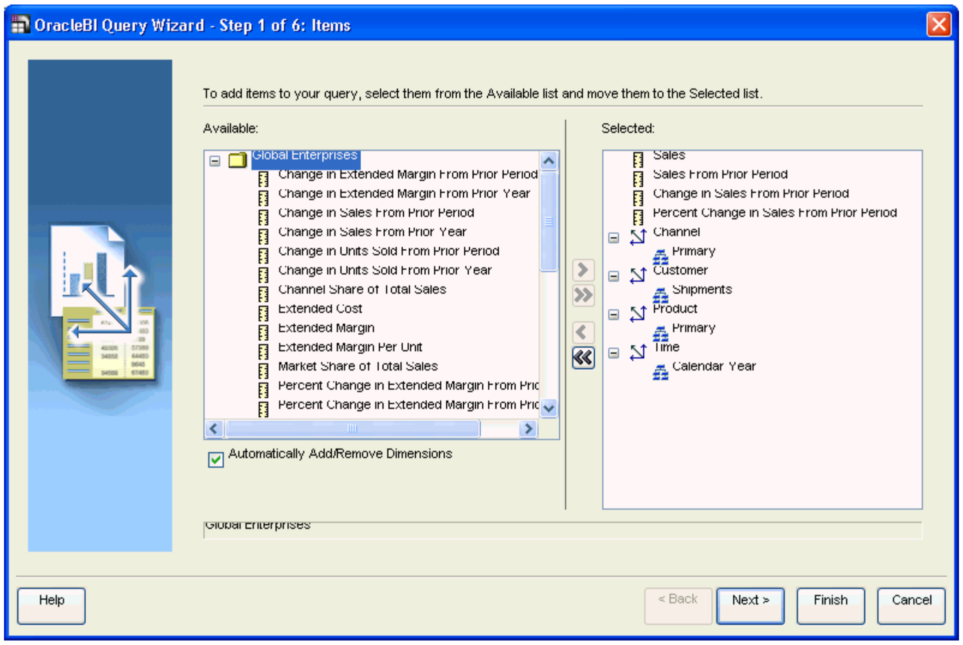
4. Follow the steps of the wizard. When you are done, the data is displayed in Excel.

5. Choose **Edit Query** from the OracleBI menu, if you want to change the data selection or any other aspect of the query.

   If this choice is not available, click any cell with data returned by the query.

Figure 5–3 shows the first page of the Query Wizard. In this example, a measure folder named "Global Enterprises" contains the measures and calculated measures in the Global analytic workspace.

Figure 5–3 Selecting Measures in the Query Wizard



OracleBI Query Wizard - Step 1 of 6: Items

To add items to your query, select them from the Available list and move them to the Selected list.

Available:

- Global Enterprises
  - Change in Extended Margin From Prior Period
  - Change in Extended Margin From Prior Year
  - Change in Sales From Prior Period
  - Change in Sales From Prior Year
  - Change in Units Sold From Prior Period
  - Change in Units Sold From Prior Year
  - Channel Share of Total Sales
  - Extended Cost
  - Extended Margin
  - Extended Margin Per Unit
  - Market Share of Total Sales
  - Percent Change in Extended Margin From Pric
  - Percent Change in Extended Margin From Pric

☑ Automatically Add/Remove Dimensions

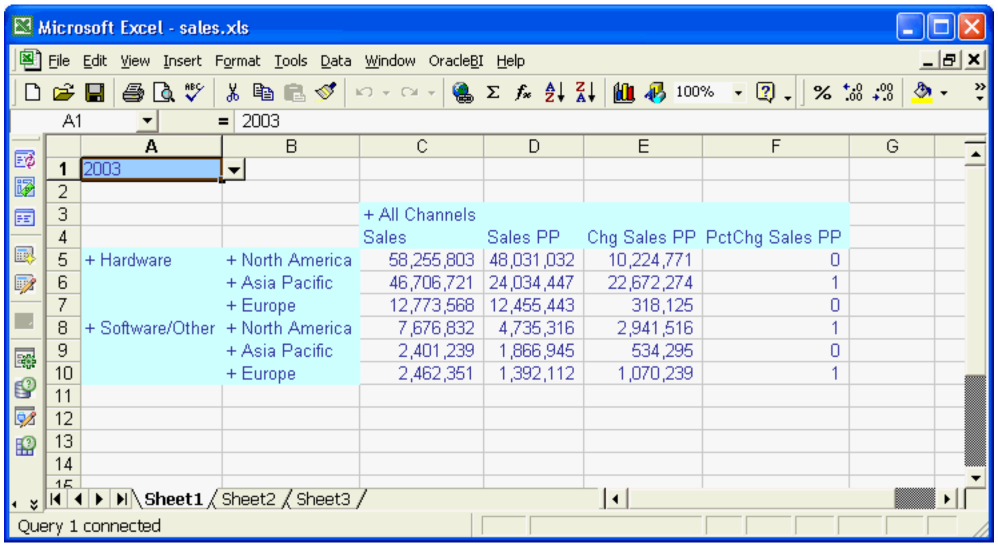
Selected:

- Sales
- Sales From Prior Period
- Change in Sales From Prior Period
- Percent Change in Sales From Prior Period
- Channel
  - Primary
- Customer
  - Shipments
- Product
  - Primary
- Time
  - Calendar Year

Global Enterprises

[Figure 5–4](#) shows the downloaded data from Oracle. It appears with the default Excel formatting for the font, size, justification, decimal places, and so forth.

Figure 5–4 Oracle Data Downloaded to a Spreadsheet



Microsoft Excel - sales.xls

File Edit View Insert Format Tools Data Window OracleBI Help
100%

A1 = 2003

	A	B	C	D	E	F	G
1	2003						
2							
3			+ All Channels				
4			Sales	Sales PP	Chg Sales PP	PctChg Sales PP	
5	+ Hardware	+ North America	58,255,803	48,031,032	10,224,771	0	
6		+ Asia Pacific	46,706,721	24,034,447	22,672,274	1	
7		+ Europe	12,773,568	12,455,443	318,125	0	
8	+ Software/Other	+ North America	7,676,832	4,735,316	2,941,516	1	
9		+ Asia Pacific	2,401,239	1,866,945	534,295	0	
10		+ Europe	2,462,351	1,392,112	1,070,239	1	
11							
12							
13							
14							

Sheet1 / Sheet2 / Sheet3

Query 1 connected

Using Excel Features on Oracle Dimensional Data

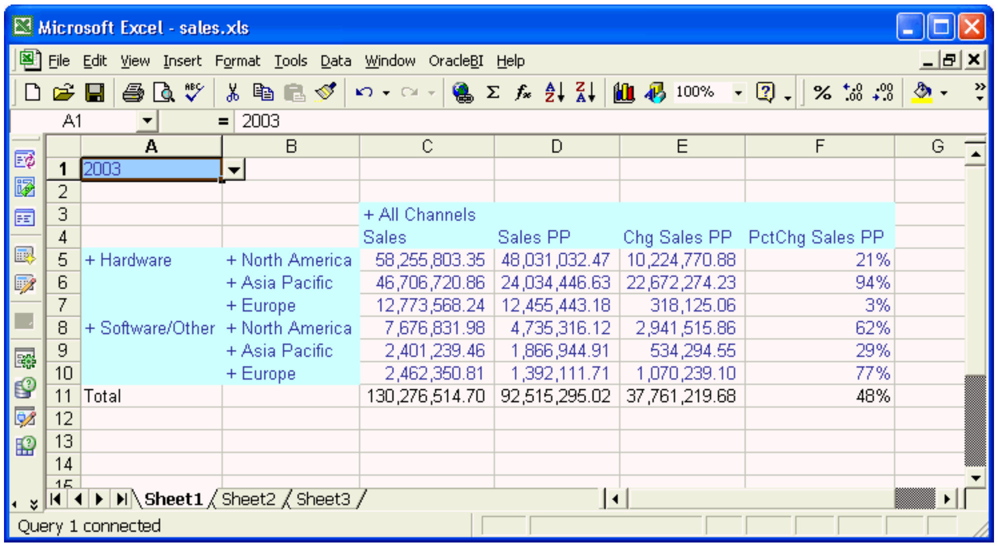
You can change all of the default settings in Excel the same as for any other data. [Figure 5–5](#) shows the same data as [Figure 5–4](#), but with the following changes:

- All data is displayed to two decimal places after using the Increase Decimal tool on the Excel toolbar.
- The PctChg Sales PP column displays the data as percentages instead of decimals after using the Percent Style tool.

- A new row contains data calculated in Excel using the Sum function on Sales, Sales PP, and Chg Sales PP, and the Average function on PctChg Sales PP. The row has the label "Total."

You can also add charts in Excel using the Chart Wizard. Figure 5–1 shows the addition of two pie charts.

Figure 5–5 Using Excel to Format the Data


Microsoft Excel - sales.xls
File Edit View Insert Format Tools Data Window OracleBI Help
100%
A1
= 2003
Sheet1 Sheet2 Sheet3
Query 1 connected

	A	B	C	D	E	F
1	2003					
2						
3			+ All Channels			
4			Sales	Sales PP	Chg Sales PP	PctChg Sales PP
5	+ Hardware	+ North America	58,255,803.35	48,031,032.47	10,224,770.88	21%
6		+ Asia Pacific	46,706,720.86	24,034,446.63	22,672,274.23	94%
7		+ Europe	12,773,568.24	12,455,443.18	318,125.06	3%
8	+ Software/Other	+ North America	7,676,831.98	4,735,316.12	2,941,515.86	62%
9		+ Asia Pacific	2,401,239.46	1,866,944.91	534,294.55	29%
10		+ Europe	2,462,350.81	1,392,111.71	1,070,239.10	77%
11	Total		130,276,514.70	92,515,295.02	37,761,219.68	48%

Creating Reports in Discoverer Plus OLAP

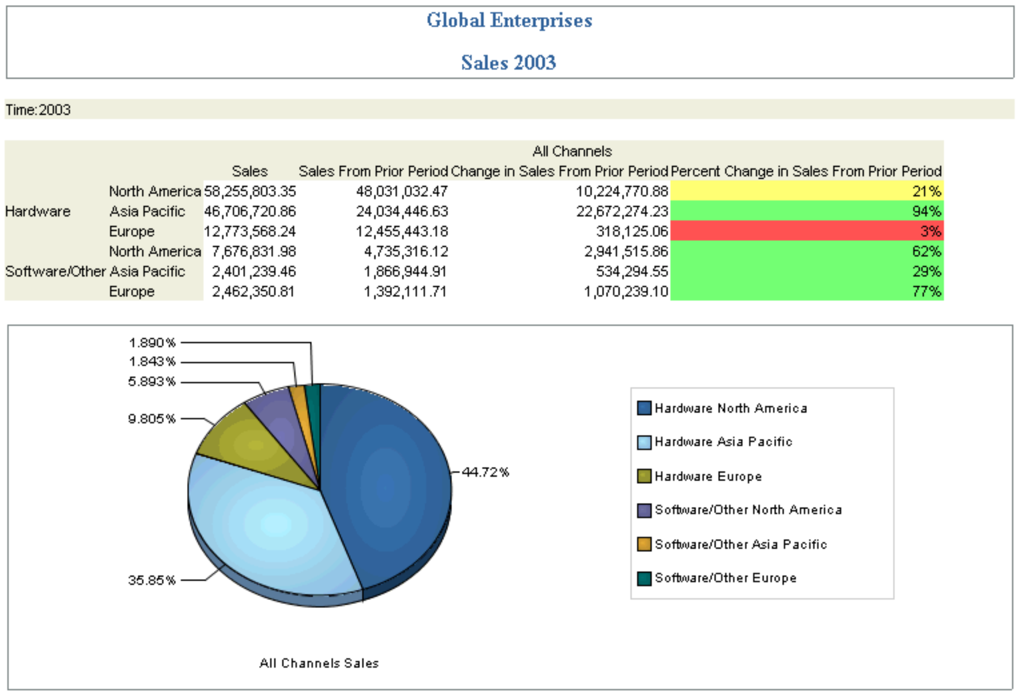
Discoverer Plus OLAP provides various wizards to guide you through the entire process of building and publishing sophisticated reports containing crosstabs and graphs. You can choose from multiple layout options to create a visual representation of the query results. You can create queries, drill, pivot, slice and dice data, add analytic calculations, graph the data, share results with other users, and export your Discoverer reports in various data formats. Discoverer reports can also be published in dashboards where other users can access them from their browsers.

You can obtain the software, tutorials, and documentation from the Oracle Technology Network at

`http://www.oracle.com/technetwork/developer-tools/discoverer/overview/index.html`

Figure 5–6 shows a report developed in Discoverer, exported in HTML format, and displayed in a browser.

Figure 5–6 Sales Report Generated by Discoverer Plus OLAP



Global Enterprises

Sales 2003

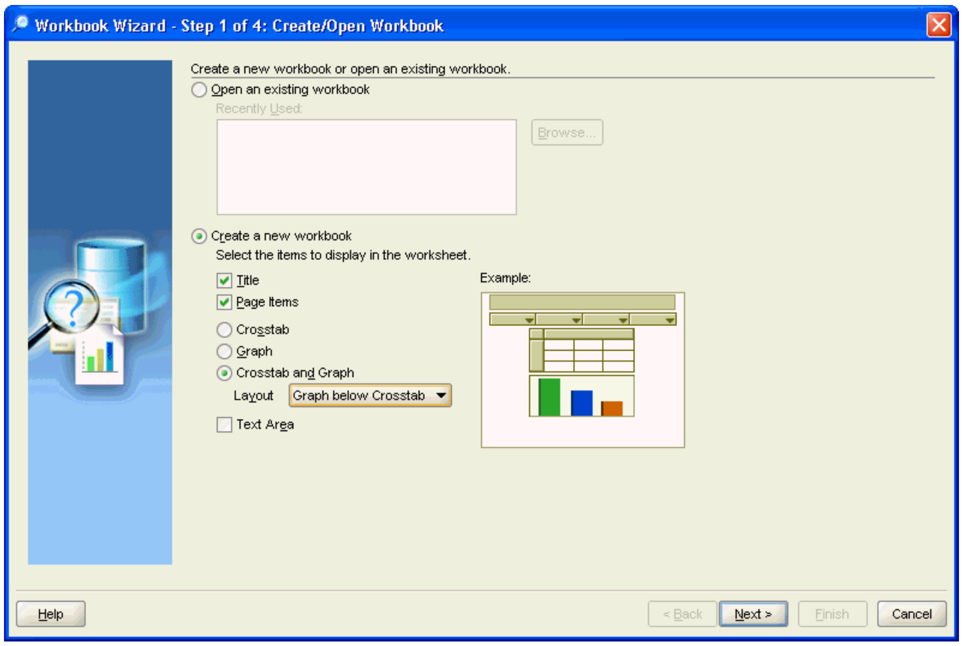
Time:2003

		All Channels			
		Sales	Sales From Prior Period	Change in Sales From Prior Period	Percent Change in Sales From Prior Period
Hardware	North America	58,255,803.35	48,031,032.47	10,224,770.88	21%
	Asia Pacific	46,706,720.86	24,034,446.63	22,672,274.23	94%
	Europe	12,773,568.24	12,455,443.18	318,125.06	3%
Software/Other	North America	7,676,831.98	4,735,316.12	2,941,515.86	62%
	Asia Pacific	2,401,239.46	1,866,944.91	534,294.55	29%
	Europe	2,462,350.81	1,392,111.71	1,070,239.10	77%

Getting Starting with Discoverer Plus OLAP

Discoverer organizes worksheets into workbooks, like a spreadsheet package. After you open Discoverer and log in, the Workbook Wizard opens automatically, as shown in Figure 5–7. You can open an existing workbook, or you can create a new workbook and define the contents of the first worksheet.

Figure 5–7 Defining a New Discoverer Workbook



Workbook Wizard - Step 1 of 4: Create/Open Workbook

Create a new workbook or open an existing workbook.

○ Open an existing workbook

Recently Used:

[✓ ● Create a new workbook]

Select the items to display in the worksheet.

☑ Title

☑ Page Items

○ Crosstab

○ Graph

[✓ ● Crosstab and Graph]

Layout: Graph below Crosstab

☐ Text Area

Creating a Query

When creating a new worksheet, you can specify in the first step of the Workbook Wizard which options you want, such as a title or a graph. The wizard then steps you through the creation of these options. You can add or delete these optional components at any time.

The Workbook Wizard uses the same query wizard as the Spreadsheet Add-In and the Measure Viewer in Analytic Workspace Manager. You select the measures, the layout, and the dimension members.

To create a new workbook and a worksheet:

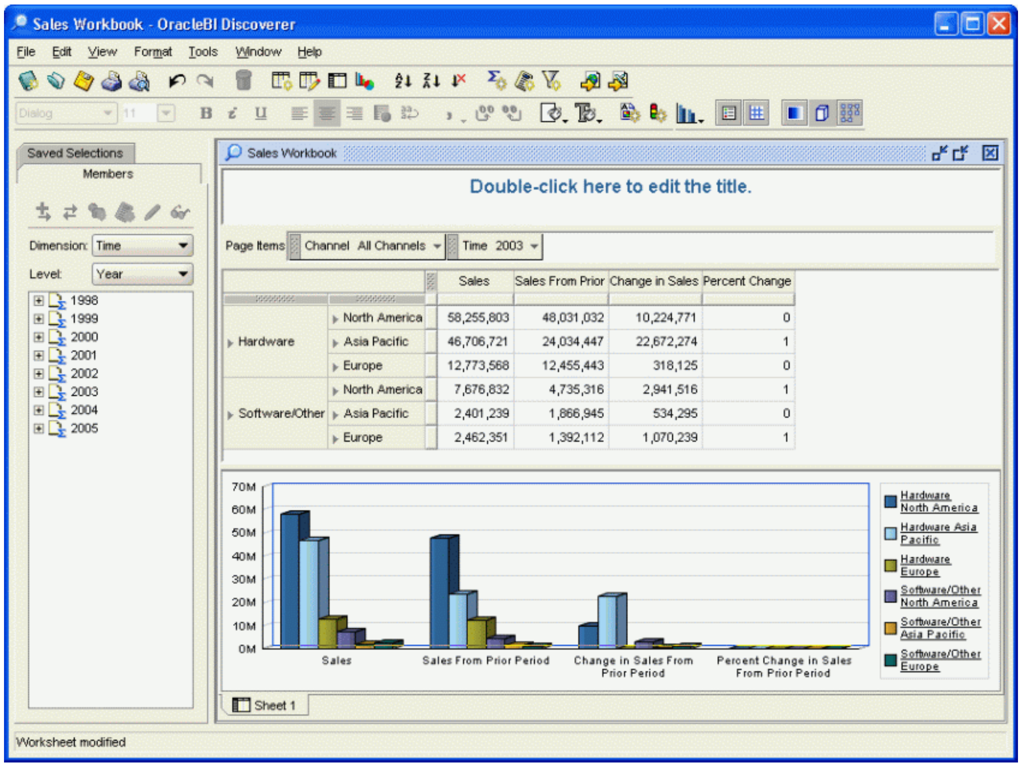
1. Open Discoverer Plus OLAP and log in to Oracle Database.

   The Workbook Wizard opens to the Create/Open Workbook page.

2. Select Create a New Workbook, the select the optional components of the first worksheet.
3. Follow the steps of the wizard. When you are done, the data is displayed in the main window as a crosstab, or a graph, or both.
4. Modify the worksheet as you like, changing the data selection or the data formats, adding calculations, and so forth.
5. To add another worksheet to the workbook, choose **Add Worksheet** from the Edit menu.

Figure 5–8 shows a worksheet that was defined in the Workbook Wizard. It appears with the default formatting for the font, justification, decimal places, and so forth.

Figure 5–8 Default Query Results in Discoverer



		Sales	Sales From Prior	Change in Sales	Percent Change
Hardware	North America	58,255,803	48,031,032	10,224,771	0
	Asia Pacific	46,706,721	24,034,447	22,672,274	1
	Europe	12,773,568	12,455,443	318,125	0
Software/Other	North America	7,676,832	4,735,316	2,941,516	1
	Asia Pacific	2,401,239	1,866,945	534,295	0
	Europe	2,462,351	1,392,112	1,070,239	1

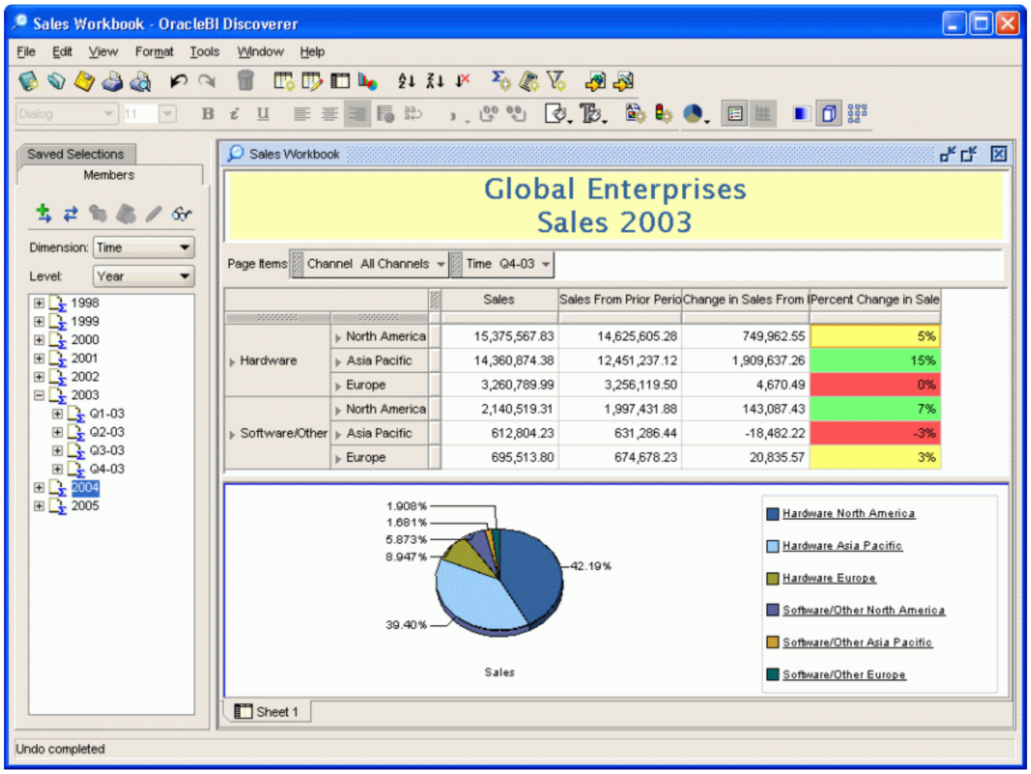
Formatting the Data in Discoverer Plus

You can change every aspect of the query and its formatting using the menus and toolbars. Figure 5–9 shows the same basic query as Figure 5–8, but with the following changes:

- A new title appears in a custom font and background color. Click the default title.
- All data is displayed to two decimal places. Use the Add Decimal tool.
- The Percent Change in Sales From Prior Period column displays the data as percentages instead of decimals. Use the Format As Percent tool.
- Stoplight formatting of the Percent Change in Sales From Prior Period column highlights products that are outside the target range in some areas. Select New Stoplight Format from the Format menu.
- The Time selection changed from 2003 to Q4-03. Use the Members tab on the Available Items pane.
- A pie chart for Sales replaced the bar charts for all measures. Use the Edit Graph Type and Properties tool.

You can also perform calculations on the data, define new calculated measures, and create saved selections of dimension members.

Figure 5–9 Changing the Default Formatting in Discoverer



Global Enterprises
Sales 2003

Page Items: Channel All Channels; Time Q4-03

		Sales	Sales From Prior Perio	Change in Sales From	Percent Change in Sale
Hardware	North America	15,375,567.83	14,625,605.28	749,962.55	5%
	Asia Pacific	14,360,874.38	12,451,237.12	1,909,637.26	15%
	Europe	3,260,789.99	3,256,119.50	4,670.49	0%
Software/Other	North America	2,140,519.31	1,997,431.88	143,087.43	7%
	Asia Pacific	612,804.23	631,286.44	-18,482.22	-3%
	Europe	695,513.80	674,678.23	20,835.57	3%

6 Enhancing Your Database With Analytic Content

Oracle OLAP provides an extensive set of analytic functions for enhancing your database with information-rich content. This chapter explains how you can use Analytic Workspace Manager to enhance your database by defining calculated measures and calculation plans.

This chapter contains the following topics:

What Is a Calculated Measure?

Calculated measures return values that are computed at run-time from data stored in one or more measures. Like relational views, calculated measures store queries against data stored in other objects. Because calculated measures do not store data, you can create dozens of them without increasing the size of the database. You can use them as the basis for defining other calculated measures, which adds depth to the types of calculations you can create in Analytic Workspace Manager.

Because calculated measures do not contain data, they are not associated with a build process. You can create a calculated measure at any time, and it is available immediately for querying.

Functions for Defining Calculations

Oracle OLAP offers an extensive range of functions and operators that can be used to define calculated measures. Analytic Workspace Manager provides a Calculation Wizard, which provides both arithmetic and analytic calculations. The calculations in the cube are performed on a cell-by-cell basis at all levels of the dimension hierarchies. The analytic functions provide the most powerful computations and fuel the most useful queries for business intelligence and similar applications. By enriching your database with an extensive list of calculated measures, you enable analysts and decision makers to make comparisons, identify trends, and make solid decisions based on the best information available.

The Calculation Wizard provides these arithmetic calculations:

- **Basic Arithmetic**: Addition, subtraction, multiplication, and division, using two measures or a measure and a number
- **Advanced Arithmetic**: Cumulative total, index, percent markup, percent variance, rank, share, variance

The Calculation Wizard provides these analytic functions:

- **Prior/Future Comparison**: Prior value, difference from prior period, percent difference from prior period, future value
- **Time Frame**: Moving average, moving maximum, moving minimum, moving total, year to date

Analytic functions provided by Oracle OLAP leverage the knowledge associated with the dimensions about levels and family relationships. Time dimensions have additional information that enables them to support time series methods such as lags, leads, moving and year-to-date calculations. Because the knowledge is stored with the dimension, you do not need to specify these relationships when creating a calculated measure.

Creating Calculated Measures

Using the Calculation Wizard, you can create calculated measures in the same cube with the source measures or in a separate cube.

To create a calculated measure:

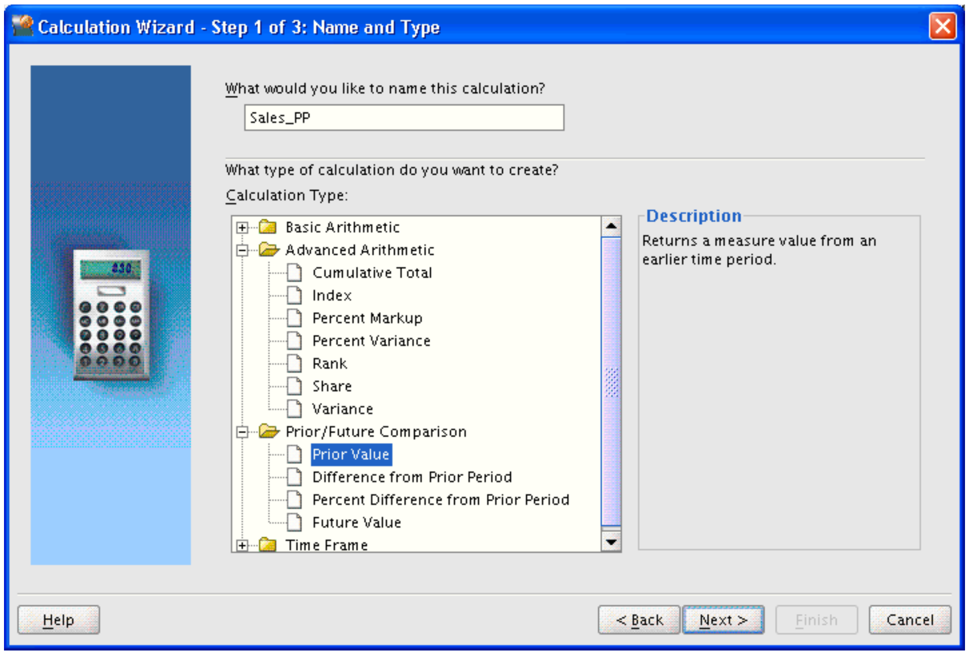
1. Expand the folder for the cube that contains the base measures that will be used in the calculation.
2. Right-click **Calculated Measures**, then choose **Create Calculated Measure** from the shortcut menu.

   The Calculation Wizard Welcome page is displayed.
3. Follow the steps of the wizard.

   Click **Help** for specific information about these choices. When you are done, the name of the new calculated measure appears as an item in the Calculated Measures folder.

Figure 6–1 displays the Name and Type page of the Calculation Wizard.

Figure 6–1 Selecting a Calculation Type


Calculation Wizard - Step 1 of 3: Name and Type

What would you like to name this calculation? Sales_PP

What type of calculation do you want to create?

Calculation Type:

- Basic Arithmetic
- Advanced Arithmetic
  - Cumulative Total
  - Index
  - Percent Markup
  - Percent Variance
  - Rank
  - Share
  - Variance
- Prior/Future Comparison
  - Prior Value
  - Difference from Prior Period
  - Percent Difference from Prior Period
  - Future Value
- Time Frame

Description

Returns a measure value from an earlier time period.

Using the Calculation Wizard

The Calculation Wizard supports all of the calculations typically in demand for business intelligence applications. The following topics describe the types of calculations available through the Calculation Wizard.

Basic Arithmetic Operations

Basic arithmetic operations enable you to perform cell-by-cell calculations on two measures or a measure and a number, using addition, subtractions, multiplication, or division.

Multiplication Example

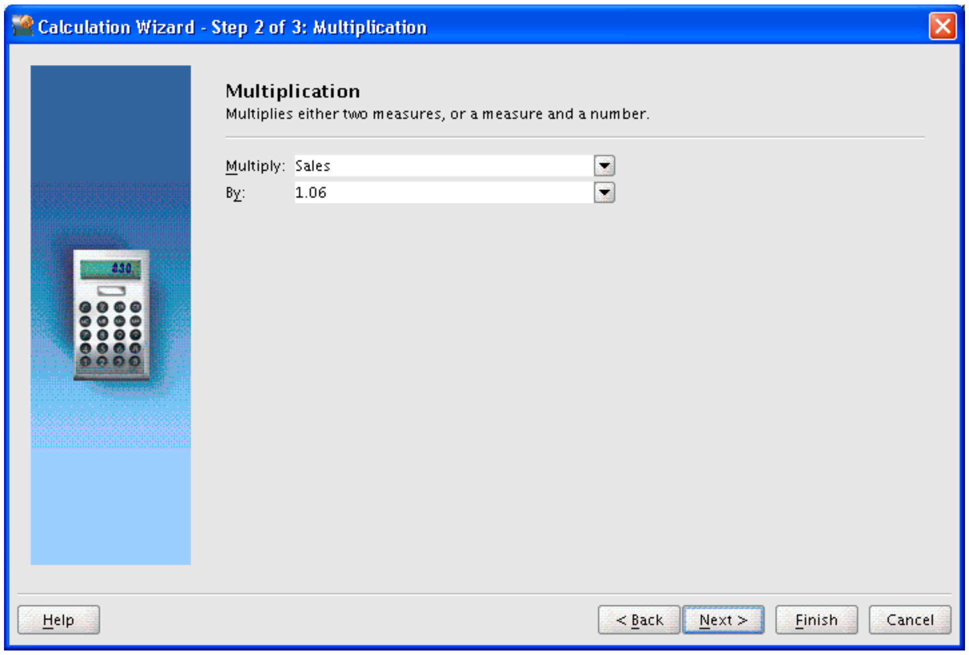
The Multiplication page defines a calculated measure using these parameters, as shown in Figure 6–2:

Multiply: Sales
By: 1.06

These are the results of a query against this calculated measure, which generates sales targets based on the results for the current year:

```
                   Sales    Sales Budget
Memory         5,272,678       5,589,038
CD/DVD        15,083,555      15,988,568
Portable PCs  19,155,814      20,305,162
Desktop PCs   67,900,544      71,974,577
Monitors       4,346,492       4,607,281
Modems/Fax     5,977,011       6,335,632
```

Figure 6–2 Defining a Calculation



Multiplication

Multiplies either two measures, or a measure and a number.

Multiply: Sales

By: 1.06

Percent Variance

Percent Variance calculates the percent difference between two measures.

Percent Variance Example

The Percent Variance page defines a calculated measure using these parameters:

Base Measure: Unit Price
Target Measure: Unit Cost

These are the results of a query against this calculated measure:

```
                 Unit Price       Unit Cost    Pct Variance
Memory                  404             366           10.51
CD/DVD                  191             156           22.55
Portable PCs          2,425           2,408            0.72
Desktop PCs           1,652           1,663          (0.66)
Monitors                388             324           19.79
Modems/Fax              111              98           14.05
```

Index

Index calculates the percent difference between the values of a measure and a selected value that serves as a base number.

Index Example

The Index page defines a calculation using these parameters:

Index: Sales
Starting at Time for: 2003
Customer: All Customers
Product: Desktop PCs
Channel: All Channels

These are the results of a query against this calculated measure, which uses Desktop PCs as the index for hardware products.

```
                Sales            Sales Index
Desktop PCs        67,900,544          100%
Portable PCs       19,155,814           28%
CD/DVD             15,083,555           22%
Modems/Fax          5,977,011            9%
Memory              5,272,678            8%
Monitors            4,346,492            6%
```

Rank

Rank orders the values of a dimension based on the values of the selected measure. When defining a rank calculation, you choose the dimension, a hierarchy, and the measure. You also choose the group in which the dimension members are ranked:

- **Total**: Ranks all members of the hierarchy.
- **Parent**: Ranks members with the same parent.
- **Level**: Ranks members at the same level.

Rank Example

The Rank page defines a calculated measure using these parameters:

Rank: Product
In: Primary
Within: Parent
Based On: Units Sold
Order: Lowest to Highest

These are the results of a query against this calculated measure in which the products are ranked from lowest to highest based on units sold.

```
                Units Sold   Rank
Portable PCs     8,259       1
Monitors        14,520       2
Memory          15,093       3
Desktop PCs     40,729       4
Modems/Fax      48,743       5
CD/DVD          64,160       6
```

Share

Share calculates the ratio of a measure's value for the current dimension member to a baseline, which is one of the following:

- **Total**: The total of all values for members on the same level as the current member.
- **Parent**: The total of all values for members on the same level as parent of the current member.
- **Level**: The total of all values for members on a specified level.
- **Member**: The value of a specified dimension member.

When creating a share calculation, you also select the measure, dimension, and hierarchy.

Share Example

The Share page defines a calculated measure using these parameters:

Share of: Sales
For: Product
In: Primary
As a Percent of: Member Total Product

These are the results of a query against this calculated measure. The Total Share column displays the percent share of the total for the selected products.

```
                       Sales           Total Share
Total Product            130,276,515           1.00
  Hardware               117,736,092           0.90
  Software/Other          12,540,422           0.10
```

Cumulative Total

Cumulative totals start with the first time period within a particular rank and calculate a running total up to the current member. The range can be all members of the level or just members with the same parent.

Cumulative Total Example

The Cumulative Total page defines a calculated measure using these parameters:

Accumulate: Sales
Over Time In: Calendar Year
Within: Parent

These are the results of a query against this calculated measure.

```
                Sales               Cumulative Sales
2003                130,276,515          698,876,935
  Q1-03              26,946,411           26,946,411
      Jan-03          8,400,440            8,400,440
      Feb-03          8,953,827           17,354,267
      Mar-03          9,592,144           26,946,411
  Q2-03              33,247,676           60,194,087
      Apr-03         10,457,165           10,457,165
      May-03         11,373,236           21,830,401
      Jun-03         11,417,275           33,247,676
  Q3-03              33,636,358           93,830,445
      Jul-03         10,705,642           10,705,642
      Aug-03         10,268,927           20,974,569
      Sep-03         12,661,790           33,636,358
  Q4-03              36,446,070          130,276,515
      Oct-03         11,705,602           11,705,602
      Nov-03         12,084,512           23,790,114
      Dec-03         12,655,955           36,446,070
```

Prior and Future Time Periods

Prior and future time period calculations are an important gauge for detecting and analyzing trends. Oracle OLAP provides several calculations under Prior/Future Comparison:

- **Prior Value**: Returns the value of a measure from an earlier time period.
- **Difference From Prior Period**: Calculates the difference between values for the current time period and an earlier time period.

- **Percent Difference From Prior Period**: Calculates the percent difference between values for the current time period and an earlier time period.
- **Future Value**: Returns the value of a measure from a later time period.

When creating a calculation for prior or future time periods, you choose the measure, the time dimension, the hierarchy, and the distance from the current period. The distance can be calculated as any of the following:

- Year Ago
- Period Ago
- Number of Years, Quarters, Months, Weeks, or Days

Prior Period Example

The Prior Value page defines a calculated measure using these parameters:

Measure: Sales
Over Time in: Calendar Year
From: Period Ago

These are the results of a query against this calculated measure. The Prior Period column shows the value of Sales for the preceding period at the same level in the Calendar Year hierarchy.

```
              Sales            Prior Period
2002          92,515,295          116,931,479
  Q1-02       21,499,270           31,846,054
  Q2-02       22,586,748           21,499,270
  Q3-02       23,845,942           22,586,748
  Q4-02       24,583,335           23,845,942
2003         130,276,515           92,515,295
  Q1-03       26,946,411           24,583,335
  Q2-03       33,247,676           26,946,411
  Q3-03       33,636,358           33,247,676
  Q4-03       36,446,070           33,636,358
```

Moving Calculations

Moving calculations are performed over the time periods surrounding the current period. They smooth the fluctuations in the data, so that you can more easily detect trends. Oracle OLAP provides several aggregation methods under Time Frame for moving calculations:

- **Moving Average**: Calculates the average value for a measure over a fixed number of time periods.
- **Moving Maximum**: Calculates the maximum value for a measure over a fixed number of time periods.
- **Moving Minimum**: Calculates the minimum value for a measure over a fixed number of time periods.
- **Moving Total**: Returns the total value for a measure over a fixed number of time periods.

You choose the measure, the time dimension, and the hierarchy. You can also select the number of previous time periods to include in the calculations.

Moving Minimum Example

The Prior Value page defines a calculated measure using these parameters:

Measure: Sales
Over Time in: Calendar Year
Include Previous: 1 period

These are the results of a query against this calculated measure for quarterly data. Each value of Minimum Sales is the smaller of the current and the previous values.

```
            Sales             Minimum Sales
Q1-03         26,946,411           24,583,335
Q2-03         33,247,676           26,946,411
Q3-03         33,636,358           33,247,676
Q4-03         36,446,070           33,636,358
Q1-04         32,977,875           32,977,875
Q2-04         35,797,920           32,977,875
```

Period to Date Calculations

Period-to-date calculations generate a running total of the data within a particular time period. Oracle OLAP provides period-to-date under Time Frame. You select the measure, the time dimension, the hierarchy, and the level.

Period to Date Example

The Period to Date page defines a calculated measure using these parameters:

Measure: Sales
Over Time in: Calendar Year
At Level: Year

These are the results of a query against this calculated measure. The Year to Date column shows a running total of sales for the months within the year.

```
                   Sales              Year to Date
2003                130,276,515          130,276,515
  Q1-03              26,946,411           26,946,411
      Jan-03          8,400,440            8,400,440
      Feb-03          8,953,827           17,354,267
      Mar-03          9,592,144           26,946,411
  Q2-03              33,247,676           60,194,087
      Apr-03         10,457,165           37,403,576
      May-03         11,373,236           48,776,812
      Jun-03         11,417,275           60,194,087
  Q3-03              33,636,358           93,830,445
      Jul-03         10,705,642           70,899,728
      Aug-03         10,268,927           81,168,656
      Sep-03         12,661,790           93,830,445
  Q4-03              36,446,070          130,276,515
      Oct-03         11,705,602          105,536,047
      Nov-03         12,084,512          117,620,559
      Dec-03         12,655,955          130,276,515
```

Nested Calculations

You can extend the variety of functions available through the Calculation Wizard by using a calculated measure as the basis for another calculated measure.

For example, the Calculation Wizard can create rank and prior period calculations. You can create a calculated measure that calculates rank, then use it to calculate the rank of the prior period.

Nested Calculations Example

The Rank page creates a Rank calculation named Units Rank using these parameters:

Rank: Product
In: Product Rollup
Within: Parent
Based on: Units Sold
Order: Lowest to Highest

The Prior Value page creates a Prior Year calculation from Units Rank:

Measure: Units Rank
Over Time in: Calendar Year
Go Back by: 1 Year

These are the results of a query against the calculated measures.

```
                           Units Sold  Units Rank   Prior Year
Q1-03     Portable PCs          2,051           1            1
          Monitors              3,153           2            2
          Memory                3,468           3            3
          Desktop PCs           7,721           4            4
          Modems/Fax           11,349           5            5
          CD/DVD               13,225           6            6
Q1-04     Portable PCs          2,082           1            1
          Monitors              3,685           2            2
          Memory                3,846           3            3
          Desktop PCs           9,429           4            4
          Modems/Fax           13,106           5            5
          CD/DVD               18,320           6            6
```

Generating Forecasts, Allocations, and Aggregations

Analytic Workspace Manager provides the tools for generating advanced analytic content:

- **Forecast**: Uses statistical time-series forecasting methods to predict future performance based on past results

  See Also: Chapter 7, "Generating Forecasts"

- **Allocation**: Distributes data down the dimension hierarchies using a selected allocation method

  See Also: Chapter 9, "Allocations"

- **Aggregation**: Consolidates data up the dimension hierarchies using a selected aggregation method

  See Also: Chapter 8, "Advanced Aggregations"

You create these types of calculations by developing a calculation plan. Calculation plans are composed of an ordered list of steps that generate additional analytical data. Each step performs a specific type of calculation. Unlike calculated measures, these steps generate data that is stored in the cube. By specifying the order in which these steps are performed, you can allow for interdependencies.

You execute calculation plans using the Maintenance Wizard, typically after loading and aggregating new data.

To create a calculation plan:

1. Expand the folder for the analytic workspace.
2. Right-click **Calculation Plans**, then choose **Create Calculation Plan** from the shortcut menu.

   The Create Calculation Plan dialog box is displayed.
3. Complete the General tab.

   Click **Help** for specific information about these choices.
4. To create a new step, click **New Step**.
5. Choose the type of step: Forecast, allocation, or aggregation.

   The New Step dialog box is displayed for that type of calculation.
6. Complete all tabs, then click **Create**.

   The new step is listed on the Calculation Plan General tab.
7. Click **Create**.

   The new calculation plan appears as an item in the Calculation Plans folder.
8. To run the calculation plan:

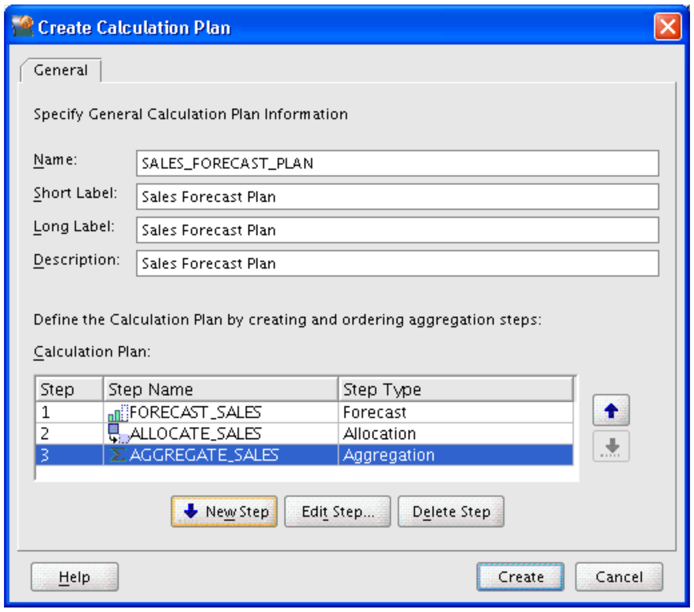
   a. Right-click it on the navigation tree and choose **Execute Calculation Plan**.

      The Maintenance wizard opens.

   b. Follow the steps of the wizard.

Figure 6–3 shows the Create Calculation Plan dialog box with three steps defined.

Figure 6–3 Creating a Calculation Plan



Specify General Calculation Plan Information

Name:	SALES_FORECAST_PLAN
Short Label:	Sales Forecast Plan
Long Label:	Sales Forecast Plan
Description:	Sales Forecast Plan

Define the Calculation Plan by creating and ordering aggregation steps:

Calculation Plan:

Step	Step Name	Step Type
1	FORECAST_SALES	Forecast
2	ALLOCATE_SALES	Allocation
3	AGGREGATE_SALES	Aggregation

New Step | Edit Step... | Delete Step

7 Generating Forecasts

Forecasting is a natural extension to the types of data analysis typically performed on the historical data stored in analytic workspaces. Using Analytic Workspace Manager, you can quickly generate forecasts of your measures. This chapter provides a basic framework for generating and using quantitative forecasting methods for those who do not have a strong statistical background. It also provides specific information about the particular forecasting engine provided with Oracle OLAP.

This chapter contains the following topics:

Introduction to Forecasting Considerations

Forecasts are predictions about future events. They provide a basis for making decisions in a timely manner, which is often in advance of the facts. There are many ways of creating forecasts, and the best method for a particular forecast depends on many factors. Consider this question: Will it rain tomorrow?

The degree of difficulty in correctly predicting tomorrow's weather depends on where you live. You may live where the weather is extremely stable, with little or no variation from one day to the next. In this situation, if it is raining today, then you can be fairly certain that it will rain tomorrow.

However, if you live where the weather is in constant flux, with sudden and dramatic changes, then today's rainfall is not a reliable predictor. You may just make an informed guess, based on your analysis of the current weather pattern, or you might consult an arthritis sufferer whose joints ache with changes in the weather. Nonetheless, all of these methods (today's rainfall, informed guess, or swollen joints) should over time prove to be more accurate than just flipping a coin.

Now consider this question: Will it rain three months from today? Instead of basing your prediction on today's weather, you must consider the frequency of rainfall for the forecast period in previous years. If you live where rainy seasons and dry seasons are clearly defined, then you can probably answer this question with relative certainty based on the season. Otherwise, your ability to predict rainfall on a particular day that far into the future may be no better than a coin toss. To make a meaningful prediction, you may need to expand the forecast period to a week or more. You may also need to expand the size of the area in which you are predicting rain from your neighborhood to a larger region.

Finally, how important is it to correctly predict the weather on a particular day and at a particular place? If accuracy is critical -- such as planning a large outdoor event -- then an accurate forecast is worth some effort, and you might try several forecasting methods to see if their predictions converge. Regardless, you might still plan to erect a tent in case you get a downpour instead of the forecasted clear skies.

This simple example demonstrates several characteristics of forecasting:

- Stable patterns in historical data are more likely to generate an accurate forecast.
- Different methods are appropriate for different forecasts, depending on how far into the future you want to make a forecast and how stable your data is.
- Some forecasting methods are experiential or qualitative (informed guess or aching joints), and others are quantitative (historical data).
- The season may be an important factor in the forecast.
- Forecasting is not 100% accurate.
- The more precise the forecast, the more prone it is to error.
- Longer-range forecasts should generate data at higher levels to offset the increasing likelihood of error.
- The degree of error may be offset by your tolerance for error.
- If you have a low tolerance for error, then you may want to make some provisions that lessen the consequences of forecasting incorrectly.

These observations may help give you a perspective on what you want to forecast, how you want to design the forecast, and how you want to use the forecast in making decisions about your business.

Choosing a General Forecasting Approach

The first step in generating a forecast is to decide how far into the future you want to make your predictions. The approach that produces the best results for short-term forecasts is not a good predictor of long-term performance. The opposite is also true.

The critical question is, of course, how far into the future these time frames reach. Is "short" five weeks or five months? Is "long" five quarters or five years? As illustrated by the rain prediction example in "Introduction to Forecasting Considerations" on page 7-1, it all depends on a variety of factors:

- What are you trying to forecast?
- How stable is the historical data?
- How are you going to use this information?

These are just a few of the questions that you must answer in order to define the forecasting time frames for your specific business. Table 7–1 provides some general guidelines for these time periods.

Table 7–1 Guidelines for Choosing a Forecasting Approach

Time Frame	Typical Forecasting Horizon	Best Approach
Short	Up to 18 months	Time Series
Medium	6 to 36 months	Causal Analysis
Long	19 months to 5 years	Expert Opinion

Time Series

Time series forecasting methods are based on the premise that you can predict future performance of a measure simply by analyzing its past results. These methods identify a pattern in the historical data and use that pattern to extrapolate future values.

Past results can, in fact, be a very reliable predictor for a short period into the future. You can generate this type of forecast very quickly and easily, and you do not need either forecasting expertise or an in-depth knowledge of your data. The modeling techniques used by the time-series methods are relatively simple and run very fast. Time-series forecasting is extremely useful when hundreds or thousands of items must be forecast.

You may also use time-series methods to generate forecast data further into the future. However, the results are not as accurate, because factors other than past performance have a greater impact over time. For example, you or your competitors may change the pricing structure or run advertising campaigns, competitive products may come onto the market, or shifts in the economy or political events may affect performance. You should consider the forecast data generated by time series methods to be one component of a medium- or long-range forecast, which may be adjusted by expert opinion and other factors.

Analytic Workspace Manager provides access to a time-series forecasting engine, which is described in this chapter.

Causal Analysis

Causal analysis takes into consideration the external factors (the causes) that can affect a forecast, as described previously under "Time Series". Statistical regression methods are the basis for causal analysis. They use the forecasts for several independent measures to forecast a dependent measure. This type of forecast requires considerable skill and understanding of forecasting methodology and the relationships between independent and dependent variables. A good regression model produces the best results for medium-range forecasts.

However, because of the time, expense, and expertise needed to develop a model, most businesses restrict regression analysis to a few key business measures. For the other measures, they use a combination of methods including time-series and expert opinion.

The forecasting engine used by Analytic Workspace Manager does not support causal analysis. The linear and nonlinear regression methods in the forecasting engine are time-series regression methods that use historical data from a single measure.

Note: Oracle Data Mining supports both time series and causal analysis methods for data stored in a relational format. This type of forecasting is done using the SQL `PREDICTION` function within a Data Mining model. Refer to *Oracle Data Mining Concepts*.

Expert Opinion

As the time horizon for the forecast moves further out into the future, expert opinion becomes the most reliable predictor. The experts, who are usually corporate executives, have their fingers on the pulse of myriad factors that may influence future performance, such as the general direction of the market and plans for new products. Customer surveys also provide input to long-term forecasts. An equivalent computer model to rival expert opinion for long-term forecasts would be too complex to generate within a usable time frame.

About the Forecasting Engine

Oracle OLAP incorporates a statistical forecasting engine that is used extensively for demand planning applications. This engine has a variety of time-series forecasting methods, which are described in "Forecasting Method Descriptions" on page 7-9.

The forecasting engine incorporates advanced filtering technology to identify and process outliers, which are data values that are extremely high or low in relation to the mean. Exception handing is a critical component of forecasting efficiency, and the forecasting engine reduces the time and money spent analyzing exceptions. This technology also enables the forecasting engine to produce accurate short-term forecasts using wildly fluctuating historical data.

Typical applications for OLAP forecasting include the following:

- Distribution requirements planning for seasonal monthly forecasts of retail sales for products reaching market saturation.
- Business planning with seasonal quarterly forecasts of expenses with upward linear trends.
- Sales quota management by forecasting exponential decay in company sales for aging products.
- Materials requirement planning with trends in raw material prices with cyclical behavior.
- Sales forecasts with exponential growth in industry sales.
- Inventory control planning by forecasting S-curve demand growth from increasing distribution.

Creating a Forecast

You can create forecasts in Analytic Workspace Manager by defining a forecast step in a Calculation Plan. These are the steps for creating a forecast. Each one is discussed in more detail in the sections that follow.

1. Creating the Forecast Time Periods
2. Creating a Forecast Measure
3. Selecting the Historical Data
4. Identifying the Levels for the Forecast
5. Creating a Forecast Step
6. Generating the Forecast Data
7. Evaluating the Forecast Results

Creating the Forecast Time Periods

The future time periods that you want to forecast must be defined as members of the time dimension in your analytic workspace. If they do not exist already, you must:

1. Add the new time periods and attributes to the relational tables in the source schema.
2. Use the Maintenance Wizard in Analytic Workspace Manager to add the new members to the Time dimension in the analytic workspace.

Use whatever mechanism guarantees that these Time dimension members are identical to those for loading actual data at a later date.

Creating a Forecast Measure

You can store the forecast data in the same measure as the actual data, or you can store it in a separate measure. If you store the forecast in the same measure, then the actual data eventually overwrites it. This prevents you from monitoring the accuracy of the forecast. For this reason, you should create a separate forecast measure in the same cube as the source measure.

To create a forecast measure:

1. In the navigation tree, expand the cube for the actual data.
2. Right-click Measures and choose **Create Measure**.
3. Complete the Create Measure property sheet. Do not map the measure to a data source.

Selecting the Historical Data

The forecasting engine needs only a year of data to detect trends and seasonality. Business cycles may take two or three years of data to detect.

If your business has experienced a paradigm shift, then you should exclude previous data from your forecast as irrelevant. The following are examples of events that might cause a paradigm shift:

- Cellular telephones on the telecommunications industry
- Digital cameras on the photography industry
- The Internet on the book and music publishing industries

You select the historical data when creating the forecast step.

Identifying the Levels for the Forecast

To generate consistent data at all levels of a hierarchy, you must generate the forecast data at a single level and use it to populate the other forecast levels by aggregation or allocation. If you generate a forecast from multiple levels, then the aggregate forecast data may be inconsistent with the lower levels of forecast data.

The "correct" levels are determined by the time frame of your forecast and by your reasons for making the forecast. For example, you may forecast Customers at the Total level for manufacturing, but at a lower level for marketing. Table 7–2 shows the recommended dimension levels for forecasting products over various time frames.

If you set the levels too low, then large variations in the data may decrease accuracy. These inaccuracies may be magnified in the aggregated forecasts. If you set the levels too high, then the aggregated forecasts may smooth out localized trends and allocate them incorrectly. You select the levels when creating the forecast step.

Table 7–2 Example of Dimension Levels for Forecasts

Time Frame	Time Level	Product Level	Other Dimension Levels
Short	Week, Biweek, or Month	UPC, SKU, NDC, ISBN	Level of interest
Medium	Month or Quarter	Brand	Level of interest
Long	Quarter or higher	Brand, Company, Market	Level of interest

Creating a Forecast Step

To create a forecast step in Analytic Workspace Manager:

1. In the navigation tree, create a new Calculation Plan or open an existing plan.
2. On the General tab of the Calculation Plan, click **New Step**, then select **New Forecast Step**.

   The New Forecast Step property pages are displayed.
3. Complete the General page. For the forecast method, select **Automatic**.

   For information about using other methods, refer to "Designing Your Own Forecast" on page 7-7. For information about completing the other fields, click **Help**.
4. Keep the default values on the Advanced Settings page unless you have expertise in time-series forecasting.
5. On the Status page, select the historical time periods and other dimension values to use as the basis for the forecast. Select only one level for each dimension.
6. Save the forecast step, then save the Calculation Plan.
7. Create allocation and aggregation steps for the forecast.

Generating the Forecast Data

If all the time periods and data are already loaded into the analytic workspace, then right-click the Calculation Plan and choose **Execute Calculation Plan**.

or

If you must load new data, then add the Calculation Plan to the regular maintenance process using the Maintenance Wizard.

Afterward, you can view the forecast data in the Measure Viewer.

Evaluating the Forecast Results

If the forecast does not initially look plausible to you, then check that there are no errors in the design of the forecast:

- Compare the first few forecast periods to the last few historical periods to verify that a discrepancy exists.
- Use the forecast step editor to check the number of forecast periods against the status of the Time dimension. The forecast periods are the last ones in status. For example, if the Time dimension has dimension members defined through the next

five months and you designed a 4-month forecast, then you must remove the last month from status. Otherwise, the forecast is based on a month of null historical data.

- Use the Measure Viewer to verify that all of the historical data has been loaded in the source measure. If several periods immediately prior to the forecast period are not loaded, then the forecast is 0.
- If you used a specific forecasting method (not Automatic):
  - Compare its results with those of the Automatic option.
  - Verify that you set Forecast Approach to Manual and Data Filter to the appropriate choice.
- If you set any of the advanced parameters, then compare the results against a forecast that uses the default settings.

A standard part of forecasting is to continually monitor the accuracy of the forecast data. The easiest way to compare the forecast data with the actual data is to set up a standard report that includes a line graph. Then you can see how closely the forecast data converges with the actual data.

Short-term forecasts should be fairly precise, with only a small difference between forecast and actual data. If this is not the case, then you should consider modifying the forecast using some of the suggestions listed previously. You may even want to create several forecasts and compare their results over time.

Medium- and long-range forecasts generated by time-series forecasting methods should be qualified by other input, such as expert opinion, because external factors affect performance in these time frames.

Designing Your Own Forecast

The OLAP forecast engine provides an Expert System that generates the best short-term forecasts over the long run, so you should use the Automatic method and the default parameters for most forecasts. However, there may be times when you should override the Expert System and design the forecast yourself.

What is the Expert System?

The Expert System supports the Automatic method by identifying the best statistical method and selecting the best parameter settings for your data. It also distinguishes outliers from factors like trend and seasonality.

The Expert System separates the data into seasonal effects and trend effects. It then uses an iterative approximation method to forecast the seasonal component of the data. After completing the trend forecast, it factors the seasonality portion into the trend forecast for all methods except Holt-Winters, which calculates its own seasonal factors.

The Expert System represents a type of artificial intelligence for statistical forecasting that has been in common use ever since computers took over the task of performing complex and lengthy numerical calculations. Instead of the analyst's having to evaluate the data and make an educated guess as to the best method, the software can quickly try all methods and select the best one based on the results.

You can override the Expert System by setting the Forecast Approach parameter to Manual. The default value of Automatic gives the Expert System the most control in overriding your choices. This is the appropriate setting when using the Automatic method, but it invalidates your attempt to design a forecast.

What is the Verification Window?

The Expert System always tests the accuracy of a forecast method using a portion of the historical data called a verification window. For the Automatic method, the Expert System uses this window to select the best statistical method. For the other methods, it verifies that your selection of a method and the parameter settings provide a good fit to the historical data.

For this test, the Expert System divides the historical time periods into two groups. The older time periods retain their role as historical data. The newer historical time periods become the "forecast" periods and form the verification window. The Expert System generates forecast data for the newer time periods, using the older time periods as the basis for the forecast.

The Expert System calculates the precision of the method by comparing the forecast data to the actual data in the verification window. The precision is the distance between the forecast data and the actual data.

The Expert System uses several standard calculations to compare the precision of different forecasting methods: Mean Absolute Deviation (MAD), Mean Absolute Percentage Error (MAPE), and Root Mean Square Error (RMSE).

When Should You Design a Forecast?

You may want to control the forecast when you have special knowledge that future performance will deviate from past results.

For example, you may recently have entered an agreement for a major national chain of stores to carry your products, so you anticipate a dramatic increase in sales. Or your company might have been an innovator in developing a new product line, but your competitors are about to introduce rival products. In this case, you expect sales to level off. You or your competitors might also be negotiating a corporate merger, and you expect that transaction to affect performance.

Under circumstances like these, your special knowledge may enable you to design a more accurate forecast than the Expert System.

Overriding the Expert System

To override the Expert System, take these steps:

1. Create or edit a forecast step, as described in "Creating a Forecast Step" on page 7-6.
2. On the General page, select the method that best describes the future performance that you expect, based on your expert knowledge.
3. On the Advanced Settings page, set **Forecast Approach** to **Manual**.
4. Set the Data Filter parameter to an appropriate setting for your data.
5. Change the Verification Window Size parameter as desired.
6. Make whatever other changes to the parameter settings are appropriate.
7. Complete the definition of the forecast, and run it as described in "Creating a Forecast" on page 7-4.

Forecasting Method Descriptions

The forecasting methods represent several basic approaches to time-series forecasting. This topic provides descriptions of the various approaches, the methods that use each approach, and the optimization parameters that apply specifically to them.

Automatic

The Expert System identifies the best fit by quickly testing each statistical method against the portion of historical data specified by the Verification Window Size parameter. The Expert System selects the method and the parameter settings that would have generated the most accurate forecast in the past. It automatically detects and handles outliers, removing noise so that it can better detect trends and seasonality.

The forecasting engine generates a forecast for every combination of dimension members. The Expert System evaluates each forecast separately and picks the best method and parameter settings for each one.

In general, Automatic is the best choice unless you have knowledge that future performance will deviate from the past. Under these special circumstances, you can substitute your own expert judgment for the Expert System.

"What is the Verification Window?" on page 7-8 provides more information about how the Expert System selects a method.

Regressions

Time series regression methods relate a variable (measure) to functions of time describing trend and seasonal components. Regression generates the most reliable forecasts when the trend or seasonal components remain constant.

OLAP forecasting provides both linear and nonlinear regression models.

Linear Regression

Linear regression attempts to fit the historical data to a straight line (y=ax+b), and extends that line into future time periods for the forecast. All data points have equal weight. This method identifies steady, long-term trends in the data.

Nonlinear Regression

Nonlinear regression attempts to fit the historical data to a curve, and extrapolates that curve into the forecast time periods. All data points have equal weight. The curved lines are defined by mathematical equations. You can choose from the following types of curves:

- **Polynomial Fit**: Fits data that fluctuates with a rise and a drop (x'=log(x); y'=log(y)).
- **Exponential Fit**: Fits data points that rise or drop at an increasingly faster rate (x'=x; y'=ln(y)).
- **Logarithmic Fit**: Fits data points that rise or drop quickly and then level off (x'=log(x); y'=y).
- **Asymptotic Fit**: Fits data points that rise or drop until they approach a fixed value and then level off (x'=1/x; y'=1/y).
- **Exponential Asymptotic Fit**: Fits data points that rise or drop at an increasingly faster rate until they approach a fixed value and then level off (x'=x; y'=ln(y/(K-y))).

For more information about the equations used by each method, refer to the topic "Equations for Forecasting Methods" in Analytic Workspace Manager Help.

Advanced Parameter for Regressions

The Cyclical Decay smoothing constant is used in the equations for linear and nonlinear regression. This constant determines how quickly a cycle reverts to the mean. A higher value implies slower decay while a lower value implies faster decay. The smaller the value, the less effect cyclical activity has on the forecast.

You can specify a maximum value and a minimum value. You can specify the same value for both the maximum and the minimum. Keep the default settings unless you have a strong background in time-series forecasting.

Exponential Smoothing

The exponential smoothing methods weight the historical data using exponentially decreasing weights. The prior period has the most weight and each period prior to it has comparatively less weight. The decline in weight is expressed mathematically as an exponential function. The smoothing parameters determine the weights.

Comparison Among Exponential Smoothing Methods

You can choose from the following methods of exponential smoothing:

- **Single Exponential Smoothing**: Identifies the percentage of weight given to the prior period and all other historical periods. It does not adjust for trend or for seasonal variance.
- **Double Exponential Smoothing**: Identifies the trend, and adjusts the forecast data to reflect this trend instead of generating a single parameter for all forecast periods.
- **Holt-Winters**: Identifies both trend and seasonal variance, and adjusts the forecast data to reflect these factors. This method is particularly sensitive to both high and low outliers. A better choice for handling seasonality is Double Exponential Smoothing with the Data Filters parameter set to Seasonal Adjustment.

Advanced Parameters for Exponential Smoothing

These smoothing constants are used in the equations for exponential smoothing methods. Keep the default settings unless you have a strong background in time-series forecasting.

- **Alpha**: Determines how responsive a forecast is to sudden jumps and drops. It is the percentage weight given to the prior period, and the remainder is distributed to the other historical periods. Alpha is used in all exponential smoothing methods.

  The lower the value of alpha, the less responsive the forecast is to sudden change. A value of 0.5 is very responsive. A value of 1.0 gives 100% of the weight to the prior period, and gives the same results as a prior period calculation. A value of 0.0 eliminates the prior period from the analysis.
- **Beta**: Determines how sensitive a forecast is to the trend. The smaller the value of beta, the less weight is given to the trend. The value of beta is usually small, because trend is a long-term effect. Beta is not used in Single Exponential Smoothing.

- **Gamma**: Determines how sensitive a forecast is to seasonal factors. The smaller the value of gamma, the less weight is given to seasonal factors. Gamma is used only by the Holt-Winters method.
- **Trend Dampening**: Determines how sensitive the forecast is to large trends in recent time periods. Dampening identifies how quickly the trend reverts to the mean. A higher value implies slower dampening while a lower value implies faster dampening. The smaller the value, the less effect the trend has on the forecast.

For each constant, you can specify a maximum value, a minimum value, and an interval. The interval is an incremental value between the maximum and minimum, which the forecasting engine uses to find the optimal value of the constant.

Advanced Parameter Descriptions

Following are descriptions of the advanced parameters that can be used with all methods.

Parameters that are specific to a particular approach are described in "Forecasting Method Descriptions" on page 7-9.

Note: When using a specific forecasting method (not Automatic), be sure to set the following parameters:

- Forecast Approach
- Data Filter

Setup Parameters

These parameters provide the forecasting engine with basic information about how you want it to approach a forecast. Always set the Forecast Approach and Data Filter parameters when using a specific forecasting method.

- **Forecast Approach**: Specifies whether the forecasting engine gives control to the Expert System.
  - **Automatic**: Give control to the Expert System. Use this setting with the Automatic method.
  - **Manual**: Give control to the user. It enables you to choose a method and set the parameters that are appropriate for the historical data. Use this setting with all methods other than Automatic.
- **Data Filter**: Identifies a basic characteristic of the data.
  - **Non-Seasonal Data**: No seasonality.
  - **Seasonal Data**: Adjust for seasonal patterns in the data. You can use this filter with Double Exponential Smoothing to get a more accurate forecast than Holt-Winters.
  - **Intermittent Data** Adjusts for sporadic or intermittent data and, if appropriate, seasonal patterns. Intermittent data has null or zero for over 50% of the values. Do not use median smoothing with this filter, because smoothing eliminates the intermittent characteristic of the data. The purpose of the intermittent data filter is to forecast intermittent demand.

    Set the Moving Total Decay parameter when using this filter.

- **Verification Window Size**: The Expert System uses the verification window to determine the best method and parameter settings, as described in "What is the Verification Window?" on page 7-8.

  The verification window is specified as a fraction of the total number of historical periods. For example, assume that you have three years of historical data for 2004, 2005, and 2006. The default window size is .3333, so the Expert System uses 1/3 of the historical data for the verification window. Thus, the data for 2004 and 2005 are used to generate a "forecast" for 2006. The difference between the forecast data and the actual data for 2006 indicates the precision of the method.

  You may want to adjust the window size, depending on the granularity of the data. For monthly data, use a window size of 20% (1/5) or more. For weekly data, use a window size of 12.5% (1/8) or more. For daily or hourly data, you can use a window size of 11.1% (1/9) or less.

General Parameters

These parameters apply to all of the specific forecasting methods.

- **Allocate Last Cycle**: Controls whether the last cycle is calculated by forecasting alone or with allocation. Allocation may reduce the risk of overadjustment for trend or seasonality.

  Allocation forecasts an average value for one period of the last cycle. That average value is then multiplied by factors to give the remaining points in that period. For example, a forecast at the day level would calculate an average for all days in the last week rather than forecasting individual days.

  Set Periodicity to a value greater than 1 when using this parameter.

- **Boundary Maximum and Minimum**: A constant that constrains the forecasting engine from occasionally generating unreasonably high or low values. The upper boundary is calculated by multiplying Boundary Maximum by the largest value in the historical series. The lower boundary is calculated by multiplying Boundary Minimum by the smallest value in the historical series.

  For example, if the Boundary Maximum parameter is 100.0 and the largest historical value 5,600, then no forecast value can be greater than 560,000. If the Boundary Minimum parameter is 0.5 and the smallest historical value 300, then no forecast value can be less than 150.

- **Moving Total Decay Maximum and Minimum**: A constant that is inversely related to noise, random deviation, and stability in the history of intermittent data. Set this value higher when the history is evolving rapidly from one cycle to the next or when the noise level is low. This parameter is used only with the Intermittent Data filter. The difference between the maximum and the minimum must be evenly divisible by 0.4.

- **Periodicity**: The number of periods in a single cycle or the number of periods in each set of nested cycles. The default value of 1 does not group the periods at all, so each period is logically independent.

  For example, if you are using Month as the base level for the forecast, and the time hierarchy has levels for Month, Quarter, and Year, then the cycles are 12 months in a year and 3 months in a quarter. For a single cycle, enter the number of periods. For nested cycles, list the cycles in parentheses from the most aggregate to the least aggregate, separated by commas, such as (12,3).

- **Trials**: The number of trials that are run to determine the best method and combination of parameter settings.

Historical Data Smoothing Parameters

These parameters help generate a smoother forecast from intermittent historical data. Alternatively, you can use the intermittent data filter to forecast intermittent demand. Do not combine the smoothing parameters with the intermittent demand filter, because these adjustments are contradictory.

- **Use Smoothed Historical Data**: Controls whether the historical data is smoothed. Smoothing is typically used for weekly or finer-grained data that has many missing values. Smoothing the historical data produces a smoother baseline forecast.
- **Interpolate Missing Values**: Specifies whether you want to smooth the data by inserting estimates for missing values instead of by averaging. This parameter is useful when missing values indicate incomplete data instead of a lack of activity.
- **Median Smoothing Window**: The number of time periods used in a median smoothing window to identify outliers and replace them with adjusted data values. Median smoothing eliminates extreme variations in the data by replacing each data point in a series by the median value of itself and its neighbors. This setting must be an odd number, so that the current time period is in the center of the window.

  The larger the window, the smoother the data. If the window is too large, smoothing may eliminate important data patterns. If the window is too small, then smoothing may include outliers that could not be filtered out. As a rule, you should not set this parameter below 3; setting it to 1 has the effect of turning off smoothing.

  For monthly data, use a maximum value of 5 to prevent excessive flattening of the data. For weekly data, use a maximum of 13. Use a longer window (15 or more) for daily or hourly data.

Case Study: Forecasting Sales for Global Enterprises

The `GLOBAL` analytic workspace has historical data from January 1998 to July 2004. Thus, the last five months of 2004 and all of 2005 is NA. This example creates a Calculation Plan that generates a four-month Sales forecast from August 2004 to December 2005. An allocation step distributes the forecast data down to the base levels of all dimensions. An aggregation step generates and stores some of the aggregate values to improve runtime performance.

Creating the Sales Forecast Target Measure

This example stores the forecast data in a separate measure from the historical data so that the results of the forecast can be evaluated more easily.

To create the target measure:

1. In the `UNITS_CUBE` folder, right-click Measures and select **Create Measure**.

   The Create Measure dialog box opens.
2. On the General page, create a measure named `SALES_FORECAST`.
3. Select **Override the Aggregation Specification of the Cube.**
4. On the Summarize To page, deselect all levels for all dimensions.
5. Click **Create**.

Creating the Calculation Plan

The Calculation Plan for this forecast has a forecast step, an allocation step, and an aggregation step.

To create a new Calculation Plan:

1. Right-click Calculation Plans and select **Create Calculation Plan**.

   The Create Calculation Plan dialog box opens.

2. Create a new plan named `SALES_PLAN`. Click **Create**.

   `SALES_PLAN` appears as a new item in the Calculation Plans folder. It does not yet contain any steps.

Creating the Sales Forecast Step

This sample forecast uses the Automatic forecast method, which takes the guesswork out of choosing an appropriate forecast method.

To create the forecast step:

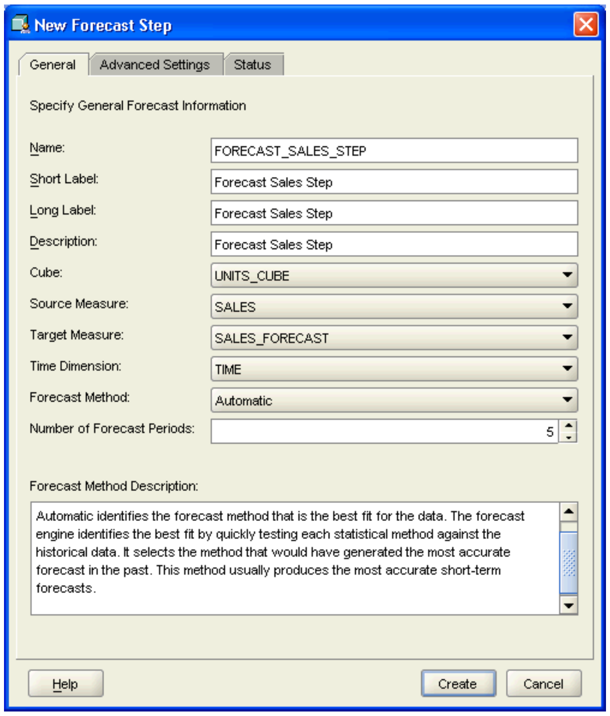
1. On the General page of `SALES_PLAN`, click **New Step**, then select **New Forecast Step**.

   The Create Forecast Step dialog box opens.

2. Complete the General page with these values, as shown in Figure 7–1

   - Name: `forecast_sales_step`
   - Cube: `UNITS_CUBE`
   - Source Measure: `SALES`
   - Target Measure: `SALES_FORECAST`
   - Time Dimension: `TIME`
   - Forecast Method: `Automatic`
   - Number of Forecast Periods: `5`

Figure 7–1 Forecasting Global Sales



Specify General Forecast Information

Name:	FORECAST_SALES_STEP
Short Label:	Forecast Sales Step
Long Label:	Forecast Sales Step
Description:	Forecast Sales Step
Cube:	UNITS_CUBE
Source Measure:	SALES
Target Measure:	SALES_FORECAST
Time Dimension:	TIME
Forecast Method:	Automatic
Number of Forecast Periods:	5

Forecast Method Description:

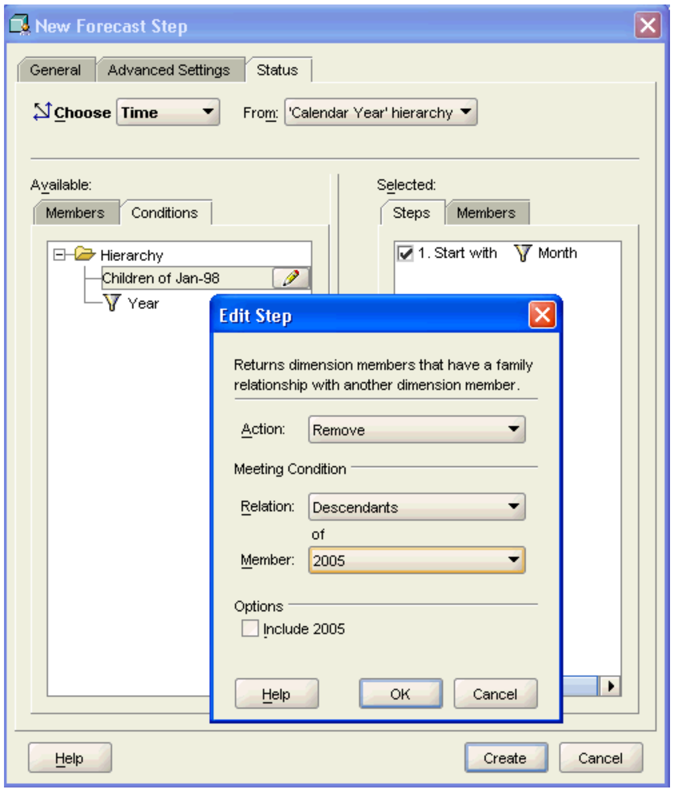
Automatic identifies the forecast method that is the best fit for the data. The forecast engine identifies the best fit by quickly testing each statistical method against the historical data. It selects the method that would have generated the most accurate forecast in the past. This method usually produces the most accurate short-term forecasts.

3. Keep the default settings on the Advanced Parameters page.
4. On the Status page, set the Time dimension:
   a. On the Selected Steps tab, click **All Levels** and select **Month** from the drop-down list.
   b. On the Available Conditions tab, expand the Hierarchy folder. Select **Children of Jan-98** and click the Edit Step icon.

      The Edit Step dialog box opens, as shown in Figure 7–2.
   c. Set Action to **Remove**, and set Relation to **Descendants**.
   d. Click **Member** and choose **More** from the list.

      The Select Members dialog box opens.
   e. Select **2005**.
   f. Click **OK** to close the Select Members dialog box, then click **OK** to close the Edit Step dialog box.
   g. Add this condition to the Selected Steps.
   h. On the Members tab, verify that only months are in the list and **Dec-04** is the last value.

Figure 7–2 Selecting Time Dimension Members



Edit Step

Returns dimension members that have a family relationship with another dimension member.

Action: Remove

Meeting Condition

Relation: Descendants

of

Member: 2005

Options

Include 2005

5. Keep the default selection, which is the top level, for the other dimensions.
6. Click **Create** to save the forecast step.
7. Click **Apply** to save the Calculation Plan.

Generating the Forecast

To generate the forecast:

1. Expand the Calculation Plans folder. Right-click **SALES_PLAN** and choose **Execute Calculation Plan SALES_PLAN**.

   The Maintenance Wizard opens, and **SALES_PLAN** is a selected target object.

2. Click **Finish**.

   The build log is displayed when the Calculation Plan is done executing.

Validating the Forecast

This forecast generated values at the base level of Time and at the top level of all other dimensions. To view the forecast data and evaluate whether you are satisfied with these results, you must select this particular portion of the data and compare it to historical results.

1. Fully expand the **UNITS_CUBE** folder, right-click the **SALES_FORECAST** measure, and choose **View Data SALES_FORECAST**.

   The Measure Data Viewer opens. No data is displayed, because the base levels for Product, Customer, and Channel are `NA`.

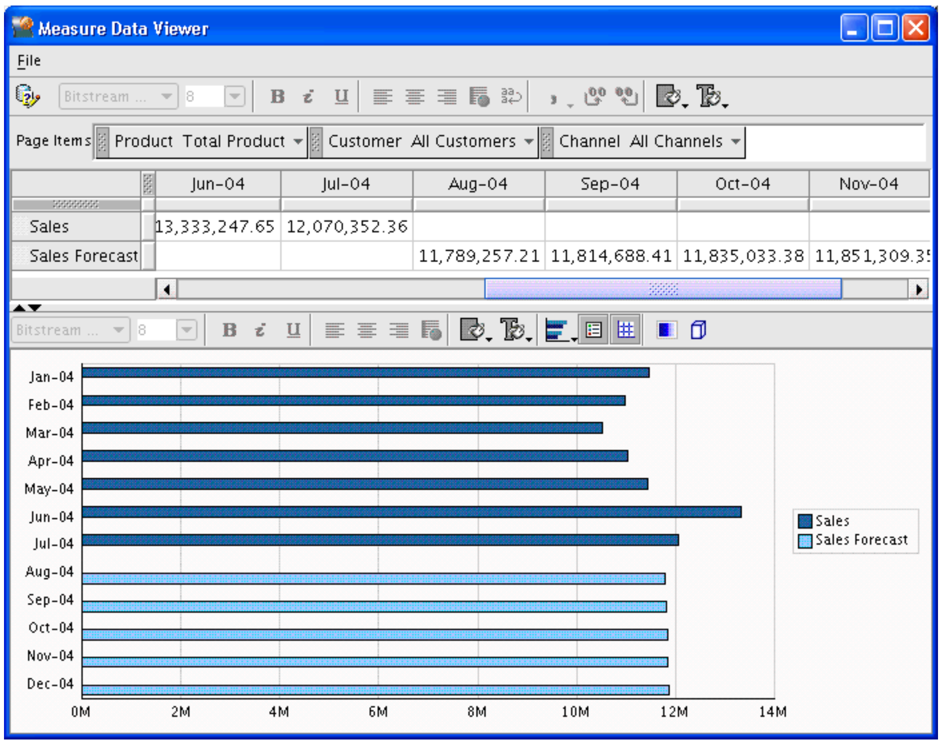
2. From the File menu, choose **Query Builder**.

The Query Builder opens.

3. On the Items tab, add Sales to the Selected list.
4. On the Layout tab, arrange the dimensions so that Measure identifies the rows and Time identifies the columns. Click **Help** for instructions.
5. On the Dimensions tab, set the status of Time:
   a. On the Steps tab, remove the initial selection.
   b. On the Conditions tab, expand the Hierarchy folder.
   c. Change **Children of 1998** to **Children of Q3-04, Q4-04**, and add this condition to the Selected Steps.
   d. On the Members tab, verify that only months are in the list from **Jul-04** to **Dec-04**.
6. Click **OK** to close the Query Builder.

Figure 7–3 shows the results of the forecast, which are displayed in the Measure Viewer.

Figure 7–3 Forecast Data Displayed in the Measure Viewer



Page Items: Product Total Product | Customer All Customers | Channel All Channels

	Jun-04	Jul-04	Aug-04	Sep-04	Oct-04	Nov-04
Sales	13,333,247.65	12,070,352.36				
Sales Forecast			11,789,257.21	11,814,688.41	11,835,033.38	11,851,309.3

Preparing the Sales Forecast Measure for Querying

This sample forecast only generated five data values, which is only a small slice of the Sales Forecast measure. Before users can query this measure, you must generate additional data from the forecast values:

1. Allocate the forecast data down the Product, Customer, and Channel dimensions.

   The forecast generated a single, top-level value for these dimensions. For the Time dimension, the forecast generated data at the detail Month level, so allocation is not possible.

2. Precompute some of the aggregate values to improve querying performance.

   The measure is not mapped to a data source, so a refresh of the cube does not aggregate a forecast measure. The data is entirely aggregated on the fly unless you precompute some of the values in a separate step. A forecast measure can use the same rules as the other measures in the cube, but you must specify the rules in an Aggregation Step.

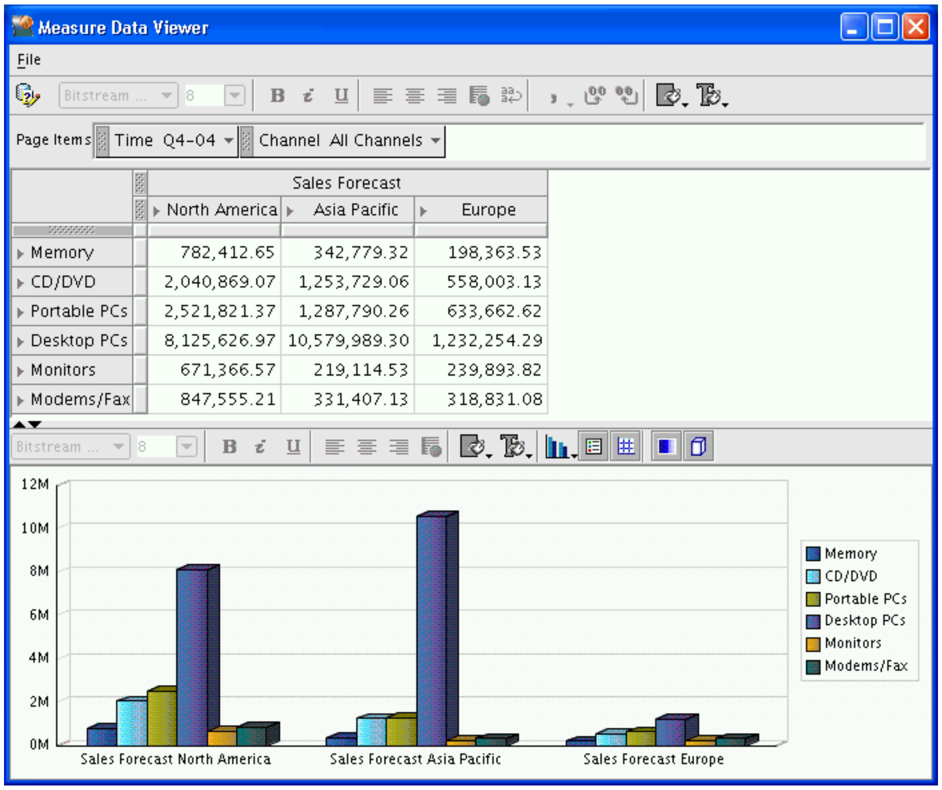
To prepare the forecast data:

1. Add an Allocation Step to the Calculation Plan.
2. Add an Aggregation Step.

   Refer to "Case Study: Allocating a Sales Forecast" on page 9-13.
3. Run the Maintenance Wizard as described in "Generating the Forecast Data" on page 7-6.

Figure 7–4 shows data in the middle levels of Time, Product, and Customer after allocation and aggregation of the forecast data.

Figure 7–4 Allocated Forecast Data



	Sales Forecast		
	North America	Asia Pacific	Europe
Memory	782,412.65	342,779.32	198,363.53
CD/DVD	2,040,869.07	1,253,729.06	558,003.13
Portable PCs	2,521,821.37	1,287,790.26	633,662.62
Desktop PCs	8,125,626.97	10,579,989.30	1,232,254.29
Monitors	671,366.57	219,114.53	239,893.82
Modems/Fax	847,555.21	331,407.13	318,831.08

8 Advanced Aggregations

A cube always returns summary data to a query as needed. While the cube may store data at the day level, for example, it returns a result at the quarter or year level without requiring a calculation in the query. This chapter explains how to optimize the unique aggregation subsystem of Oracle OLAP to provide the best performance for both data maintenance and querying.

This chapter contains the following topics:

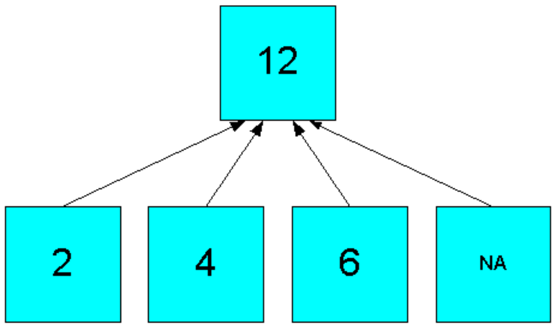
What is Aggregation?

Aggregation is the process of consolidating multiple values into a single value. For example, data can be collected on a daily basis and aggregated into a value for the week, the weekly data can be aggregated into a value for the month, and so on. Aggregation allows patterns in the data to emerge, and these patterns are the basis for analysis and decision making. When you define a data model with hierarchical dimensions, you are providing the framework in which aggregate data can be calculated.

Aggregation is frequently called summarization, and aggregate data is called summary data. While the most frequently used aggregation operator is Sum, there are many other operators, such as Average, First, Last, Minimum, and Maximum. Oracle OLAP also supports weighted and hierarchical methods. Following are some simple diagrams showing how the basic types of operators work. For descriptions of all the operators, refer to "Aggregation Operators" on page 8-3.

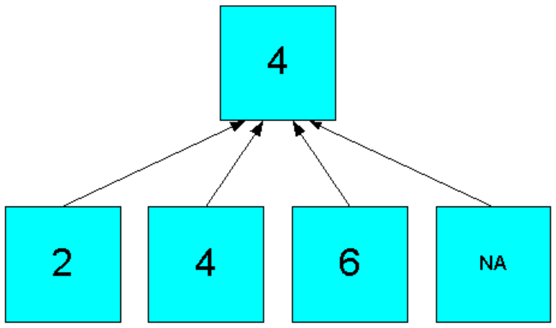
Figure 8–1 shows a simple hierarchy with four children and one parent value. Three of the children have values, while the fourth is empty. This empty cell has a null or `NA` value. The Sum operator calculates a value of (2 + 4 + 6)=12 for the parent value.

Figure 8–1 Summary Aggregation in a Simple Hierarchy



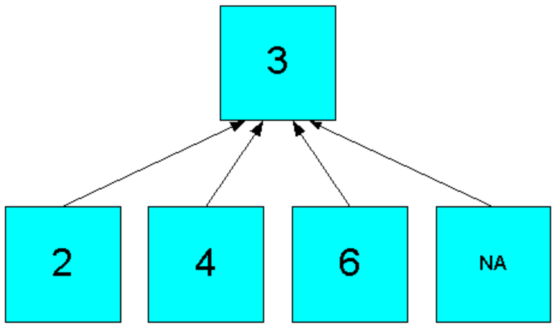
The Average operator calculates the average of all real data, producing an aggregate value of ((2 + 4 + 6)/3)=4, as shown in Figure 8–2.

Figure 8–2 Average Aggregation in a Simple Hierarchy



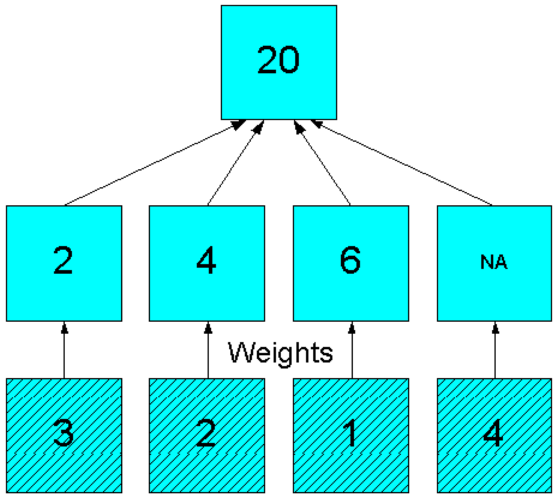
The hierarchical operators include null values in the count of cells. In Figure 8–3, the Hierarchical Average operator produces an aggregate value of ((2 + 4 + 6 +NA)/4)=3.

Figure 8–3 Hierarchical Average Aggregation in a Simple Hierarchy



The weighted operators use the values in another measure to generate weighted values before performing the aggregation.Figure 8–4 shows how the simple sum of 12 in Figure 8–1 changes to 20 by using weights ((3*2) + (2*4) + (NA*6) +(4*NA)).

Figure 8–4 Weighted Sum Aggregation in a Simple Hierarchy



Aggregation Operators

Analytic workspaces provide an extensive list of aggregation methods, including weighted, hierarchical, and weighted hierarchical methods.

Basic Operators

The following are descriptions of the basic aggregation operators:

- **Average**: Adds non-null data values, then divides the sum by the number of data values that were added together.
- **First Non-NA Data Value**: Returns the first real data value.
- **Last Non-NA Data Value**: Returns the last real data value.
- **Maximum**: Returns the largest data value among the children of each parent.
- **Minimum**: Returns the smallest non-null data value among the children of each parent.
- **Nonadditive**: Does not aggregate the data.
- **Sum**: Adds data values.

Scaled and Weighted Operators

These are the scaled and weighted aggregation operators.

These operators require a measure providing the weight or scale values in the same cube. In a weight measure, an NA (null) is calculated like a `1`. In a scale measure, an NA is calculated like a `0`.

The weighted operators use outer joins, as described in "When Does Aggregation Order Matter?" on page 8-4.

- **Scaled Sum**: Adds the value of a weight object to each data value, then adds the data values.
- **Weighted Average**: Multiplies each data value by a weight factor, adds the data values, and then divides that result by the sum of the weight factors.

- **Weighted First**: Multiplies the first non-null data value by its corresponding weight value.
- **Weighted Last**: Multiplies the last non-null data value by its corresponding weight value.
- **Weighted Sum**: Multiplies each data value by a weight factor, then adds the data values.

Hierarchical Operators

The following are descriptions of the hierarchical operators. They include all cells identified by the hierarchy in the calculations, whether the cells contain data or not .

Hierarchical Average and the Hierarchical Weighted operators use outer joins.

- **Hierarchical Average**: Adds data values, then divides the sum by the number of the children in the dimension hierarchy. Unlike Average, which counts only non-null children, hierarchical average counts all of the logical children of a parent, regardless of whether each child does or does not have a value.
- **Hierarchical First Member**: Returns the first data value in the hierarchy, even when that value is null.
- **Hierarchical Last Member**: Returns the last data value in the hierarchy, even when that value is null.
- **Hierarchical Weighted Average**: Multiplies non-null child data values by their corresponding weight values, then divides the result by the sum of the weight values. Unlike Weighted Average, Hierarchical Weighted Average includes weight values in the denominator sum even when the corresponding child values are null.
- **Hierarchical Weighted First**: Multiplies the first data value in the hierarchy by its corresponding weight value, even when that value is null.
- **Hierarchical Weighted Last**: Multiplies the last data value in the hierarchy by its corresponding weight value, even when that value is null.

When Does Aggregation Order Matter?

The OLAP engine aggregates a cube across one dimension at a time. When the aggregation operators are the same for all dimensions, the order in which they are aggregated may or may not make a difference in the calculated aggregate values, depending on the operator.

You should specify the order of aggregation when a cube uses multiple aggregation methods. The only exceptions are that you can combine Sum and Weighted Sum, or Average and Weighted Average, when the weight measure is only aggregated over the same dimension. For example, a weight measure used to calculate weighted averages across Customer is itself only aggregated across Customer.

The weight operators are uncompressible for the specified dimension and all preceding dimensions. For a compressed cube, you should list the weighted operators as early as possible to minimize the number of outer joins. For example, suppose that a cube uses Weighted Sum across Customer, and Sum across all other dimensions. Performance is best if Customer is aggregated first.

Using the Same Operator for All Dimensions of a Cube

The following information provides guidelines for when you must specify the order of the dimensions as part of defining the aggregation rules for a cube.

Order Has No Effect

When these operators are used for all dimension of a cube, the order does not affect the results:

- Maximum
- Minimum
- Sum
- Hierarchical First Member
- Hierarchical Last Member
- Hierarchical Average

Order Changes the Aggregation Results

Even when these operators are used for all dimensions of a cube, the order can affect the results:

- Average
- First Non-NA Data Value
- Last Non-NA Data Value
- Weighted First
- Weighted Last
- Hierarchical Weighted First
- Hierarchical Weighted Last
- Scaled Sum

Order May Be Important

When the following weighted operators are used for all dimensions of a cube, the order affects the results only if the weight measure is aggregated over multiple dimensions:

- Weighted Average
- Weighted Sum
- Hierarchical Weighted Average

Example: Mixing Aggregation Operators

Even though you can use the Sum and Maximum operators alone without ordering the dimensions, you cannot use them together without specifying the order. The following figures show how they calculate different results depending on the order of aggregation. Figure 8–5 shows a cube with two dimensions. Sum is calculated first across one dimension of the cube, then Maximum is calculated down the other dimension.

Figure 8–5 Sum Method Followed by Maximum Method

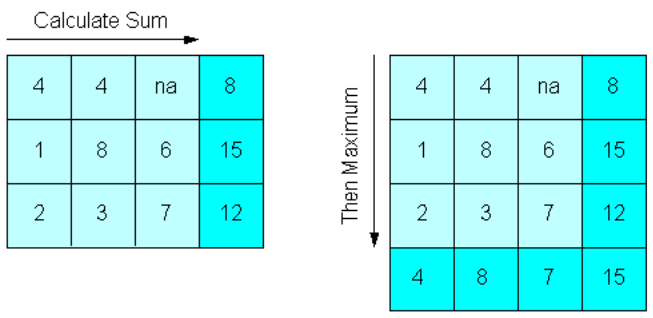
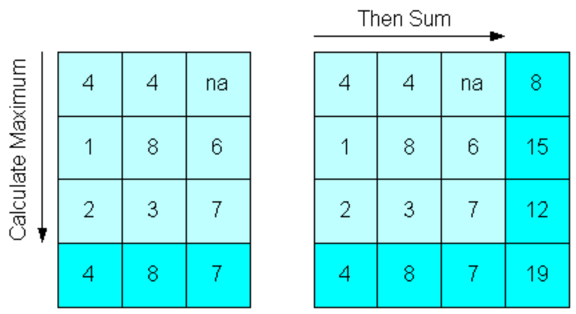


Figure 8–6 shows the same cube, except Maximum is calculated first down one dimension of the cube, then Sum is calculated across the other dimension. The maximum value of the sums in Figure 8–5 is 15, while the sum of the maximum values in Figure 8–6 is 19.

Figure 8–6 Max Method Followed by Sum Method



Aggregating Compressed Cubes

Compressed composites are used to store extremely sparse data. Use this aggregation strategy for compressed cubes:

- Identify the dimension with the most members. If several dimensions have about the same number, then choose the dimension with the most levels. Do not pre-aggregate this dimension.
- Pre-aggregate all other dimensions up to, but not including, the top level, unless the next level down has a large number of members.

You can adjust these basic guidelines to the particular characteristics of your data. For example, you may skip levels that are seldom queried from pre-aggregation. Or you may need to pre-aggregate a level with a large number of child values, to provide acceptable run-time performance.

Aggregating Uncompressed Cubes

Uncompressed cubes are used to store data that is either moderately sparse or dense. The strategy for aggregating noncompressed cubes is called skip-level aggregation, because some levels are stored and others are skipped until runtime. The success of this strategy depends on choosing the right levels to skip, which are those that can be calculated quickly in response to a query.

Selecting Dimensions for Skip-Level Aggregation

As a general rule, you should skip levels for only one or two dimensions and for no more than half of the dimensions of the cube. Choose the dimensions with the most levels in their hierarchies for skip-level aggregation.

Slower varying dimensions take longer to aggregate because the data is scattered throughout its storage space. If you are optimizing for data maintenance, then fully aggregate the faster varying dimensions and use skip-level aggregation on the slower varying dimensions.

Selecting the Levels to Skip

You can identify the best levels to skip by determining the ratio of dimension members at each level, and keeping the ratio of members to be rolled up on the fly at approximately 10:1 or less. This ratio assures that all answer sets can be returned quickly. Either a data value is stored in the analytic workspace so it can simply be retrieved, or it can be calculated quickly from 10 stored values.

This 10:1 rule is best applied with some judgment. You might want to permit a higher ratio for levels that you know are seldom accessed. Or you might want to store levels at a lower ratio if you know they have heavy use. Generally, you should strive for a lower ratio instead of a higher one to maintain the best performance.

Aggregation rules identify how and when the aggregate values are calculated. You define the aggregation rules for each cube, and you can override these rules by defining new ones for a particular measure.

Aggregating a Slice of a Measure

The aggregation rules defined for a cube or a measure are always performed over all dimension members. You can perform a partial aggregation only in a calculation plan and only for regular composites.

Note: Do not set status for a compressed cube. All members must be in status.

To aggregate over a portion of a measure, you select the dimension members that identify the cells containing the source data, using the Status page of the Aggregation property sheet. You do not need to select the target cells. All of the cells identified by the ancestors of the selected dimension members are aggregated, either when you execute the cube script or when a user queries the measure.

When you select the dimension members, they are **in status**. This means that the dimension members have been selected for use in a calculation, a query, or other data manipulation. Likewise, **out of status** means that the dimension members have been excluded from use.

Figure 8–7 shows an aggregation in which the 12 months of 2006 are in status. Neither the quarters nor the year are in status, but aggregates are generated for all levels.

Figure 8–7 Sum Aggregation With All Source Values in Status

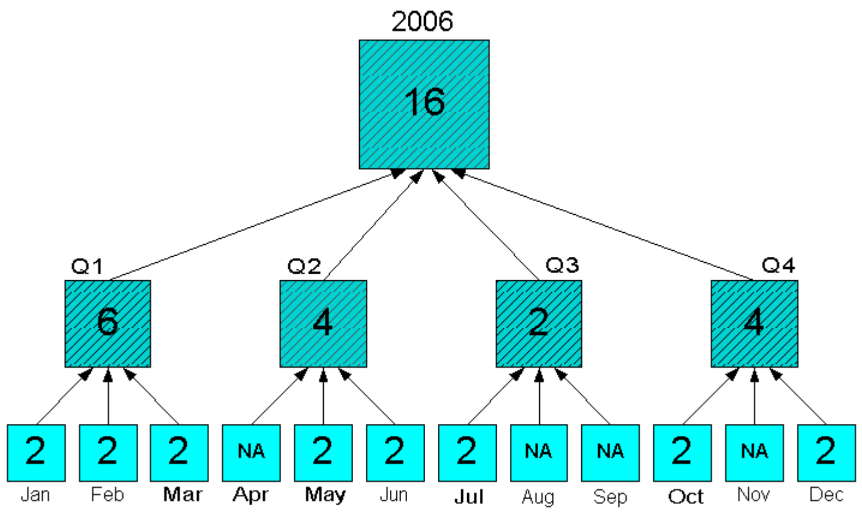
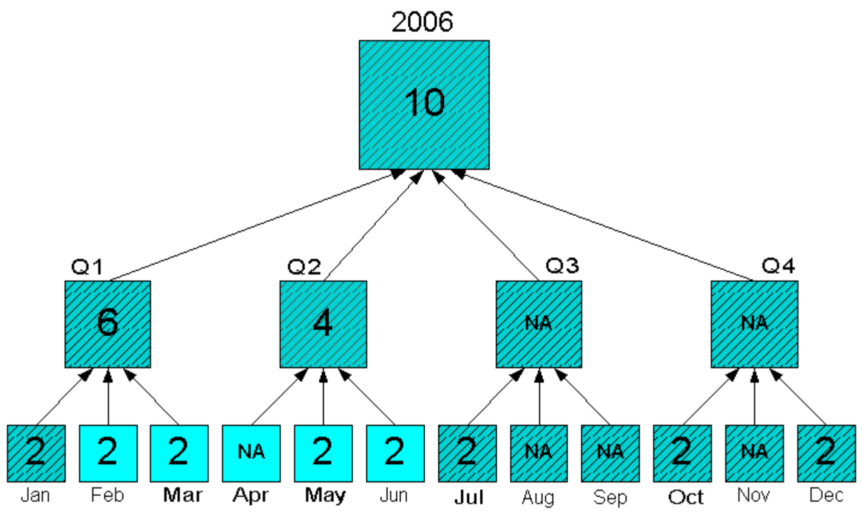


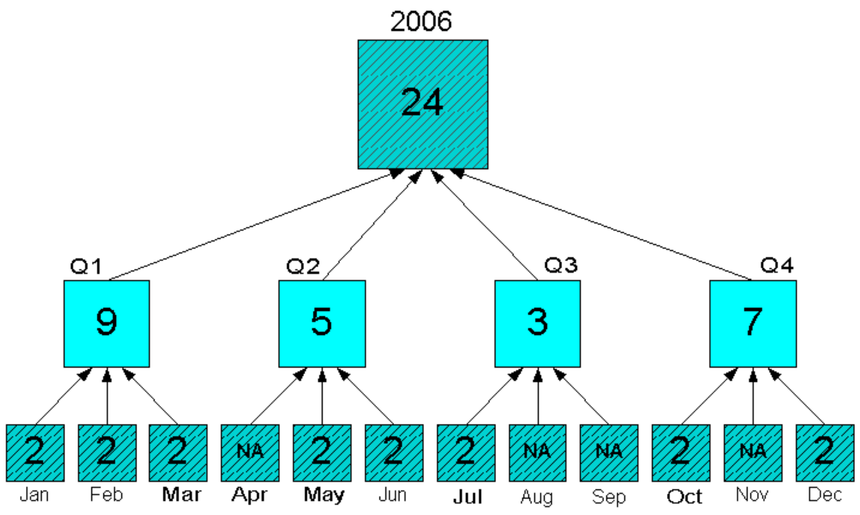
Figure 8–8 shows the same portion of data, but with only Feb to Jun in status. Aggregates are calculated only for Q1, Q2, and 2006. Note that Jan is included in the aggregation, even though it is out of status. The aggregation engine adds the ancestors, then the children to status before aggregating the data, as a means of maintaining the integrity of the data. The values for Q3 and Q4 are not included in the aggregation.

Figure 8–8 Sum Aggregation With Some Source Values Out of Status



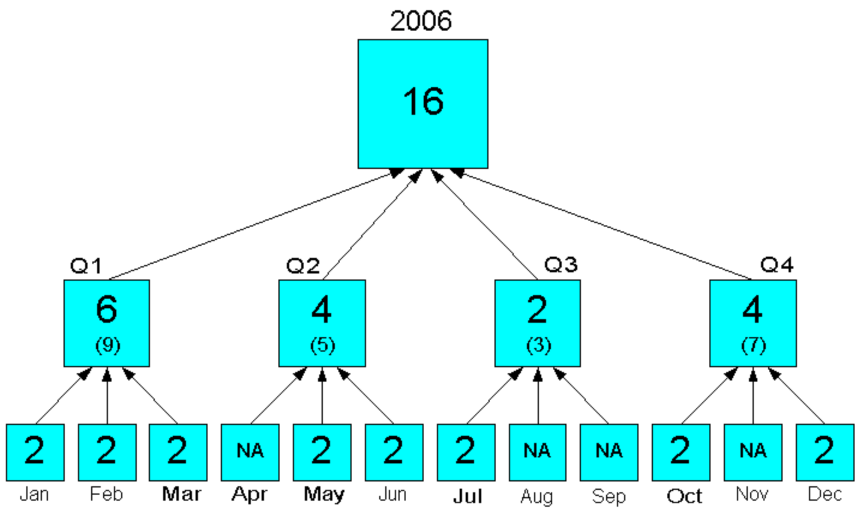
You may need to aggregate data that is stored in the middle of a hierarchy, perhaps if the data for a particular measure is not available or needed at the base level. You must be sure that the cells with the data are the lowest levels in the hierarchy in status. Figure 8–9 shows quarterly forecast data in status and aggregated to the year. The monthly values are not in status, and thus are excluded from the aggregation.

Figure 8–9 Sum Aggregation From the Quarterly Level



Aggregation begins at the lowest level in status and rolls up the hierarchy. The aggregate values overwrite any pre-existing values higher in the hierarchy. Figure 8–10 shows that when the Month level is in status, those values overwrite the forecast values at the Quarter level. The status of Quarter and Year has no effect on the aggregation.

Figure 8–10 Sum Aggregation From the Month Level Overwrites Quarters



Improving Aggregation Performance

The previous guidelines provide an approach to aggregation that should help you meet these basic goals:

- Finish Data Updates on Time
- Keep Within Allocated Resources
- Provide Good Response Time

If you anticipate problems with one or more of these goals, then you should keep them in mind while devising your aggregation rules. Otherwise, you may need to make adjustments after the initial build, if you experience problems meeting all of these goals.

Often the problem can be solved by changing factors other than the aggregation rules, as described in the following topics.

Note: Be sure to run the Sparsity Advisor so that the data is structured in the most efficient way. Refer to "Choosing a Data Storage Strategy" on page 3-15.

Finish Data Updates on Time

Most organizations allocate a batch window in which all data maintenance must be complete. If you are unable to finish refreshing the data in the allotted time, then you can make the following adjustments.

Be sure that you have set the database initialization parameters correctly for data maintenance, as described in "Setting Database Initialization Parameters" on page 12-1. You can make significant improvements in build performance by setting `SGA_TARGET`, `PGA_AGGREGATE_TARGET`, and `JOB_QUEUE_PROCESSES`.

After the initial build, you can save time by aggregating only newly loaded values, instead of aggregating all of them again. Partial aggregation is a choice you can make in the Maintenance Wizard.

Analytic workspaces are stored in partitioned tables, and you can create partitioned cubes. You can use these partitions to distribute the data across several disks, thus avoiding bottlenecks in I/O operations.

See Also: Chapter 12, "Administering Oracle OLAP"

Keep Within Allocated Resources

Your analytic workspace must fit within the allocated resources. The more levels of aggregate data that you store, the larger the tablespaces must be to store the analytic workspace.

The data type is an important consideration when estimating the size of an analytic workspace. The most commonly used data types for measures are `NUMBER` and `DECIMAL`. The difference in size is significant: an unscaled `NUMBER` value is 22 bytes and a `DECIMAL` value is 8 bytes.

Provide Good Response Time

An analytic workspace must provide good performance for end users. When pre-aggregation is done correctly, the response time for queries does not noticeably slow down. Analytic workspaces are optimized for multidimensional calculations, so that run-time summarizations should be extremely fast. However, runtime performance suffers if the wrong choices are made.

If response time is poor, then review the decisions you made in skipping levels and find those that should be pre-aggregated. Try to identify and pre-aggregate those areas of the data that are queried heavily. Check the level on which you partitioned the cube. Remember that all levels above the partition are calculated on the fly. When partitioning over Time, the Month level is a much better choice than Day.

Read the recommendations given in the previous topics. The savings in maintenance time and disk storage may be used to pre-aggregate more of the data.

9 Allocations

In Analytic Workspace Manager, you can create forecasts, set goals, and create budgets at a high level, and then allocate those numbers down a hierarchy to see how those numbers impact the contributing values.

This chapter contains the following topics:

What Is an Allocation?

Allocations distribute aggregate level data to detail level data, sometimes using an existing set of data as the basis for the allocation. This technology is often used in forecasting and budgeting systems. An example of a financial allocation is the automated distribution of a bonus pool, based on the current salaries and performance ratings of the employees.

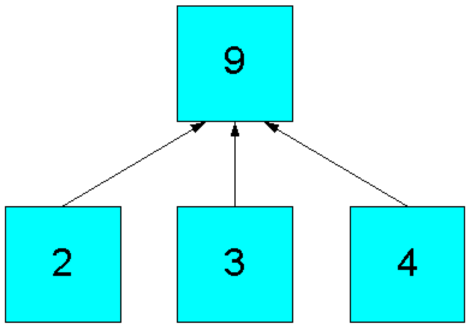
You can think of allocations as inverse aggregations.

- In aggregations, a group of child values are aggregated into a single parent value using an aggregation method, such as Sum.
- In allocations, a parent value is distributed to a group of child cells using an allocation method that is the inverse of the aggregation method, such as Average.

One important difference between aggregation and allocation is that an aggregation has one defined answer. An allocation has many possible answers for the same source value.

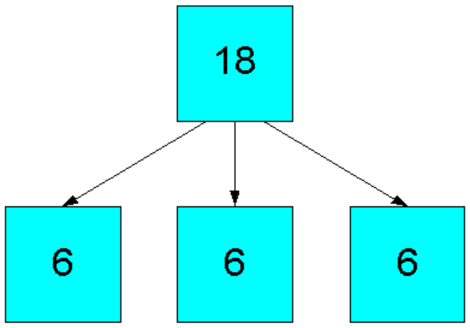
For example, consider the hierarchy in Figure 9–1. The value 9 is derived by aggregating the values 2, 3 and 4 using the Sum operator.

Figure 9–1 Aggregation in a Simple Hierarchy



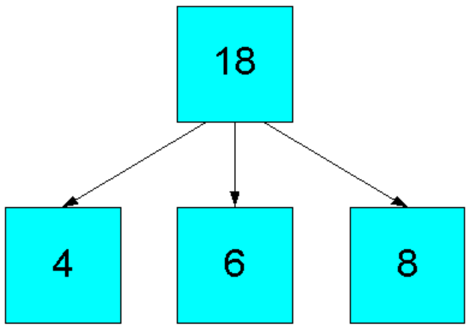
Now change the value of 9 to 18 and allocate the results to the children. The Even allocation operator divides the source value evenly by the number of children, and so assigns each child a value of 6, as shown in Figure 9–2.

Figure 9–2 Even Allocation In a Simple Hierarchy



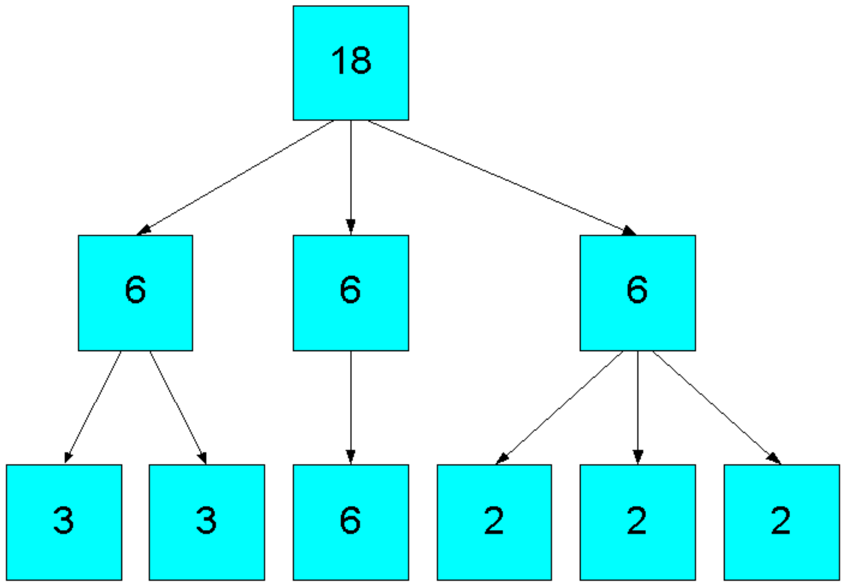
In contrast, the Proportional allocation operator divides the value into proportions based on the current value of each target cell, and so assigns values of 4, 6 and 8, as shown in Figure 9–3.

Figure 9–3 Proportional Allocation In a Simple Hierarchy



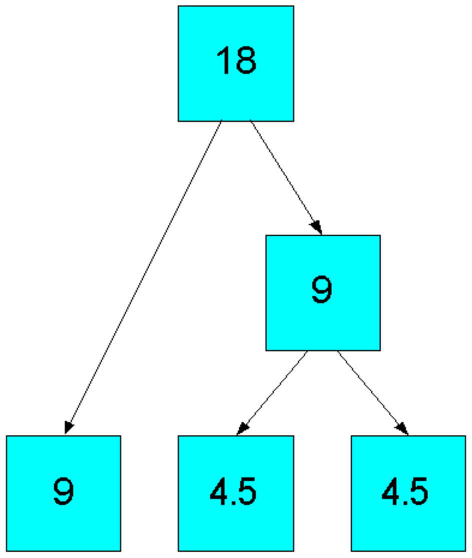
The previous examples show direct allocation, that is, where there is a parent-child relation between the source cell and the target cells. However, most hierarchies have multiple levels, and an allocation may assign values down the hierarchy, as shown in Figure 9–4.

Figure 9–4 Even Allocation in a Multilevel Hierarchy



Next, consider a skip level hierarchy. The source value is allocated down the hierarchy, as shown in Figure 9–5. The relationship of the target cell to the allocation source, not the hierarchical level of a cell, determines the allocation. Note that, as the result of an intermediate value in one branch, the base-level cells are allocated different values than in the simple hierarchy shown in Figure 9–2.

Figure 9–5 Even Allocation in a Skip Level Hierarchy



Creating Measures to Support an Allocation

Source, basis, and target are the most fundamental terms for describing allocation. You may use the same measure for all three roles or assign a different measure to each role. All allocation operators require a source and a target, but some operators do not use a basis. You can also multiply the results of an allocation by a weight measure.

Source Measures

The source measure contains the set of numbers that you want to allocate. You may use an existing measure, or you may perform some computation on existing data to construct new source values. For instance, you might want to budget 30 percent growth over the next year and perform an allocation to see the sales targets required for each product to meet that budget. You would create a calculated measure based on actual sales to use as the allocation source. Alternatively, you might generate a forecast at the middle or top of a hierarchy and then allocate the forecast results down to the lower levels.

Basis Measures

Depending on the type of allocation, the basis measure may identify which cells are the targets of an allocation, and what proportion of the allocation each target cell receives. Different operators use the basis measure in different ways, as illustrated by the diagrams of Even and Proportional operators in "What Is an Allocation?" on page 9-1. Note that a basis measure is not used by the hierarchical operators. Refer to "Allocation Operators" on page 9-9 for descriptions of all the operators and their use of a basis measure.

The basis measure can be the same as the target measure, or it can be a different measure. For example, suppose you want to calculate the sales of each individual product for an increase in total sales of 15 percent. You would create a calculated measure from Sales that contains the desired aggregate values, and use it as the allocation source. By using the original Sales measure as both the target and the allocation basis, and allocating with the Proportional distribution method, you can generate the individual product sales figures that are needed to produce the desired total sales figure.

If, however, you want to write the results of the allocation to a completely new measure, you would still use the Sales measure as the basis. The new target enables you to preview the allocated results before overwriting the original data. Similarly, you may want to allocate data into a Budget target measure and use an Actuals measure as the basis of the allocation.

Target Measures

The target measure stores the results of an allocation. By default, the target and the basis are the same measure. However, you may prefer to use a different target measure so that you can preview the results of an allocation before overwriting any original values.

Weight Measures

You can perform a calculation on the allocated values before they are stored in the target measure. For example, you might need to convert Sales numbers to a different currency. You might create a budget in US dollars, and then translate the allocation target into local currencies. To accomplish this, you would multiply the target values by a weight measure that contains the currency translation rates.

Selecting Dimension Members for an Allocation

You can perform an allocation over an entire measure or over selected branches of the hierarchy. You must restrict the allocation to a portion of the measure under these circumstances:

- You want to allocate some of the values at the top of the hierarchy, but not all the values.

  For example, you may need to restrict the Time dimension to a few future periods to prevent allocating over all the historical data.
- You want to allocate some values that are in the middle of the hierarchy.

  For example, you may have generated a forecast at the Month level of Time and the Brand level of Product, and you want to allocate those numbers down to the base.
- You want to allocate down to the middle of the hierarchy, not to the base.

  For example, you do not want to proliferate data to the Day level of Time and the SKU level of Product, because you are setting sales quotas, which do not need that level of detail.

Identifying the Sources and Targets

The dimension members that you select for the allocation is used to identify the source and the target cells. The selection must include:

- In the source measure, the cells at the top of the hierarchy that contain the values to be allocated.
- In the target measure, the cells down the hierarchy that are allocated values.

Figure 9–6 shows a portion of a Time hierarchy with the source allocation values at the Quarter level. How the allocation is performed depends on which members are selected (or **in status**). Table 9–1 describes various status settings and their effect on the allocation.

Figure 9–6 Allocating at the Quarter Level

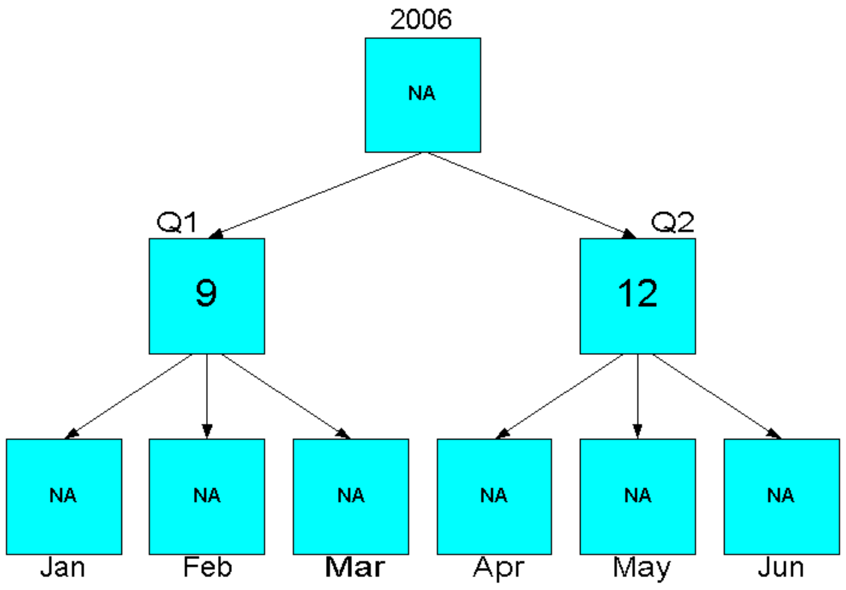


Table 9–1 Results of Status on Allocation at the Quarter Level

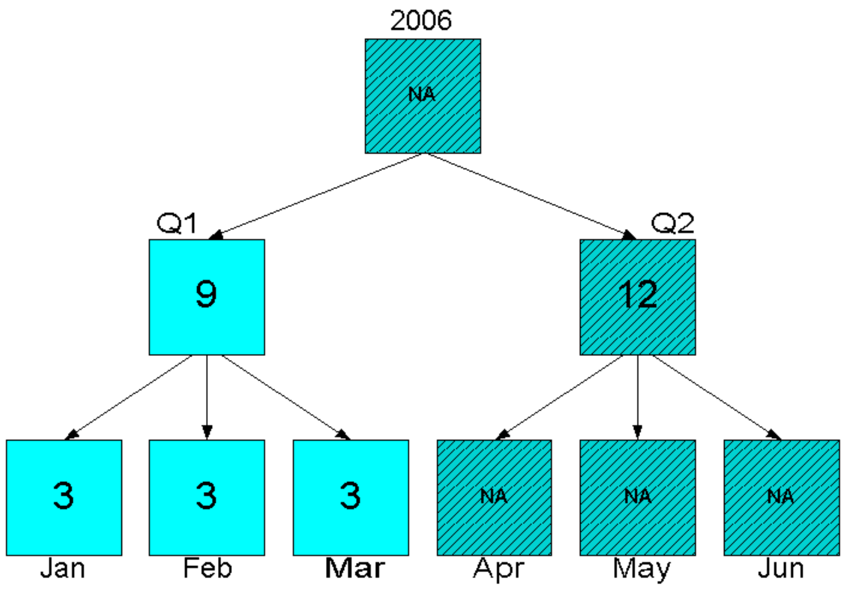
Status	Allocation	Explanation
All	None	The top member of the hierarchy (2006) has no value, so there is no source value to allocate.

Table 9–1 (Cont.) Results of Status on Allocation at the Quarter Level

Status	Allocation	Explanation
All quarters	None	The children of Q1 and Q2 are not in status, so there is no target for allocation.
All quarters, all months	Jan to Jun	Q1 and Q2 are in status, so the value 9 is allocated to Jan, Feb, and Mar, and the value 12 is allocated to Apr, May, and Jun.
Q1, Jan to Mar	Jan to Mar	Q1 and its children are in status, so the value 9 is allocated to Jan, Feb, and Mar. Q2 is not in status and is not allocated.

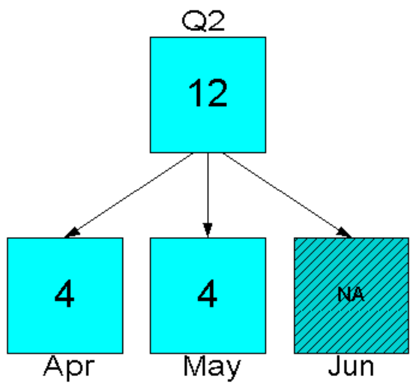
Figure 9–7 shows the correct status for allocating only Q1.

Figure 9–7 Status for Allocating One Mid-Level Branch of a Hierarchy



When calculating the allocation, the OLAP engine expands the current status to include siblings, if necessary. Figure 9–8 shows an even allocation when Q2, Apr and May are in status. Jun is not a target and does not get a value. Nonetheless, the engine divides the allocated value of 12 by all three children, not just the two targets, to calculate the values for Apr and May.

Figure 9–8 Even Allocation to Selected Child Members

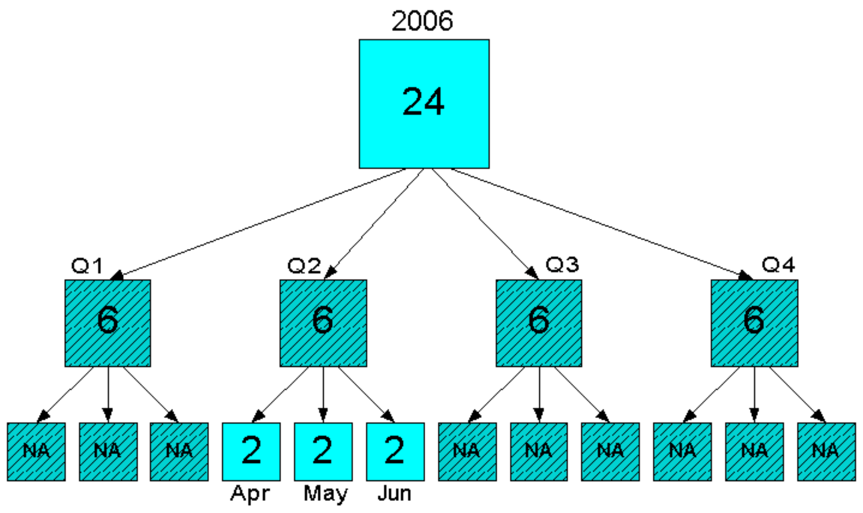


Identifying the Allocation Path

When the allocation path from the source to the target cells is not defined by the current status, the engine may populate the siblings of cells along the path. This information is important only if you want to avoid overwriting existing values or unnecessarily proliferating data.

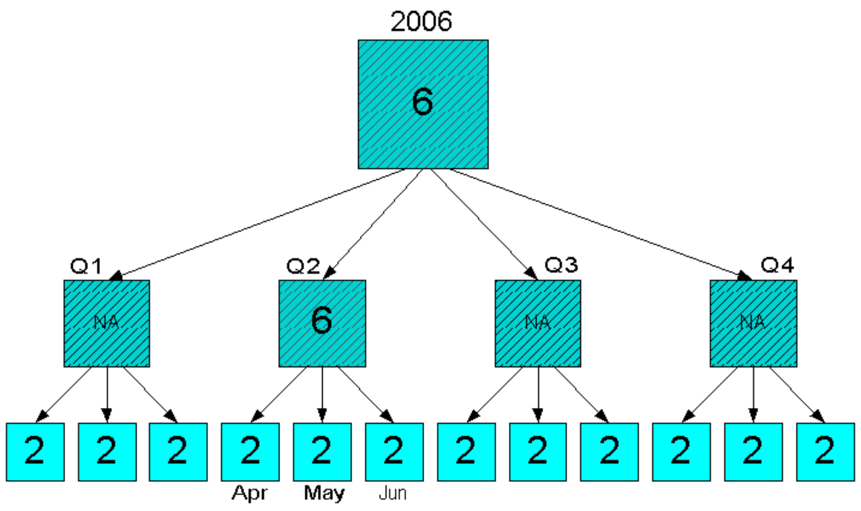
Figure 9–9 shows the results of an allocation from 2006 to the three months in Q2. Only 2006, Apr, May, and Jun are in status. This status does not define a path from the source to the target. Because the Quarter level is on the path to the target, all of the quarters are allocated a value.

Figure 9–9 Even Allocation Without a Defined Allocation Path



However, when Q2 is included in status, it is the only quarter to get an allocated value, as shown in Figure 9–10.

Figure 9–10 Even Allocation With a Defined Allocation Path



Creating an Allocation

You can create allocations in Analytic Workspace Manager by defining an allocation step in a Calculation Plan. Take these steps:

1. Create the source, basis, target, and weight measures. They must be in the same cube. The source, basis, and weight measures can be either stored measures or calculated measures. The target measure must be a stored measure.
2. Create an allocation step:
   a. In the navigation tree, create a new Calculation Plan or open an existing plan.
   b. On the General tab of the Calculation Plan property page, click **New Step**, then choose **New Allocation Step**.

      The New Allocation Step property pages are displayed.
   c. Complete the General page, being careful to select the correct source, target, and basis measures.
   d. On the Rules page, use the up- and down-arrows to list the dimensions in the order you want them calculated. If you assign different operators to different dimensions, then the allocated values may be different depending on the order.
   e. Select an operator for each dimension that you want to allocate, and a weight measure if desired.
   f. On the Status page, select the members for each dimension of the measure. To allocate values from the top down to the base, retain the default selection of All Levels. Otherwise, select the dimension members with the source data and the target members.

      Refer to "Selecting Dimension Members for an Allocation" on page 9-4 for information on selecting the dimension values.
   g. Click **Create** to save the allocation step, then **Apply** to save the Calculation Plan.
3. To allocate the data, right-click the Calculation Plan in the navigation tree, then choose **Execute Calculation Plan**.

4. To view the results of the allocation, right-click the target measure and choose **View Data**.

Note: Always follow an Allocation Step with an Aggregation Step.

Allocation Operators

Allocation operators determine the methodology for distributing source values to their targets. There are three basic types of allocation operators: Copy, Even Distribution, and Proportional Distribution.

Within these basic types are regular operators and hierarchical operators. The regular operators only assign values to cells identified by the basis measure as having a value. The hierarchical operators do not use a basis measure. They assign values to all target cells.

Note: The hierarchical operators may increase the size of a measure dramatically by allocating values to previously empty cells. Be careful to set the status of all dimensions.

Copy Operators

These are the copy operators:

- **Copy**: Copies the allocation source to all of the target cells that have a basis value that is not NA (null).
- **Hierarchical Copy**: Copies the allocation source to all of the target cells specified by the hierarchy, regardless of the basis value.
- **Minimum**: Copies the allocation source to the target that has the smallest basis value.
- **Maximum**: Copies the allocation source to the target that has the largest basis value.
- **First non-NA Data Value**: Copies the allocation source to the first target cell that has a non-NA basis value.
- **Hierarchical First Member**: Copies the allocation source to the first target cell specified by the hierarchy, regardless of the basis value.
- **Last non-NA Data Value**: Copies the allocation source to the last target cell that has a non-NA basis value.
- **Hierarchical Last Member**: Copies the allocation source to the last target cell specified by the hierarchy, regardless of the basis value.

Even Distribution Operators

These are the even distribution operators:

Even: Divides the allocation source by the number of target cells that have non-NA basis values and applies the quotient to each target cell.

Hierarchical Even: Divides the allocation source by the number of target cells, including the ones that have NA values, and applies the quotient to each target cell.

Proportional Distribution Operator

The proportional distribution operator is:

Proportional: Divides the allocation source by the sum of the basis values, then multiplies the quotient by the individual basis value for each target cell.

Relationships Between Allocation and Aggregation Operators

The allocation system operates as the logical inverse of the aggregation system. In other words, if you allocate down from a middle level of a hierarchy, you can aggregate up to the top of the hierarchy using an aggregation operator that corresponds to the allocation operator. Table 9–2 shows the correspondence between allocation operators and aggregation operators.

Table 9–2 Corresponding Allocation and Aggregation Operators

Allocation Operator	Aggregation Operator
Copy	Average
Hierarchical Copy	Average
Minimum	Minimum
Maximum	Maximum
First non-NA Data Value	First Non-NA Data Value
Last non-NA Data Value	Last Non-NA Data Value
Hierarchical First Member	Hierarchical First Member
Hierarchical Last Member	Hierarchical Last Member
Even	Sum or Average
Hierarchical Even	Hierarchical Average
Proportional	Sum

Case Study: Allocating a Budget

This example creates a sales budget that is 10% higher than the previous year's sales. It uses a calculated measure to generate the increase, then distributes the total increase evenly down the dimension hierarchies.

Creating the Source Measure

To create the source measure:

1. Expand the `UNITS_CUBE` folder, right-click **Calculated Measures**, and choose **Create Calculated Measure**.

   The Calculation Wizard opens.

2. Complete the Name and Type page with these values:

   - Name: `sales_py`
   - Calculation Type: **Prior Value** (under Prior/Future Comparison)

3. Complete the Prior Value page with these values:

   - Measure: **Sales**
   - Over Time in: **Calendar Year**

- Go back by: **1 Year**

4. Create a second calculated measure with the name SALES_BUDGET.
5. For the calculation, expand the Basic Arithmetic folder and choose Multiplication.
6. On the Multiplication page, multiply SALES_PY by 1.06 for a 6% increase in Sales over the prior year.

Creating the Target Measure

This example stores the allocated data in a separate measure from the source data to assure that the allocated data does not overwrite any source data.

To create the target measure:

1. In the UNITS_CUBE folder, right-click Measures and select **Create Measure**.

   The Create Measure dialog box opens.
2. On the General page, create a measure named ALLOC_SALES_BUDGET.
3. Select **Override the Aggregation Specification of the Cube.**
4. On the Summarize To page, deselect all levels for all dimensions.

The measure is not mapped to a data source, so no aggregation needs to be done during regular builds. Instead, aggregation is defined in the Calculation Plan. The aggregation step is not shown in this example; refer to "Case Study: Forecasting Sales for Global Enterprises" on page 7-13 for an example that shows forecasting, allocation, and aggregation.

Creating the Calculation Plan

Budget Plan has an allocation step and an aggregation step (not shown).

To create a new Calculation Plan:

1. Right-click Calculation Plans and select **Create Calculation Plan**.

   The Create Calculation Plan dialog box opens.
2. Create a new plan named BUDGET_PLAN. Click **Create**.

   BUDGET_PLAN appears as a new item in the Calculation Plans folder. It does not yet contain any steps.

Creating the Allocate Budget Step

The SALES_BUDGET calculated measures generates data at all levels. The allocation redistributes the data from the top of the hierarchy to the lowest levels and stores it in the target measure.

To create an allocation step:

1. On the General page of Sales Plan, click **New Step**, then select **New Allocation Step**.

   The Create Allocation Step dialog box opens.
2. Complete the General page with these values:
   - Name: allocate_budget_step
   - Cube: UNITS_CUBE

- Source Measure: `SALES_BUDGET`
- Target Measure: `ALLOC_SALES_BUDGET`
- Basis Measure: `SALES_BUDGET`

3. On the Rules page, assign **Hierarchical Even** for the Time operator. For the other dimensions, assign the **Proportional** operator.
4. On the Status page, keep the default status of **All Levels** for all dimensions.
5. Click **Create** to save the allocation step.
6. Click **Apply** to save the Calculation Plan.

Generating and Validating the Allocation

To generate the allocation:

1. Expand the Calculation Plans folder. Right-click `BUDGET_PLAN` and choose **Execute Calculation Plan BUDGET_PLAN**.

   The Maintenance Wizard opens, and `BUDGET_PLAN` is a selected target object.

2. Click **Finish**.

   The build log is displayed when the Calculation Plan is done executing.

To view the allocation results, take these steps:

1. Fully expand the `UNITS_CUBE` folder, right-click the `ALLOC_SALES_BUDGET` measure, and choose **View Data ALLOC_SALES_BUDGET**.

   The Measure Data Viewer opens. No data is displayed, because the top dimension levels provide the source data, not the allocated data.

2. From the File menu, choose **Query Builder**.

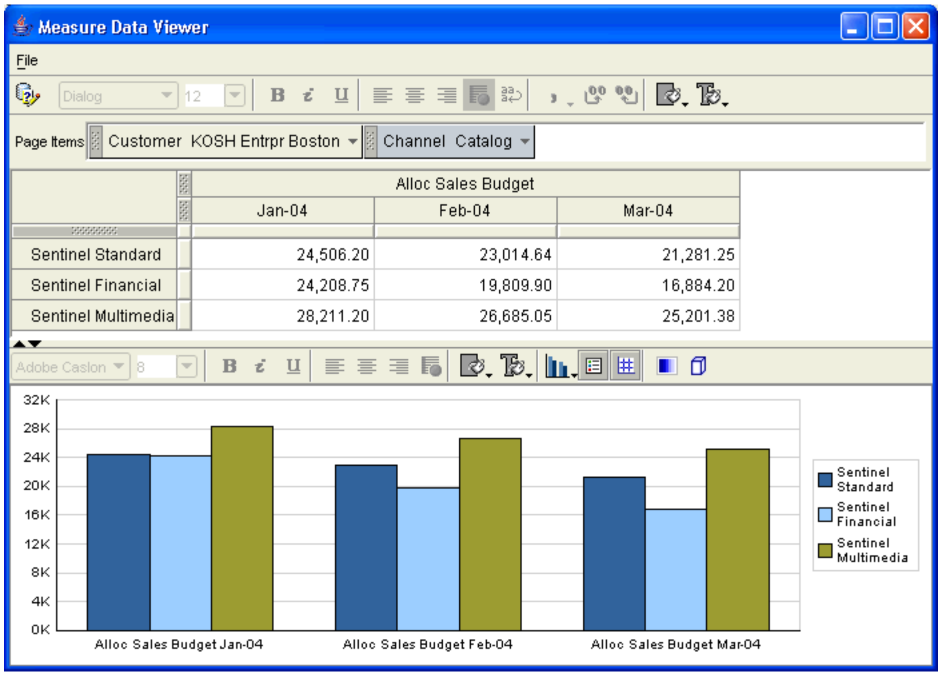
   The Query Builder opens.

3. On the Layout tab, switch Product and Customer. Click **Help** for instructions.
4. On the Dimensions tab, set the status of all dimensions to the base level. You may wish to select just a few values from these lists. For Time, limit the months to 2004, since that it is only allocated year.
5. Click **OK** to close the Query Builder.

Note: Always follow an Allocation Step with an Aggregation Step.

Figure 9–11 shows a sample of the allocated data. The allocated data should be aggregated from these base levels to the top by an aggregation step.

Figure 9–11 Allocated Sales Budget Data



	Alloc Sales Budget		
	Jan-04	Feb-04	Mar-04
Sentinel Standard	24,506.20	23,014.64	21,281.25
Sentinel Financial	24,208.75	19,809.90	16,884.20
Sentinel Multimedia	28,211.20	26,685.05	25,201.38

Case Study: Allocating a Sales Forecast

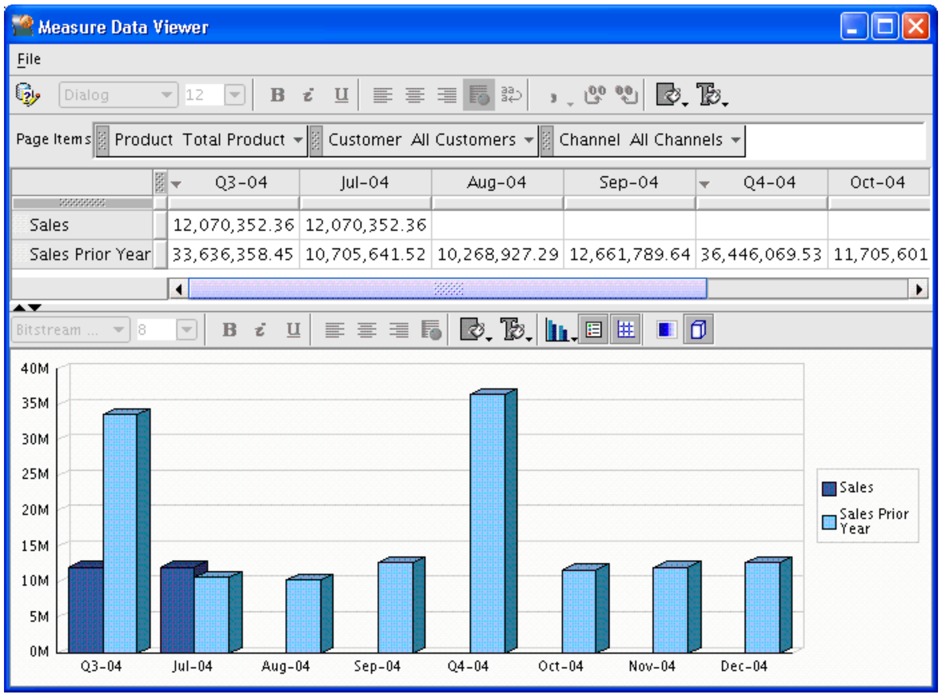
"Case Study: Forecasting Sales for Global Enterprises" on page 7-13 generates a four-month Sales forecast from August 2004 to December 2005. The forecast data is at the Month level for Time, and at the topmost level for Product, Customer, and Channel. An allocation step distributes the forecast data down to the base levels of these three dimensions.

Creating an Allocation Basis Measure

This example uses the Proportional method to distribute the values based on the sales performance for the previous year. The Proportional method uses another measure as the basis for the allocation. This example uses a calculated measure for sales values for the prior year as the basis measure. If you did not create `Sales_PY` for "Case Study: Allocating a Budget" on page 9-10, you should do so now.

Figure 9–12 compares Prior Year Sales to Sales. The Prior Year Sales measure has data for the forecast periods, while the Sales measure does not.

Figure 9–12 Creating a Basis Measure for Allocating Forecast Data



Page Items: Product Total Product | Customer All Customers | Channel All Channels

	Q3-04	Jul-04	Aug-04	Sep-04	Q4-04	Oct-04
Sales	12,070,352.36	12,070,352.36				
Sales Prior Year	33,636,358.45	10,705,641.52	10,268,927.29	12,661,789.64	36,446,069.53	11,705,601

Creating the Allocate Sales Forecast Step

The forecast created the data only for a single level of each dimension. Only Time is populated at the base level. The data must be allocated to the base levels of the other dimensions before it can be aggregated by the OLAP engine.

To create an allocation step:

1. On the General page of Sales Plan, click **New Step**, then select **New Allocation Step**.

   The Create Allocation Step dialog box opens.

2. Complete the General page with these values:

   - Name: `allocate_sales_forecast_step`
   - Cube: `UNITS_CUBE`
   - Source Measure: `SALES_FORECAST`
   - Target Measure: `SALES_FORECAST`
   - Basis Measure: `SALES_PY`

3. On the Rules page, select **None** for the Time operator. For the other dimensions, select **Proportional**.

4. On the Status page, set the dimension status using conditions:

   - Time: Start with **Month**

     On the Members tab, verify that only months are listed.

   - Customer: Start with **All Levels**

     On the Members tab, verify that all members are listed.

   - Product: Start with **Descendants of Total Product**

On the Members tab, verify that all members except `Total Products` are listed.

- Channel: Start with **All Levels**

  On the Members tab, verify that all members are listed.

5. Click **Create** to save the allocation step.
6. Click **Apply** to save the Calculation Plan.

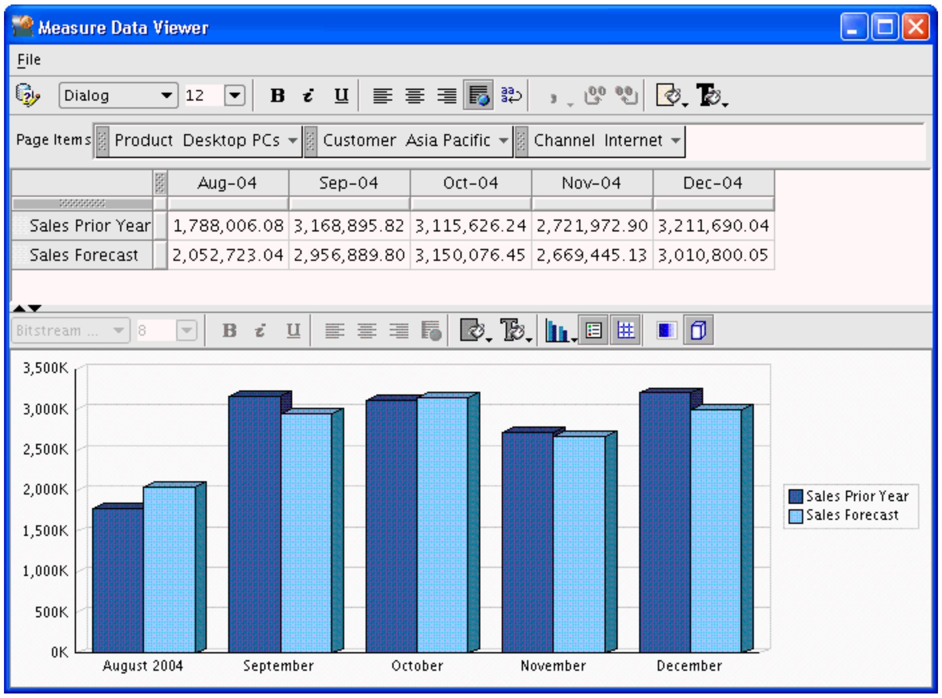
Generating and Validating the Allocation

Rerun the Calculation Plan, as described in "Generating and Validating the Allocation" on page 9-12. Both the forecast step and the allocation step are executed.

To view the allocation results, use the Measure Viewer to see the data in the Sales Forecast measure. The allocation populated all levels of Product, Customer, and Channel.

Figure 9–13 compares the forecast data with the basis measure at the middle levels of the dimensions used in the allocation.

Figure 9–13 Forecast Data After Allocation



Page Items: Product Desktop PCs; Customer Asia Pacific; Channel Internet

	Aug-04	Sep-04	Oct-04	Nov-04	Dec-04
Sales Prior Year	1,788,006.08	3,168,895.82	3,115,626.24	2,721,972.90	3,211,690.04
Sales Forecast	2,052,723.04	2,956,889.80	3,150,076.45	2,669,445.13	3,010,800.05

10 Developing Reports and Dashboards

You can use any SQL development tool or application to create reports and dashboards populated with data from OLAP cubes. This chapter shows the basic steps for working with the tools provided with Oracle Database: Oracle Business Intelligence Publisher (BI Publisher) and Oracle Application Express. You can try these tools, or you can apply the methods shown here to your favorite SQL tool.

This chapter contains the following topics:

- Developing SQL Applications for Dimensional Data
- Developing a Report Using BI Publisher
- Developing a Dashboard Using Application Express

See Also: Chapter 4, "Querying Dimensional Objects Using SQL"

Developing SQL Applications for Dimensional Data

You can use any SQL query against a cube as the content for a report or dashboard. Both BI Publisher and Application Express contain a Query Builder, which you can use to develop queries against both relational and dimensional objects. You can also cut-and-paste queries from a SQL script or another source, which is the method used in this chapter.

If your goal is to create static reports and dashboards, then you do not need to read any further. You can start developing OLAP applications immediately using your favorite tool. This chapter explains how to create applications with dynamic content. It focuses on ways to leverage the unique capabilities of cubes and dimensions to create drillable reports and graphs using a single query. In this chapter, you learn how to create two types of drillable interfaces:

- **Choice Lists**: You can create a drop-down list for each dimension to drill on the dimensions in a report or dashboard.
- **Linked Dimension Columns**: In Application Express, you can add links to the dimension columns of a crosstab to drill down to the bottom of a hierarchy, and use a Reset button to return to the top level.

These user interfaces set the values of bind variables in the `WHERE` clause of the source query. When a user changes the current selection in a choice list or clicks a link in a crosstab, that action dynamically changes the value of the variable. When the variable changes, so does the condition of the query and the contents of the report or dashboard.

When the variable sets the value of a parent column in a cube view, users can drill on a parent to view its children.

Example 10–1 shows a basic SQL query against a cube view of the Units Cube in the Global sample schema. The query selects the `SALES` measure and three calculated measures that use `SALES` as the basis for the calculations:

- `SALES_PP`: Sales from the prior period.
- `SALES_CHG_PP`: Difference in sales between the current period and the prior period.
- `SALES_PCT_CHG_PP`: Percent difference in sales between the current period and the prior period.

This query is used in the sample applications developed in this chapter. The parent columns for the Product, Customer, and Time dimensions support drilling in these applications. The Channel dimension remains anchored at the `TOTAL_CHANNEL` level.

Example 10–1   SQL Query Against the Sales Cube

```
SELECT product_ldsc "Product",
       customer_ldsc "Customer",
       time_ldsc "Time",
       round(sales) "Sales",
       round(sales_pp) "Prior Period",
       round(sales_chg_pp) "Change",
       round(sales_pct_chg_pp * 100) "Percent Change"
/* From cube view */
    FROM units_cube_cubeview
/* Filters on all dimensions */
    WHERE product_primary_prnt = 'TOTAL_PRODUCT_1'
       AND customer_shipments_prnt = 'TOTAL_CUSTOMER_1'
       AND time_calendar_yea_prnt = 'YEAR_4'
       AND channel_level = 'TOTAL_CHANNEL'
    ORDER BY product, customer, time_end_date;

Product         Customer        Time          Sales Prior Period     Change Percent Change
--------------- --------------- -------- ---------- ------------ ---------- --------------
Hardware        North America   Q1-01      15029369     16225669   -1196300             -7
Hardware        North America   Q2-01      14873260     15029369    -156108             -1
Hardware        North America   Q3-01      15951726     14873260    1078465              7
Hardware        North America   Q4-01      17228528     15951726    1276802              8
Hardware        Asia Pacific    Q1-01       6282186      7040505    -758319            -11
Hardware        Asia Pacific    Q2-01       6661802      6282186     379616              6
Hardware        Asia Pacific    Q3-01       6936356      6661802     274554              4
Hardware        Asia Pacific    Q4-01       7371841      6936356     435486              6
Hardware        Europe          Q1-01       4183166      4509289    -326123             -7
Hardware        Europe          Q2-01       4165212      4183166     -17954              0
Hardware        Europe          Q3-01       4521921      4165212     356710              9
Hardware        Europe          Q4-01       4785698      4521921     263777              6
Software/Other  North America   Q1-01       1229455      1260900     -31445             -2
Software/Other  North America   Q2-01       1233716      1229455       4261              0
Software/Other  North America   Q3-01       1333084      1233716      99368              8
Software/Other  North America   Q4-01       1446759      1333084     113675              9
Software/Other  Asia Pacific    Q1-01        519562       517674       1888              0
Software/Other  Asia Pacific    Q2-01        509705       519562      -9857             -2
Software/Other  Asia Pacific    Q3-01        566745       509705      57040             11
Software/Other  Asia Pacific    Q4-01        596995       566745      30250              5
Software/Other  Europe          Q1-01        351592       364827     -13235             -4
Software/Other  Europe          Q2-01        354732       351592       3140              1
Software/Other  Europe          Q3-01        381837       354732      27105              8
Software/Other  Europe          Q4-01        416232       381837      34395              9

24 rows selected.
```

See Also: Chapter 6, "Enhancing Your Database With Analytic Content," for information about calculated measures

Developing a Report Using BI Publisher

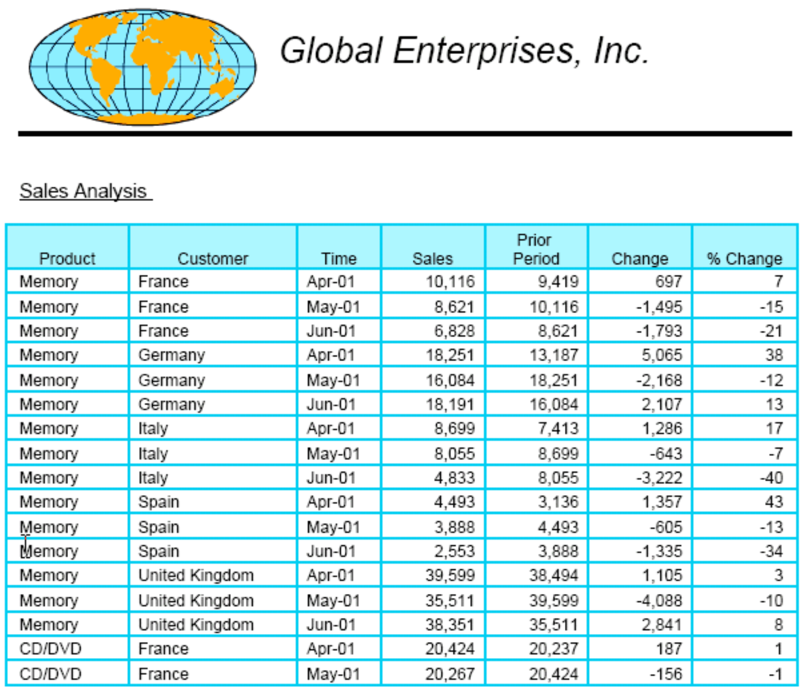
BI Publisher is an efficient, scalable reporting solution for generating and delivering information through a variety of distribution methods. It reduces the high costs associated with the development and maintenance of business documents, while increasing the efficiency of reports management. BI Publisher generates reports in a variety of formats, including HTML, PDF, and Excel.

If you have not used BI Publisher, you can download the software, tutorials, and full documentation from the Oracle Technology Network at

`http://www.oracle.com/technetwork/middleware/bi-publisher/overview/index.html`

Figure 10–1 shows a report in PDF format based on the query shown in Example 10–1. When generating a report for distribution, you can select any combination of Products, Customers, and Time Periods from the choice lists. The selection for this report is Hardware products, customers in Europe, and months in Q2-01. This chapter explains how you can create a report like this one using drillable dimensions.

Figure 10–1 Sales Report in BI Publisher



Global Enterprises, Inc.

Sales Analysis

Product	Customer	Time	Sales	Prior Period	Change	% Change
Memory	France	Apr-01	10,116	9,419	697	7
Memory	France	May-01	8,621	10,116	-1,495	-15
Memory	France	Jun-01	6,828	8,621	-1,793	-21
Memory	Germany	Apr-01	18,251	13,187	5,065	38
Memory	Germany	May-01	16,084	18,251	-2,168	-12
Memory	Germany	Jun-01	18,191	16,084	2,107	13
Memory	Italy	Apr-01	8,699	7,413	1,286	17
Memory	Italy	May-01	8,055	8,699	-643	-7
Memory	Italy	Jun-01	4,833	8,055	-3,222	-40
Memory	Spain	Apr-01	4,493	3,136	1,357	43
Memory	Spain	May-01	3,888	4,493	-605	-13
Memory	Spain	Jun-01	2,553	3,888	-1,335	-34
Memory	United Kingdom	Apr-01	39,599	38,494	1,105	3
Memory	United Kingdom	May-01	35,511	39,599	-4,088	-10
Memory	United Kingdom	Jun-01	38,351	35,511	2,841	8
CD/DVD	France	Apr-01	20,424	20,237	187	1
CD/DVD	France	May-01	20,267	20,424	-156	-1

Creating an OLAP Report in BI Publisher

A report consists of a **report entry**, which you create in BI Publisher, and a **layout template**, which you create using an application such as Microsoft Word or Adobe Acrobat. You can organize your reports in folders.

BI Publisher is a middleware application and can derive data from multiple sources. These procedures assume that you can access one or more cubes from BI Publisher. If you cannot, contact your BI Publisher administrator about defining a new data source.

To create a report entry:

1. Open a browser to the BI Publisher home page and log in.
2. Click **My Folders**.
3. Open an existing folder.

   or

   To create a new folder:

   a. Click **Create a New Folder**.

   b. Enter a name for the folder in the text box, such as `OLAP Reports`.

   c. Click **Create**.
4. Click the new folder to open it.
5. Create a new report:

   a. Click **Create a New Report**.

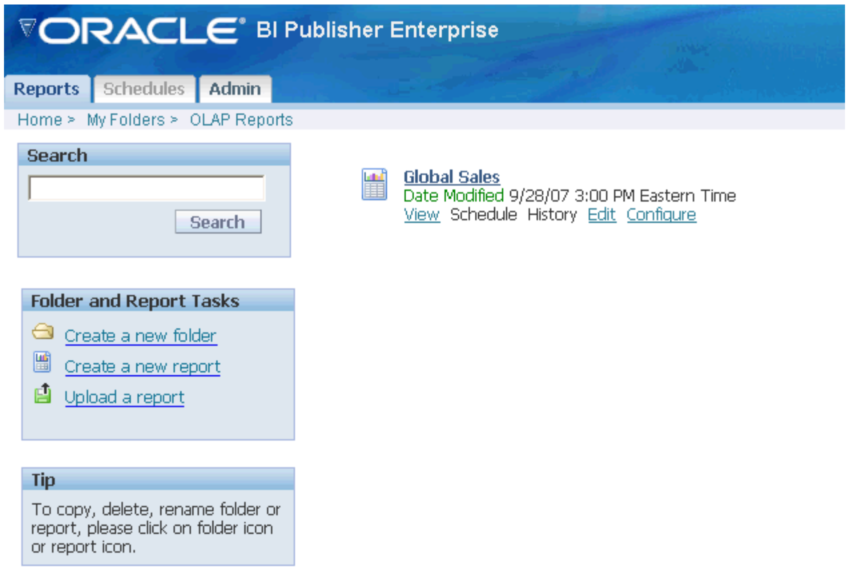
   b. Enter a report name in the text box.

      This example creates a report named `Global Sales`.

   c. Click **Create**.

   The new report appears in the folder, as shown in Figure 10–2.

Figure 10–2 Creating a New Report


ORACLE BI Publisher Enterprise
Reports
Schedules
Admin
Home > My Folders > OLAP Reports
Search
Search
Global Sales
Date Modified 9/28/07 3:00 PM Eastern Time
View Schedule History Edit Configure
Folder and Report Tasks
Create a new folder
Create a new report
Upload a report
Tip
To copy, delete, rename folder or report, please click on folder icon or report icon.

To configure the report entry:

1. To define the contents of the report, click **Edit**.

   The Report Editor opens.
2. For General Settings, enter a description and select a default data source.

   If the list does not include a connection to the database and schema containing your cubes, contact your BI Publisher administrator.
3. Select Data Model, then click **New**.

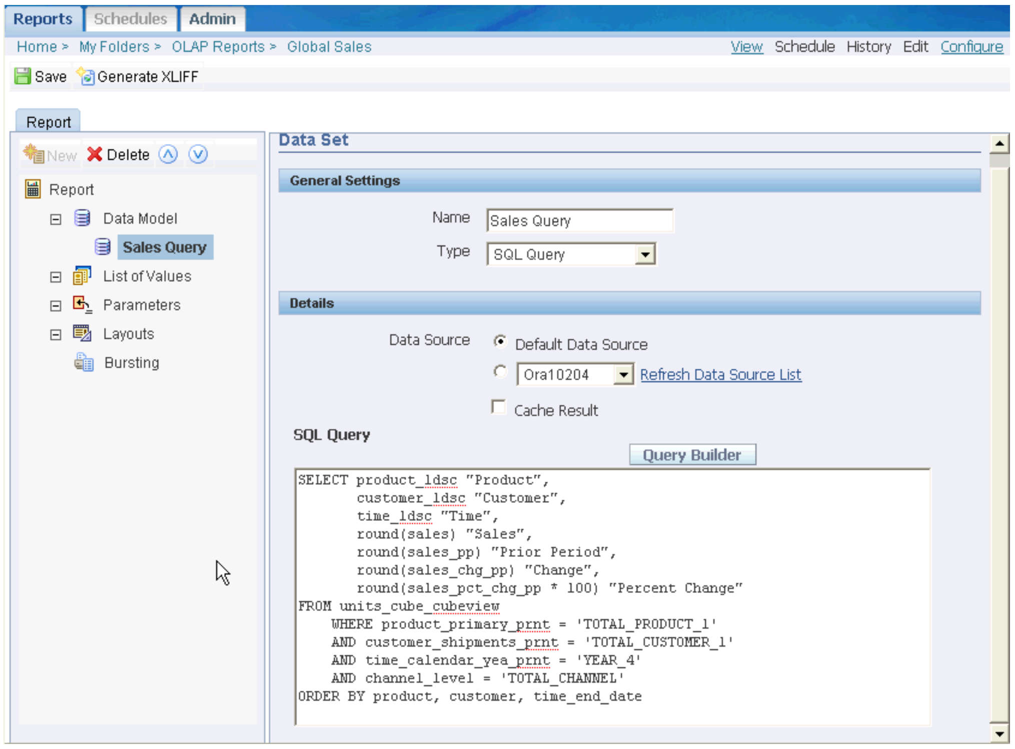
The Data Set page opens.

4. Enter a name for the data set and enter a SQL query like the one shown in Example 10–1. Do not end semicolon.
5. Click **Save**.
6. Click **View**.

   BI Publisher checks the report definition for errors. If there are none, then it generates the XML for the report.

Figure 10–3 shows the Report Editor with the Data Set page displayed.

Figure 10–3 Creating a Data Model in the BI Publisher Report Editor



Name: Sales Query
Type: SQL Query
Data Source: (●) Default Data Source ( ) Ora10204 Refresh Data Source List
[ ] Cache Result

SQL Query

```
SELECT product_ldsc "Product",
       customer_ldsc "Customer",
       time_ldsc "Time",
       round(sales) "Sales",
       round(sales_pp) "Prior Period",
       round(sales_chg_pp) "Change",
       round(sales_pct_chg_pp * 100) "Percent Change"
FROM units_cube_cubeview
    WHERE product_primary_prnt = 'TOTAL_PRODUCT_1'
    AND customer_shipments_prnt = 'TOTAL_CUSTOMER_1'
    AND time_calendar_yea_prnt = 'YEAR_4'
    AND channel_level = 'TOTAL_CHANNEL'
ORDER BY product, customer, time_end_date
```

Creating a Template in Microsoft Word

BI Publisher does not contain formatting tools. Instead, it enables you to design a report using familiar desktop applications. This example uses Microsoft Word. A report template can contain:

- Static text and graphics that you enter like any other Word document.
- Dynamic fields such as the date and time or page numbers, which are processed by Word.
- Codes that identify the XML tags for your data, which are processed by BI Publisher. When BI Publisher generates a report, it replaces the codes with the data identified by these tags.

You can format all parts of the report template in Word, selecting the fonts, text and background colors, table design, and so forth.

Example 10–2 shows the XML for a row of data returned by the sample query. The tags match the column names in the select list, except that underscores replace the spaces. The tags are `Product`, `Customer`, `Time`, `Sales`, `Prior_Period`, `Change`, and `Percent_Change`. XML tags are case-sensitive. You use the HTML tag names as the codes in the Word document.

Example 10–2 XML for a SQL Query

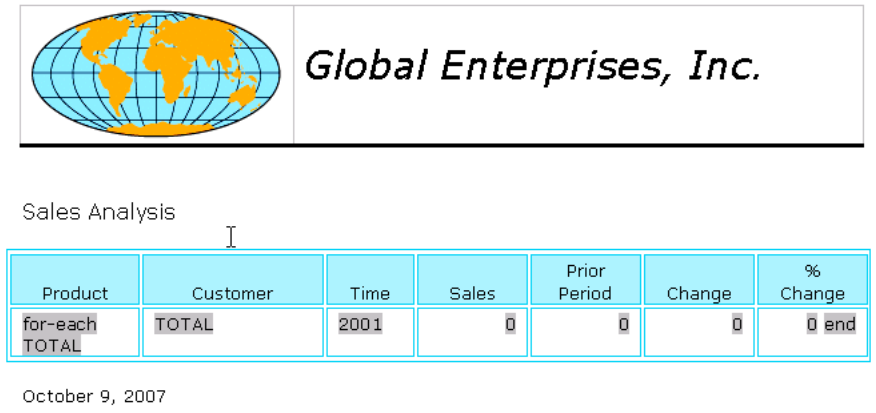
```
<ROW>
   <Product>Hardware</Product>
   <Customer>North America</Customer>
   <Time>Q1-01</Time>
   <Sales>15029369</Sales>
   <Prior_Period>16225669</Prior_Period>
   <Change>-1196300</Change>
   <Percent_Change>-7</Percent_Change>
</ROW>
```

Figure 10–4 shows the Word document that is used as the template for the sample report. It contains these elements:

- A table used to format the banner, which consists of a graphic, the company name, and a horizontal line. (Static)
- The name of the report. (Static)
- A table for the query results that contains two rows:
  - A heading row. (Static)
  - A body row containing text form fields, which identify the XML tags and the appropriate formatting for the data. BI Publisher replaces these fields with data from the query. Note that the first and last columns contain two fields. The first (`for each`) and last (`end`) fields identify the range of repeating columns. (Dynamic)
- A date field. Word updates this field with the current date. (Dynamic)

This example uses a blank Word template, but you could use a template with, for example, the banner already defined.

Figure 10–4 Sample Report Template Created in Word for BI Publisher



Global Enterprises, Inc.

Sales Analysis

Product	Customer	Time	Sales	Prior Period	Change	% Change
for-each TOTAL	TOTAL	2001	0	0	0	0 end

October 9, 2007

The following procedure defines the template manually. Alternatively, you can use a Word plugin called Oracle BI Publisher Desktop. On the BI Publisher My Folders page, click **Template Builder** to download the plugin.

To create a BI Publisher template in Word:

1. Open a new document in Word.
2. Compose the page according to your preferences.
3. For the query results, create a table.

   The table shown in Figure 10–4 is very simple. You can use much more elaborate formatting if you wish, including nested columns and tables.
4. From the View menu, choose **Toolbars**, then **Forms**.

   The Forms toolbar opens.
5. Enter a field in the body row of each column:
   a. Position the cursor in the appropriate cell.
   b. On the Forms toolbar, click the **Text Form Field** icon.

      The Text Form Field Options dialog box opens.
   c. Choose an appropriate Type, generally **Regular Text** for dimension labels and **Number** for measures.
   d. Enter a default value and a format.
   e. Click **Add Help Text**.

      The Form Field Help Text dialog box opens.
   f. Type the appropriate XML tag in the Type Your Own box, using the format `<?`*`tag`*`?>`.

      Enter the tag name exactly as it appears in the XML report. For example, enter `<?Product?>` for the XML tag `<Product>`.
   g. Click **OK** to close the Form Field Help dialog box.
   h. Click **OK** to close the Text Form Field Options dialog box.
6. Insert an additional form field at the beginning of the first column:
   a. In the Text Form Field Options dialog box, enter any default value, such as `For-Each`.
   b. In the Form Field Help Text dialog box, enter this text:

      ```
      <?for-each:ROW?>
      ```
7. Insert an additional form field at the end of the last column:
   a. In the Text Form Field Options dialog box, enter any default value, such as `End`.
   b. In the Form Field Help Text dialog box, enter this text:

      ```
      <?end for-each?>
      ```
8. Make any additional formatting changes in Word, such as the appropriate justification of the table headings and data columns.
9. Save the document as an RTF file.

Generating a Formatted Report

After creating a report template in Word, you can upload it to BI Publisher and associate it with your report definition. Then you can generate reports in a variety of formats.

Create a Report Layout:

1. Open the report editor in BI Publisher.
2. Select **Layouts**.

   The Create Layouts page opens.
3. Click **New**.

   The Layout page opens.
4. Enter a name and select **RTF** for the template type.
5. Select **Layouts** again, and select the new layout as the default template for this report.
6. Under Manage Template Files, click **Browse**. Select the RTF file you created.
7. Click **Upload**.

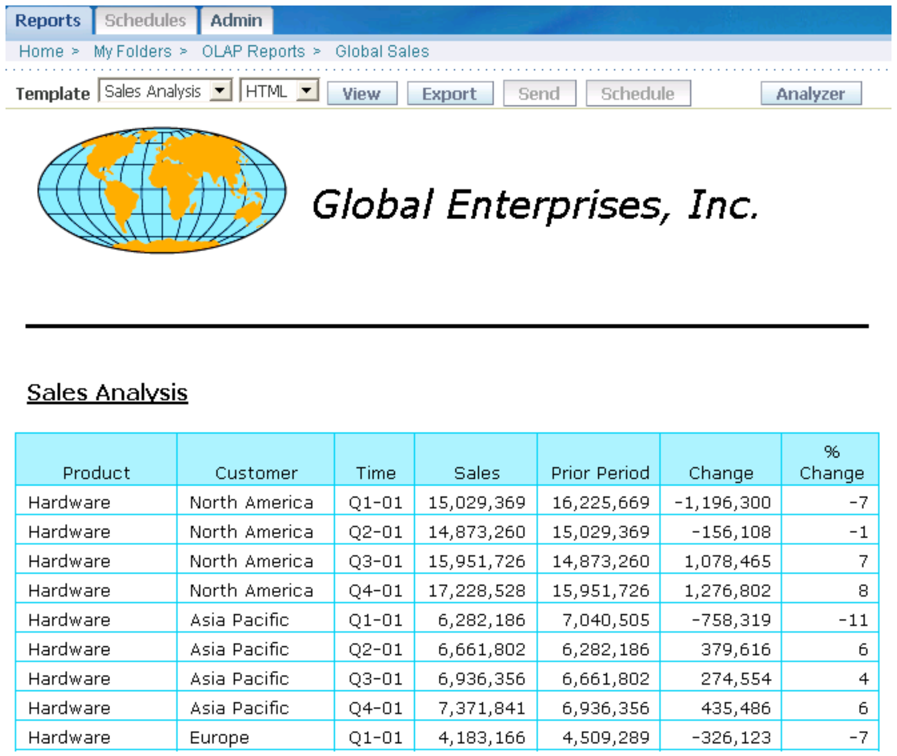
   The uploaded file is listed under Manage Template Files. Whenever you change the file in Word, upload it again. Otherwise, BI Publisher continues to use its copy of the previous version.
8. Click **Save**.
9. Click **View**.

   The report is displayed.
10. To change the format, select a new format from the list and click **View**.

    To see the XML, select **Data**.

Figure 10–5 shows the report in HTML format.

Figure 10–5 BI Publisher Report Displayed in HTML Format



Sales Analysis

Product	Customer	Time	Sales	Prior Period	Change	% Change
Hardware	North America	Q1-01	15,029,369	16,225,669	-1,196,300	-7
Hardware	North America	Q2-01	14,873,260	15,029,369	-156,108	-1
Hardware	North America	Q3-01	15,951,726	14,873,260	1,078,465	7
Hardware	North America	Q4-01	17,228,528	15,951,726	1,276,802	8
Hardware	Asia Pacific	Q1-01	6,282,186	7,040,505	-758,319	-11
Hardware	Asia Pacific	Q2-01	6,661,802	6,282,186	379,616	6
Hardware	Asia Pacific	Q3-01	6,936,356	6,661,802	274,554	4
Hardware	Asia Pacific	Q4-01	7,371,841	6,936,356	435,486	6
Hardware	Europe	Q1-01	4,183,166	4,509,289	-326,123	-7

Adding Dimension Choice Lists

You can add choice lists for the dimensions to a report. When generating a report, you can change the selection of data without changing the query. To add choice lists, take these steps:

- Create one or more Lists of Values (LOV) to be displayed in the menu.
- Create menus for displaying the LOVs.
- Edit the query to use the bind variables created for the menus.

Creating a List of Values

For an LOV, use a SQL query that selects the dimension keys that you want to display. Include the long description and dimension key columns. This example creates a list for the Product Primary hierarchy using a dimension view:

```
SELECT product_ldsc, product
    FROM product_dimview
    WHERE product_primary_prnt = 'TOTAL_PRODUCT_1'
       OR product = 'TOTAL_PRODUCT_1'
    ORDER BY product_level, product_ldsc;

PRODUCT_LDSC         PRODUCT
-------------------- --------------------
Hardware             CLASS_2
Software/Other       CLASS_3
Total Product        TOTAL_PRODUCT_1
```

To create a list of values:

1. Open the Report Editor in BI Publisher.

2. Select **List of Values**, then click **New**.

   The List of Values page opens.

3. Define the list:

   a. Enter a name for the list, such as `Product_LOV`.

   b. For the type, select **SQL Query**.

   c. Enter a query against a dimension view, as shown previously.

4. Click **Save**.

Repeat these steps for the other dimensions. This example uses lists for Product, Customer, and Time.

Creating a Menu

In BI Publisher, a menu is a type of parameter. Creating a parameter automatically creates a bind variable that you can use in the query for the report.

To create a menu:

1. Select Parameters, then click **New**.

   The Parameter page opens.

2. Define the parameter:

   a. For the Identifier, enter a name such as `product`.

      This is the case-sensitive name of the bind variable that you use in the query.

   b. Select an appropriate data type, typically String.

   c. For the Default Value, enter the dimension key used in the `WHERE` clause of the LOV query.

      The menu initially displays the label for this key.

   d. For the Parameter Type, select **Menu**.

   e. Select the appropriate List of Values.

   f. Clear all options.

3. Click **Save**.

Repeat these steps for the other dimensions. This example uses menus for Product, Customer, and Time.

Editing the Query

To activate the menus, you change the `WHERE` clause in the query so that the report uses the bind variables. The value of a bind variable is the current menu choice.

This is the format for the conditions of the `WHERE` clause:

```
parent_column = :bind_variable
```

In this example, the `WHERE` clause uses the bind variables for Time, Product, and Customer:

```
WHERE product_primary_prnt = :product
    AND customer_shipments_prnt = :customer
    AND time_calendar_yea_prnt = :time
    AND channel_level = 'TOTAL_CHANNEL'
```

To edit the query:

1. Under Data Model, select the data set you defined for this report.

   The Data Set page opens.

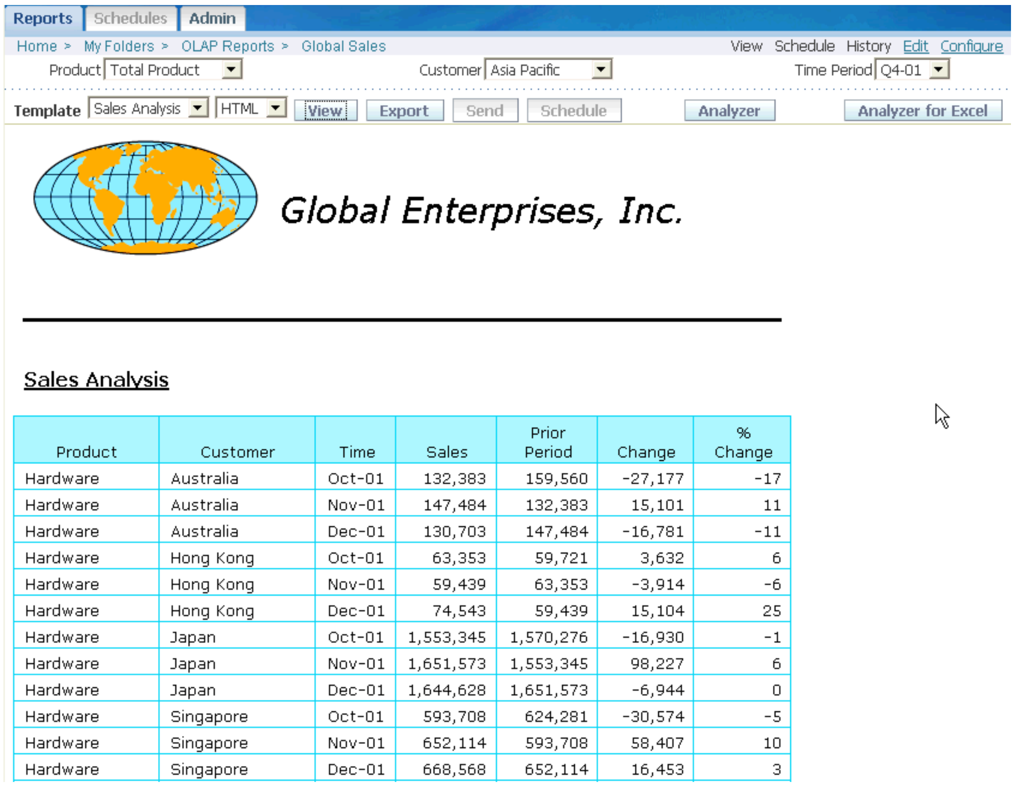
2. In the SQL Query box, edit the `WHERE` clause to use the bind variables created by the parameter definitions.

3. Click **Save**.

Figure 10–6 shows a report in HTML format displayed in BI Publisher. The choice lists for Product, Customer, and Time appear across the top. The crosstab lists the Hardware products, the countries in Asia Pacific, and the months in Q4-01. To see a different selection of data, you can choose a Time Period, Product, and Customer from the menus, then click **View**. This report was generated by the same report entry, using the same query, as the reports shown in Figure 10–1 and Figure 10–5.

You can continue working on this report, adding charts and other tables.

Figure 10–6 Sales Report With Choice Lists in BI Publisher



Sales Analysis

Product	Customer	Time	Sales	Prior Period	Change	% Change
Hardware	Australia	Oct-01	132,383	159,560	-27,177	-17
Hardware	Australia	Nov-01	147,484	132,383	15,101	11
Hardware	Australia	Dec-01	130,703	147,484	-16,781	-11
Hardware	Hong Kong	Oct-01	63,353	59,721	3,632	6
Hardware	Hong Kong	Nov-01	59,439	63,353	-3,914	-6
Hardware	Hong Kong	Dec-01	74,543	59,439	15,104	25
Hardware	Japan	Oct-01	1,553,345	1,570,276	-16,930	-1
Hardware	Japan	Nov-01	1,651,573	1,553,345	98,227	6
Hardware	Japan	Dec-01	1,644,628	1,651,573	-6,944	0
Hardware	Singapore	Oct-01	593,708	624,281	-30,574	-5
Hardware	Singapore	Nov-01	652,114	593,708	58,407	10
Hardware	Singapore	Dec-01	668,568	652,114	16,453	3

Developing a Dashboard Using Application Express

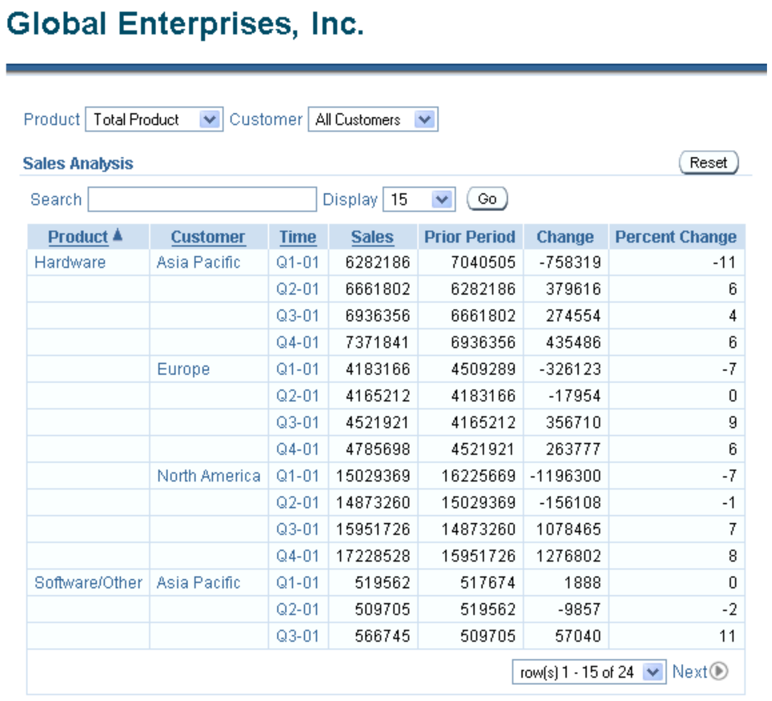
Oracle Application Express is a rapid Web application development tool for Oracle Database. Application Express offers built-in features such as user interface themes, navigational controls, form handlers, and flexible reports, which simplify the development process. You can easily create dashboards from your cubes that display the rich analytical content generated by Oracle OLAP.

If you have not used Application Express, you can download the software, tutorials, and full documentation from the Oracle Technology Network at

`http://www.oracle.com/technetwork/developer-tools/apex/overview/index.html`

Figure 10–7 shows a crosstab with display lists for Product and Customer, and links in all three dimension columns. Choosing a new Product or Customer changes the related column to show the children for the selected key. Clicking a dimension key in any column displays its children. The Reset button refreshes the page with the initial selection of data.

Figure 10–7 Drillable Dimensions in Application Express



Global Enterprises, Inc.

Product: Total Product
Customer: All Customers

Sales Analysis

Search
Display: 15

Product	Customer	Time	Sales	Prior Period	Change	Percent Change
Hardware	Asia Pacific	Q1-01	6282186	7040505	-758319	-11
		Q2-01	6661802	6282186	379616	6
		Q3-01	6936356	6661802	274554	4
		Q4-01	7371841	6936356	435486	6
	Europe	Q1-01	4183166	4509289	-326123	-7
		Q2-01	4165212	4183166	-17954	0
		Q3-01	4521921	4165212	356710	9
		Q4-01	4785698	4521921	263777	6
	North America	Q1-01	15029369	16225669	-1196300	-7
		Q2-01	14873260	15029369	-156108	-1
		Q3-01	15951726	14873260	1078465	7
		Q4-01	17228528	15951726	1276802	8
Software/Other	Asia Pacific	Q1-01	519562	517674	1888	0
		Q2-01	509705	519562	-9857	-2
		Q3-01	566745	509705	57040	11

row(s) 1 - 15 of 24

Creating an OLAP Application in Application Express

In Application Express, the Administrator creates a **workspace** in which you can develop your Web applications. An **application** consists of one or more HTML pages, a **page** consists of regions that identify specific locations on the page, and a **region** contains a report (crosstab), a chart, or some other item.

Application Express runs in Oracle Database. If your dimensional objects are stored in a different database, then you must use a database link in your queries. The following procedure assumes that you have a workspace and access to at least one cube. It creates an application with one page containing a crosstab.

To create a Web page from a SQL query:

1. Open a browser to the Application Express home page and log in.
2. Click the **Application Builder** icon.

   The Application Builder opens.
3. Click **Create**.

   The Create Application wizard opens.
4. Select **Create Application**, then **Next**.
5. On the Name page, enter a title for the application such as `Global Dashboard` and select **From Scratch**.

6. On the Pages page, select the **Report** page type, then define the page:

   a. For Page Source, select **SQL Query**.

   b. For Page Name, enter a name such as `Sales Analysis`.

      This title is displayed on the page.

   c. For Query, enter a SQL `SELECT` statement for your cube, like the one shown in Example 10–1. Do not include an `ORDER BY` clause or a semicolon.

   d. Click **Add Page**.

      The page definition appears in the Create Application Box.

7. Click **Next**, then complete the Create Application wizard according to your own preferences.

   This example was created with no tabs, no shared components, no authentication, and Theme 15 (Light Blue).

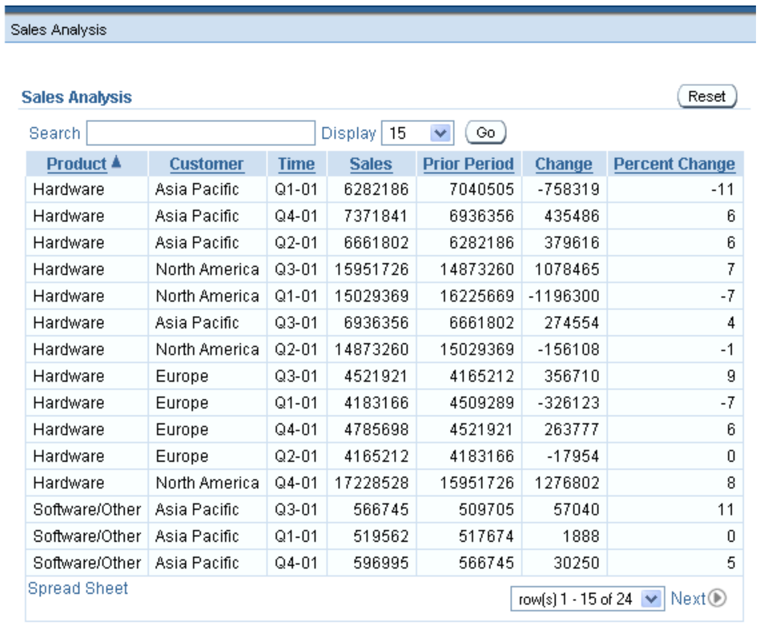
8. On the Confirm page, click **Create**.

9. On the Application Builder home page, click the **Run Application** icon.

   Tip: To continue working on this page, click the **Edit Page 1** link at the bottom of the display.

Figure 10–8 shows the results of the query displayed in Application Express. Several items are automatically added to the page: breadcrumbs, Search box, Display list, Go button, Reset button, and Spread Sheet link. This application only needs the Reset button, so you can delete the other items if you wish.

Figure 10–8 Basic Sales Report in Application Express



Product	Customer	Time	Sales	Prior Period	Change	Percent Change
Hardware	Asia Pacific	Q1-01	6282186	7040505	-758319	-11
Hardware	Asia Pacific	Q4-01	7371841	6936356	435486	6
Hardware	Asia Pacific	Q2-01	6661802	6282186	379616	6
Hardware	North America	Q3-01	15951726	14873260	1078465	7
Hardware	North America	Q1-01	15029369	16225669	-1196300	-7
Hardware	Asia Pacific	Q3-01	6936356	6661802	274554	4
Hardware	North America	Q2-01	14873260	15029369	-156108	-1
Hardware	Europe	Q3-01	4521921	4165212	356710	9
Hardware	Europe	Q1-01	4183166	4509289	-326123	-7
Hardware	Europe	Q4-01	4785698	4521921	263777	6
Hardware	Europe	Q2-01	4165212	4183166	-17954	0
Hardware	North America	Q4-01	17228528	15951726	1276802	8
Software/Other	Asia Pacific	Q3-01	566745	509705	57040	11
Software/Other	Asia Pacific	Q1-01	519562	517674	1888	0
Software/Other	Asia Pacific	Q4-01	596995	566745	30250	5

Adding Dimension Choice Lists

Like BI Publisher, Application Express enables you to drill on the dimensions by adding choice lists of dimension keys. The dashboard user can choose a particular item from the list and dynamically change the selection of data displayed in one or more graphics and crosstabs on the page. To implement a choice list, take these steps:

- Create a new region on the page to display the list.
- Create a list of values (LOV).
- Create a list item with a bind variable to display the LOV.
- Create an unconditional branch for the list.
- Edit the query to use the bind variable.

The Page Definition is where you can create new pages and edit existing ones, including adding new graphical items and modifying existing ones. The items are organized in three columns: Page Rendering, Page Processing, and Shared Components.

To open the Page Definition:

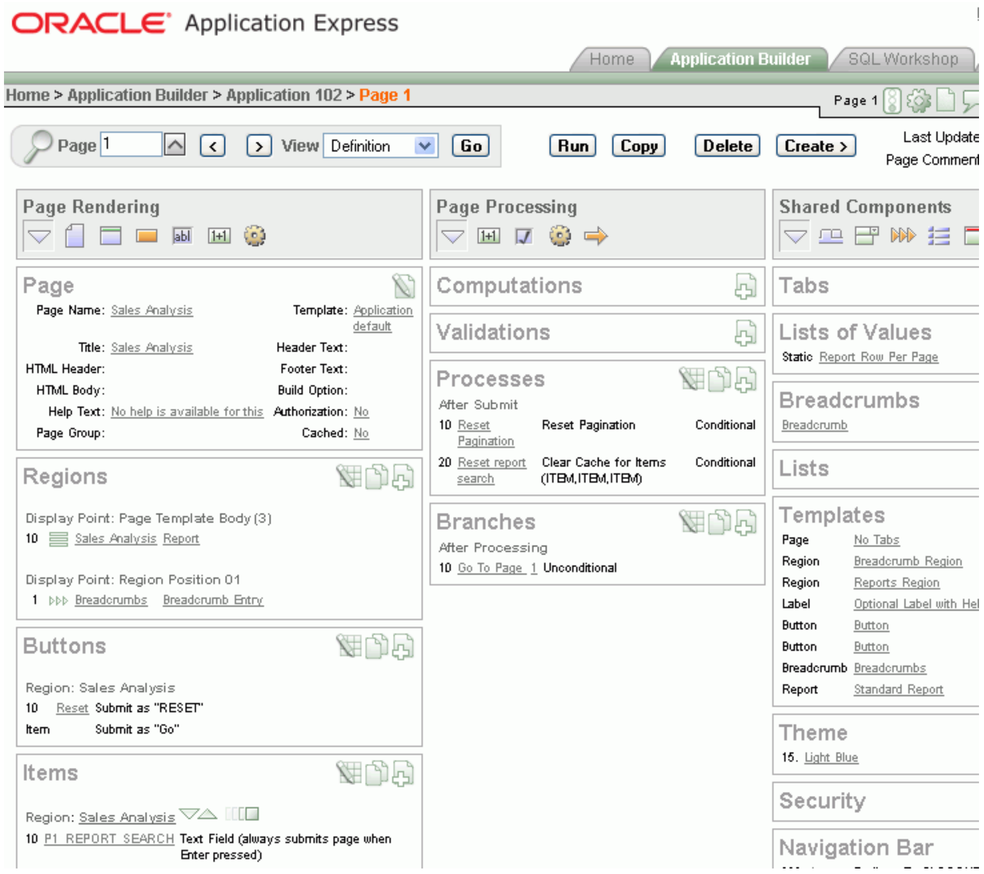
After running the application, click the **Edit Page** link at the bottom of the page.

or

On the Application home page, click the icon for the page where the report is defined.

Figure 10–9 shows an area of the Page Definition.

Figure 10–9 Application Express Page Definition



Home > Application Builder > Application 102 > Page 1

Page 1 View Definition

Page Rendering

Page

Page Name:	Sales Analysis	Template:	Application default
Title:	Sales Analysis	Header Text:	
HTML Header:		Footer Text:	
HTML Body:		Build Option:	
Help Text:	No help is available for this	Authorization:	No
Page Group:		Cached:	No

Regions

Display Point: Page Template Body (3)
10 Sales Analysis Report

Display Point: Region Position 01
1 Breadcrumbs Breadcrumb Entry

Buttons

Region: Sales Analysis
10 Reset Submit as "RESET"
Item Submit as "Go"

Items

Region: Sales Analysis
10 P1_REPORT_SEARCH Text Field (always submits page when Enter pressed)

Page Processing

Computations

Validations

Processes

After Submit

10	Reset Pagination	Reset Pagination	Conditional
20	Reset report search	Clear Cache for Items (ITEM,ITEM,ITEM)	Conditional

Branches

After Processing
10 Go To Page 1 Unconditional

Shared Components

Tabs

Lists of Values

Static Report Row Per Page

Breadcrumbs

Breadcrumb

Lists

Templates

Page	No Tabs
Region	Breadcrumb Region
Region	Reports Region
Label	Optional Label with Hel
Button	Button
Button	Button
Breadcrumb	Breadcrumbs
Report	Standard Report

Theme

15. Light Blue

Security

Navigation Bar

Creating a Region

You can create the choice list in a plain HTML area at the top of the page.

To create an empty HTML region:

1. On the Page Definition under Regions, click the **Create** icon.

   The Create Region wizard opens.

2. On the Region pages, select **HTML**, click **Next**, then select **HTML** again.
3. On the Display Attributes page, enter a descriptive title and select an appropriate template and location on the page for the lists.

   For this example, the title is `lov_region`, the region template is **No Template**, and the location is **Page Template Body (1 items below region content)**. The name can be displayed on the rendered page, but it is hidden in this example.

4. Click **Next**, then **Create Region**.

   The new region appears on the Page Definition under Regions.

Creating a List of Values

For a list of values, use a SQL query like the one shown here. Include the description and key columns from the dimension view. This query creates a list for the Global Customer Shipments hierarchy:

```
SELECT customer_ldsc, customer
    FROM customer_dimview
    WHERE customer_shipments_prnt = 'TOTAL_CUSTOMER_1'
    OR customer = 'TOTAL_CUSTOMER_1'
    ORDER BY customer_level, customer_ldsc;

CUSTOMER_LDSC        CUSTOMER
-------------------- --------------------
Asia Pacific         REGION_8
Europe               REGION_9
North America        REGION_10
All Customers        TOTAL_CUSTOMER_1
```

To create a List of Values

1. On the Page Definition under Lists of Values, click the **Create** icon.

   The Create List of Values wizard opens.

2. On the Source page, select **From Scratch.**
3. On the Name and Type page, enter a descriptive name and select **Dynamic**.

   This example uses the name `CUSTOMER_LOV`.

4. On the Query page, enter a query like the one shown previously. Do not use a semicolon.
5. Click **Create List of Values**.

   The new LOV appears in the Page Definition under List of Values.

For additional LOVs, repeat these steps. This example creates LOVs for the Product and Customer dimensions.

Creating the Choice List

For a choice list, you create a list item that displays the LOV.

To create a list item:

1. On the Page Definition under Items, click the **Create** icon.

The Create Item wizard opens.

2. On the Item Type page, select **Select List.**
3. For Control Type, select **Select List with Submit.**
4. On the Display Position and Name page:
   - Enter a name that identifies the dimension, such as `P1_CUSTOMER` for the name of the Customer bind variable. `P1` is the page number, and `CUSTOMER` identifies the Customer dimension.
   - Choose the new HTML region for the location of the list.
5. On the List of Values page, set these values:
   - Named LOV to the List Of Values created for this dimension, such as `CUSTOMER_LOV`.
   - Display Null Option to **No**.
6. Select the Item attributes according to your own preferences.
7. On the Source page, enter the name of the top dimension key for the default value.

   For the Global Customer dimension, the value is `TOTAL_CUSTOMER_1`.
8. Click **Create Item**.

Repeat these steps for other lists. This example creates lists for the Product and Customer dimensions.

To activate the list item:

1. On the Page Definition under Branches, click the **Create** icon.

   The Edit Branch wizard opens.
2. On the Point and Type page, accept the default settings.
3. On the Target page:
   - Set Branch Target to **Page in This Application**.
   - Set Page to the page with the list item, which is `1` in this example.
   - Select **Reset Pagination For This Page**.
4. On the Branch Conditions page, accept the default settings to create an unconditional branch.
5. Click **Create Branch**.

   The Edit Branch page closes, and you return to the Page Definition. The new unconditional branch is listed under Branches.

Editing the Query

This is the format for the dynamic conditions in the `WHERE` clause:

```
parent_column = NVL(:bind_variable, 'top_key')
```

The `NVL` function substitutes the name of the top dimension key in the hierarchy for null values. The dimension keys at the top have no parent key.

To edit the query:

1. Open the Page Definition.

2. Under Regions, click the **Edit Region** link. In this example, the region is named Sales Report.

   The Edit Region page opens.

3. Under Source, modify the query:

   - Change the `WHERE` clause to use the bind variables.
   - Delete the outer `SELECT` added by Application Express.

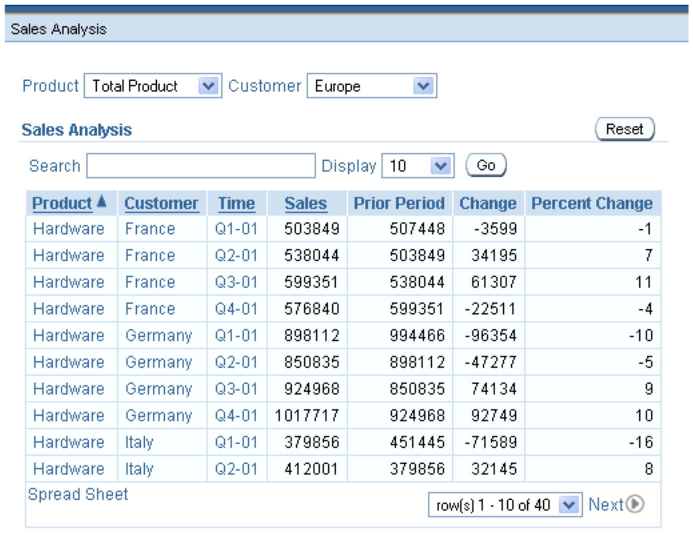
4. Click **Apply Changes**.

For this example, the `WHERE` clause now looks like this:

```
WHERE product_primary_prnt = NVL(:P1_PRODUCT, 'TOTAL_PRODUCT_1')
       AND customer_shipments_prnt = NVL(:P1_CUSTOMER, 'TOTAL_CUSTOMER_1')
       AND time_calendar_yea_prnt = 'YEAR_4'
       AND channel_level = 'TOTAL_CHANNEL'
```

Figure 10–10 shows the modified page with choice lists for Product and Customer.

Figure 10–10 Dashboard With Choice Lists for Drilling



Product	Customer	Time	Sales	Prior Period	Change	Percent Change
Hardware	France	Q1-01	503849	507448	-3599	-1
Hardware	France	Q2-01	538044	503849	34195	7
Hardware	France	Q3-01	599351	538044	61307	11
Hardware	France	Q4-01	576840	599351	-22511	-4
Hardware	Germany	Q1-01	898112	994466	-96354	-10
Hardware	Germany	Q2-01	850835	898112	-47277	-5
Hardware	Germany	Q3-01	924968	850835	74134	9
Hardware	Germany	Q4-01	1017717	924968	92749	10
Hardware	Italy	Q1-01	379856	451445	-71589	-16
Hardware	Italy	Q2-01	412001	379856	32145	8

Drilling on Dimension Columns

You can enable users to drill down from the top of a hierarchy to the detail level using a single query. To implement drilling in Application Express, take these steps:

- Create hidden items with bind variables.
- Edit the query to use the bind variables.
- Add links to the dimension columns of the crosstab.

This example adds drilling to all displayed dimensions.

Creating Hidden Items

You can create various types of items in Application Express that provide bind variables. They store the session state for a particular element, in this case, the current selection of a parent dimension key.

Each dimension that supports drilling needs a bind variable. In this example, Product and Customer already have bind variables created with the list items. Time is the only displayed dimension in the report that does not have a bind variable. Because links in the Time dimension column provide the user interface for changing the session state, Time does not need any other graphical user interface. A hidden item serves the purpose.

To create a hidden item:

1. Open the Page Definition.
2. Under Items, click the **Create** icon.

   The Create Item wizard opens.
3. On the Item Type page, select **Hidden**.
4. On the Display Position and Name page:
   - Enter a name that identifies the dimension, such as `P1_TIME` for the name of the Time bind variable.
   - Choose the region where the report is defined.
5. On the Source page, enter the dimension key at the top of the hierarchy as the default value.

   `TOTAL` is the top of all hierarchies in the Global schema. For this example, Time is set to `YEAR_4` to restrict the selection to 2001.
6. Click **Create Item**.
7. Repeat these steps for any other dimensions that supports drilling only on the column links.

   For this example, a hidden item is defined for Time.

Editing the Query

To add column links to a report, you must change two areas of the `SELECT` statement:

- Select list: Application Express manages only those columns that appear in the select list. You can choose to display or hide the columns. For defining the column links, add the key and parent columns in the cube view to the query select list.
- `WHERE` clause: Add the bind variables for the hidden items like you did for the choice lists in "Editing the Query" on page 10-10.

Example 10–3 shows the modified sample query.

Example 10–3 Revised Query for Column Links in Application Express

```
SELECT product_ldsc "Product",
       customer_ldsc "Customer",
       time_ldsc "Time",
       round(sales) "Sales",
       round(sales_pp) "Prior Period",
       round(sales_chg_pp) "Change",
       round(sales_pct_chg_pp * 100) "Percent Change",
/* Add dimension keys and parents */
       product product_key,
       product_primary_prnt product_parent,
       customer customer_key,
       customer_shipments_prnt customer_parent,
       time time_key,
       time_calendar_yea_prnt time_parent
/* From cube view */
    FROM units_cube_cubeview
/* Use parent columns and bind variables for drilling */
    WHERE product_primary_prnt = NVL(:P1_PRODUCT, 'TOTAL_PRODUCT_1')
       AND customer_shipments_prnt = NVL(:P1_CUSTOMER, 'TOTAL_CUSTOMER_1')
       AND time_calendar_yea_prnt = NVL(:P1_TIME, 'YEAR_4')
       AND channel_level = 'TOTAL_CHANNEL'
```

Adding Links to the Dimension Columns

When a dashboard user clicks a linked dimension key in the crosstab, the value of the bind variable changes, causing the crosstab to change also. After drilling down a hierarchy, the user can restore the display to its original selection of data by pressing the Reset button. To implement these column links, you must add the column links and activate the Reset button.

To add a link to a dimension column:

1. Open the Page Definition.
2. Under Regions, click the **Report** link.

   The Report Attributes page opens.
3. Under Column Attributes, modify the report display:
   - Clear the Show check boxes for columns that you want to hide, such as the dimension key and parent columns.
   - Set the Sort and Sort Sequence check boxes for appropriate sorting for the report. In this example, the sort order is `Product` (1), `Customer` (2), and `Time` (3).
4. Click the **Edit** icon for a dimension column.

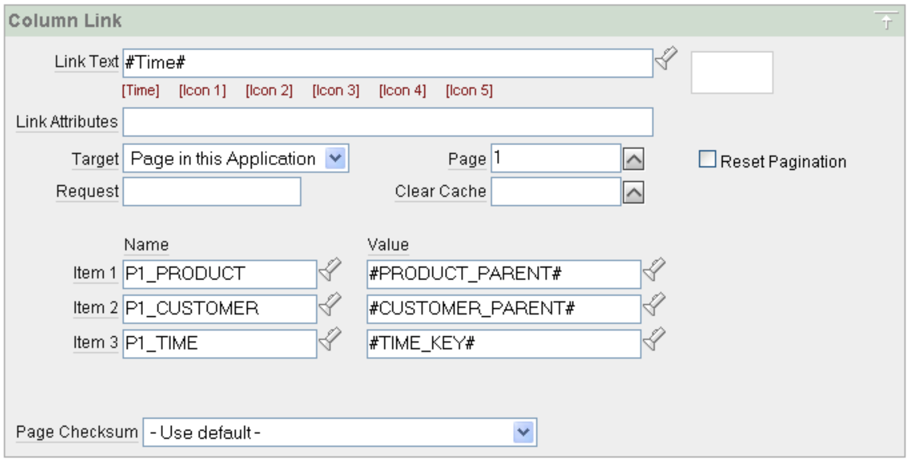
   The Column Attributes page opens.
5. Under Column Link, define the link as follows:
   - Link Text: Choose the dimension name.
   - Page: Enter the page number.
   - Name: List the dimensions in the order they appear in the report. **Item** is the name of the bind variable. **Value** is the key column for the dimension being defined or the parent column for the other dimensions.

   Figure 10–11 shows the link definition for the Time dimension.
6. Click **Apply Changes**.

   The Column Attributes page closes, and you return to the Report Attributes page.
7. Define links on the other dimension columns.
8. Click **Apply Changes**.

   The Report Attributes page closes, and you return to the Page Definition.

Figure 10–11 Definition of the Time Link


Column Link

Link Text	#Time#		
	[Time] [Icon 1] [Icon 2] [Icon 3] [Icon 4] [Icon 5]		
Link Attributes			
Target	Page in this Application	Page	1
Request		Clear Cache	

☐ Reset Pagination

	Name	Value
Item 1	P1_PRODUCT	#PRODUCT_PARENT#
Item 2	P1_CUSTOMER	#CUSTOMER_PARENT#
Item 3	P1_TIME	#TIME_KEY#

Page Checksum: - Use default -

To activate the Reset button:

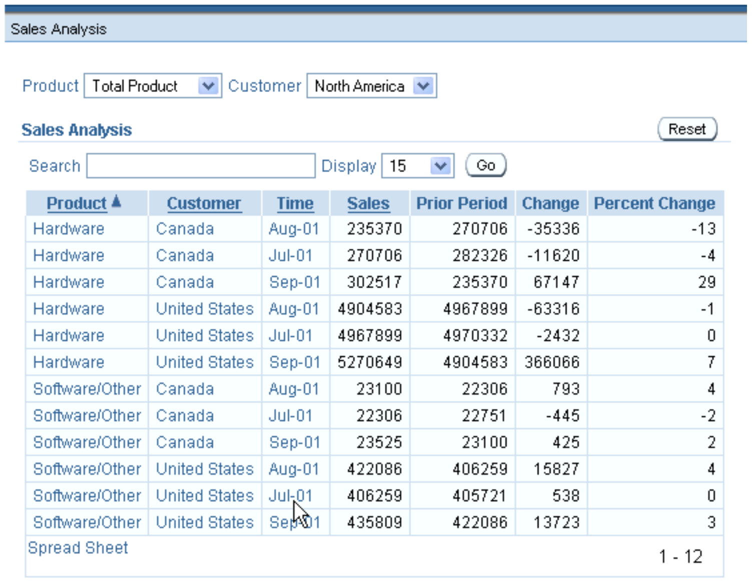
1. Open the Page Definition.
2. Under Branches, click the **Go to Page conditional** link.

   The Reset button was created on the page automatically along with its conditional branch. The Edit Branch page opens.
3. Under Action, set Clear Cache to the page number (in this example, `1`).
4. Under Conditions, set When Button Pressed to **RESET**.
5. Click **Apply Changes**.

   The Edit Branch page closes, and you return to the Page Definition.
6. Click **Run** to display the page.

Figure 10–12 shows the finished page displaying months in Q3.01. You can continue working on this application, adding more reports and charts to the page. For the SQL queries providing data to those reports and charts, you can reuse the same bind variables for the dimensions.

Figure 10–12 Sales Report With Column Links in Application Express



Sales Analysis

Product: Total Product | Customer: North America

Sales Analysis

Search | Display: 15 | Go | Reset

Product ▲	Customer	Time	Sales	Prior Period	Change	Percent Change
Hardware	Canada	Aug-01	235370	270706	-35336	-13
Hardware	Canada	Jul-01	270706	282326	-11620	-4
Hardware	Canada	Sep-01	302517	235370	67147	29
Hardware	United States	Aug-01	4904583	4967899	-63316	-1
Hardware	United States	Jul-01	4967899	4970332	-2432	0
Hardware	United States	Sep-01	5270649	4904583	366066	7
Software/Other	Canada	Aug-01	23100	22306	793	4
Software/Other	Canada	Jul-01	22306	22751	-445	-2
Software/Other	Canada	Sep-01	23525	23100	425	2
Software/Other	United States	Aug-01	422086	406259	15827	4
Software/Other	United States	Jul-01	406259	405721	538	0
Software/Other	United States	Sep-01	435809	422086	13723	3

Spread Sheet

1 - 12

11

Developing Java Applications for OLAP

This chapter presents the rich development environment and the powerful tools that you can use to create OLAP-aware applications in Java. It includes the following topics:

- Building Analytical Java Applications
- Introducing OracleBI Beans
- Building Java Applications That Manage Analytic Workspaces

Building Analytical Java Applications

Java is the language of the Internet. Using Java, application developers can write standalone Java applications (which can be launched from a browser with Java's WebStart technology) or HTML applications that access live data from Oracle Database, through servlets, JavaServer Pages (JSP), and Oracle User Interface XML (UIX).

About Java

Java is the preferred programming language for an ever-increasing number of professional software developers. For those who have been programming in C or C++, the move to Java is easy because it provides a familiar environment while avoiding many of the shortcomings of the C language. Developed by Sun Microsystems, Java is fast superseding C++ and Visual Basic as the language of choice for application developers, for the following reasons:

- Object oriented. Java enables application developers to focus on the data and methods of manipulating that data, rather than on abstract procedures; the programmer defines the desired object rather than the steps needed to create that object. Almost everything in Java is defined as an object.
- Platform independent. The Java compiler creates byte code that is interpreted at runtime by the Java Virtual Machine (JVM). As the result, the same software can run on all Windows, Linux, Unix, and Macintosh platforms where the JVM has been installed. All major browsers have the JVM built in.
- Network based. Java was designed to work over a network, which enables Java programs to handle remote resources as easily as local resources.
- Secure. Java code is either trusted or untrusted, and access to system resources is determined by this characteristic. Local code is trusted to have full access to system resources, but downloaded remote code (that is, an applet) is not trusted. The Java "sandbox" security model provides a very restricted environment for untrusted code.

The Java Solution for OLAP

To develop an OLAP application, you can use the Java programming language. Java enables you to write applications that are platform-independent and easily deployed over the Internet.

The OLAP API is a Java-based application programming interface that provides access to dimensional data for analytical business applications. Java classes in the OLAP API provide all of the functions required of an OLAP application: Connection to an OLAP instance; authentication of user credentials; access to data in the RDBMS controlled by the permissions granted to those credentials; and selection and manipulation of that data for business analysis.

OracleBI Beans simplifies application development by providing these functions as JavaBeans. Moreover, OracleBI Beans includes JavaBeans for presenting the data in graphs and crosstabs.

Note: Oracle JDeveloper and OracleBI Beans are not packaged with the Oracle RDBMS.

The OLAP API has a companion interface that can be used to build applications for OLAP DBAs. The OLAP Analytic Workspace Java API is a set of Java classes and an XML schema for designing, building, and updating analytic workspaces in the Oracle Database. For more information, see "Building Java Applications That Manage Analytic Workspaces" on page 11-5.

Oracle Java Development Environment

Oracle JDeveloper provides an integrated development environment (IDE) for developing Java applications. Although third-party Java IDEs can also be used effectively, only JDeveloper achieves full integration with the Oracle Database and OracleBI Beans wizards. The following are a few JDeveloper features:

- Remote graphical debugger with break points, watches, and an inspector.
- Multiple document interface (MDI)
- *Codecoach* feature that helps you to optimize your code
- Generation of 100% Pure Java applications, applets, servlets, Java beans, and so forth with no proprietary code or markers
- Oracle Database browser

Note: Oracle JDeveloper is an application and is not packaged with Oracle Database.

Introducing OracleBI Beans

OracleBI Beans provides reusable components that are the basic building blocks for OLAP decision support applications. Using OracleBI Beans, developers can rapidly develop and deploy new applications, because these large functional units have already been developed and tested — not only for their robustness, but also for their ease of use. And because OracleBI Beans provides a common look and feel to OLAP applications, the learning curve for end users is greatly reduced.

OracleBI Beans includes the following:

- **Presentation beans** display the data in a rich variety of formats so that trends and variations can easily be detected. Among the presentation beans currently available are Graph and Crosstab.
- **Data beans** acquire and manipulate the data. The data beans use the OLAP API to connect to a data source, define a query, manipulate the resultant data set, and return the results to the presentation beans for display. Data beans include a QueryBuilder, a CalcBuilder, and a Metadata Manager.
- **Persistence Service** is a set of packages that support the storage and retrieval of objects in the OracleBI Beans Catalog, not only so that you can save your work, but also so that you can share the work with others who have access to the Catalog.

OracleBI Beans can be incorporated in a Java client or an HTML client application. Java clients best support users who do immersed analyses, that is, use the system for extensive periods of time with a lot of interaction. For example, users who create reports benefit from a Java client. HTML clients best support remote users who use a low bandwidth connection and have basic analytical needs. Thin clients can be embedded in a portal or other Web site for these users.

Metadata

The OLAP API and OracleBI Beans use the logical model that is projected by the **Active Catalog** to obtain the information they need about dimensional objects defined in analytic workspaces. They use OLAP Catalog metadata to obtain information about dimensional objects defined in Oracle relational data warehouses.

OracleBI Beans generates additional metadata to support its additional functionality. This additional metadata is contained in the OracleBI Beans Catalog. The Metadata Manager presents applications with a consolidated view of metadata from the Active Catalog, OLAP Catalog, and the OracleBI Beans Catalog. For example, in the QueryBuilder, the measures obtained from the Active Catalog and the custom measures obtained from the OracleBI Beans Catalog appear together.

Navigation

The presentation beans support navigation techniques such as drilling, pivoting, and paging.

- **Drilling** displays lower-level values that contribute to a higher-level aggregate, such as the cities that contribute to a state total.
- **Pivoting** rotates the data cube so that the dimension members that labeled a graph series now label groups, or the dimension members that labeled columns in a crosstab now label rows instead. For example, if products label the rows and regions label the columns, then you can pivot the data cube so that products label the columns and regions label the rows.
- **Paging** handles additional dimensions by showing each member in a separate graph, crosstab, or table rather than nesting them in the columns or rows. For example, you might want to see each time period in a separate graph rather than all time periods on the same graph.

Formatting

The presentation beans enable you to change the appearance of a particular display. In addition, the values of the data itself can affect the format.

- Number formatting. Numerical displays can be modified by changing their scale, number of decimal digits and leading zeros, currency symbol, negative notation, and so forth.
- Stoplight formatting. The formatting of the cell background color, border, font, and so forth can be data driven so that outstanding or problematic results stand out visually from the other data values.

Graphs

The Graph bean presents data in a large selection of two- and three-dimensional business graph types, such as bar, area, line, pie, ring, scatter, bubble, pyramid, and stock market. Most graph types have several subtypes, such as clustered bar, stacked bar, and percent bar.

Bar, line, and area graphs can be combined so that individual rows in the data cube can be specified as one of these graph types. You can also assign marker shape and type, data line type, color, fill color, and width and on a row-by-row basis, depending on the type of graph.

The graph image can be exported in PNG and other image formats.

Users can zoom in and out of selected areas of a graph. They can also scroll across the axes.

Crosstabs

The Crosstab bean presents data in a two-dimensional grid similar to a spreadsheet. Multiple dimensions can be nested along the rows or columns, and additional dimensions can appear as separate pages. Among the available customizations are: Font style, size, and color; data-driven formatting, stoplight reporting, and underlining; individual cell background colors; border formats; and text alignment.

Users can navigate through the data using either a mouse or the keyboard.

Data Beans

The data beans use the OLAP API to provide the basic services needed by an application. They enable clients to identify a database, present credentials for accessing that database, and make a connection. The application can then access the metadata and identify the available data. Users can select the measures they want to see and the specific slice of data that is of interest to them. That data can then be modified and manipulated.

Wizards

OracleBI Beans offers wizards that can be used both by application developers in creating an initial environment and by end users in customizing applications to suit their particular needs. The wizards lead you step-by-step so that you provide all of the information needed by an application. The following are some of the tasks that can be done using wizards.

- **Building a query**. Fact tables and materialized views often contain much more data than users are interested in viewing. Fetching vast quantities of data can also degrade performance unnecessarily. In addition to selecting measures, you can limit the amount of data fetched in a query by selecting dimension members from a list or using a set of conditions. Selections can be saved, and these saved selections can be used again just by picking their names from a list.

OracleBI Beans takes advantage of all of the new OLAP functions in the database, including ranking, lag, lead, and windowing. End users can create powerful queries that ask sophisticated analytical questions, without knowing SQL at all.

- **Generating custom measures**. You can define new "custom" measures whose values are calculated from data stored within the database. For example, a user might create a custom measure that shows the percent of change in sales from a year ago. The data in the custom measure would be calculated using the lag method on data in the Sales measure. Because a DBA cannot anticipate and create all of the calculations required by all users, OracleBI Beans enables users to create their own.

JSP Tag Library

OracleBI Beans includes an extensive JSP tag library that enables the development of applications without writing custom code. After you use wizards to create the presentations that are needed for an application, you can use JSP tags to insert the presentations in HTML pages and to create additional pages for the user interface.

The tags in this library are grouped in the following categories:

- General tags. Used to represent objects such as graphs, crosstabs, formatting tools, explorers for the OracleBI Beans Catalog, and controls for displaying messages; also includes a tag that lets you link the queries of graphs and crosstabs.
- Dialog and wizard tags. Used to create user interface elements that let end users manipulate presentations. For example, these tags let users change the type of a graph or export crosstab data.
- List tags. Used to create lists that let end users perform the following kinds of tasks: Modify queries by selecting dimensions or measures; browse for graphs or crosstabs in the Catalog; and navigate pages in an application.

OracleBI Beans also includes an extensive UIX tag library.

Building Java Applications That Manage Analytic Workspaces

The Analytic Workspace application programming interface is a companion API to the OLAP API and OracleBI Beans. You can use the Analytic Workspace API to build Java applications that create and maintain analytic workspaces.

The Analytic Workspace API provides a set of Java classes that:

- Create a logical dimensional model of cubes, dimensions, measures, and attributes
- Define a set of mappings for loading data from relational columns into objects in the logical model
- Define the aggregation rules for data in the logical model
- Define advanced analytics such as allocations, forecasts, and models on objects in the logical model
- Instantiate the logical model in an analytic workspace

The Analytic Workspace API supports two deployment modes: It can be embedded in a Java application; or it can be used to generate XML that is executable by the `DBMS_AW_XML.EXECUTE` PL/SQL function. `DBMS_AW_XML.EXECUTE` can process any XML document that has been validated against the OLAP XML schema.

See Also:

- *Oracle OLAP Analytic Workspace Java API Reference*
- *Oracle OLAP Reference* for information on `DBMS_AW_XML.EXECUTE`

12 Administering Oracle OLAP

This chapter describes the various administrative tasks that are associated with Oracle OLAP. It contains the following topics:

Setting Database Initialization Parameters

Table 12–1 identifies the parameters that affect the performance of Oracle OLAP. Alter your server parameter file or `init.ora` file to these values, then restart your database instance. You can monitor the effectiveness of these settings and adjust them as necessary.

See Also:

- *Oracle Database Performance Tuning Guide* for information about tuning parameter settings
- *Oracle Database Reference* for descriptions of individual parameters

Table 12–1 Initial Settings for Database Parameter Files

Parameter	Default Value	Recommended Setting	Description
`JOB_QUEUE_PROCESSES`	0	Number of CPUs, plus one additional process for every three CPUs; in a multi-core CPU, each core counts as a CPU For example, `JOB_QUEUE_PROCESSES=5` for a four-processor computer	Controls the degree of parallelism in OLAP builds
`PGA_AGGREGATE_TARGET`	10 MB or 20% SGA	50% of physical memory to start, then tune as indicated by performance statistics	
`SGA_TARGET`	0	25% or less of physical memory to start, then tune as indicated by performance statistics	
`SESSIONS`	Derived	2.5 * maximum number of simultaneous OLAP users	Provides sufficient background processes for each user
`UNDO_MANAGEMENT`	`MANUAL`	`AUTO`	Specifies use of an undo tablespace
`UNDO_TABLESPACE`	Derived	Name of the undo tablespace, which must already be defined	Identifies the undo tablespace defined for OLAP use, as shown in "Creating an Undo Tablespace" on page 12-3

To set the system parameters:

1. Open the `init.ora` initialization file in a text editor.
2. Add or change the settings in the file.
3. Stop and restart the database, using commands such as the following. Be sure to identify the initialization file in the `STARTUP` command.

```
SQLPLUS '/ AS SYSDBA'
SHUTDOWN IMMEDIATE
STARTUP pfile=$ORACLE_BASE/admin/orcl/pfile/init.ora.724200516420
```

Parameter Settings for BI Beans

OracleBI Beans performs best when the configuration parameters for the database are optimized for its use. During installation of Oracle Database, an OLAP configuration table is created and populated with `ALTER SESSION` commands that have been tested to optimize the performance of OracleBI Beans. Each time OracleBI Beans opens a session, it executes these `ALTER SESSION` commands.

If a database instance is being used only to support Java applications that use OracleBI Beans, then you can modify your server parameter file or `init.ora` file to include these settings. Alternatively, you might want to include some of the settings in the server parameter file and leave others in the table, depending upon how your database instance is going to be used. These are your choices:

- Keep all of the parameters in the configuration table, so that they are set as part of the initialization of a OracleBI Beans session. This method fully isolates these configuration settings solely for OracleBI Beans. (Default)
- Add some of the configuration parameters to the server parameter file or `init.ora` file, and delete those rows from the configuration table. This is useful if your database is being used by other applications that require the same settings.

- Add all of the configuration parameters to the server parameter file or `init.ora` file, and delete all rows from the configuration table. This is the most convenient if your database instance is being used only by OracleBI Beans.

Regardless of where these parameters are set, you should check the Oracle Technology Network for updated recommendations.

> **See Also:** *Oracle Database SQL Reference* for descriptions of initialization parameters that can be set by the `ALTER SESSION` command

Storage Management

Analytic workspaces are stored in the owner's default tablespace, unless the owner specifies otherwise. All tablespaces for OLAP use should specify `EXTENT MANAGEMENT LOCAL`. Tablespaces created using default parameters may use resources inefficiently. You should create undo, permanent, and temporary tablespaces that are appropriate for storing analytic workspaces.

Creating an Undo Tablespace

Create an undo tablespace with the `EXTENT MANAGEMENT LOCAL` clause, as shown in this example:

```
CREATE UNDO TABLESPACE olapundo DATAFILE '$ORACLE_BASE/oradata/undo.dbf'
    SIZE 64M REUSE AUTOEXTEND ON NEXT 8M
    MAXSIZE UNLIMITED EXTENT MANAGEMENT LOCAL;
```

After creating the undo tablespace, change your system parameter file to include the following settings, then restart the database as described in "Setting Database Initialization Parameters" on page 12-1.

```
UNDO_TABLESPACE=tablespace
UNDO_MANAGEMENT=AUTO
```

Creating Permanent Tablespaces for OLAP Use

Each dimensional object occupies at least one extent. A fixed extent size may waste most of the allocated space. For example, if an object is 64K and the extents are set to a uniform size of 1M (the default), then only a small portion of the extent is used.

Create permanent tablespaces with the `EXTENT MANAGEMENT LOCAL` and `SEGMENT SPACE MANAGEMENT AUTO` clauses, as shown in this example:

```
CREATE TABLESPACE glo DATAFILE '$ORACLE_BASE/oradata/glo.dbf'
   SIZE 64M REUSE AUTOEXTEND ON NEXT 8M MAXSIZE UNLIMITED
   EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;
```

Creating Temporary Tablespaces for OLAP Use

Oracle OLAP uses the temporary tablespace to store all changes to the data in a cube, whether the changes are the result of a data load or data analysis. Saving the cube moves the changes into the permanent tablespace and clears the temporary tablespace.

This usage creates numerous extents within the tablespace. A temporary tablespace suitable for use by Oracle OLAP should specify the `EXTENT MANAGEMENT LOCAL` clause and a `UNIFORM SIZE` clause with a small size, as shown in this example:

```
CREATE TEMPORARY TABLESPACE glotmp TEMPFILE '$ORACLE_BASE/oradata/glotmp.tmp'
   SIZE 50M REUSE AUTOEXTEND ON NEXT 5M MAXSIZE UNLIMITED
   EXTENT MANAGEMENT LOCAL UNIFORM SIZE 256K;
```

Spreading Data Across Storage Resources

Oracle Database provides excellent storage management tools to simplify routine tasks. Automatic Storage Management (ASM) provides a simple storage management interface that virtualizes database storage into disk groups. You can manage a small set of disk groups, and ASM automates the placement of the database files within those disk groups.

ASM spreads data evenly across all available storage resources to optimize performance and utilization. After you add or drop disks, ASM automatically rebalances files across the disk group.

Because OLAP is part of Oracle Database, you can use ASM to manage both relational and dimensional data.

ASM is highly recommended for analytic workspaces. A system managed with ASM is faster than a file system and easier to manage than raw devices. ASM optimizes the performance of analytic workspaces both on systems with Oracle RAC and those without Oracle RAC.

However, you do not need ASM to use Oracle OLAP. You can still spread your data across multiple disks, just by defining the tablespaces like in this example:

```
CREATE TABLESPACE glo DATAFILE
   'disk1/oradata/glo1.dbf' SIZE 64M REUSE AUTOEXTEND ON NEXT 8M MAXSIZE 1024M
   EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;

ALTER TABLESPACE glo ADD DATAFILE
   'disk2/oradata/glo2.dbf' SIZE 64M REUSE AUTOEXTEND ON NEXT 8M MAXSIZE 1024M,
   'disk3/oradata/glo3.dbf' SIZE 64M REUSE AUTOEXTEND ON NEXT 8M
      MAXSIZE UNLIMITED;
```

Security of Multidimensional Data in Oracle Database

Your company's data is a valuable asset. The information must be secure, private, and protected. Analytic data is particularly vulnerable because it is highly organized, easy to navigate, and summarized into meaningful units of measurement.

When you use Oracle OLAP, your data is stored in the database. It has the security benefits of Oracle Database, which leads the industry in security. You do not need to expose the data by transferring it to a standalone database. You do not need to administer security on a separate system. And you do not need to compromise your data by storing it in a less secure environment than Oracle Database.

Security Management

Because you have just one system to administer, you do not have to replicate basic security tasks such as these:

- Creating user accounts
- Creating and administering rules for password protection
- Securing network connections
- Detecting and eliminating security vulnerabilities
- Safeguarding the system from intruders

The cornerstone of data security is the administration of user accounts and roles. Users open a connection with Oracle Database with a user name and password, and they have access to both dimensional and relational objects in the same session.

Users by default have no access rights to an analytic workspace or any other data type in another user's schema. The owner or an administrator must grant them, or a role to which they belong, any access privileges.

Granting Querying Privileges

To access the cubes in an analytic workspace, users must have the `SELECT` privilege on the table in which the analytic workspace is stored. The name of the table is the name of the analytic workspace with an `AW$` prefix. For example, the `GLOBAL` analytic workspace is stored in the `AW$GLOBAL` relational table.

To access the relational views of dimensional objects, users must have `SELECT` privileges explicitly on those views.

Dictionary Views and System Tables

Oracle Database data dictionary views and system tables contain extensive information about analytic workspaces.

Static Data Dictionary Views

Among the static views of the database data dictionary are several that provide information about analytic workspaces. Table 12–2 provides brief descriptions of them. All data dictionary views have corresponding `DBA` and `USER` views.

Table 12–2 Static Data Dictionary Views for OLAP

View	Description
`ALL_AWS`	Describes all analytic workspaces accessible to the current user.
`ALL_AW_OBJ`	Describes the current objects in all analytic workspaces accessible to the current user.
`ALL_AW_PROP`	Describes the properties defined in all analytic workspaces accessible to the current user.
`ALL_AW_PS`	Describes the page spaces currently in use by all analytic workspaces accessible to the current user.

See Also:

- "Querying the Active Catalog" on page 4-14 for a list of views that describe OLAP dimensional objects
- *Oracle Database Reference* for full descriptions of all data dictionary views

System Tables

The `SYS` user owns several tables associated with analytic workspaces. Table 12–3 provides brief descriptions.

Important: These tables are vital for the operation of Oracle OLAP. Do not delete them or attempt to modify them directly without being fully aware of the consequences.

Table 12–3 OLAP Tables Owned By SYS

Table	Description
`AW$`	Maintains a record of all analytic workspaces in the database, recording its name, owner, and other information.
`AW$AWCREATE`	Stores the `AWCREATE` analytic workspace, which contains programs for using OLAP Catalog metadata in Oracle Database 10*g* Release 10.1.0.2 and earlier releases. It exists only for backward compatibility.
`AW$AWCREATE10G`	Stores the `AWCREATE10G` analytic workspace, which contains programs for using OLAP Catalog metadata in Oracle Database 10*g* Release 10.1.0.3. The OLAP Catalog is not used by later releases. It exists only for backward compatibility.
`AW$AWMD`	Stores the `AWMD` analytic workspace, which contains programs for creating metadata catalogs.
`AW$AWREPORT`	Stores the `AWREPORT` analytic workspace, which contains a program named `AWREPORT` for generating a summary space report.
`AW$AWXML`	Stores the `AWXML` analytic workspace, which contains programs for creating and managing analytic workspaces for Oracle Database 10*g* Release 10.1.0.4 and later.
`AW$EXPRESS`	Stores the `EXPRESS` analytic workspace. It contains objects and programs that support basic operations. `EXPRESS` is used any time a session is open.
`AW_OBJ$`	Describes the objects stored in analytic workspaces.
`AW_PRG$`	Stores program data. Not currently used.
`AW_PROP$`	Stores analytic workspace object properties.
`PS$`	Maintains a history of all page spaces. A page space is an ordered series of bytes equivalent to a file. Oracle OLAP manages a cache of workspace pages. Pages are read from storage in a table and written into the cache in response to a query. The same page can be accessed by several sessions. The information stored in `PS$` enables Oracle OLAP to discard pages that are no longer in use, and to maintain a consistent view of the data for all users, even when the workspace is being modified during their sessions. When changes to a workspace are saved, unused pages are purged and the corresponding rows are deleted from `PS$`.

Analytic Workspace Tables

Analytic workspaces are stored in tables in the Oracle database. The names of these tables always begin with `AW$`.

For example, if the `GLOBAL` user creates two analytic workspaces, one named `MARKETING` and the other named `FINANCIALS`, then these tables are created in the `GLOBAL` schema:

```
AW$FINANCIALS
AW$MARKETING
```

The tables store all of the object definitions and data.

Build Logs

When submitting a maintenance task to the job queue, be sure to note the job number so that you can verify that the job completed successfully. Runtime messages are stored in a table named `XML_LOAD_LOG`, which is owned by `OLAPSYS`. You must have either the `OLAP_USER` or the `OLAP_DBA` role to access this file.

Messages in `XML_LOAD_LOG` are identified by the digits in the job number. The following SQL statement returns the messages for job `54`:

```
SELECT xml_message FROM olapsys.xml_load_log WHERE xml_loadid='54';
```

You can manage these jobs using tools such as Oracle Enterprise Manager Scheduler or the `DBMS_SCHEDULER` PL/SQL package. Example 12–1 shows a sample log file.

Partitioned Cubes and Parallelism

Cubes are often partitioned to improve build and maintenance times. For information about creating a partitioned cube, refer to "Choosing a Data Storage Strategy" on page 3-15.

Creating and Dropping Partitions

The OLAP engine automatically creates and drops partitions as part of data maintenance, as members are added and deleted from the partitioning dimension.

For example, assume that in the sample Global analytic workspace, the Units cube is partitioned on the Time dimension, using the Calendar hierarchy, and at the Calendar Quarter level. The OLAP engine creates a partition for each Calendar Quarter and its children. The default top partition contains Calendar Years and all members of the Fiscal hierarchy. If Global has three years of data, then the Units cube has 13 partitions: Four bottom partitions for each Calendar Year, plus the top partition.

A data refresh typically creates new time periods and deletes old ones. Whenever a Calendar Quarter value is loaded into the Time dimension, a corresponding new partition is added to the cube. Whenever a Calendar Quarter value is deleted from the Time dimension, the corresponding empty partition is deleted from the cube.

Parallelism

You can improve the performance of data maintenance by enabling parallel processing. There are two levels of parallelism:

- Parallel job execution: Loading and aggregating the data using multiple processes.
- Parallel update: Moving the data from temporary to permanent tablespaces using multiple processes.

This number of parallel processes is controlled by these factors:

- The number of objects that can be aggregated in parallel. Each cube and each partition (including the top partition) can use a separate process.

  You can control the number of partitions in a cube on the Implementation Details tab of the cube property sheet in Analytic Workspace Manager.
- The number of simultaneous database processes the user is authorized to run.

This number is controlled by the `JOB_QUEUE_PROCESSES` parameter. The setting for this parameter is based on the number of processors, as described in "Setting Database Initialization Parameters" on page 12-1. You can obtain the current parameter setting with the following SQL command:

```
SHOW parameter job_queue_processes
```

- For parallel update, the number of processes you allocate to the job. You can specify the number of processes in the Maintenance Wizard of Analytic Workspace Manager when specifying the task processing options.

Suppose that a cube is partitioned on the Quarter level of Time, and the cube contains three years of data. The cube has 3*4=12 bottom partitions, `JOB_QUEUE_PROCESSES` is set to 8, and you set the parallelism option to 4 for the build. Oracle Database processes the cube in this way:

1. Load and build the dimensions of the cube serially using a single process.
2. Load and build the 12 bottom partitions in parallel using 4 processes. As soon as one process finishes, another begins until all 12 are complete.

   This cube could use the 8 processes allowed by `JOB_QUEUE_PROCESSES`, but it is limited to 4 by the build setting.
3. Load and build the top partition.

Oracle Database allocates the specified number of processes regardless of whether all of them can be used simultaneously at any point in the job. For example, if your job can use up to three processes, but you specify five, then two of the processes allocated to your job cannot be used by it or by any other job.

If Oracle Database is installed with Oracle RAC, then a script submitted to the job queue is distributed across all nodes in the cluster. The performance gains can be significant. For example, a job running on four nodes in a cluster may run up to four times faster than the same job running on a single computer.

The build log is stored in a file named `XML_LOAD_LOG`, which is owned by `OLAPSYS`. Example 12–1 shows excerpts from the job log for a completed build that used two parallel processes.

Example 12–1 Job Log Verifying Parallel Processing

```
SQL> SELECT xml_message FROM olapsys.xml_load_log WHERE xml_loadid='4';

XML_MESSAGE
-----------------------------------------------------------------------------------------------
11:13:02 Job# AWXML$_4 to Build(Refresh) Analytic Workspace GLOBAL.GLOBAL Submitted to the Queue.
11:13:05 Started Build(Refresh) of GLOBAL.GLOBAL Analytic Workspace.
11:13:10 Attached AW GLOBAL.GLOBAL in RW Mode.
11:13:10   Started Loading Dimensions.
11:13:20     Started Loading Dimension Members.
               .
               .
               .
11:13:27     Finished Loading Dimension Members.
11:13:27     Started Loading Hierarchies.
               .
               .
               .
11:13:33     Finished Loading Hierarchies.
11:13:33     Started Loading Attributes.
               .
```

```
                .
                .
11:13:44      Finished Loading Attributes.
11:13:44    Finished Loading Dimensions.
11:13:44    Started Updating Partitions.
11:13:46    Finished Updating Partitions.
11:14:37 Detached AW GLOBAL.GLOBAL.
11:14:37 Starting Parallel Processing.
11:14:42 Attached AW GLOBAL.GLOBAL in MULTI Mode.
11:14:44      Started Load of Measures: UNIT_PRICE, UNIT_COST from Cube PRICE_AND_COST_CUBE.CUBE.
11:14:45      Finished Load of Measures: UNIT_PRICE, UNIT_COST from Cube PRICE_AND_COST_CUBE.CUBE.
              Processed 5046 Records. Rejected 0 Records.
11:14:45      Started Auto Solve for Measures: UNIT_COST from Cube PRICE_AND_COST_CUBE.CUBE.
11:14:46      Finished Auto Solve for Measures: UNIT_COST from Cube PRICE_AND_COST_CUBE.CUBE.
11:14:46      Started Auto Solve for Measures: UNIT_PRICE from Cube PRICE_AND_COST_CUBE.CUBE.
11:14:46      Finished Auto Solve for Measures: UNIT_PRICE from Cube PRICE_AND_COST_CUBE.CUBE.
11:14:53 Attached AW GLOBAL.GLOBAL in MULTI Mode.
11:14:54      Started Load of Measures: UNITS, SALES from Cube UNITS_CUBE.CUBE.
11:15:17      Finished Load of Measures: UNITS, SALES from Cube UNITS_CUBE.CUBE.
              Processed 222589 Records.Rejected 0 Records.
11:15:17      Started Auto Solve for Measures: SALES from Cube UNITS_CUBE.CUBE.
11:15:26      Finished Auto Solve for Measures: SALES from Cube UNITS_CUBE.CUBE.
11:15:26      Started Auto Solve for Measures: UNITS from Cube UNITS_CUBE.CUBE.
11:15:35      Finished Auto Solve for Measures: UNITS from Cube UNITS_CUBE.CUBE.
11:14:38 Started 1 Finished 0 out of 2 Tasks.
11:14:38 Running Jobs: AWXML$_4_1.
11:14:38 Started 2 Finished 0 out of 2 Tasks.
11:14:38 Running Jobs: AWXML$_4_1, AWXML$_4_2.
11:14:38 Started 2 Finished 0 out of 2 Tasks.
11:14:38 Running Jobs: AWXML$_4_1, AWXML$_4_2. Waiting for Tasks to Finish...
11:15:39 Finished Parallel Processing.
11:15:39 Completed Build(Refresh) of GLOBAL.GLOBAL Analytic Workspace.
```

Monitoring Analytic Workspaces

Oracle Database provides various tools to help you diagnose performance problems. As an Oracle DBA, you may find these tools useful in tuning the database:

- Oracle Enterprise Manager Database Control (Database Control) is a general database management and administration tool. In addition to facilitating basic tasks like adding users and modifying datafiles, Database Control presents a graphic overview of a database's current status. It also provides an interface to troubleshooting and performance tuning utilities.
- Automatic Workload Repository collects database performance statistics and metrics for analysis and tuning, shows the exact time spent in the database, and saves session information.
- Automatic Database Diagnostic Monitor watches database performance statistics to identify bottlenecks, analyze SQL statements, and offer suggestions to improve performance.

Oracle Database also provides system views to help you diagnose performance problems. The following topics identify views that are either specific to OLAP or provide database information that is pertinent to OLAP.

Dynamic Performance Views

Each Oracle Database instance maintains fixed tables that record current database activity. These tables collect data on internal disk structures and memory structures. Among them are tables that collect data on Oracle OLAP.

These tables are available to users through a set of dynamic performance views. By monitoring these views, you can detect usage trends and diagnose system bottlenecks. Table 12–4 provides a brief description of each view. Global dynamic performance views (`GV$`) are also provided.

> **See Also:** *Oracle Database Reference* for full descriptions of the OLAP dynamic performance views.

Table 12–4 OLAP Dynamic Performance Views

View	Description
`V$AW_AGGREGATE_OP`	Lists the aggregation operators available in analytic workspaces.
`V$AW_ALLOCATE_OP`	Lists the allocation operators available in analytic workspaces.
`V$AW_CALC`	Collects information about the use of cache space and the status of dynamic aggregation.
`V$AW_LONGOPS`	Collects status information about SQL fetches.
`V$AW_OLAP`	Collects information about the status of active analytic workspaces.
`V$AW_SESSION_INFO`	Collects information about each active session.

Table 12–5 describes some other dynamic performance views that are not specific to OLAP, but which you may want to use when tuning your database for OLAP.

Table 12–5 Selected Database Performance Views

View	Description
`V$LOG`	Displays log file information from the control file.
`V$LOGFILE`	Contains information about redo log files.
`V$PGASTAT`	Provides PGA memory usage statistics as well as statistics about the automatic PGA memory manager when `PGA_AGGREGATE_TARGET` is set.
`V$ROWCACHE`	Displays statistics for data dictionary activity. Each row contains statistics for one data dictionary cache.
`V$SYSSTAT`	Lists system statistics.

Basic Queries for Monitoring the OLAP Option

The following queries extract OLAP information from the data dictionary.

More complex queries are provided in a script that you can download from the Oracle OLAP Web site on the Oracle Technology Network. For descriptions of these scripts and download instructions, refer to "OLAP DBA Scripts" on page 12-12.

Is the OLAP Option Installed in the Database?

The OLAP option is provided with Oracle Database Enterprise Edition. To verify that the OLAP components have been installed, issue this SQL command:

```
SELECT comp_name, version, status FROM dba_registry WHERE comp_name LIKE '%OLAP%';
```

```
COMP_NAME                 VERSION      STATUS
------------------------- ------------ -----------
OLAP Analytic Workspace   10.2.0.4.0   VALID
Oracle OLAP API           10.2.0.4.0   VALID
OLAP Catalog              10.2.0.4.0   VALID
```

What Analytic Workspaces are in the Database?

The DBA_AWS view provides information about all analytic workspaces. Use the following SQL command to get a list of names, their owners, and the version:

```
SELECT owner, aw_name, aw_version FROM dba_aws;

OWNER      AW_NAME                        AW_VERSION
---------- ------------------------------ ----------
SYS        EXPRESS                        9.1
SYS        AWMD                           9.1
SYS        AWCREATE                       9.1
SYS        AWCREATE10G                    9.1
SYS        AWXML                          9.1
SYS        AWREPORT                       9.1
GLOBAL     GLOBAL                         10.2
```

See Also: "System Tables" on page 12-5 for descriptions of the analytic workspaces owned by SYS.

How Big is the Analytic Workspace?

To find out the size in bytes of the tablespace extents for a particular analytic workspace, use the following SQL statements, replacing GLOBAL with the name of your analytic workspace.

```
SELECT extnum, sum(dbms_lob.getlength(awlob)) bytes FROM global.aw$global
     GROUP BY extnum;

    EXTNUM      BYTES
---------- ----------
         0   80254708
```

To see the size of the LOB table containing an analytic workspace, use a SQL command like the following, replacing GLOBAL.AW$GLOBAL with the qualified name of your analytic workspace.

```
SELECT ROUND(sum(dbms_lob.getlength(awlob))/1024,0) kb
     FROM global.aw$global;

        KB
----------
     78374
```

When Were the Analytic Workspaces Created?

The DBA_OBJECTS view provides the creation date of the objects in your database. The following SQL command generates an easily readable report for analytic workspaces.

```
SELECT owner, object_name, created, status FROM dba_objects
      WHERE object_name LIKE 'AW$%' AND object_name!='AW$'
      GROUP BY owner, object_name, created, status
      ORDER BY owner, object_name;

OWNER      OBJECT_NAME     CREATED   STATUS
```

```
---------- --------------- --------- -------
GLOBAL     AW$GLOBAL       10-AUG-07 VALID
SYS        AW$AWCREATE     01-AUG-07 VALID
SYS        AW$AWCREATE10G  01-AUG-07 VALID
SYS        AW$AWMD         01-AUG-07 VALID
SYS        AW$AWREPORT     01-AUG-07 VALID
SYS        AW$AWXML        01-AUG-07 VALID
SYS        AW$EXPRESS      01-AUG-07 VALID
```

OLAP DBA Scripts

You can download a file that contains several SQL scripts from the Oracle OLAP Web site on the Oracle Technology Network. These scripts typically extract information from two or more system views and generate a report that may be useful in monitoring and tuning a database. To download the file, use this URL:

`http://www.oracle.com/technetwork/database/options/olap/olap-dba-scripts-393636.zip`

Table 12–6 describes these scripts. For more information, refer to the `README` file provided with the scripts.

Table 12–6 OLAP DBA Scripts

SQL Script	Description
`aw_objects_in_cache`	Identifies the objects in the buffer cache that are related to analytic workspaces.
`aw_reads_writes`	Tallies the reads from temporary and permanent tablespaces, the writes to cache, and the rows processed in analytic workspaces.
`aw_segment_size`	Calculates the size of analytic workspace segments in tablespaces on disk.
`aw_size`	Displays the amount of disk space used by each analytic workspace.
`aw_tablespaces`	Provides extensive information about the tablespaces used by analytic workspaces.
`aw_total_size`	Tallies the sizes of all analytic workspaces accessible to the current user.
`aw_users`	Identifies the users of analytic workspaces.
`aw_wait_events`	Describes the wait events experienced by users of analytic workspaces over the previous hour.
`buffer_cache_hits`	Calculates the buffer cache hit ratio.
`cursor_parameters`	Indicates whether the database parameters that limit the number of open cursors are set too low.
`olap_hit_ratio`	Identifies the PGA, OLAP page pool, and OLAP hit/miss ratio for every user of analytic workspaces in the database.
`olap_pga_performance`	Determines how much PGA is in use, the size of the OLAP page pool, and the hit/miss ratio for OLAP pages for each user.
`olap_pga_use`	Determines how much PGA is consumed by the OLAP page pool to perform operations on analytic workspaces.
`session_resources`	Identifies the use of cursors, PGA, and UGA for each open session.
`shared_pool_hits`	Calculates the shared pool hit ratio.

Scripts for Monitoring Performance

Several of the scripts listed in "OLAP DBA Scripts" on page 12-12 provide detailed information about the use of memory and other database resources by OLAP sessions. You can use these scripts as is, or you can use them as the starting point for developing your own scripts.

Example 12–2 shows the information returned by the session_resources script. It lists the use of resources such as cursors, PGA, and UGA.

Example 12–2 Querying Session Resources

```
SQL> @session_resources

USERNAME             NAME                                VALUE
-------------------- ------------------------------ ----------
GLOBAL:95            opened cursors cumulative             101
                     opened cursors current                  3
                     session cursor cache count             31
                     session cursor cache hits              68
                     session pga memory                1219292
                     session pga memory max            1219292
                     session stored procedure space          0
                     session uga memory                 432700
                     session uga memory max             432700


9 rows selected.
```

Scripts for Monitoring Disk Space

Several of the scripts listed in "OLAP DBA Scripts" on page 12-12 provide detailed information about the use of disk space by analytic workspaces. Example 12–3 shows the information returned by the `aw_size` script. It lists all of the analytic workspaces in the database, the disk space they consume, and the tablespaces in which they are stored.

Example 12–3 Querying the Use of Disk Space By Analytic Workspaces

```
SQL> @aw_size

Analytic Workspace                          On Disk MB Tablespace
---------------------------------------- --------------- --------------------
GLOBAL.GLOBAL                                     239.38 GLOBAL
SYS.AWCREATE                                        9.81 SYSAUX
SYS.AWCREATE10G                                     1.38 SYSAUX
SYS.AWMD                                            7.00 SYSAUX
SYS.AWREPORT                                        1.50 SYSAUX
SYS.AWXML                                          12.00 SYSAUX
SYS.EXPRESS                                         2.69 SYSAUX
                                         ---------------
Total Disk:                                       273.75

7 rows selected.
```

Backup and Recovery

You can backup and recover analytic workspaces using the same tools and procedures as the rest of your database.

Oracle Recovery Manager (RMAN) is a powerful tool that simplifies, automates, and improves the performance of backup and recovery operations. RMAN enables one time backup configuration, automatic management of backups, and archived logs based on a user-specified recovery window, restartable backups and restores, and test restore/recovery.

RMAN implements a recovery window to control when backups expire. This lets you establish a period of time during which it is possible to discover logical errors and fix the affected objects by doing a database or tablespace point-in-time recovery. RMAN also automatically expires backups that are no longer required to restore the database to a point-in-time within the recovery window. Control file auto backup also allows for restoring or recovering a database, even when an RMAN repository is not available.

Export and Import

You can copy analytic workspaces in several different ways, either to replicate them on another computer or to back them up.

- **Data Pump**. Analytic workspaces are copied with the other objects in a schema or database export. Use the `expdp/impdp` database utilities.
- **Transportable Tablespaces**. Analytic workspaces are copied with the other objects to a transportable tablespace. However, you can only transport the tablespace to the same platform (for example, from Linux to Linux, Solaris to Solaris, or Windows to Windows) because the OLAP `DECIMAL` data type is hardware dependent. Use the `expdp/impdp` database utilities. Transportable tablespaces are much faster than dump files.
- **XML Templates**. A template saves the XML definition of objects in an analytic workspace. You can save the entire analytic workspace, or individual cubes, dimensions, and calculated measures. Using a saved template, you can create a new analytic workspace exactly like an existing one. The template does not save any data, nor does it save any customizations to the analytic workspace. You can copy a template to a different platform.

The owner of an analytic workspace can create an XML template, or export the schema to a dump file. Only users with the `EXP_FULL_DATABASE` privilege or a privileged user (such as `SYS` or a user with the `DBA` role) can export the full database or create a transportable tablespace.

See Also:

- "Using Templates to Re-Create Dimensional Objects" on page 3-21 for information about XML templates
- *Oracle Database Utilities* for information about Oracle Data Pump and the `expdp/impdp` commands

A
Designing a Dimensional Model

This guide uses the Global schema for its examples. This appendix explores the business requirements of the fictitious Global Computing Company and discusses how the design of a data model emerges from these requirements.

This appendix contains the following topics:

Case Study Scenario

The fictional Global Computing Company was established in 1990. Global Computing distributes computer hardware and software components to customers on a worldwide basis. The Sales and Marketing department has not been meeting its budgeted numbers. As a result, this department has been challenged to develop a successful sales and marketing strategy.

Global Computing operates in an extremely competitive market. Competitors are numerous, customers are especially price-sensitive, and profit margins tend to be narrow. In order to grow profitably, Global Computing must increase sales of its most profitable products.

Various factors in Global Computing's current business point to a decline in sales and profits:

- Traditionally, Global Computing experiences low third-quarter sales (July through September). However, recent sales in other quarters have also been lower than expected. The company has experienced bursts of growth but, for no apparent reason, has had lower first-quarter sales during the last two years as compared with prior years.
- Global has been successful with its newest sales channel, the Internet. Although sales within this channel are growing, overall profits are declining.
- Perhaps the most significant factor is that margins on personal computers - previously the source of most of Global Computing's profits - are declining rapidly.

Global Computing needs to understand how each of these factors is affecting its business.

Current reporting is done by the IT department, which produces certain standard reports on a monthly basis. Any ad hoc reports are handled on an as-needed basis and are subject to the time constraints of the limited IT staff. Complaints have been widespread within the Sales and Marketing department, with regard to the delay in response to report requests. Complaints have also been numerous in the IT department, with regard to analysts who change their minds frequently or ask for further information.

The Sales and Marketing department has been struggling with a lack of timely information about what it is selling, who is buying, and how they are buying. In a meeting with the CIO, the VP of Sales and Marketing states, "By the time I get the information, it's no longer useful. I'm only able to get information at the end of each month, and it doesn't have the details I need to do my job."

Reporting Requirements

When asked to be more specific about what she needs, the Vice President of Sales and Marketing identifies the following requirements:

- Trended sales data for specific customers, regions, and segments.
- The ability to provide information and some analysis capabilities to the field sales force. A Web interface would be preferred, since the sales force is distributed throughout the world.
- Detail regarding mail-order, phone, and e-mail sales on a monthly and quarterly basis, as well as a comparison to past time periods. Information must identify when, how, and what is being sold by each channel.
- Margin information on products in order to understand the dollar contribution for each sale.
- Knowledge of percent change versus the prior and year-ago period for sales, units, and margin.
- The ability to perform analysis of the data by ad hoc groupings.

The CIO has discussed these requirements with his team and has come to the conclusion that a standard reporting solution against the production order entry system would not be flexible enough to provide the required analysis capabilities. The reporting requirements for business analysis are so diverse that the projected cost of development, along with the expected turnaround time for requests, would make this solution unacceptable.

The CIO's team recommends using an analytic workspace to support analysis. The team suggests that the Sales and Marketing department's IT group work with Corporate IT to build an analytic workspace that meets their needs for information analysis.

Business Goals

The development team identifies the following high-level business goals that the project must meet:

- Global Computing's strategic goal is to increase company profits by increasing sales of higher margin products and by increasing sales volume overall.
- The Sales and Marketing department objectives are to:
  - Analyze industry trends and target specific market segments
  - Analyze sales channels and increase profits
  - Identify product trends and create a strategy for developing the appropriate channels

Information Requirements

Once you have established business goals, you can determine the type of information that helps achieve these goals. To understand how end users examine the data in the analytic workspace, it is important to conduct extensive interviews. From interviews with key end users, you can determine how they look at the business, and what types of business analysis questions they want to answer

Business Analysis Questions

Interviews with the VP of Sales and Marketing, salespeople, and market analysts at Global Computing reveal the following business analysis questions:

- What products are profitable?
- Who are our customers, and what and how are they buying?
- What accounts are most profitable?
- What is the performance of each distribution channel?
- Is there still a seasonal variance to the business?

We can examine each of these business analysis questions in detail.

What products are profitable?

This business analysis question consists of the following questions:

- What is the percent of total sales for any item, product family, or product class in any month, quarter or year, and in any distribution channel? How does this percent of sales differ from a year ago?
- What is the unit price, unit cost, and margin for each unit for any item in any particular month? What are the price, cost, and margin trends for any item in any month?
- What items were most profitable in any month, quarter, or year, in any distribution channel, and in any geographic area or market segment? How did profitability change from the prior period? What was the percent change in profitability from the prior period?
- What items experienced the greatest change in profitability from the prior period?
- What items contributed the most to total profitability in any month, quarter, or year, in any distribution channel, and in any geographic area or market segment?
- What items have the highest per unit margin for any particular month?
- In summary, what are the trends?

Who are our customers, and what and how are they buying?

This business analysis question consists of the following questions:

- What were sales for any item, product family, or product class in any month, quarter, or year?
- What were sales for any item, product family, or product class in any distribution channel, geographic area, or market segment?
- How did sales change from the prior period? What was the percent change in sales from the prior period?
- How did sales change from a year ago? What was the percent change in sales from a year ago?

- In summary, what are the trends?

Which accounts are most profitable?

This business analysis question consists of the following questions:

- Which accounts are most profitable in any month, quarter, or year, in any distribution channel, by any item, product family, or product class?
- What were sales and extended margin (gross profit) by account for any month, quarter, or year, for any distribution channel, and for any product?
- How does account profitability compare to the prior time period?
- Which accounts experienced the greatest increase in sales as compared to the prior period?
- What is the percent change in sales from the prior period? Did the percent change in profitability increase at the same rate as the percent change in sales?
- In summary, what are the trends?

What is the performance of each distribution channel?

This business analysis question consists of the following questions:

- What is the percent of sales to total sales for each distribution channel for any item, product family, or product class, or for any geographic area or market segment?
- What is the profitability of each distribution channel: direct sales, catalog sales, and the Internet?
- Is the newest distribution channel, the Internet, "cannibalizing" catalog sales? Are customers simply switching ordering methods, or is the Internet distribution channel reaching additional customers?
- In summary, what are the trends?

Is there still a seasonal variance to the business?

This business analysis question consists of the following questions:

- Are there identifiable seasonal sales patterns for particular items or product families?
- How do seasonal sales patterns vary by geographic location?
- How do seasonal sales patterns vary by market segment?
- Are there differences in seasonal sales patterns as compared to last year?

Summary of Information Requirements

By examining the types of analyses that users wish to perform, we can identify the following key requirements for analysis:

- Global Computing has a strong need for profitability analysis. The company must understand profitability by product, account, market segment, and distribution channel. It also needs to understand profitability trends.
- Global Computing needs to understand how sales vary by time of year. The company must understand these seasonal trends by product, geographic area, market segment, and distribution channel.

- Global Computing has a need for ad hoc sales analysis. Analysis must identify what products are sold to whom, when these products are sold, and how customers buy these products.
- The ability to perform trend analysis is important to Global Computing.

Identifying Required Business Facts

The key analysis requirements reveal the business facts that are required to support analysis requirements at Global Computing.

These facts are ordered by time, product, customer shipment or market segment, and distribution channel:

Sales
Units
Change in sales from prior period
Percent change in sales from prior period
Change in sales from prior year
Percent change in sales from prior year
Product share
Channel share
Market share
Extended cost
Extended margin
Extended margin change from prior period
Extended margin percent change from prior period
Units sold, change from prior period
Units sold, percent change from prior period
Units sold, change from prior year
Units sold, percent change from prior year

These facts are ordered by item and month:

Unit price
Unit cost
Margin per unit

Designing a Dimensional Model for Global Computing

"Business Goals" on page A-2 identifies the business facts that support analysis requirements at Global Computing. Next, we identify the dimensions, levels, and attributes in a data model. We will also identify the relationships within each dimension. The resulting data model is used to design the Global schema, the dimensional model, and the analytic workspace.

Identifying Dimensions

Four dimensions are used to organize the facts in the database:

- Product shows how data varies by product.
- Customer shows how data varies by customer or geographic area.
- Channel shows how data varies according to each distribution channel.
- Time shows how data varies over time.

Identifying Levels

Now that we have identified dimensions, we can identify the levels of summarization within each dimension. Analysis requirements at Global Computing reveal that:

- There are three distribution channels: Sales, Catalog, and Internet. These three values are the lowest level of detail in the data warehouse and are grouped in the Channel level. From the order of highest level of summarization to the lowest level of detail, levels are Total and Channel.
- Global performs customer and geographic analysis along the line of shipments to customers and by market segmentation. Shipments and Segment are two hierarchies in the Customer dimension. In each case, the lowest level of detail in the data model is the Ship To location.
  - When analyzing along the line of customer shipments, the levels of summarization are (highest to lowest): Total, Region, Warehouse, and Ship To.
  - When analyzing by market segmentation, the levels of summarization are (highest to lowest): Total, Market Segment, Account, and Ship To.
- The Product dimension has four levels (highest to lowest): Total, Class, Family, and Item.
- The Time dimension has four levels (highest to lowest): Total, Year, Quarter, and Month.

All dimensions have a Total level as the highest level of summarization. Adding this highest level provides additional flexibility as application users analyze data.

Identifying Hierarchies

We will identify the hierarchies that organize the levels within each dimension. To identify hierarchies, we group the levels in the correct order of summarization and in a way that supports the identified types of analysis.

For the Channel and Product dimensions, Global Computing requires only one hierarchy for each dimension. For the Customer dimension, Global Computing requires two hierarchies. Analysis within the Customer dimension tends to be either by geographic area or market segment. Therefore, we organize levels into two hierarchies, Shipments and Segment. Analysis over time also requires two hierarchies, a Calendar hierarchy and a Fiscal hierarchy.

Identifying Stored Measures

"Identifying Required Business Facts" on page A-5 lists 21 business facts that are required to support the analysis requirements of Global Computing. Of this number, only four facts must be acquired from the transactional database:

- Units
- Sales
- Unit Price
- Unit Cost

All of the other facts can be derived from these basic facts. The derived facts can be calculated in the analytic workspace on demand. If experience shows that some of these derived facts are being used heavily and the calculations are putting a noticeable load on the system, then some of these facts can be calculated and stored in the analytic workspace as a data maintenance procedure.

Glossary

Active Catalog

A set of relational views that expose the standard form metadata stored in analytic workspaces, where it can be accessed by SQL. Applications that use OracleBI Beans query the Active Catalog.

See also **database standard form**.

additive

Describes a measure or fact that can be summarized through addition, such as a `SUM` function. An additive measure is the most common type. Examples include sales, cost, and profit.

Contrast with **nonadditive**.

aggregation

The process of consolidating data values into a single value. For example, sales data could be collected on a daily basis and then be aggregated to the week level, the week data could be aggregated to the month level, and so on. The data can then be referred to as aggregate data.

The term aggregation is often used interchangeably with summarization, and aggregate data is used interchangeably with summary data. However, there are a wide range of aggregation methods available in addition to `SUM`.

allocation

The process of distributing aggregate data down a hierarchy to the detail level, sometimes using an existing set of data as the basis for the allocation. Allocation is often used in forecasting and budgeting systems. An example of a financial allocation is the automated distribution of a bonus pool, based on the current salaries and performance ratings of the employees.

analytic workspace

A container for storing related dimensional objects, such as dimensions and cubes. An analytic workspace is stored in a relational table.

See also **cube**.

ancestor

A dimension member at a higher level of aggregation than a particular member. For example, in a Time dimension, the year 2007 is the ancestor of the day 06-July-07. The member immediately above is the parent. In a dimension hierarchy, the data value of the ancestor is the aggregated value of the data values of its descendants.

Contrast with **descendant**. See also **hierarchy**, **level**, **parent**.

attribute

A database object related to an OLAP cube dimension. An attribute stores descriptive characteristics for all dimension members, or members of a particular hierarchy, or only members at a particular level of a hierarchy.

When the values of an attribute are unique, they provide supplementary information that can be used for display (such as a descriptive name) or in analysis (such as the number of days in a time period). When the values of an attribute apply to a group of dimension members, they enable users to select data based on like characteristics. For example, in a database representing footwear, you might use a color attribute to select all boots, sneakers, and slippers of the same color.

base level data

See **detail data**.

base measure

See **measure**.

calculated measure

A stored expression that executes in response to a query. For example, a calculated measure might generate the difference in costs from the prior period by using the `LAG_VARIANCE` function on the `COSTS` measure. Another calculated measure might calculate profits by subtracting the `COSTS` measure from the `SALES` measure. The expression resolves only the values requested by the query.

See also **expression**, **measure**.

cell

A single data value of an expression. In a dimensioned expression, a cell is identified by one value from each of the dimensions of the expression. For example, if you have a measure with the dimensions `MONTH` and `CUSTOMER`, then each combination of a month and a customer identifies a separate cell of that measure.

child

A dimension member that is part of a more aggregate member in a hierarchy. For example, in a Time dimension, the month Jan-06 might be the child of the quarter Q1-2006. A dimension member can be the child of a different parent in each hierarchy.

Contrast with **parent**. See also **descendant**, **hierarchy**.

composite

A compact format for storing sparse multidimensional data. Oracle OLAP provides two types of composites: a compressed composite for extremely sparse data, and a regular composite for moderately sparse data.

See also **dimension**, **sparsity**.

compressed cube

A cube with very sparse data that is stored in a compressed composite.

See also **composite**.

compression

See **compressed cube**.

cube

An organization of measures with identical dimensions and other shared characteristics. The edges of the cube contain the dimension members, and the body of the cube contains the data values. For example, sales data can be organized into a cube whose edges contain values from the Time, Product, and Customer dimensions and whose body contains Volume Sales and Dollar Sales data.

custom measure

See **calculated measure**.

custom member

A dimension member whose data is calculated from the values of other members of the same dimension using the rules defined in a model.

See **model**.

data source

A relational table, view, synonym, or other database object that provides detail data for cubes and cube dimensions.

data warehouse

A database designed for query and analysis rather than transaction processing. A data warehouse usually contains historical data that is derived from transaction data, but it can include data from other sources. It separates analysis workload from transaction workload and enables a business to consolidate data from several sources.

database standard form

An analytic workspace that has been constructed with a specific set of objects, such as hierarchy dimensions, level dimensions, parent relations, and level relations. Each object must be defined with a set of properties that identify its role and its relationships with other objects in the analytic workspace. Standard form is required for an analytic workspace to be accessible to OLAP tools, however, it is not a prerequisite for multidimensional analysis.

denormalized

Permit redundancy in a table. Contrast with **normalize**.

derived measure

See **calculated measure**.

descendant

A dimension member at a lower level of aggregation than a particular member. For example, in a Time dimension, the day 06-July-07 is the descendant of year 2007. The member immediately below is the child. In a dimension hierarchy, the data values of the descendants roll up into the data values of the ancestors.

Contrast with **ancestor**. See also **aggregation**, **child**, **hierarchy**, **level**.

detail data

Data at the lowest level, which is acquired from another source.

Contrast with **aggregation**.

dimension

A structure that categorizes data. Among the most common dimensions for sales-oriented data are Time, Geography, and Product. Most dimensions have hierarchies and levels.

In a cube, a dimension is a list of values at all levels of aggregation. It is an index for identifying the values of a measure. For example, if Sales data has a separate sales figure for each month, then the data has a Time dimension that contains month values, which organize the data by month.

In a relational table, a dimension is a type of object that defines hierarchical (parent/child) relationships between pairs of column sets.

See also **hierarchy**.

dimension key

See **dimension member**.

dimension member

One element in the list that composes a cube dimension. For example, a Time dimension might have dimension members for days, months, quarters, and years.

dimension table

A relational table that stores all or part of the values for a dimension in a star or snowflake schema. Dimension tables typically contain columns for the dimension keys, levels, and attributes.

dimension value

See **dimension member**.

dimension view

A relational view of data in an analytic workspace that contains the same types of data as a dimension table in a star schema, that is, columns for dimension members and attributes. A dimension view typically lists all dimension members in the key column, regardless of their level in the dimension hierarchy.

See also **dimension table**, **star schema**.

drill

To navigate from one item to a set of related items. Drilling typically involves navigating up and down through the levels in a hierarchy.

Drilling down expands the view to include child values that are associated with parent values in the hierarchy.

Drilling up collapses the list of descendant values that are associated with a parent value in the hierarchy.

EIF file

A specially formatted file for transferring data between analytic workspaces, or for storing versions of an analytic workspace (all of it or selected objects) outside the database.

embedded total

A list of dimension members at all levels of a hierarchy, such that the aggregate members (totals and subtotals) are interspersed with the detail members. For example, a Time dimension might contain dimension members for days, months, quarters, and years.

expression

A combination of one or more values (typically provided by a measure or a calculated measure), operators, and functions that evaluates to a value. An expression generally assumes the data type of its components.

The following are examples of expressions, where `SALES` is a measure: `SALES`, `SALES*1.05`, `TRUNC(SALES)`.

fact

See **measure**.

fact table

A table in a star schema that contains factual data. A fact table typically has two types of columns: those that contain facts and those that are foreign keys to dimension tables. The primary key of a fact table is usually a composite key that is made up of all of its foreign keys.

A fact table might contain either detail facts or aggregated facts. Fact tables that contain aggregated facts are typically called summary tables or materialized views. A fact table usually contains facts with the same level of aggregation.

See also **materialized view**.

hierarchy

A way to organize data at different levels of aggregation. Hierarchies are used to define data aggregation; for example, in a Time dimension, a hierarchy might be used to aggregate data from days to months to quarters to years. Hierarchies are also used to define a navigational drill path.

In a relational table, hierarchies can be defined as part of a dimension object.

See also **level-based hierarchy**, **ragged hierarchy**, **skip-level hierarchy**, **value-based hierarchy**.

key

A column or set of columns included in the definition of certain types of integrity constraints. Keys describe the relationships between the different tables and columns of a relational database.

See also **dimension member**.

leaf data

See **detail data**.

level

A named position in a hierarchy. For example, a Time dimension might have a hierarchy that represents data at the month, quarter, and year levels. The levels might be named Month, Quarter, and Year. The names provide an easy way to reference a group of dimension members at the same distance from the base.

level-based hierarchy

A hierarchy composed of levels. For example, Time is always level based with levels such as Month, Quarter, and Year. Most hierarchies are level based.

See also **value-based hierarchy**.

mapping

The definition of the relationship and data flow between source and target objects. For example, the metadata for a cube includes the mappings between each measure and the columns of a fact table or view.

materialized view

A database object that provides access to aggregate data and can be recognized by the automatic refresh and the query rewrite subsystems.

measure

Data that represents a business measure, such as sales or cost data. You can select, display, and analyze the data in a measure. The terms **measure** and **fact** are synonymous; measure is more commonly used in a multidimensional environment and fact is more commonly used in a relational environment.

Measures are dimensional objects that store data, such as Volume Sales and Dollar Sales. Measures belong to a cube.

See also **calculated measure**, **fact**, **cube**.

measure folder

A database object that organizes and label groups of measures. Users may have access to several schemas with measures named Sales or Costs, and measure folders provide a way to differentiate among them.

measure view

A relational view of data in analytic workspace that contains the same types of data as a fact table in a star schema. However, in addition to the base-level facts, a measure view also contains derived data, such as aggregates and inter-row calculations.

See also **fact table**, **star schema**.

model

A set of inter-related equations specified using the members of a particular dimension. Line item dimensions often use models to calculate the values of dimension members.

See also **custom member**. Contrast with **calculated measure**.

NA value

A special data value that indicates that data is "not available" (`NA`) or null. It is the value of any cell to which a specific data value has not been assigned or for which data cannot be calculated.

See also **cell**, **sparsity**.

nonadditive

Describes a measure or fact that cannot be summarized through addition, such as Unit Price. Maximum is an example of a nonadditive aggregation method.

Contrast with **additive**.

normalize

In a relational database, the process of removing redundancy in data by separating the data into multiple tables. Contrast with **denormalized**.

object type

In Oracle object technology, a form of user-defined data type that is an abstraction of a real-world entity. An object type is a schema object with the following components:

- A name, which identifies the object type uniquely within a schema
- Attributes, which model the structure and state of the real-world entity
- Methods, which implement the behavior of the real-world entity, in either PL/SQL or Java

OLAP

Online Analytical Processing. OLAP functionality is characterized by dynamic, dimensional analysis of historical data, which supports activities such as the following:

- Calculating across dimensions and through hierarchies
- Analyzing trends
- Drilling up and down through hierarchies
- Rotating to change the dimensional orientation

Contrast with **OLTP**.

OLAP DML

The internal data definition and manipulation language for analytic workspaces.

OLTP

Online Transaction Processing. OLTP systems are optimized for fast and reliable transaction handling. Compared to data analysis systems, most OLTP interactions involve a relatively small number of rows, but a larger group of tables.

Contrast with **OLAP**.

on the fly

Calculated at run-time in response to a specific query. In a cube, calculated measures and custom members are typically calculated on the fly. Aggregate data can be precomputed, calculated on the fly, or a combination of the two methods.

Contrast with **precompute**.

page

A unit for swapping data in and out of memory.

Also called a block.

page space

A grouping of related data pages.

parent

A dimension member immediately above a particular member in a hierarchy. In a dimension hierarchy, the data value of the parent is the aggregated total of the data values of its children.

Contrast with **child**. See also **hierarchy**, **level**.

parent-child relation

A one-to-many relationship between one parent and one or more children in a hierarchical dimension. For example, New York (at the state level) might be the parent of Albany, Buffalo, Poughkeepsie, and Rochester (at the city level).

See also **child**, **parent**.

precalculate

See **precompute**.

precompute

Calculate and store as a data maintenance procedure. In a cube, aggregate data can be precomputed, calculated on the fly, or a combination of the two methods.

Contrast with **on the fly**.

ragged hierarchy

A hierarchy that contains at least one member with a different base level, creating a "ragged" base level for the hierarchy. Organization dimensions are frequently ragged.

refresh

Load new and changed values from the source tables and recompute the aggregate values.

skip-level hierarchy

A hierarchy that contains at least one member whose parents are multiple levels above it, creating a hole in the hierarchy. For example, in a Geography dimension with levels for City, State, and Country, Washington D.C. is a city that does not have a State value; its parent is United States at the Country level.

snowflake schema

A type of star schema in which the dimension tables are partly or fully normalized.

See also **normalize**, **star schema**.

solved data

A result set in which all derived data has been calculated. Data fetched from an cube is always fully solved, because all of the data in the result set is calculated before it is returned to the SQL-based application. The result set from the cube is the same whether the data was precomputed or calculated on the fly.

See also **on the fly**, **precompute**.

source

See **data source**.

sparsity

A concept that refers to multidimensional data in which a relatively high percentage of the combinations of dimension values do not contain actual data.

There are two types of sparsity:

- Controlled sparsity occurs when a range of values of one or more dimensions has no data; for example, a new measure dimensioned by Month for which you do not have data for past months. The cells exist because you have past months in the Month dimension, but the cells are empty.

- Random sparsity occurs when nulls are scattered throughout a measure, usually because some combinations of dimension members never have any data. For example, a district might only sell certain products and never have sales data for the other products.

Some dimensions may be sparse while others are dense. For example, every time period may have at least one data value across the other dimensions, making Time a dense dimension. However, some products may not be sold in some cities, and may not be available anywhere for some time periods; both Product and Geography may be sparse dimensions.

See also **composite**.

standard form

See **database standard form**.

star query

A join between a fact table and a number of dimension tables. Each dimension table is joined to the fact table using a primary key to foreign key join, but the dimension tables are not joined to each other.

star schema

A relational schema whose design represents a dimensional data model. The star schema consists of one or more fact tables and one or more dimension tables that are related through foreign keys.

See also **snowflake schema**.

status

The list of currently accessible values for a given dimension. The status of a dimension persists within a particular session, and does not change until it is changed deliberately. When an analytic workspace is first attached to a session, all members are in status.

See also **dimension member**.

summary

See **aggregation**.

update window

The length of time available for loading new data into a database.

value-based hierarchy

A hierarchy defined only by the parent-child relationships among dimension members. The dimension members at a particular distance from the base level do not form a meaningful group for analysis, so the levels are not named. 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.

See also **hierarchy**, **level-based hierarchy**.

Index