Oracle® OLAP

User's Guide

26ai
G44122-01
October 2025

Oracle OLAP User's Guide, 26ai

G44122-01

Copyright © 2003, 2025, Oracle and/or its affiliates.

Primary Author: David McDermid

Contributors: David Bardwell, Donna Carver, Ken Chen, Sandeep Desai, Dave DeDonato, Bud Endress, Scott Feinstein, David Greenfield, Marty Gubar, AA Hopeman, Christopher Kearney, Pam Montalto, Anne Murphy, Zhiqi Qiu, Marty Roth

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface

Audience i
Documentation Accessibility i
Related Documents i
Conventions ii

Changes in This Release for Oracle OLAP User's Guide

Changes in Oracle AI Database 26ai i

1 Overview

1.1 OLAP Technology in the Oracle Database 1
1.1.1 Full Integration of Multidimensional Technology 1
1.1.2 Ease of Application Development 2
1.1.3 Ease of Administration 2
1.1.4 Security 2
1.1.5 Unmatched Performance and Scalability 2
1.1.6 Reduced Costs 3
1.2 Developing Reports and Dashboards Using SQL Tools and Application Builders 3
1.3 Overview of the Dimensional Data Model 5
1.3.1 Cubes 6
1.3.2 Measures 7
1.3.3 Dimensions 7
1.3.4 Hierarchies and Levels 8
1.3.4.1 Level-Based Hierarchies 8
1.3.4.2 Value-Based Hierarchies 8
1.3.5 Attributes 8

2 Getting Started with Oracle OLAP

2.1 Installing the Sample Schema 1
2.2 Database Management Tasks 1
2.3 Granting Privileges to DBAs and Application Developers 2

Copyright © 2003, 2025, Oracle and/or its affiliates.

2.4 Getting Started with Analytic Workspace Manager 3
2.4.1 Installing Analytic Workspace Manager 3
2.4.2 Opening Analytic Workspace Manager 3
2.4.3 Defining a Database Connection 4
2.4.4 Opening a Database Connection 5
2.4.5 Showing the Analytic Workspace Attachment Modes 5
2.4.6 Installing Plug-ins 5
2.5 Upgrading Metadata From Oracle OLAP 10g 6

3 Creating Dimensions and Cubes

3.1 Designing a Dimensional Model for Your Data 1
3.2 Introduction to Analytic Workspace Manager 2
3.3 Creating a Dimensional Data Store Using Analytic Workspace Manager 3
3.3.1 Adding Functionality to Dimensional Objects 4
3.3.2 When Does Analytic Workspace Manager Save Changes? 4
3.4 Creating Dimensions 4
3.4.1 Requirements of a Dimension 5
3.4.1.1 Dimension Members Must Be Unique 6
3.4.1.2 Time Dimensions Have Special Requirements 6
3.4.2 Creating a Dimension 6
3.4.3 Creating Levels 7
3.4.4 Creating Hierarchies 8
3.4.5 Creating Attributes 9
3.4.5.1 Automatically Defined Attributes 9
3.4.5.2 User-Defined Attributes 10
3.4.5.3 Unique Key Attributes 11
3.4.6 Creating Measure Dimensions 11
3.4.7 Mapping Dimensions 12
3.4.7.1 Dimension Mapping Window 13
3.4.7.2 Source Data Query 14
3.4.8 Loading Data Into Dimensions 15
3.4.9 Displaying the Dimension View 16
3.4.10 Displaying the Default Hierarchy 17
3.5 Creating Cubes 17
3.5.1 Creating Measures 19
3.5.2 Mapping Cubes 19
3.5.3 Partitioning a Cube 24
3.5.3.1 Selecting Partitions 25
3.5.3.2 Analyzing Partition Members 27
3.5.4 Loading Data Into Cubes 28
3.5.5 Displaying the Data in a Cube 31

Copyright © 2003, 2025, Oracle and/or its affiliates.

3.5.6 Displaying the Cube View Descriptions 31
3.6 Choosing a Data Maintenance Method 32
3.6.1 Creating and Executing Custom Cube Scripts 33
3.6.1.1 Creating Cube Scripts 33
3.6.1.2 Running a Cube Script 34
3.6.2 Creating and Executing Maintenance Scripts 35
3.6.2.1 Creating Maintenance Scripts 35
3.6.2.2 Running Maintenance Scripts 36
3.6.3 Adding Materialized View Capability to a Cube 36
3.7 Supporting Multiple Languages 38
3.8 Defining Measure Folders 39
3.9 Saving and Re-Creating Dimensional Objects with Object Definitions 40
3.9.1 Creating Dimensional Objects From XML Templates 41
3.9.2 Saving Object Definitions to XML Templates 41
3.9.3 Creating Analytic Workspaces from EIF Files 42
3.9.4 Saving Analytic Workspaces to EIF Files 42
3.10 Copying and Pasting Dimensional Objects 42

4 Querying Dimensional Objects

4.1 Exploring the OLAP Views 2
4.1.1 Cube Views 2
4.1.1.1 Discovering the Names of the Cube Views 2
4.1.1.2 Discovering the Columns of a Cube View 2
4.1.1.3 Displaying the Contents of a Cube View 3
4.1.2 Dimension and Hierarchy Views 4
4.1.2.1 Discovering the Names of Dimension and Hierarchy Views 4
4.1.2.2 Discovering the Columns of a Dimension View 4
4.1.2.3 Displaying the Contents of a Dimension View 5
4.1.2.4 Discovering the Columns of a Hierarchy View 6
4.1.2.5 Displaying the Contents of a Hierarchy View 6
4.2 Creating Basic Queries 7
4.2.1 Applying a Filter to Every Dimension 7
4.2.2 Allowing the Cube to Aggregate the Data 10
4.2.3 Query Processing 11
4.3 Creating Hierarchical Queries 11
4.3.1 Drilling Down to Children 12
4.3.2 Drilling Up to Parents 12
4.3.3 Drilling Down to Descendants 13
4.3.4 Drilling Up to Ancestors 13
4.4 Using Calculations in Queries 13
4.5 Using Attributes for Aggregation 15

Copyright © 2003, 2025, Oracle and/or its affiliates.

4.5.1 Aggregating Measures Over Attributes 15
4.5.2 Aggregating Calculated Measures Over Attributes 16
4.6 Joining Cubes to Tables and Views 17
4.7 Viewing Execution Plans 18
4.7.1 Generating Execution Plans 18
4.7.2 Types of Execution Plans 20
4.8 Querying the Data Dictionary 20

5 Enhancing Your Database with Analytic Content

5.1 What Is a Calculated Measure? 1
5.2 Functions for Defining Calculations 1
5.2.1 Arithmetic Operators 2
5.2.2 Analytic Functions 2
5.2.3 Single-Row Functions 2
5.3 Creating Calculated Measures 3
5.3.1 Modifying a Template 5
5.3.2 Choosing a Range of Time Periods 6
5.4 Using Calculation Templates 6
5.4.1 Arithmetic Calculations 7
5.4.2 Index 7
5.4.3 Prior and Future Periods 8
5.4.4 Period to Date 9
5.4.5 Share 10
5.4.6 Rank 10
5.4.7 Parallel Period 11
5.4.8 Moving Calculations 12
5.4.9 Cumulative Calculations 13
5.4.10 Nested Calculations 13
5.5 Creating User-Defined Expressions 14
5.5.1 Using the OLAP Expression Syntax 15
5.5.2 Expression Syntax Example Using an Arithmetic Operator 15
5.5.3 Free-Form Calculation Example Using an Analytic Function 16
5.5.4 Expression Syntax Analytic Functions 16
5.6 Creating Calculated Measures Using the OLAP DML 18
5.6.1 Selecting an OLAP DML Calculation Type 18
5.6.2 OLAP DML Expression Examples 19
5.6.3 OLAP DML Function Example 19

6 Developing Reports and Dashboards

6.1 Developing OLAP Applications 1

Copyright © 2003, 2025, Oracle and/or its affiliates.

6.2 Developing a Report Using BI Publisher 3
6.2.1 Creating an OLAP Report in BI Publisher 4
6.2.2 Creating a Template in Microsoft Word 5
6.2.3 Generating a Formatted Report 8
6.2.4 Adding Dimension Choice Lists in BI Publisher 9
6.2.4.1 Creating a List of Values for a BI Publisher Report 9
6.2.4.2 Creating a Menu 10
6.2.4.3 Editing the Query in BI Publisher 11
6.3 Developing a Dashboard Using Application Express 12
6.3.1 Creating an OLAP Application in Application Express 13
6.3.2 Adding Dimension Choice Lists in Application Express 15
6.3.2.1 Creating a Region 16
6.3.2.2 Creating a List of Values in Application Express 16
6.3.2.3 Creating the Choice List 17
6.3.2.4 Editing the Query in Application Express 18
6.3.3 Drilling on Dimension Columns 19
6.3.3.1 Creating Hidden Items 19
6.3.3.2 Editing the Query to Use Bind Variables 19
6.3.3.3 Adding Links to the Dimension Columns 20

7 Administering Oracle OLAP

7.1 Setting Database Initialization Parameters 1
7.2 Storage Management 2
7.2.1 Creating an Undo Tablespace 3
7.2.2 Creating Permanent Tablespaces for OLAP Use 3
7.2.3 Creating Temporary Tablespaces for OLAP Use 3
7.2.4 Spreading Data Across Storage Resources 3
7.3 Dictionary Views and System Tables 4
7.3.1 Static Data Dictionary Views 4
7.3.2 System Tables 5
7.3.3 Analytic Workspace Tables 6
7.3.4 Maintenance Logs 6
7.4 Partitioned Cubes and Parallelism 7
7.4.1 Querying Metadata for Cube Partitioning 7
7.4.2 Creating and Dropping Partitions 7
7.4.3 Parallelism 8
7.5 Analyzing Cubes and Dimensions 10
7.6 Monitoring Analytic Workspaces 11
7.6.1 Dynamic Performance Views 12
7.6.2 Basic Queries for Monitoring the OLAP Option 12
7.6.2.1 Is the OLAP Option Installed in the Database? 13

Copyright © 2003, 2025, Oracle and/or its affiliates.

7.6.2.2 What Analytic Workspaces Are in the Database? 13
7.6.2.3 How Big Is the Analytic Workspace? 13
7.6.2.4 When Were the Analytic Workspaces Created? 14
7.6.3 OLAP DBA Scripts 14
7.6.4 Scripts for Monitoring Performance 15
7.6.5 Monitoring Disk Space 15
7.7 About Backing Up and Recovering Analytic Workspaces 16
7.8 About Copying Analytic Workspaces 16
7.9 About Saving Dimensional Object Definitions 17
7.9.1 About XML Templates 17
7.9.2 About EIF Files 18
7.10 Cube Materialized Views 18
7.10.1 Acquiring Information From the Data Dictionary 19
7.10.1.1 Identifying Cube Materialized Views 19
7.10.1.2 Identifying the Refresh Logs 19
7.10.2 Initiating a Data Refresh 20
7.10.2.1 Using DBMS_CUBE 20
7.10.2.2 Using DBMS_MVIEW 20
7.10.3 Refresh Methods 21
7.10.3.1 Refresh Method Descriptions 21
7.10.3.2 Fast Solve Refreshes 22
7.10.4 Using Query Rewrite 22
7.10.5 Acquiring Additional Information About Cube Materialized Views 23

8 Security

8.1 Security of Multidimensional Data in Oracle Database 1
8.1.1 Security Management 1
8.1.2 Types of Security 2
8.1.3 About the Privileges 2
8.1.4 Layered Security 2
8.2 Setting Object Security 3
8.2.1 Using SQL to Set Object Security 3
8.2.1.1 Setting Object Security on an Analytic Workspace 3
8.2.1.2 Setting Object Security on Dimensions 3
8.2.1.3 Setting Object Security on Cubes 4
8.2.2 Using Analytic Workspace Manager to Set Object Security 5
8.2.2.1 Setting Object Security on an Analytic Workspace 5
8.2.2.2 Setting Object Security on Dimensions 6
8.2.2.3 Setting Object Security on Cubes 7
8.3 Creating Data Security Policies on Dimensions and Cubes 7

Copyright © 2003, 2025, Oracle and/or its affiliates.

8.4 Creating OLAP Data Security Roles 10

9 Advanced Aggregations

9.1 What Is Aggregation? 1
9.2 Aggregation Operators 3
9.2.1 Basic Operators 3
9.2.2 Scaled and Weighted Operators 3
9.2.3 Hierarchical Operators 4
9.3 When Does Aggregation Order Matter? 4
9.3.1 Using the Same Operator for All Dimensions of a Cube 5
9.3.1.1 Order Has No Effect 5
9.3.1.2 Order Changes the Aggregation Results 5
9.3.1.3 Order May Be Important 5
9.3.2 Example: Mixing Aggregation Operators 6
9.4 Example: Aggregating the Units Cube 6
9.4.1 Selecting the Aggregation Operators and Hierarchies 7
9.4.2 Choosing the Percentage of Precomputed Values 7

A Designing a Dimensional Model

A.1 Case Study Scenario A-1
A.1.1 Reporting Requirements A-2
A.1.2 Business Goals A-2
A.1.3 Information Requirements A-2
A.1.3.1 Business Analysis Questions A-3
A.1.3.2 Summary of Information Requirements A-4
A.2 Identifying Required Business Facts A-5
A.3 Designing a Dimensional Model for Global Computing A-5
A.3.1 Identifying Dimensions A-5
A.3.2 Identifying Levels A-6
A.3.3 Identifying Hierarchies A-6
A.3.4 Identifying Stored Measures A-6

B Keyboard Shortcuts

B.1 Menu Bar B-1
B.2 Navigation Tree B-1
B.3 Property Sheets B-2
B.4 Shuttle Keys B-2
B.5 Mapping Canvas B-2

Copyright © 2003, 2025, Oracle and/or its affiliates.

Glossary

Index

Copyright © 2003, 2025, Oracle and/or its affiliates.

Preface

Oracle OLAP User's Guide explains how SQL applications can extend their analytic processing capabilities and manage summary data by using the OLAP option of Oracle Database. It also provides information about managing resources for OLAP.

The preface contains these topics:

- Audience
- Documentation Accessibility
- Related Documents
- Conventions

- Audience
- Documentation Accessibility
- Related Documents
- Conventions

Audience

This manual is intended for DBAs who perform these tasks:

- Develop and manage a data warehouse
- Create and maintain dimensional data objects
- 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.

Accessible Access to Oracle Support

Oracle customers who have purchased support 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 about the OLAP option, see the following manuals in the Oracle Database 12*c* documentation set:

- *Oracle Database SQL Language Reference*

Copyright © 2003, 2025, Oracle and/or its affiliates.

Contains complete syntax descriptions of the SQL CREATE, ALTER, and DELETE syntax for managing cubes, cube dimensions, and other dimensional database objects.

- *Oracle Database Reference*

  Contains full descriptions of the data dictionary views for cubes, cube dimensions, and other dimensional database objects.

- *Oracle Database PL/SQL Packages and Types Reference*

  Contains full descriptions of `DBMS_CUBE` and several other PL/SQL packages for managing cubes.

- *Oracle OLAP DML Reference*

  Contains a complete description of the OLAP Data Manipulation Language (OLAP DML).

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.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Changes in This Release for Oracle OLAP User's Guide

This preface contains:

- Changes in Oracle AI Database 26ai
  The following are the changes in Oracle Database release 26ai.

Changes in Oracle AI Database 26ai

The following are the changes in Oracle Database release 26ai.

See Desupported Features in Oracle AI Database 26ai for a complete list of desupported features in this release.

- Deprecated Features

Deprecated Features

Oracle recommends that you do not use deprecated features or values in new applications. Support for deprecated features is for backward compatibility only. For more information about deprecated features, see *Oracle Database Upgrade Guide*.

Deprecation of Oracle OLAP

Analytic workspaces, the OLAP DML programming language, financial reporting, and the OLAP Java API continue to be deprecated in Oracle AI Database 26ai.

Be aware that OLAP will not be supported beyond the term of the current release (Oracle AI Database 26ai) premier support. Oracle strongly recommends that you do not start new projects using OLAP and begin migrating applications using OLAP to alternatives now. If your application requires an in-database dimensional model, then consider using Oracle Analytic Views. Analytic views provide a dimensional semantic model, calculations, and query semantics using data in Oracle Database. When used with columnar tables, analytic views provide query performance similar to the OLAP Option. If your application requires support for advanced dimensional analytics, what-if analysis, or forecasting, then consider Oracle Essbase. Oracle Essbase is a multidimensional database management system with support for complex dimensional business analytics.

Copyright © 2003, 2025, Oracle and/or its affiliates.

1

Overview

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

- OLAP Technology in the Oracle Database
- Developing Reports and Dashboards Using SQL Tools and Application Builders
- Overview of the Dimensional Data Model

- OLAP Technology in the Oracle Database
- Developing Reports and Dashboards Using SQL Tools and Application Builders
- Overview of the Dimensional Data Model

1.1 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
- Ease of Application Development
- Ease of Administration
- Security
- Unmatched Performance and Scalability
- Reduced Costs

1.1.1 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:

- Cubes and other dimensional objects are first class data objects represented in the Oracle data dictionary.
- Cubes and other dimensional objects are supported by standard SQL syntax in the CREATE, ALTER, DROP, and SELECT statements.
- The OLAP engine runs within the kernel of Oracle Database.
- Dimensional objects are stored in Oracle Database in their native multidimensional format.

Copyright © 2003, 2025, Oracle and/or its affiliates.

- Data security is administered in the standard way, by granting and revoking privileges to Oracle Database users and roles.

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

1.1.2 Ease of Application Development

Oracle OLAP makes it easy to enrich your database and your applications with interesting analytic content. Native SQL access to Oracle multidimensional objects and calculations greatly eases the task of developing dashboards, reports, business intelligence (BI) and analytical applications of any type compared to systems that offer proprietary interfaces. Moreover, SQL access means that the power of Oracle OLAP analytics can be used by *any* database application, not just by the traditional, limited collection of OLAP applications.

1.1.3 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.

A major administrative advantage of Oracle's embedded OLAP technology is automated cube maintenance. With standalone OLAP servers, the burden of refreshing the cube is entirely the responsibility of the administrator. This can be a complex and potentially error-prone job. You must create procedures to extract the changed data from the relational source, move the data from the source system to the system running the standalone OLAP server, load and rebuild the cube. You must take responsibility for the security of the deltas (changed values) during this process as well.

With Oracle OLAP, in contrast, cube refresh is handled entirely by the Oracle database. The database tracks the staleness of the dimensional objects, automatically keeps track of the deltas in the source tables, and automatically applies only the changed values during the refresh process. You simply schedule the refresh at appropriate intervals, and Oracle Database takes care of everything else.

1.1.4 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.

1.1.5 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 as needed all aggregates in the defined multidimensional space, it delivers unmatched performance for typical business intelligence applications.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Oracle OLAP queries take advantage of Oracle shared cursors, dramatically reducing memory requirements and increasing performance.

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

1.1.6 Reduced Costs

All these features add up to reduced costs. Administrative costs are reduced because existing personnel skills can be leveraged. Moreover, the Oracle database can manage the refresh of dimensional objects, a complex task left to administrators in other systems. Standard security reduces administration costs as well. Application development costs are reduced because the availability of a large pool of application developers who are SQL knowledgeable, and a large collection of SQL-based development tools means applications can be developed and deployed more quickly. Any SQL-based development tool can take advantage of Oracle OLAP. Hardware costs are reduced by Oracle OLAP's efficient management of aggregations, use of shared cursors, and Oracle RAC, which enables highly scalable systems to be built from low-cost commodity components.

1.2 Developing Reports and Dashboards Using SQL Tools and Application Builders

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 a tool 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, Sales Analysis, and Product 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.

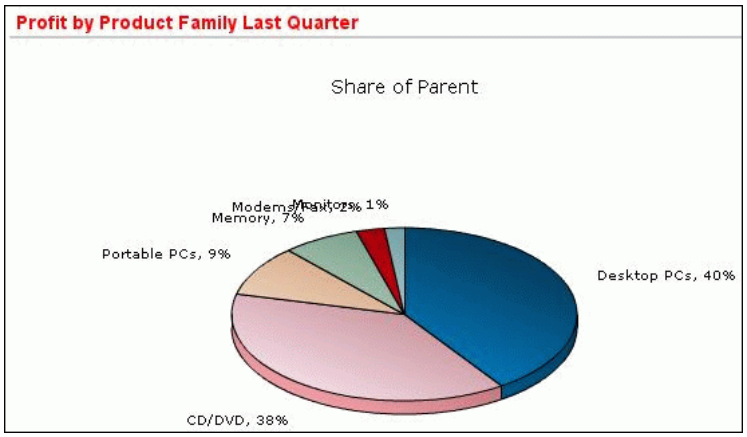
The dial displays the quarterly profit margin. To the right is a bar chart that compares current profits with year-ago profits.

Copyright © 2003, 2025, Oracle and/or its affiliates.

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.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Figure 1-3 Ranking of Percent Change in Year-to-Date Profits From Year Ago

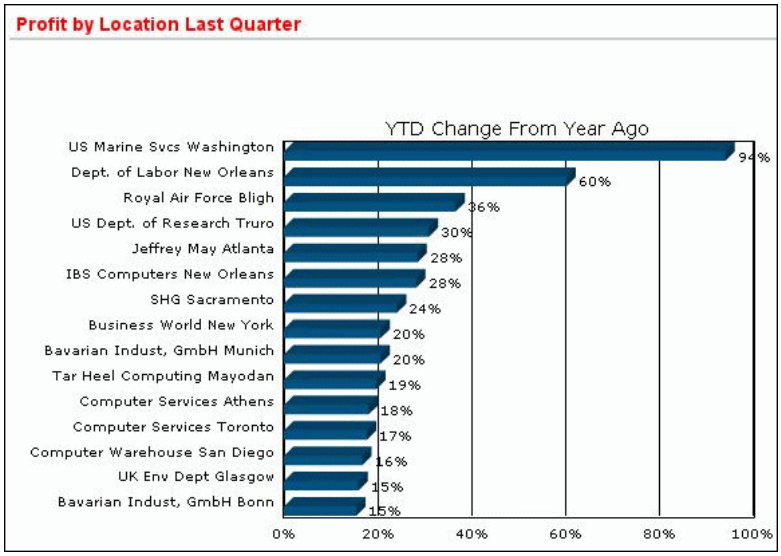


Figure 1-4 compares current profits with year-to-date, year-to-date year ago, the change between year-to-date and year-to-date year ago, and percent change between year-to-date and year-to-date year-ago profits. The cross-tabular report features interactive drilling, so that decision makers can easily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

Figure 1-4 Year-to-Date Profits Compared to Year Ago

Profit Reporting

Time	Product	Customer	Profit	YTD	YTD Yr Ago	YTD Chg Yr Ago	YTD % Chg Yr Ago
1998	Total Product	Total Customer	7,249,296	7,249,296			
1999	Total Product	Total Customer	9,190,282	9,190,282	7,249,296	1,940,986	2677
2000	Total Product	Total Customer	8,880,369	8,880,369	9,190,282	-309,913	-337
2001	Total Product	Total Customer	8,658,271	8,658,271	8,880,369	-222,098	-250
2002	Total Product	Total Customer	6,854,325	6,854,325	8,658,271	-1,803,945	-2083
2003	Total Product	Total Customer	8,730,695	8,730,695	6,854,325	1,876,370	2737
2004	Total Product	Total Customer	11,175,647	11,175,647	8,730,695	2,444,952	2800
2005	Total Product	Total Customer	10,544,532	10,544,532	11,175,647	-631,115	-565
2006	Total Product	Total Customer	11,024,547	11,024,547	10,544,532	480,015	455

1.3 Overview of the Dimensional Data Model

Dimensional objects are an integral part of 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 objects and the OLAP engine are designed to solve complex queries in real time.

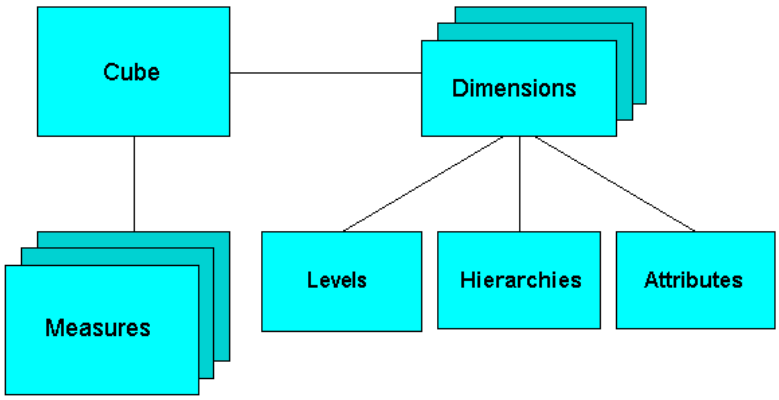
The dimensional objects include cubes, measures, dimensions, attributes, levels, and hierarchies. The simplicity of the model is inherent because it defines objects that represent

Copyright © 2003, 2025, Oracle and/or its affiliates.

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 1-5 shows the general relationships among dimensional objects.

Figure 1-5 Diagram of the OLAP Dimensional Model



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. Cubes are the physical implementation of the dimensional model, and thus are highly optimized for dimensional queries. The OLAP engine leverages this innate dimensionality 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. The technology that underlies cubes is based on an indexed multidimensional array model, which provides direct cell access.

The OLAP engine manipulates dimensional objects in the same way that the SQL engine manipulates relational objects. However, because the OLAP engine is optimized to calculate analytic functions, and dimensional objects are optimized for analysis, analytic and row functions can be calculated much faster in OLAP than in SQL.

The dimensional model enables Oracle OLAP to support high-end business intelligence tools and applications such as OracleBI Discoverer Plus OLAP, OracleBI Spreadsheet Add-In, OracleBI Suite Enterprise Edition, BusinessObjects Enterprise, and Cognos ReportNet.

- Cubes
- Measures
- Dimensions
- Hierarchies and Levels
- Attributes

1.3.1 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 can easily be analyzed and displayed together.

A cube usually corresponds to a single fact table or view.

Copyright © 2003, 2025, Oracle and/or its affiliates.

1.3.2 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 detail 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, 2006, at 2:34 p.m. But they might want to find out which color of dress was most popular in the summer of 2006 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, the Customer dimension could be rolled up into regions, and the Product dimension 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.

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.

1.3.3 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-06), a customer (Warren Systems), a product (Portable PCs), and a channel (Catalog).

Base-level dimension values correspond to the unique keys of a fact table.

A measure dimension is a dimension that has measures as dimension members. With a measure dimension, you can generate calculated measures for all of the measures in the cube simultaneously. Also, you do not have to create a new set of calculated measures for each measure that you add to the cube. The existing calculated measures apply to the new measure in the measure dimension. This is especially useful if you create new measures frequently.

Copyright © 2003, 2025, Oracle and/or its affiliates.

1.3.4 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
- Value-Based Hierarchies

1.3.4.1 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.

Each level typically corresponds to a column in a dimension table or view. The base level is the primary key.

1.3.4.2 Value-Based Hierarchies

Although hierarchies are typically composed of named 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.

1.3.5 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?

Copyright © 2003, 2025, Oracle and/or its affiliates.

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.

Each attribute typically corresponds to a column in dimension table or view.

Copyright © 2003, 2025, Oracle and/or its affiliates.

2

Getting Started with Oracle OLAP

This chapter describes the preliminary steps you should take to use Oracle OLAP. It assumes that you have installed Oracle Database 12*c* 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 Querying Dimensional Objects.

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
- Upgrading Metadata From Oracle OLAP 10g

- Installing the Sample Schema
- Database Management Tasks
- Granting Privileges to DBAs and Application Developers
- Getting Started with Analytic Workspace Manager
- Upgrading Metadata From Oracle OLAP 10g

2.1 Installing the Sample Schema

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

`http://www.oracle.com/technetwork/database/enterprise-edition/downloads/global-11g-schema-1-128202.zip`

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

2.2 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".

Copyright © 2003, 2025, Oracle and/or its affiliates.

2.3 Granting Privileges to DBAs and Application Developers

Anyone who must create or manage dimensional objects in Oracle Database needs the necessary privileges. These privileges are different from those needed just to query the data stored in dimensional objects. The security system is discussed in Security.

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

To administer data security:

- `OLAP_XS_ADMIN` role

To create cube materialized views in the user's own schema:

- `CREATE MATERIALIZED VIEW` privilege
- `CREATE DIMENSION` privilege
- `ADVISOR` privilege

To create cube materialized views in different schemas:

- `CREATE ANY MATERIALIZED VIEW` privilege
- `CREATE ANY DIMENSION` privilege
- `ADVISOR` 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 Administering Oracle OLAP.

If the source tables are in a different schema, then the owner of the dimensional objects must have `READ` or `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;
GRANT OLAP_XS_ADMIN TO GLOBAL;
```


Copyright © 2003, 2025, Oracle and/or its affiliates.

2.4 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
- Opening Analytic Workspace Manager
- Defining a Database Connection
- Opening a Database Connection
- Showing the Analytic Workspace Attachment Modes
- Installing Plug-ins

2.4.1 Installing Analytic Workspace Manager

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

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

> **See Also**
>
> The installation guide for your client platform.

2.4.2 Opening Analytic Workspace Manager

Use the appropriate procedure for your platform.

On Windows, to open Analytic Workspace Manager:

- From the Start menu, select **Oracle - *Oracle_home***, then **Integrated Management Tools**, and then **OLAP Analytic Workspace Manager and Worksheet**.

On Linux, to open Analytic Workspace Manager:

- From the shell command line, enter this command:

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

Figure 2-1 shows the initial display.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Figure 2-1 Opening Analytic Workspace Manager

If Analytic Workspace Manager does not have access to the Internet, the property viewer shows links to several useful sites. It also shows an exception, because Analytic Workspace Manager cannot display the OLAP home page. To connect to the Internet, you typically need to identify the proxy server.

To identify the proxy server:

1. From the Tools menu, select **Configuration** to display the Configuration dialog box.
2. Under OLAP Home Page Settings, enter the address of the proxy server.
3. Enter the port number for the proxy server, if it is not default port 80.
4. Click **OK** to save these settings. The OLAP Home page appears the next time you start Analytic Workspace Manager.

2.4.3 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.

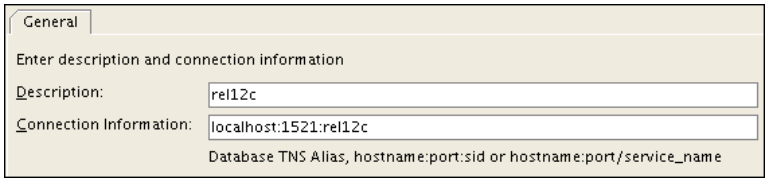
To define a database connection:

1. Right-click the top Databases folder in the navigation tree, then select **New Database Connection** from the shortcut menu.
2. Complete the New Database Connection dialog box.

Figure 2-2 shows the connection information on the General tab of the New Database Connection dialog box.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Figure 2-2 Defining a Database Connection



2.4.4 Opening a Database Connection

To connect to a database:

1. Click the plus icon (+) next to a database connection in the navigation tree.
2. Supply your database user name and password in the Connect to Database dialog box.

2.4.5 Showing the Analytic Workspace Attachment Modes

You can specify an analytic workspace attachment mode when you open an analytic workspace. The modes are the following:

- Read only

  In this mode a user can view the analytic workspace objects and data but cannot create or change objects. The user can export an object by copying it or saving it as a template. Any number of users can open an analytic workspace in Read Only mode.

- Read Write

  In this mode a user can view the analytic workspace objects and data and create or change objects. The user can export or import an object. Only one user can open an analytic workspace in Read Write mode but any number of other users can open it in Read Only mode. This is the default mode.

- Read Write Exclusive

  In this mode a user has the same access rights as in Read Write mode but no one else can open the analytic workspace. This mode is not available if another user has the analytic workspace open.

To specify showing attachment modes:

1. From the Tools menu, select **Configuration**.

   The Configuration dialog box opens.

2. Select **Show Analytic Workspace Attachment Options**. Click **OK**.

2.4.6 Installing Plug-ins

Plug-ins extend the functionality of Analytic Workspace Manager. Plug-ins are distributed as JAR files. Any Java developer can create a plug-in. 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 must identify their location to Analytic Workspace Manager.

Copyright © 2003, 2025, Oracle and/or its affiliates.

To use plug-ins:

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

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

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

To see a list of the currently installed plug-ins:

- On the Help menu, click **About** and then click **Plugins**.

Some Analytic Workspace Manager plug-ins are available for download from the Oracle Technology Network (OTN).

To download plug-ins from OTN:

- In a web browser, go the Oracle OLAP Downloads page at

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

2.5 Upgrading Metadata From Oracle OLAP 10*g*

You can upgrade an Oracle OLAP 10*g* analytic workspace to OLAP 11*g* or 12*c* by saving the objects as an XML template and importing the XML into a different schema. The original analytic workspace remains accessible and unchanged by the upgrade process.

Prerequisites:

- The OLAP 10*g* analytic workspace can use OLAP standard form metadata.
- The original relational source data must be available to load into the new analytic workspace. If the data is in a different schema or the table names are different, then you must remap the dimensional objects to the new relational sources after the upgrade.
- You can create the OLAP 12*c* analytic workspace in the same schema as the OLAP 10*g* analytic workspace. However, if you choose to create the OLAP 12*c* analytic workspace in a different schema, you must grant the new user the appropriate privileges as described in "Granting Privileges to DBAs and Application Developers".

To upgrade an OLAP 11*g* analytic workspace:

1. Open Analytic Workspace Manager for Oracle Database 12*c* Release 1.
2. If necessary, create a new database connection to the database instance with the analytic workspace. See "Defining a Database Connection".
3. Open the database connection. On the Connect to Database dialog box, select **OLAP 11g/12c** for the Cube Type. See "Opening a Database Connection".
4. Expand the navigation tree until the name of the analytic workspace appears.

Copyright © 2003, 2025, Oracle and/or its affiliates.

5. Right-click the analytic workspace and select **Create 12c Upgrade Template for 11g Analytic Workspace**. Save the XML template to a file.

   The Create 12c Upgrade Template for 12*c* Analytic Workspace dialog box appears if any subobjects, such as a level and a hierarchy, have the same name.

   Duplicate object names are changed automatically for the upgrade. You cannot edit the names now, but you can change them later.

6. Click **Close** to close the dialog box.
7. Right-click the connection in the tree and select **Disconnect Database**.
8. Right-click the connection again and select **Connect Database**.
9. On the Connect to Database dialog box, log in with the new user name and select **OLAP 11g/12c** for the Cube Type.
10. Expand the tree, right-click **Analytic Workspaces** under the new schema, and select **Create Analytic Workspace From Template**.
11. Open the upgrade template that you created previously.

    The Correct Duplicate Names From Analytic Workspace Template Import dialog box appears if any objects, such as a cube, dimensions, or the analytic workspace, duplicate object names that already exist in the schema.

12. Enter new names to resolve any conflicts, then click **OK**.
13. Before loading the data, you may want to browse the dimensional objects and make any changes to the object names, cube partitioning, or aggregation strategy.
14. Load data into the new analytic workspace as described in "Loading Data Into Cubes". Select all objects for maintenance.

See Also

DBMS_CUBE in the *Oracle Database PL/SQL Packages and Types Reference* for upgrading in PL/SQL.

Copyright © 2003, 2025, Oracle and/or its affiliates.

3

Creating Dimensions and Cubes

This chapter explains how to design a data model and create dimensions and cubes using Analytic Workspace Manager. It contains the following topics:

- Designing a Dimensional Model for Your Data
- Introduction to Analytic Workspace Manager
- Creating a Dimensional Data Store Using Analytic Workspace Manager
- Creating Dimensions
- Creating Cubes
- Choosing a Data Maintenance Method
- Supporting Multiple Languages
- Defining Measure Folders
- Saving and Re-Creating Dimensional Objects with Object Definitions
- Copying and Pasting Dimensional Objects

- Designing a Dimensional Model for Your Data
- Introduction to Analytic Workspace Manager
- Creating a Dimensional Data Store Using Analytic Workspace Manager
- Creating Dimensions
- Creating Cubes
- Choosing a Data Maintenance Method
- Supporting Multiple Languages
- Defining Measure Folders
- Saving and Re-Creating Dimensional Objects with Object Definitions
- Copying and Pasting Dimensional Objects

3.1 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 should 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 in a star or snowflake schema, then you have the elements of a dimensional model:

- Fact tables correspond to cubes.
- Data columns in the fact tables correspond to measures.

Copyright © 2003, 2025, Oracle and/or its affiliates.

- 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" for an introduction to dimensional objects

Designing a Dimensional Model for a case study of developing a dimensional model for the Global analytic workspace

3.2 Introduction to Analytic Workspace Manager

Analytic Workspace Manager is the primary tool for creating, developing, and managing dimensional objects in Oracle Database. Your goal in using Analytic Workspace Manager is to create a dimensional data store that supports business analysis. This data store can stand alone or store summary data as part of a relational data warehouse.

Populating dimensional objects involves a physical transformation of the data. The first step in that transformation is defining the cubes, measures, dimensions, levels, hierarchies, and attributes. Afterward, you can map these dimensional objects to their relational data sources. The data loading process transforms the data from a relational format into a dimensional format.

Using Analytic Workspace Manager, you can:

- Develop a dimensional model of your data.
- Instantiate that model as dimensional objects.
- Load data from relational tables into those objects.
- Define information-rich calculations.
- Create materialized views that can be used by the database refresh system.
- Automatically generate relational views of the dimensional objects.

You can load data from these sources in the database:

- Tables
- Views
- Synonyms

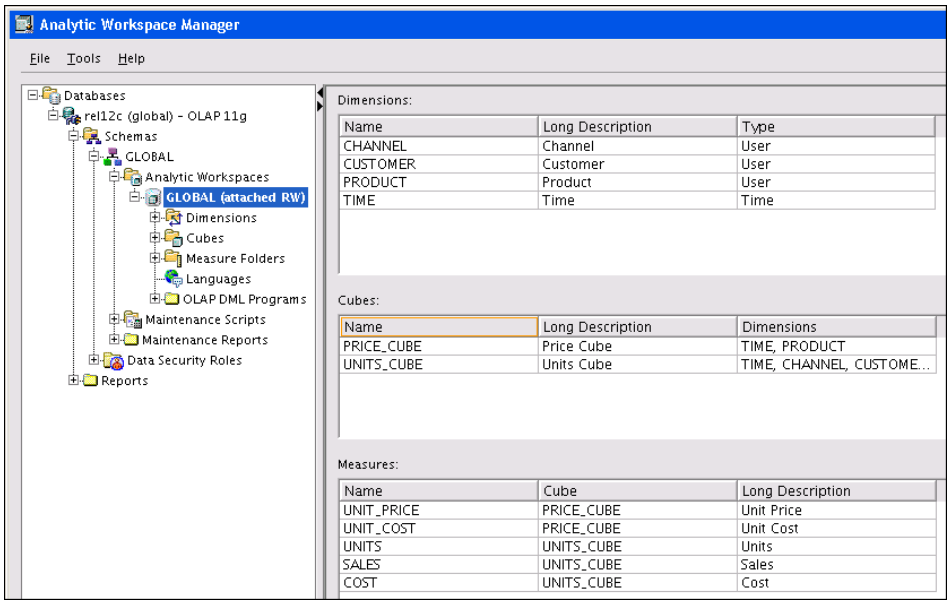
Copyright © 2003, 2025, Oracle and/or its affiliates.

You must have SELECT privileges on the relational data sources so you can load the data into the dimensions and cubes. This chapter assumes that you have a star, snowflake, or other relational schema that supports dimensional objects.

Figure 3-1 shows the main window of Analytic Workspace Manager. It contains 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.

Figure 3-1 Analytic Workspace Manager Main Window



3.3 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 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 an analytic workspace in the database:
   a. In the navigation tree, expand the folders until you see the schema where you want to create the analytic workspace.
   b. Right-click Analytic Workspaces, then click **Create Analytic Workspace**.
   c. Complete the Create Analytic Workspace dialog box, then select **Create**.

      If the Attach Workspace dialog box appears, select the **Read Write** or **Read Write Exclusive** attachment mode.

Copyright © 2003, 2025, Oracle and/or its affiliates.

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

3. Define the dimensions for the data.

   See "Creating Dimensions".

4. Define the cubes for the data.

   See "Creating Cubes".

5. Load data into the cubes and dimensions.

   See "Loading Data Into Cubes".

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
- When Does Analytic Workspace Manager Save Changes?

3.3.1 Adding Functionality to Dimensional Objects

In addition to the basic steps, you can add functionality to the cubes in these ways:

- Develop custom cube scripts to customize the builds.

  See "Creating and Executing Custom Cube Scripts".

- Generate materialized views that support automatic refresh and query rewrite.

  See "Adding Materialized View Capability to a Cube".

- Support multiple languages by adding translations of metadata and attribute values.

  See "Supporting Multiple Languages".

- Define measure folders to simplify access for end users.

  See "Defining Measure Folders".

3.3.2 When Does Analytic Workspace Manager Save 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 **Create** in the Create Dimension dialog box, the 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**.

3.4 Creating Dimensions

Dimensions are lists of unique values that identify and categorize data. They form the edges of a cube, and thus of the measures within the cube. In a report, the dimension values (or their descriptive attributes) provide labels for the rows and columns.

Copyright © 2003, 2025, Oracle and/or its affiliates.

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

- 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.
- List or flat dimensions that have no levels or hierarchies.

You define a dimension as a User, Time, or Measure dimension. Detail-level dimension values typically correspond to the unique keys of a fact table. A measure dimension has measures as dimension members.

This section has the following topics:

- Requirements of a Dimension
- Creating a Dimension
- Creating Levels
- Creating Hierarchies
- Creating Attributes
- Creating Measure Dimensions
- Mapping Dimensions
- Loading Data Into Dimensions
- Displaying the Dimension View
- Displaying the Default Hierarchy
- Requirements of a Dimension
- Creating a Dimension
- Creating Levels
- Creating Hierarchies
- Creating Attributes
- Creating Measure Dimensions
- Mapping Dimensions
- Loading Data Into Dimensions
- Displaying the Dimension View
- Displaying the Default Hierarchy

3.4.1 Requirements of a Dimension

Dimensions must meet the following requirements:

- Dimension Members Must Be Unique
- Time Dimensions Have Special Requirements
- Dimension Members Must Be Unique
- Time Dimensions Have Special Requirements

Copyright © 2003, 2025, Oracle and/or its affiliates.

3.4.1.1 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. If you have any doubt that the values are unique across all levels, then keep the default choice of surrogate keys.

- **Source keys** are read from the relational sources without modification. To use the same exact keys as the source data, 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 dimension table might have a Day column with values of `1` to `366` and a Week column with values of `1` to `52`. Unless you take steps to assure uniqueness, the values from the Week column overwrite the first 52 Day values.
- **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 `DAY_1` and `WEEK_1`, instead of a single member named `1`. A dimension that has surrogate keys must be defined with at least one level-based hierarchy.

Analytic Workspace Manager creates surrogate keys unless you specify otherwise.

3.4.1.2 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 comparisons with earlier or later time periods. If this information is not available, then you can define Time as a User dimension, but it cannot 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.

3.4.2 Creating a Dimension

This section describes how to create a standard User or Time dimension. See "Creating Measure Dimensions" for information on creating a measure dimension.

To create a dimension:

1. Expand the folder for the analytic workspace.
2. Right-click **Dimensions**, then select **Create Dimension**.

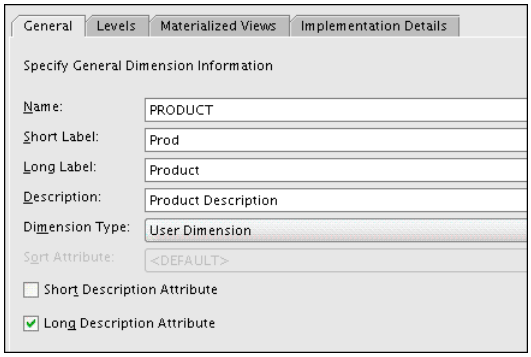
   The Create Dimension dialog box appears.
3. Complete the General tab.
4. If the keys in the source table are unique across levels, you can change the default setting on the Implementation Details tab.
5. Click **Create**.

   The dimension appears as a subfolder under Dimensions.

Figure 3-2 shows the creation of the Product dimension.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Figure 3-2 Creation of the Product Dimension



3.4.3 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 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 select **Create Level**.

   The Create Level dialog box appears.
3. Complete the General tab of the Create Level dialog box.
4. Click **Create**.

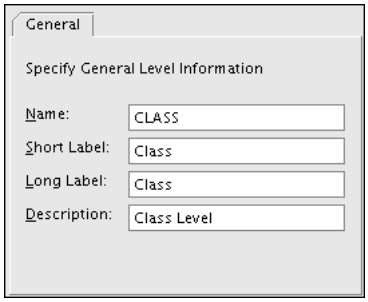
   The level appears as an item in the Levels folder.

> **Tip**
>
> Alternatively, you can create levels in the Create Dimension dialog box Levels tab.

Figure 3-3 shows the creation of the Class level for the Product dimension.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Figure 3-3 Creation of the Class Level



3.4.4 Creating Hierarchies

Dimensions can have one or more hierarchies. They can be level based or value based.

Level-Based 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. Ragged hierarchies are not supported for cube materialized views.
- **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. Skip-level hierarchies are not supported for cube materialized views.

Multiple hierarchies for a dimension typically share the base-level dimension members and then branch into separate hierarchies. They can share the top level if they use all the same base members and use the same aggregation operators. Otherwise, they need different top levels to store different aggregate values. For example, a Customer dimension may have multiple hierarchies that include all base-level customers and are summed to a shared top level. However, a Time dimension with calendar and fiscal hierarchies must aggregate to separate Calendar Year (January to December) and Fiscal Year (July to June) levels, because they use different selections of base-level members.

Value-Based Hierarchies

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 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 the source keys, because surrogate keys are formed with the names of the levels.

Copyright © 2003, 2025, Oracle and/or its affiliates.

To create a hierarchy:

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

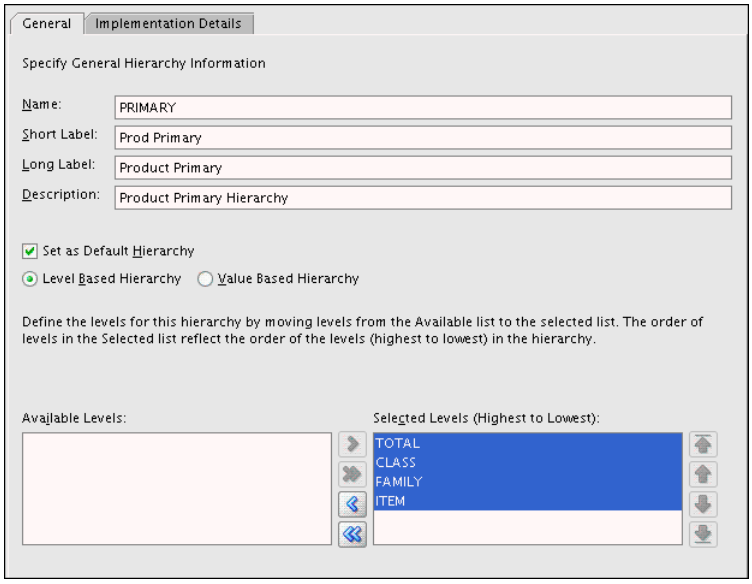
   The Create Hierarchy dialog box appears.
3. Complete the General tab of the Create Hierarchy dialog box.

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

   The hierarchy appears as an item in the Hierarchies folder.

Figure 3-4 shows the creation of the Primary hierarchy for the Product dimension.

Figure 3-4 Creation of the Product Primary Hierarchy



3.4.5 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
- User-Defined Attributes
- Unique Key Attributes

3.4.5.1 Automatically Defined Attributes

Analytic Workspace Manager creates some attributes automatically when creating a dimension. These attributes have a unique type, such as "Long Description."

All dimensions can be 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

Copyright © 2003, 2025, Oracle and/or its affiliates.

columns. However, if your source tables include only one set of descriptions, then you can create and map just one description attribute. If you map both the long and short description attributes to the same column, the data is loaded twice.

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

3.4.5.2 User-Defined Attributes

You can create additional "User" attributes that provide supplementary information about the dimension members, such as the addresses and telephone numbers of customers, or the color and sizes of products.

To create an attribute:

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

   The Create Attribute dialog box appears.
3. Complete the General tab of the Create Attribute dialog box.

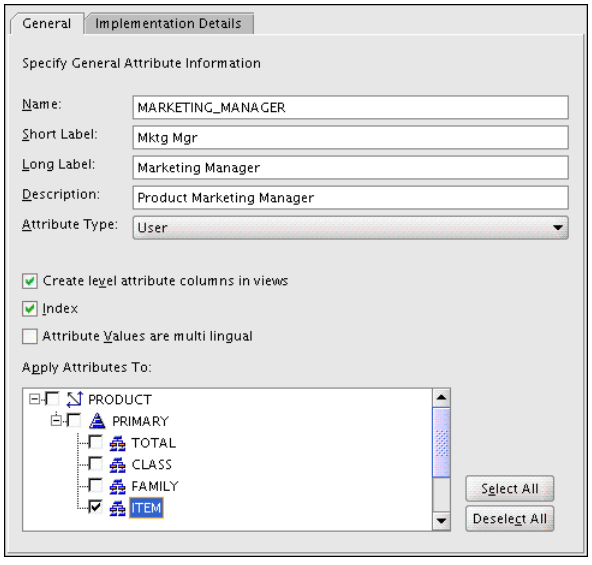
   Some attributes apply to all dimension members, and others apply to only one level. Your selection in the Apply Attributes To box controls the mapping of the attribute to one column or to multiple columns.

   Click **Help** for information about these choices.
4. To change the data type from the default choice of `VARCHAR2`, complete the Implementation Details tab.
5. Click **Create**.

   The attribute appears as an item in the Attributes folder.

Figure 3-5 shows the creation of the Marketing Manager attribute for the Product dimension. Notice that this attribute applies only to the Item level.

Figure 3-5 Creation of the Product Marketing Manager Attribute



Copyright © 2003, 2025, Oracle and/or its affiliates.

3.4.5.3 Unique Key Attributes

Materialized views require that each dimension of the cube have unique key attributes. These attributes store the original key values of the source dimensions, which may have been changed when creating the embedded total dimensions of the cubes.

Analytic Workspace Manager automatically creates unique key attributes for the dimensions of a cube materialized view. You do not create or manage them manually.

3.4.6 Creating Measure Dimensions

A measure dimension enables you to generate calculated measures for all of the measures in the cube simultaneously. Before creating a measure dimension you must first create a fact view. The fact view pivots a fact table so that the measures identify rows instead of columns.

To create a measure dimension:

1. From the Tools menu, select **Create Fact View with Measure Dimension**.

   The Create Fact View with Measure Dimension dialog box appears.
2. Complete the Create Fact View with Measure Dimension dialog box.
   a. From the Schema list, select a schema.
   b. From the Object list, select a fact table.
   c. In the Fact View Name field, keep the default name or enter a different name.
   d. In the table of the columns of the fact table, select the columns for the measures that you want the measure dimension to have.
   e. *Optional*: To automatically create a table for the measure dimension, select the **Create Measure Dimension Table** option.
   f. Click **Create**.
3. Expand the folder for the analytic workspace.
4. Right-click **Dimensions**, then select **Create Dimension**.

   The Create Dimension dialog box appears.
5. Complete the General tab. For the Dimension Class Type, be sure to select **Measure Dimension**.

   A measure dimension is a flat dimension, with no levels or hierarchies.
6. Click **Create**.

   The dimension appears as a subfolder under Dimensions.

After creating the measure dimension, create a cube and add the dimension to it.

> **Note**
>
> If you create a new column in the fact table and you want to add it to the measure dimension, then must create the fact view for the fact table again and maintain the measure dimension and the cube.

Copyright © 2003, 2025, Oracle and/or its affiliates.

To add a measure to the measure dimension:

1. From the Tools menu, select **Create Fact View with Measure Dimension**.

   The Create Fact View with Measure Dimension dialog box appears.
2. Complete the Create Fact View with Measure Dimension dialog box.
   a. From the Schema list, select a schema.
   b. From the Object list, select the fact table that you used to create the measure dimension.
   c. In the Fact View Name field, keep the default name or enter a different name.
   d. In the table of the columns of the fact table, select the columns for the measures that you want the measure dimension to have.
   e. *Optional*: To automatically create a table for the measure dimension, select the **Create Measure Dimension Table** option.
   f. Click **Create**.
3. Right-click the measure dimension and then select **Maintain Dimension**.
4. Right-click the cube that has the measure dimension and then select **Maintain Cube**.

See Also

- "Creating Cubes"

3.4.7 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.

SQL Data Types for Dimensions

You can map dimensions and levels to columns having these SQL data types, which are converted to text during a data load:

- VARCHAR2
- NVARCHAR2
- NUMBER
- INTEGER
- DECIMAL
- CHAR
- NCHAR
- DATE
- TIMESTAMP
- TIMESTAMP WITH TIMEZONE

Copyright © 2003, 2025, Oracle and/or its affiliates.

- TIMESTAMP WITH LOCAL TIMEZONE

You can map attributes to the same data types as cubes and measures, as described in "Data Types".

- Dimension Mapping Window
- Source Data Query

3.4.7.1 Dimension Mapping Window

The mapping window has a tabular view and a graphical view. You can switch between the two views, using the icons at the top of the canvas.

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

You can use the OLAP expression syntax when mapping dimensions in the tabular view. This capability enables you to create the top level of a dimension without having a source column in the dimension table.

You can also map attributes from different tables. OLAP automatically joins the tables on columns with the same name.

Click **Help** on the Mapping window for more information.

To map a dimension:

1. In the navigation tree, expand the dimension folder and click **Mappings**.

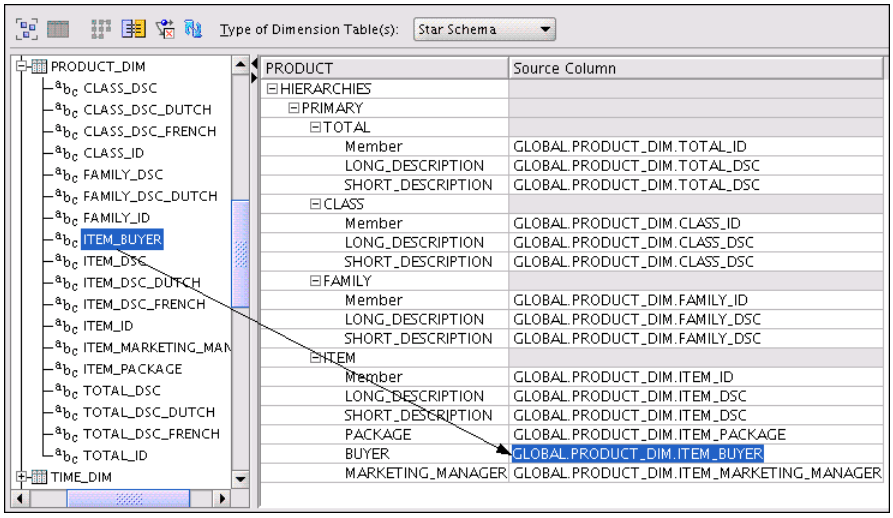
   The Mapping window contains a schema navigation tree on the left and a mapping table for the dimension with rows for the levels and their attributes. This is the tabular view.
2. For normalized dimension tables, select **Snowflake Schema** for the Type of Dimension Table.
3. To enlarge the Mapping Window, drag the divider to the left.
4. In the schema tree, expand the tables, views, or synonyms that contain the dimension members and attributes.
5. Drag-and-drop the source columns onto the appropriate cells in the mapping table for the dimension.

   Map a measure dimension to the measure dimension table. Specify `measure_id` as the member value.
6. After you have mapped all levels and attributes, click **Apply**.
7. Drag the divider back to the right to reveal the navigation tree.

Figure 3-6 shows the Product dimension mapped in the tabular view. The arrow highlights how the `PRODUCT_DIM.ITEM_BUYER` column maps to the `PRODUCT.ITEM.BUYER` attribute.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Figure 3-6 Product Dimension Mapped in Tabular View



To map a top level without a relational source:

1. Create the dimension and its levels (including the top level), hierarchies, and attributes.
2. Map the dimension as described previously for all but the top level.
3. Enter an expression in the OLAP expression syntax for the top level.

Example 3-1 Creating a Top Level for the Global Time Dimension

This example shows a top level for all years in the Time dimension. The mapping expressions used for a Total level (that is, all years) in the Time dimension might look like this:

```
Member: 'TOTAL'
LONG_DESCRIPTION: 'Total'
SHORT_DESCRIPTION: 'Total'
END_DATE: TO_DATE('31-Dec-2007', 'dd-mon-yyyy')
TIME_SPAN: 3646
```

`Member`, `LONG_DESCRIPTION`, and `SHORT_DESCRIPTION` are set to literal strings, `END_DATE` uses the `TO_DATE` function, and `TIME_SPAN` is set to a number.

3.4.7.2 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.

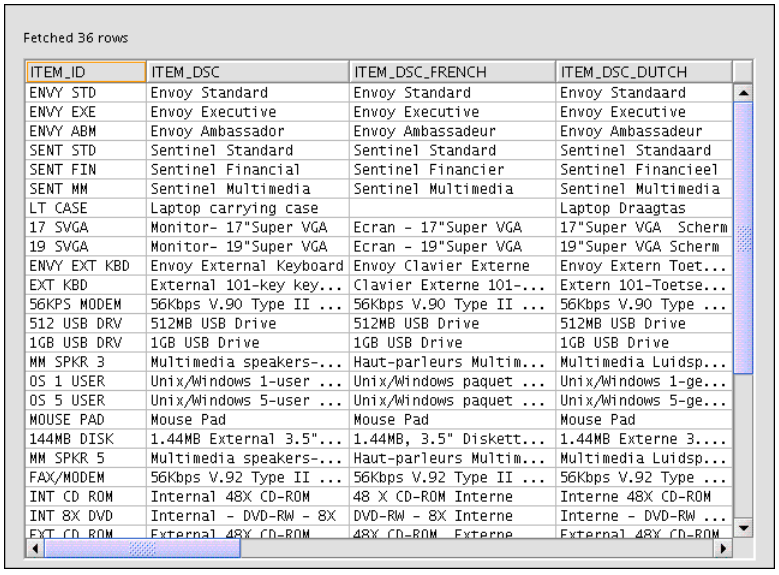
To see the values in a particular source table or view:

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

Figure 3-7 shows the data stored in the `PRODUCT_DIM` table.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Figure 3-7 Data in the PRODUCT_DIM Table



Fetched 36 rows

ITEM_ID	ITEM_DSC	ITEM_DSC_FRENCH	ITEM_DSC_DUTCH
ENVY STD	Envoy Standard	Envoy Standard	Envoy Standaard
ENVY EXE	Envoy Executive	Envoy Executive	Envoy Executive
ENVY ABM	Envoy Ambassador	Envoy Ambassadeur	Envoy Ambassadeur
SENT STD	Sentinel Standard	Sentinel Standard	Sentinel Standaard
SENT FIN	Sentinel Financial	Sentinel Financier	Sentinel Financieel
SENT MM	Sentinel Multimedia	Sentinel Multimedia	Sentinel Multimedia
LT CASE	Laptop carrying case		Laptop Draagtas
17 SVGA	Monitor- 17"Super VGA	Ecran - 17"Super VGA	17"Super VGA Scherm
19 SVGA	Monitor- 19"Super VGA	Ecran - 19"Super VGA	19"Super VGA Scherm
ENVY EXT KBD	Envoy External Keyboard	Envoy Clavier Externe	Envoy Extern Toet...
EXT KBD	External 101-key key...	Clavier Externe 101-...	Extern 101-Toetse...
56KPS MODEM	56Kbps V.90 Type II ...	56Kbps V.90 Type II ...	56Kbps V.90 Type ...
512 USB DRV	512MB USB Drive	512MB USB Drive	512MB USB Drive
1GB USB DRV	1GB USB Drive	1GB USB Drive	1GB USB Drive
MM SPKR 3	Multimedia speakers-...	Haut-parleurs Multim...	Multimedia Luidsp...
OS 1 USER	Unix/Windows 1-user ...	Unix/Windows paquet ...	Unix/Windows 1-ge...
OS 5 USER	Unix/Windows 5-user ...	Unix/Windows paquet ...	Unix/Windows 5-ge...
MOUSE PAD	Mouse Pad	Mouse Pad	Mouse Pad
144MB DISK	1.44MB External 3.5"...	1.44MB, 3.5" Diskett...	1.44MB Externe 3....
MM SPKR 5	Multimedia speakers-...	Haut-parleurs Multim...	Multimedia Luidsp...
FAX/MODEM	56Kbps V.92 Type II ...	56Kbps V.92 Type II ...	56Kbps V.92 Type ...
INT CD ROM	Internal 48X CD-ROM	48 X CD-ROM Interne	Interne 48X CD-ROM
INT 8X DVD	Internal - DVD-RW - 8X	DVD-RW - 8X Interne	Interne - DVD-RW ...
EXT CD ROM	External 48X CD-ROM	48X CD-ROM Externe	External 48X CD-ROM

3.4.8 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 shown here. They are discussed in "Choosing a Data Maintenance Method".

To load data into the dimensions:

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

   The Maintenance Wizard opens on the Select Objects page.
3. Select one or more dimensions from Available Target Objects and use the shuttle buttons to move them to Selected Target Objects.
4. 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.
5. 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 Product dimension has been selected for maintenance. All the Product dimension members and attributes are fetched from the mapped relational sources.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Figure 3-8 Loading Dimension Values into the Product Dimension

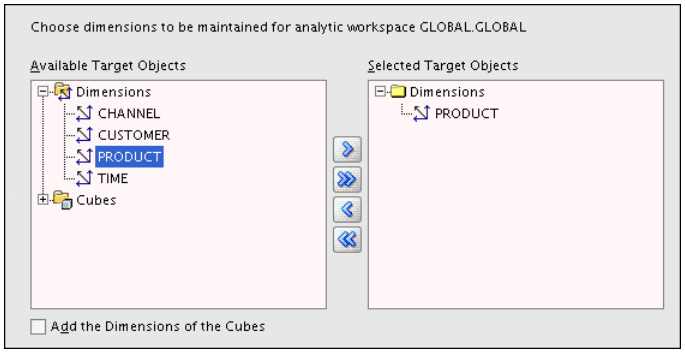
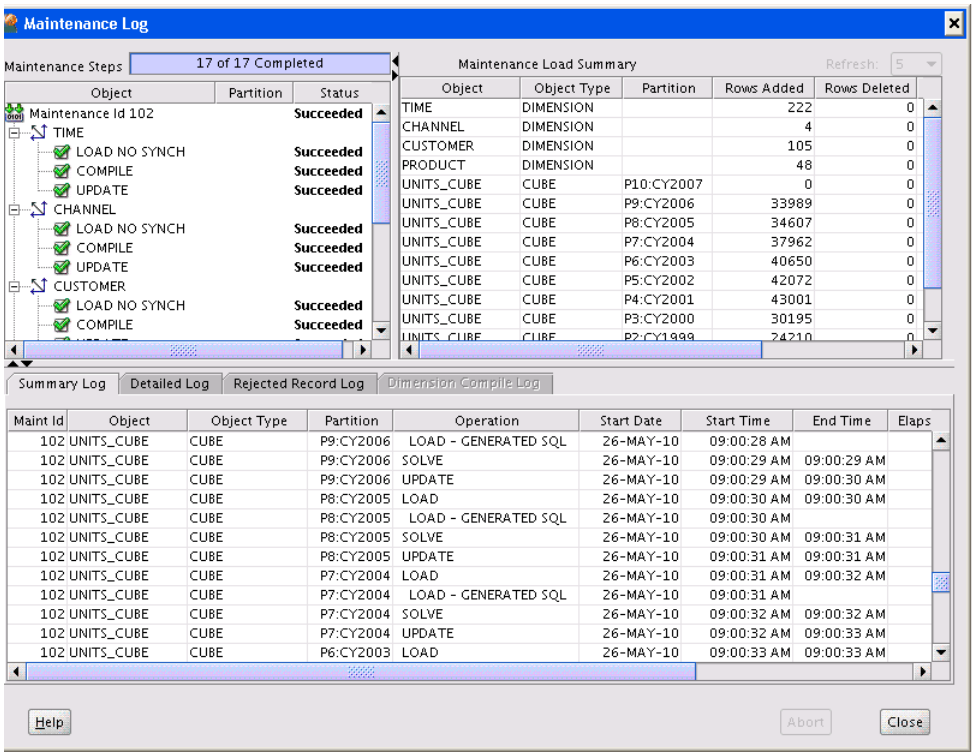


Figure 3-9 shows the Maintenance log for a dimension displayed by Analytic Workspace Manager. It refreshes throughout the build to provide you with the most up-to-date information.

Figure 3-9 Maintenance Log for the Product Dimension



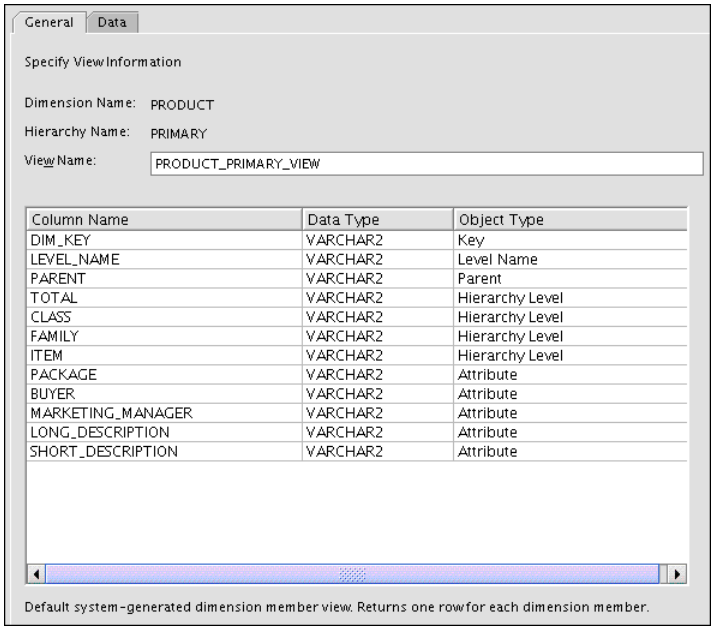
3.4.9 Displaying the Dimension View

The Maintenance Wizard automatically generates relational views of dimensions and hierarchies. Querying Dimensional Objects describes these views and explains how to query them.

Figure 3-10 shows the description of the relational view of the Product Primary hierarchy. You can view the data on the Data tab.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Figure 3-10 Product Primary Hierarchy View



3.4.10 Displaying the Default Hierarchy

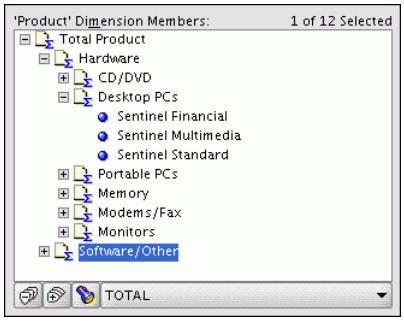
After loading a dimension, you can display the default hierarchy.

To display the default hierarchy:

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

Figure 3-11 shows the Primary hierarchy of the Product dimension.

Figure 3-11 Displaying the Product Primary Hierarchy



3.5 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.

Copyright © 2003, 2025, Oracle and/or its affiliates.

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 select **Create Cube**.

   The Create Cube dialog box appears.
3. On the General tab, enter a name for the cube and select its dimensions.

   Select **Enable SQL Expressions** to allow Analytic Workspace Manager to create additional calculated measures as needed in processing a calculated measure. Enabling SQL expressions is especially useful if you are using the Oracle Business Intelligence Enterprise Edition (OBIEE) Plug-in for Analytic Workspace Manager to export the cube to OBIEE.
4. On the Aggregation tab, click the Rules subtab and select an aggregation method for each dimension. If the cube uses multiple methods, then you may need to specify the order in which the dimensions are aggregated to get the desired results.

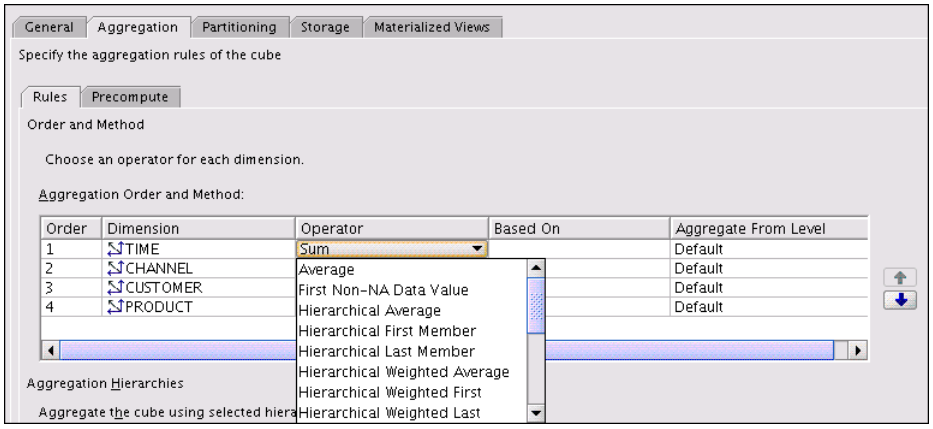
   You can ignore the bottom of the tab, unless you want to exclude a hierarchy from the aggregation.

   For a measure dimension, the aggregation operator is non-additive.
5. If you run the advisors after mapping the cube, Oracle OLAP can determine the best partitioning and storage options. Alternatively, to define these options yourself, complete the Partitioning and Storage tabs before creating the cube.
6. Click **Create**. The cube appears as a subfolder under **Cubes**.

Figure 3-12 shows the Rules subtab for the Units cube with the list of operators displayed.

> **See Also**
>
> "Aggregation Operators" for descriptions of the aggregation operators.

Figure 3-12 Selecting an Aggregation Operator



Copyright © 2003, 2025, Oracle and/or its affiliates.

- Creating Measures
- Mapping Cubes
- Partitioning a Cube
- Loading Data Into Cubes
- Displaying the Data in a Cube
- Displaying the Cube View Descriptions

3.5.1 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 characteristics of a measure are inherited from the cube.

> **Note**
>
> The cube for a measure dimension has only one measure, which Analytic Workspace Manager creates automatically.

To create a measure:

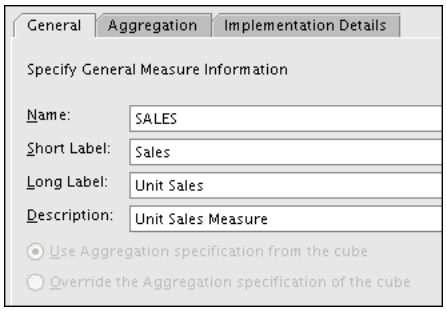
1. Expand the folder for the cube that has the dimensions of the measure.
2. Right-click **Measures**, then select **Create Measure**.

   The Create Measure dialog box appears.
3. On the General tab, enter a name for the measure.
4. Click **Create**.

   The measure appears in the navigation tree as an item in the Measures folder.

Figure 3-13 shows the General tab of the Create Measure dialog box.

Figure 3-13 Creating the Sales Measure



3.5.2 Mapping Cubes

You use the same interface to map cubes as you did to map dimensions, as described in "Mapping Dimensions". You can map a cube directly to a single fact table, or you can create more complex mappings using the OLAP expression syntax, which supports expressions, join conditions, and filters.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Although the dimension columns in a fact table typically contain only key values at the detail level, you can also map cubes to summary tables that contain the values from multiple levels. For example, a Time column might contain days, months, quarters, and years; a Geography column might contain cities, states, and countries. When a build rolls up the data in the cube from the detail level, the calculated values overwrite the loaded summary values, thereby correcting any inconsistencies.

Data Types

You can map cubes and measures to columns having these SQL data types:

- NUMBER
- INTEGER
- DECIMAL
- BINARY_FLOAT
- BINARY_DOUBLE
- VARCHAR2
- NVARCHAR2
- CHAR
- NCHAR
- DATE
- TIMESTAMP
- TIMESTAMP WITH TIMEZONE
- TIMESTAMP WITH LOCAL TIMEZONE
- INTERVAL YEAR TO MONTH
- INTERVAL DAY TO SECOND

Expressions

You can use the OLAP expression syntax when mapping cubes in the tabular view. This capability enables you to perform tasks like these as part of data maintenance, without any intermediate staging of the data:

- Perform calculations on the relational data using any combination of functions and operators available in the OLAP expression syntax.
- Create measures that are more aggregate than their relational sources. For example, suppose the Time dimension has columns for Day, Month, Quarter, and Year, and the fact table for Sales is related to Time by the Day foreign key column. In a basic mapping, you would store data in the cube at the Day level. However, you could aggregate it to the Month level during the data refresh. Using a technique called one-up mapping, you would map the cube to the Month column for Time, and specify a join between the dimension table and the fact table on the Day columns.

> **Note**
>
> You cannot map a measure dimension to an expression. You must map it to a column.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Join Conditions

In the tabular view, the mapping for each dimension includes a join condition. In the basic case where you are mapping the foreign keys in a fact table to the primary keys in the related dimension tables, you can leave the join condition blank. Analytic Workspace Manager derives this information from the relational source tables when you save the mapping.

For example, Analytic Workspace Manager provides this join condition for the `TIME` dimension in the `UNITS_CUBE` mapping:

```
GLOBAL.TIME_DIM.MONTH_ID = GLOBAL.UNITS_FACT.MONTH_ID
```

> **Note**
>
> The join condition for a measure dimension must be a simple equijoin.

Filters

A filter applies a `WHERE` clause to the query that loads data from the relational source into the cube. You can use a filter to limit the rows to those matching a certain condition. This filter restricts the data to the year 2007:

```
GLOBAL.UNITS_FACT.MONTH_ID LIKE '2007%'
```

You can also use a filter to join two or more tables containing the measures. This filter joins the `UNITS_FACT` and `PRICE_FACT` tables in the Global schema on the Time (`MONTH_ID`) and Product (`ITEM_ID`) dimensions:

```
GLOBAL.PRICE_FACT.MONTH_ID=GLOBAL.UNITS_FACT.MONTH_ID AND
GLOBAL.PRICE_FACT.ITEM_ID=GLOBAL.UNITS_FACT.ITEM_ID
```

Aggregate Functions

The aggregate function specifies how the fact table data is loaded into the cube. You select an aggregate function from the Group By list. The aggregate functions are the following:

- SUM
- AVG
- MAX
- MIN
- COUNT

To map a cube:

1. In the navigation tree, expand the cube folder and click **Mappings**.

   The Mapping window contains a schema navigation tree on the left and a mapping table for the cube and its dimensions. This is the tabular view.

   The level of a dimension from which values are aggregated is indicated by the symbol . You specify the level in the Aggregate From Level column on the Rules subtab of the Aggregation tab of the property sheet of a cube.
2. To enlarge the Mapping window, drag the divider to the left.

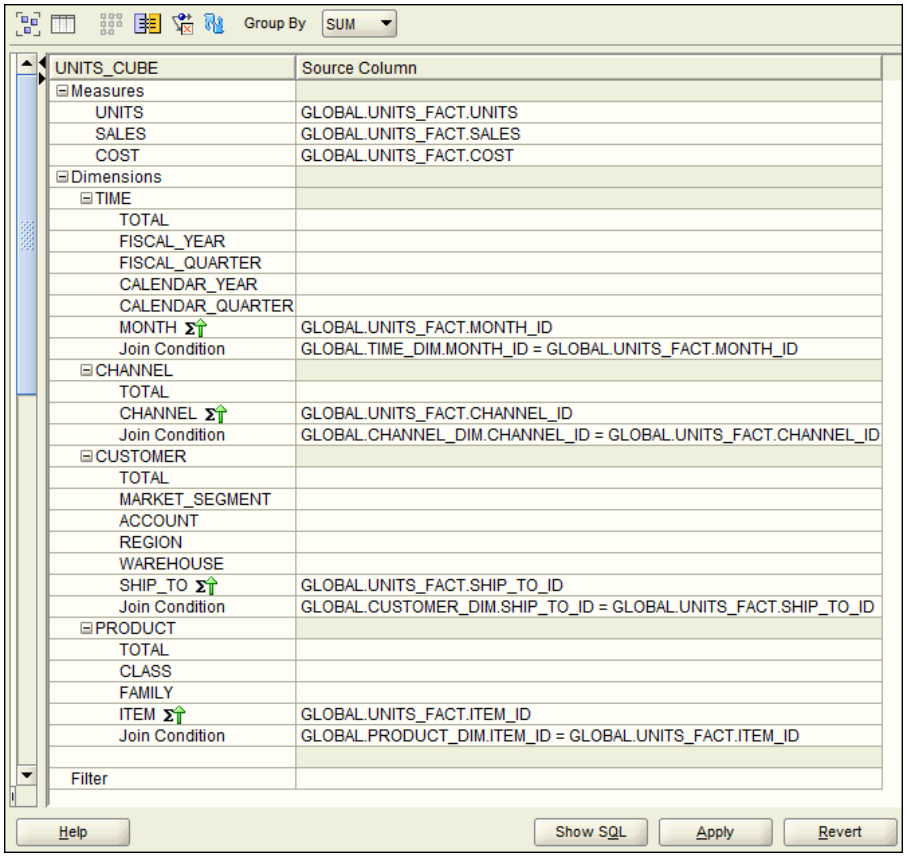
Copyright © 2003, 2025, Oracle and/or its affiliates.

3. In the schema tree, expand the tables, views, or synonyms that contain the data for the measures.
4. Drag-and-drop the source columns onto the appropriate cells in the mapping table for the cube.

   Map a measure dimension to the measure dimension fact view. See "Creating Measure Dimensions" for information on creating the measure dimension fact view. From the measure dimension fact view columns, specify `MEASURE_VALUE` as the source column for the measure of the cube and specify `MEASURE_DIM` as the source column for the measure dimension of the cube.
5. *Optional*: To see the SQL statements for the mapping, click **Show SQL**. You can save the SQL to a file or to the clipboard.
6. After you have mapped all dimensions and measures, click **Apply**.
7. Drag the divider back to the right to reduce the size of the Mapping window.

Figure 3-14 shows the mapping canvas with the Units cube mapped to columns in the `UNITS_FACT` table. After you save the mappings, Analytic Workspace Manager provides the join conditions for base-level mappings such as the ones shown here.

Figure 3-14 Units Cube Mapped in the Tabular View



To calculate the facts of a measure as they are loaded into a cube:

1. Create the cube.
2. Map all dimensions and measures to the source tables.

Copyright © 2003, 2025, Oracle and/or its affiliates.

3. Edit the mapping of the measure to include a calculation in the OLAP expression syntax.

   For example, you might change `UNITS_FACT.SALES` to `UNITS_FACT.SALES*1.06`.

   You can use row expressions, column expressions, and conditions, but not nested SQL queries.

To map a cube above the detail level:

1. Create the cube dimensions with the desired levels and map them to the source dimension table.
2. Create the cube and its measures.
3. Map each measure to its source column in the fact table.
4. For dimensions that are not being consolidated, map the detail level to its source column in the fact table, the same as you would in a basic cube mapping.
5. For dimensions being consolidated:
   a. Map the dimension to the appropriate column in the dimension table, not to the fact table. In the previous scenario, you would map the Month level of the Time dimension to the Month column of the Time dimension table. For example, you would map Month to time_dim.month_column.
   b. Enter a join condition between the fact table and the dimension table at the detail level. For example, `time_dim.day_key = fact_tbl.day_foreign_key`.

To map measures to different tables:

1. Create the cube dimensions with the desired levels and map them to the source dimension table.
2. Create the cube and its measures.
3. Map each measure to its source column in the appropriate table.
4. Map the detail level of each of the dimensions to its source column in each of the tables. When you drop the additional source column names, you are asked whether to add or replace the existing mapping. Select **Add**.

Example 3-2 Mapping Measures to Different Tables

This example maps the two measures of a cube to columns in two different fact tables. The data for `UNIT_PRICE` is in the `UNITS_FACT` table, and the data for `UNITS_SOLD` is in the `PRICE_FACT` table. The following mapping identifies the dimension keys in both tables for `MONTH` and `PRODUCT`.

```
UNIT_PRICE: GLOBAL.PRICE_FACT.UNIT_PRICE
UNITS_SOLD: GLOBAL.UNITS_FACT.UNITS
MONTH:      GLOBAL.PRICE_FACT.MONTH_ID
            GLOBAL.UNITS_FACT.MONTH_ID
PRODUCT:    GLOBAL.PRICE_FACT.ITEM_ID
            GLOBAL.UNITS_FACT.ITEM_ID
```

The next example maps one measure of a cube to columns in two different fact tables. The data for North America is in the `AMERICA` table, and the data for Europe is in the `EMEA` table. The following mapping for the `UNITS_SOLD` measure of `UNION_CUBE` creates a union of the two fact columns.

```
UNITS_SOLD: GLOBAL.AMERICA.UNITS
            GLOBAL.EMEA.UNITS
TIME:       GLOBAL.AMERICA.MONTH_ID
            GLOBAL.EMEA.MONTH_ID
```


Copyright © 2003, 2025, Oracle and/or its affiliates.

```
CHANNEL:    GLOBAL.AMERICA.CHANNEL_ID
            GLOBAL.EMEA.CHANNEL_ID
CUSTOMER:   GLOBAL.AMERICA.SHIP_TO_ID
            GLOBAL.EMEA.SHIP_TO_ID
PRODUCT:    GLOBAL.AMERICA.ITEM_ID
            GLOBAL.EMEA.ITEM_ID
```

3.5.3 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.

You can select multiple hierarchies and multiple levels of a hierarchy for partitioning.

You select partitions and specify properties of them on the Partitioning tab of the property sheet for a cube. You can also view information about the partitions to help you decide on a partitioning strategy.

> **Note**
>
> Cubes are partitioned by default.

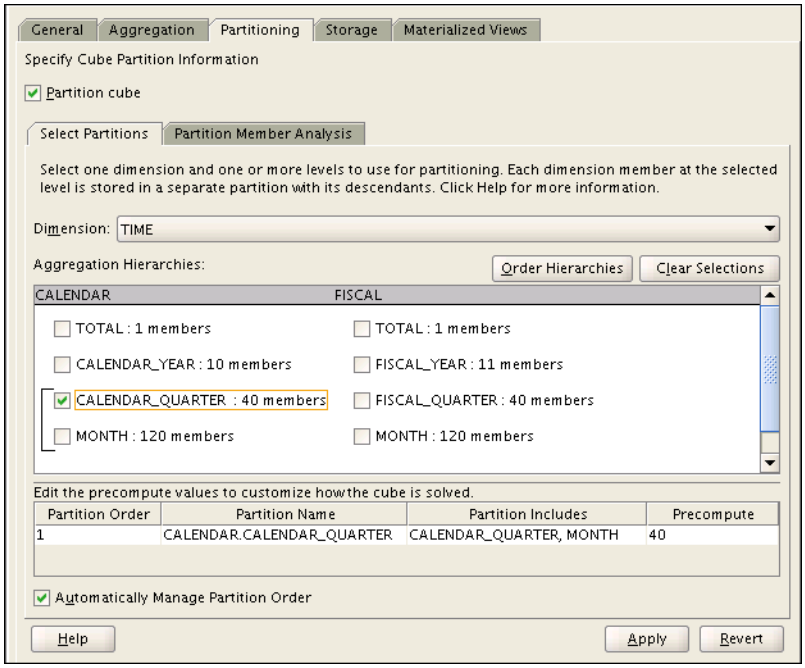
To select partitions:

1. In the navigation tree, select a cube.
2. In the property sheet, select the Partitioning tab.

   The Partitioning tab appears, as shown in Figure 3-15.
3. Select **Partition Cube** and the Select Partitions subtab.
4. Complete the Select Partitions subtab.
5. *Optional*: To view information about the partitions, select the Partition Member Analysis subtab.
6. To apply the partitioning to the cube, click **Apply**.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Figure 3-15 Selecting Partitions



- Selecting Partitions
- Analyzing Partition Members

> **See Also**
>
> - Selecting Partitions
> - Analyzing Partition Members

3.5.3.1 Selecting Partitions

You select the dimension and levels to be used for partitioning on the Select Partitions subtab. This section describes the following choices you can make on the subtab.

Dimension

A dimension for partitioning the cube. The dimension must have at least one level-based hierarchy. In developing a partitioning strategy, you typically want the members to be distributed evenly, such that each partition has about the same amount of data as the others, to support the best performance. You can switch among dimensions without losing your selections in Aggregation Hierarchies, and so you can freely explore your data. By default, partitions are created on a time dimension.

Aggregation Hierarchies

From the hierarchies and their levels for the selected dimension, you select the levels for partitioning. If the dimension has multiple hierarchies and you are partitioning on only one of them, choose the one that has the most members; it should be defined as the default

Copyright © 2003, 2025, Oracle and/or its affiliates.

hierarchy. After you make a selection, brackets enclose the levels that will be stored in the same partition.

Each dimension member at the selected level is stored in a separate partition, along with its descendants. Any dimension members that are at higher levels or are not in the hierarchy are stored together, unless you select multiple levels for partitioning.

Choose the levels with care to distribute the data evenly across the partitions. For example, if the time dimension has 10 years of data at the year, quarter, month, and day levels, then you might partition at the quarter level. This choice creates 40 partitions, one for each quarter and its descendants (months and days). The 10 members at the year level are stored together in a separate partition. If the data is very sparse, then you might partition by year instead of quarter.

Another example is a time dimension with two hierarchies, calendar and fiscal, with month and day levels in both hierarchies. In this scenario, you might partition on the month, calendar year, and fiscal year levels.

The goal is to create partitions that fit in memory, which optimizes performance. The more memory your computer has, the larger the partitions can be and still achieve this goal.

Order Hierarchies

You can change the aggregation order of the hierarchies for the selected dimension.

Clear Selections

You can delete all hierarchy selections from the current display. Any selected hierarchies in other dimensions are unaffected.

Edit the Precompute Values

You can edit the percentage of values that are calculated and stored during data maintenance. The remaining members are calculated on demand in response to a query. In general, you should precompute the values that are queried most frequently.

A value of 0 does not create any aggregate values; they are calculated at run-time to provide the answer sets to queries. The result of 0% pre-aggregation is the fastest maintenance, the least storage space, but the slowest query response time. A value of 100 creates all of the aggregate values, which are simply fetched in response to queries. The result of 100% pre-aggregation is the longest maintenance, the most storage space, but the fastest query response time. Most DBAs choose values between these two extremes to balance the performance requirements for queries with the limitations of a data maintenance window.

A value of 1 only creates 1% of the aggregate values, but also creates the data structures for storing and tracking the aggregates. Thus, the amount of time to calculate this small percentage is correspondingly longer.

You may want to adjust the percentages over time to balance runtime performance with maintenance restrictions on time and disk space.

- **Partition Order**: The order in which the partitions are aggregated.
- **Partition Name**: Name assigned to the partition.
- **Partition Includes**: Levels included in the partition.
- **Precompute**: The percentage of precomputed values in this partition. You can edit this value unless Disable Editing of Cube Precompute Values is selected in the Configuration dialog box.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Automatically Manage Partition Order

You can enable Oracle OLAP to determine the optimal aggregation order. Do *not* select this option when the aggregation order changes the results. Order is important for some aggregation operators, such as Average, and when a cube uses multiple aggregation methods, such as Hierarchical Last Member for Time and Sum for all other dimensions.

This option appears only when the Show Automatic Partitioning Order Check Box is selected in the Analytic Workspace Manager Configuration dialog box.

See Also

- "When Does Aggregation Order Matter?"

3.5.3.2 Analyzing Partition Members

The Partition Member Analysis subtab shows how the members of the selected dimension are distributed among the partitions. Use this information to create a partitioning strategy with approximately an even number of dimension members in each partition.

The information appears in tabular and graphic formats.

Table

The table provides this information about the specified partitions:

- **Partition Name**: Name of the partition, as shown in the **Select Partitions** subtab.
- **Number Partitions**: Number of partitions created by partitioning on the selected level.
- **Total Members**: Total number of dimension members being distributed across the partitions. This number includes the members at the level selected for partitioning and their children at levels included in the partition.
- **Minimum Members**: Minimum number of dimension members assigned to a partition.
- **Maximum Members**: Maximum number of dimension members assigned to a partition.
- **Average Members**: Average number of dimension members assigned to a partition.
- **Standard Deviation**: Amount of variation among the partitions from the average. A lower standard deviation is better than a high standard deviation.

Graph

The graph illustrates the partition selected in the table. It provides a visual representation of the number of members in each partition and their level in the dimension hierarchy.

A tool bar enables you to make temporary changes to the graph. The text tools are disabled. You can use these tools:

- **Fill Color**: Changes the background color surrounding the graph.
- **Graph Type**: Provides a variety of standard graph types, as described in Table 3-1.
- **Legend**: Controls whether the legend is displayed.
- **Grid Lines**: Controls whether horizontal grid lines are displayed on graphs with an X/Y axis.

Copyright © 2003, 2025, Oracle and/or its affiliates.

- **Gradient Effect**: Controls whether colored areas appear solid or with a slight variation in color.
- **3-D Effect**: Controls whether the graph appears flat or three-dimensional.

Table 3-1 Partitioning Graph Types

Graph Type	Usage
Bar	Comparisons (default)
Horizontal Bar	Comparisons
Pie	Percentage or comparisons of percentages; relationship between the parts and the whole
Line	Trends over time; rate of data change
Area	Trends over time; rate of data change
Combination	Trends over time; effect of one variable on another
Scatter	Correlations of two or three measures
Stock	Stock prices over time
Circular	Cyclical or directional patterns
Pareto	Highest and lowest contributors to a total; ranking
3-D	Three-dimensional comparison

3.5.4 Loading Data Into Cubes

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.

To load data into a cube:

1. In the navigation tree, right-click the Cubes folder or the name of a particular cube.
2. Select **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 loaded, you can omit them from Selected Target Objects.

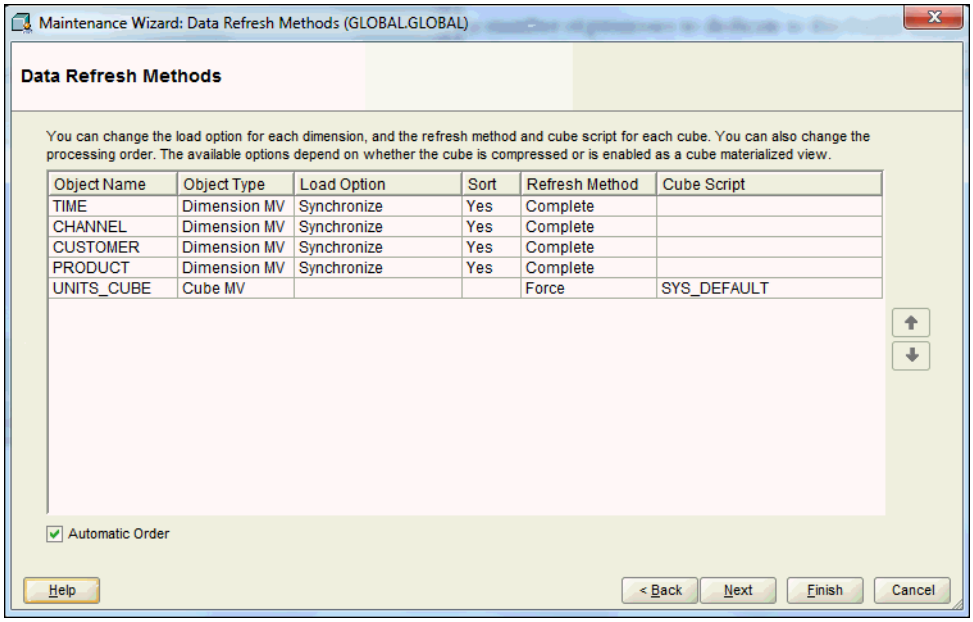
   If you click **Next**, the Data Refresh Methods page appears.
4. The Data Refresh Methods page identifies the cubes and dimensions included in the build, the load options, sort order, refresh methods, and the cube script that defines the steps of the build.

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

   Figure 3-16 shows the Data Refresh Methods page.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Figure 3-16 Selecting Build Options



If you click **Next**, the Processing Options page appears.

5. On the Processing Options page, you can keep the default values.

   If you click **Next**, the Scheduling page appears.

6. On the Scheduling page, you can specify task processing options. 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.

7. Click **Finish**.

Figure 3-17 shows the build submitted immediately to the Oracle job queue.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Figure 3-17 Selecting the Scheduling Options

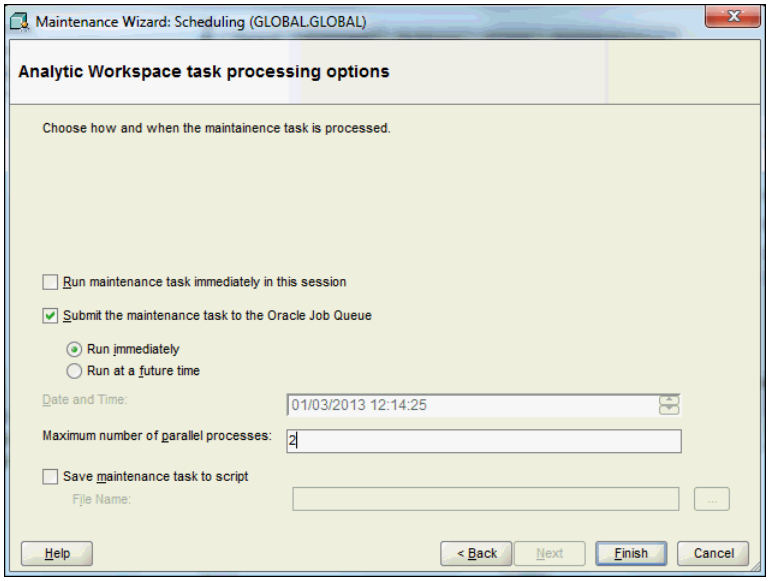
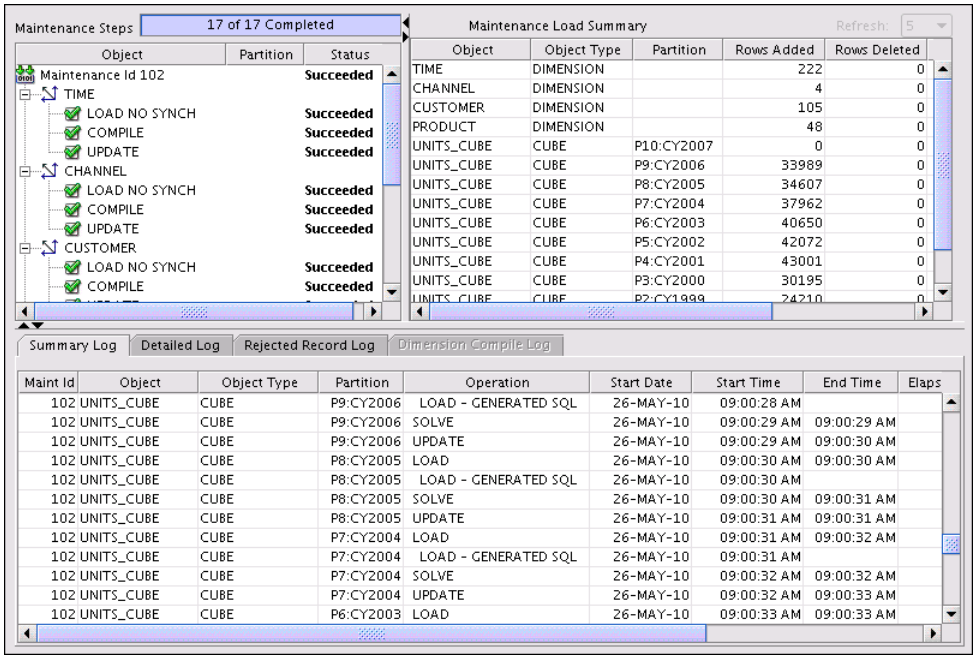


Figure 3-18 shows the maintenance log displayed by Analytic Workspace Manager for a cube. The log refreshes throughout the build to provide you with the most up-to-date information. The maintenance log appears automatically for maintenance tasks that run immediately in the session. When you submit a job to the Oracle job queue, you can track its progress through the various reports in the Maintenance Reports folder: Jobs Scheduled, Jobs Running, and Jobs History. The reports in Jobs Running and Jobs History are the same as the one shown in Figure 3-18.

Figure 3-18 Maintenance Log for the Units Cube



Copyright © 2003, 2025, Oracle and/or its affiliates.

3.5.5 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. Select **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 want, you can use multiple dimensions to label the columns and rows, by nesting one dimension under another.

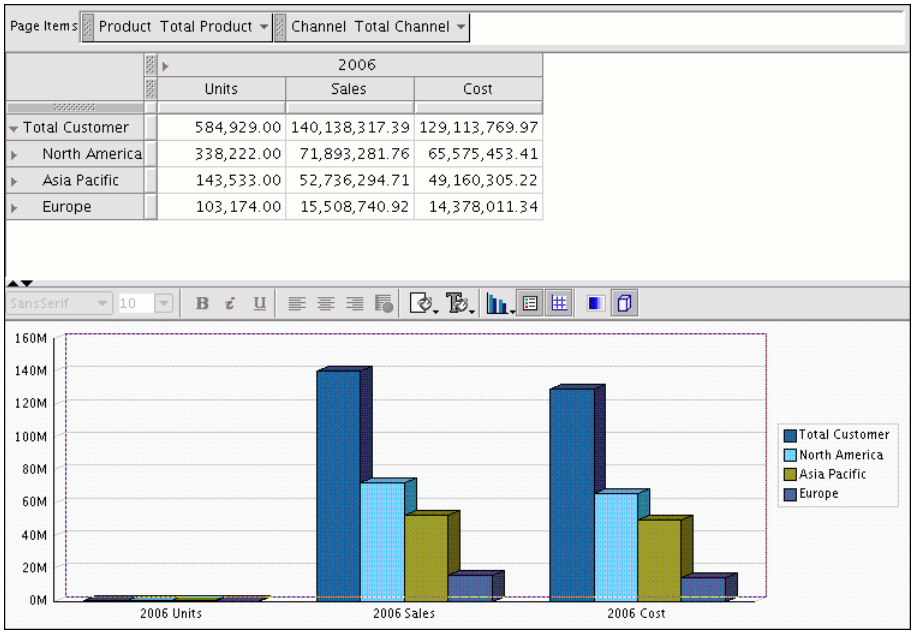
To change the default display:

- 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, select **Query Builder** from the File menu.

Figure 3-19 shows the Units cube in the Measure Viewer.

Figure 3-19 Displaying the Units Cube



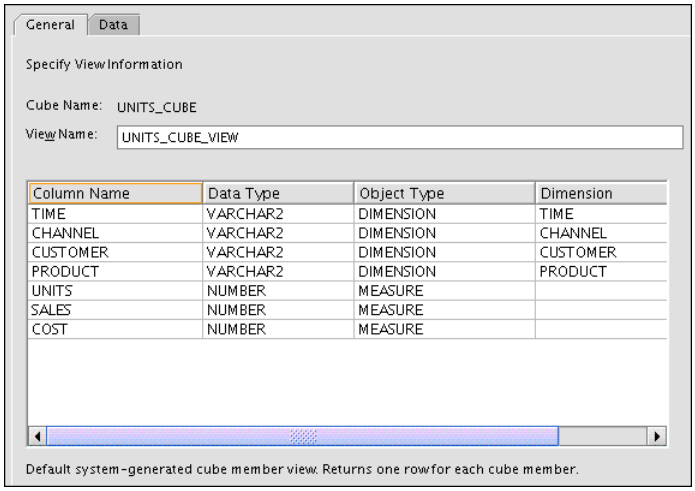
3.5.6 Displaying the Cube View Descriptions

The Maintenance Wizard automatically generates relational views of a cube. Querying Dimensional Objects describes these views and explains how to query them.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Figure 3-20 shows the description of the relational view of the Units cube.

Figure 3-20 Description of the Units Cube View



3.6 Choosing a Data Maintenance Method

While developing a dimensional model of your data, mapping and loading each object immediately after you create it is a good idea. That way, you can detect and correct any errors that you made to the object definition or the mapping.

However, in a production environment, you want to perform routine maintenance as quickly and easily as possible. For this stage, you can choose among data maintenance methods.

You can refresh all cubes using the Maintenance Wizard. This wizard enables you to refresh a cube immediately, or submit the refresh as a job to the Oracle job queue, or generate a PL/SQL script. You can run the script manually or using a scheduling utility, such as Oracle Enterprise Manager Scheduler or the `DBMS_SCHEDULER` PL/SQL package.

The generated script calls the `BUILD` procedure of the `DBMS_CUBE` PL/SQL package. You can modify this script or develop one from the start using this package.

The data for a partitioned cube is loaded and aggregated in parallel when multiple processes have been allocated to the build. You are able to see this in the build log.

In addition, each cube can support these data maintenance methods:

- Custom cube scripts
- Maintenance scripts
- Cube materialized views

If you are defining cubes to replace existing materialized views, then you use the materialized views as an integral part of data maintenance. Materialized view capabilities restrict the types of analytics that can be performed by a custom cube script.

- Creating and Executing Custom Cube Scripts
- Creating and Executing Maintenance Scripts
- Adding Materialized View Capability to a Cube

Copyright © 2003, 2025, Oracle and/or its affiliates.

See Also

- "Maintenance Logs"
- "Parallelism"

3.6.1 Creating and Executing Custom Cube Scripts

A cube script is an ordered list of steps that prepare a cube for querying. Each step represents a particular data transformation. By specifying the order in which these steps are performed, you can allow for interdependencies.

You can choose from these step types:

- **Clear Data**: Clears the data from the entire cube, from selected measures, or from selected portions of the cube. You can clear just the detail data (called **leaves**) for a fast refresh, just the aggregate data, or both for a complete refresh. Clearing old data values is typically done before loading new values.
- **Load**: Loads the data from the source tables into the cube. You can load all measures in the cube or just selected measures.
- **Aggregation**: Generates aggregate values using the rules defined for the cube. You can aggregate the entire cube, selected measures, or selected portions of the cube.
- **Analyze**: Generates optimizer statistics, which can improve the performance of some types of queries. For more information, see "Analyzing Cubes and Dimensions". Generating statistics is typically done immediately after data maintenance.
- **OLAP DML**: Run a command or program in the OLAP DML.
- **PL/SQL**: Run a PL/SQL command or script. You can run a PL/SQL script, for example, at the beginning of data maintenance to initiate a refresh of the relational source tables.

If a cube is used to support advanced analytics in a cube script, then it cannot be enhanced as a cube materialized view, as described in "Adding Materialized View Capability to a Cube". In this case, you are responsible for detecting when the data in the cube is stale and must be refreshed.

- Creating Cube Scripts
- Running a Cube Script

3.6.1.1 Creating Cube Scripts

To create a cube script:

1. Expand the folder for a cube that is not defined as a cube materialized view.
2. Right-click **Cube Scripts**, then select **Create Cube Script**.

   The Create Cube Script dialog box appears.
3. On the General tab, enter a name for the cube script.
4. To create a step, click **New Step**.
5. Select the type of step.

   The New Step dialog box appears for that type of step.
6. Complete the tabs, then click **OK**.

Copyright © 2003, 2025, Oracle and/or its affiliates.

The step is listed on the Cube Script General tab.

7. Click **Create**.

   The cube script appears as an item in the Cube Script folder.

8. To run the cube script:

   a. Right-click the cube script on the navigation tree, and select **Run Cube Script**.

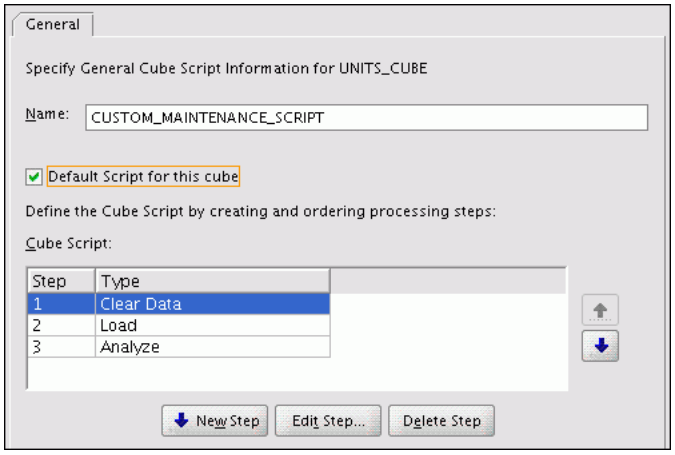
      The Maintenance Wizard opens.

   b. Follow the steps of the wizard.

   c. To view the results, right-click the cube and select **View Data**.

Figure 3-21 shows the Create Cube Script dialog box, in which several steps have been defined.

Figure 3-21 Creating a Cube Script



3.6.1.2 Running a Cube Script

Each cube automatically has a default cube script named `LOAD_AND_AGGREGATE` that loads the data and aggregates it using the rules defined on the cube. You can define any number of additional scripts and designate one as the default cube script. All methods of refreshing a cube runs the default cube script. You can run other cube scripts manually using the Maintenance Wizard.

To manually run a custom cube script:

1. Expand the Cube Scripts folder for the cube.
2. Right-click the cube script and select **Run Cube Script** to open the Maintenance Wizard.
3. Follow the steps of the Maintenance Wizard.

To run a custom cube script as the default script:

1. Expand the Cube Scripts folder for the cube.
2. Select the cube script so the General tab appears.
3. Select Default Script For This Cube and click **Apply**.
4. Open the Maintenance Wizard anywhere on the navigation tree and select the cube.

Copyright © 2003, 2025, Oracle and/or its affiliates.

5. Follow the steps of the Maintenance Wizard.

To run a cube script as a step in a maintenance script:

1. Create a maintenance script.
2. Add the cube script as a step.
3. Run the maintenance script.

3.6.2 Creating and Executing Maintenance Scripts

A maintenance script is an ordered list of steps for maintaining multiple cubes in a schema. By using a maintenance script, you can manage interdependencies among the cubes.

To load and aggregate a cube or a dimension, add it as a step. For more control over the maintenance of a particular cube or dimension, either create a cube script or enter the individual steps directly into the maintenance script:

- Clear Data
- Load
- Aggregation
- Analyze
- OLAP DML
- PL/SQL

These are the same steps described in "Creating and Executing Custom Cube Scripts".

- Creating Maintenance Scripts
- Running Maintenance Scripts

3.6.2.1 Creating Maintenance Scripts

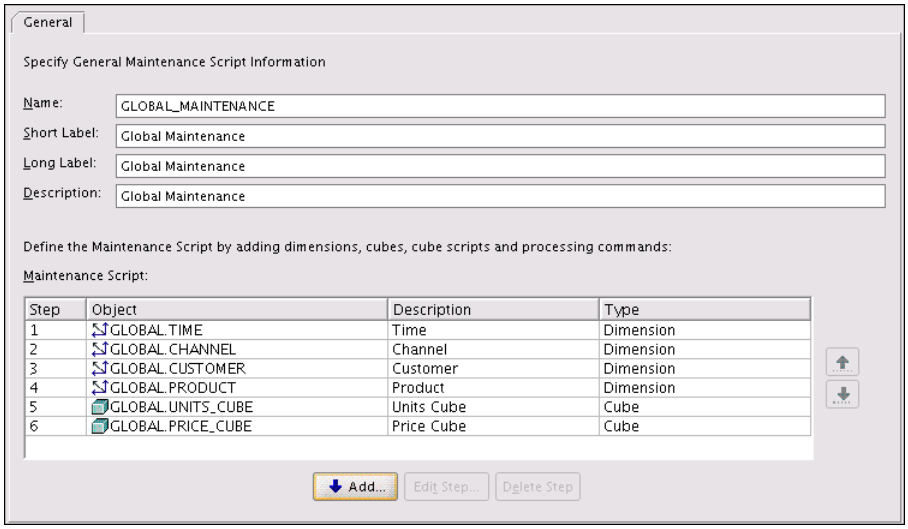
To create a maintenance script:

1. In the navigation tree, right-click Maintenance Scripts, then select **Create Maintenance Script** to display the Create Maintenance Script dialog box.
2. Enter the name, labels, and description on the General tab.
3. To create a new step, click **Add**, then select the type of step from the list.
4. Create additional steps as desired. You can edit, delete, or re-order the steps at any time.
5. Click **Create**. The new maintenance script appears as an object in the Maintenance Scripts folder.

Figure 3-22 shows the General tab of the Create Maintenance Script dialog box.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Figure 3-22 Creating a Maintenance Script



3.6.2.2 Running Maintenance Scripts

To run a maintenance script:

1. Expand the Maintenance Scripts folder.
2. Right-click the script, then select **Run Maintenance Script**.
3. The Maintenance Wizard opens.
4. Follow the steps of the Maintenance Wizard.

3.6.3 Adding Materialized View Capability to a Cube

Oracle OLAP cubes can be enhanced with materialized view capabilities. Cubes can be incrementally refreshed through the Oracle Database materialized view subsystem, and they can serve as targets for transparent rewrite of queries against the source tables. A cube that has been enhanced in this way is called a **cube materialized view**.

The OLAP dimensions associated with a cube materialized view are also defined with materialized view capabilities.

A cube must conform to these requirements, before it can be designated as a cube materialized view:

- All dimensions of the cube have at least one level and one level-based hierarchy. Ragged and skip-level hierarchies are not supported. The dimensions must be mapped.
- All dimensions of the cube use the same aggregation operator, which is either `SUM`, `MIN`, or `MAX`.
- The cube has one or more dimensions and one or more measures.
- The cube is fully defined and mapped. For example, if the cube has five measures, then all five are mapped to the source tables.
- The data type of the cube is `NUMBER`, `VARCHAR2`, `NVARCHAR2`, or `DATE`.

Copyright © 2003, 2025, Oracle and/or its affiliates.

- The source detail tables support dimension and rely constraints. If they have not been defined, then use the Relational Schema Advisor on the Materialized Views tab of the cube property sheet to generate a script that defines them on the detail tables.
- The cube is compressed.
- The cube can be enriched with calculated measures, but it cannot support more advanced analytics in a cube script.

> **See Also**
>
> "Cube Materialized Views"

To add materialized view capabilities:

1. In the navigation tree, select a cube.

   The property sheets for the cube are displayed.
2. Select the Materialized Views tab.
3. Review the checklist and, if some tests failed, fix the cause of the problem.

   You cannot define a cube materialized view until the cube is valid.
4. For automatic refresh, complete just the top half page. For query rewrite, complete the entire page.

   Click **Help** for information about the choices on this page.
5. Click **Apply**.

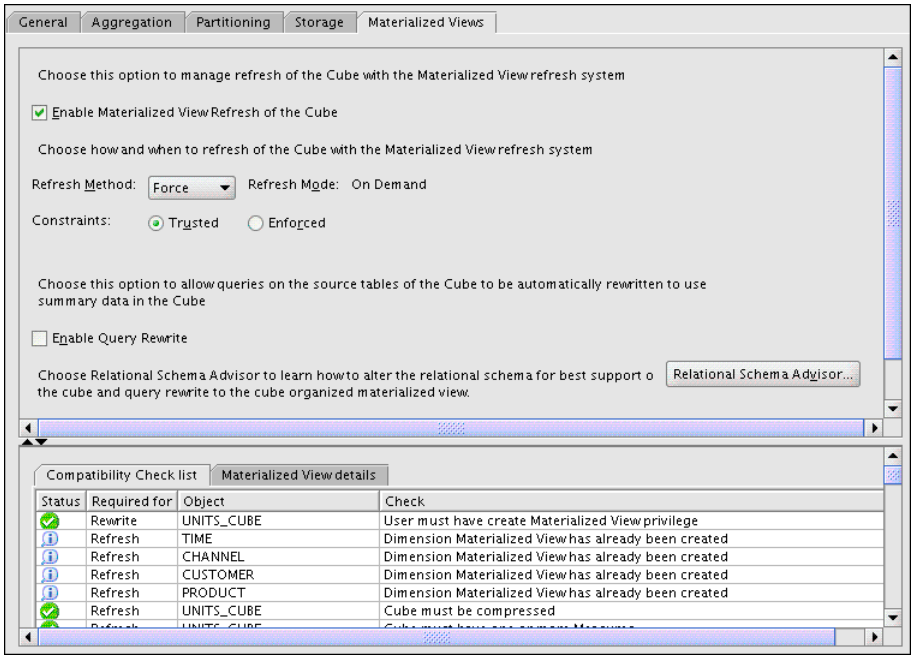
The cube materialized views appear in the same schema as the analytic workspace. A materialized view is created for the cube and each of its dimensions. Unlike traditional materialized views, cube materialized views do not use relational tables to store data; the data is stored in the backing cube. A `CB$` prefix identifies the tables as cube materialized views.

The initial state of a new materialized view is invalid, so it does not support query rewrite until after it is refreshed. You can specify the first refresh time on the Materialized View tab of the cube, or you can run the Maintenance Wizard.

Figure 3-23 shows the Materialized View tab of the Units Cube.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Figure 3-23 Defining a Materialized View



3.7 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.

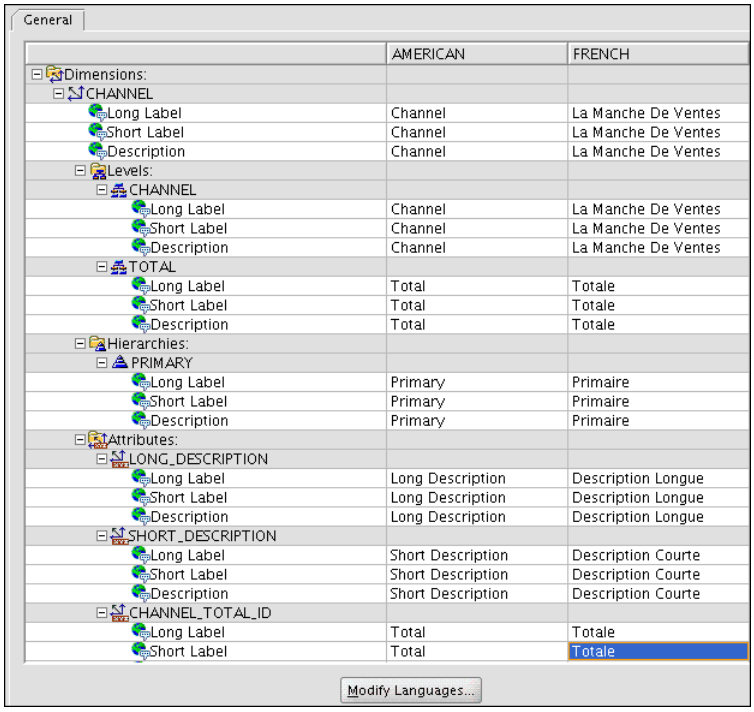
To add support for multiple languages:

1. In the navigation tree, expand the folder for the analytic workspace.
2. Select Languages to display its property page.
3. On the General tab, click **Modify Languages**.
4. On the Modify Languages dialog box, select the languages that the analytic workspace must support. Use the shuttle keys to move them to the Selected Languages box.
5. Click **OK** to return to the Languages property page.
6. Enter the translations of the various labels and descriptions. Each language has a column where you can enter this information.
7. For each dimension, open the Mappings window. Map the attributes to the source columns for each language.

Figure 3-24 shows the addition of French to the analytic workspace.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Figure 3-24 Adding a Language



3.8 Defining Measure Folders

Measure folders organize and label groups of measures. Users may have access to several analytic workspaces or relational schemas with measures named Sales or Costs, and measure folders provide a way for applications to differentiate among them.

To create a measure folder:

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

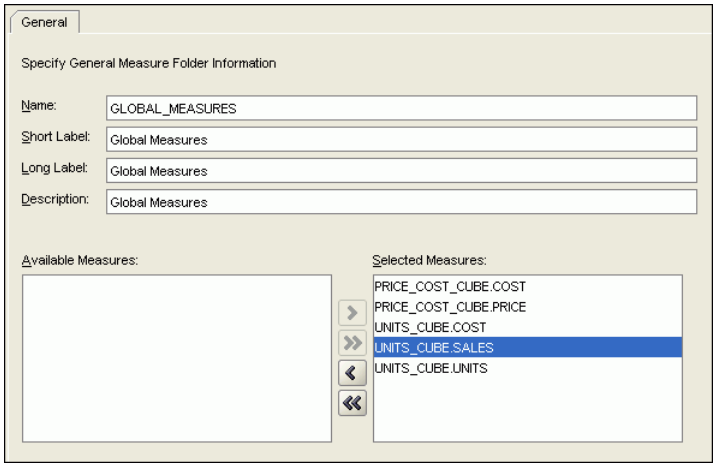
   Click **Help** for specific information about these choices.

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

Figure 3-25 shows creation of a measure folder.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Figure 3-25 Creating a Measure Folder



3.9 Saving and Re-Creating Dimensional Objects with Object Definitions

Analytic Workspace Manager enables you to save all or part of the data model as a template. You can save a template to a file or to a table. The template contains the XML definitions of the dimensional objects, such as dimensions, levels, hierarchies, attributes, and measures. Only the metadata is saved, not the data.

Template files are small, so you can easily distribute them by email or on a website, just as the templates for Global and Sales History are distributed on the Oracle website. A template saved to a table is available to any user of the database who has permission to see it. Oracle OLAP saves templates to the CUBE_TEMPLATES table.

To re-create the dimensional objects, you simply identify the templates in Analytic Workspace Manager.

You can also save an analytic workspace to, or create one from, an EIF file. EIF files are specially formatted files for copying analytic workspaces. They save the definitions of OLAP DML objects and optionally save the data also.

This section has the following topics:

- Creating Dimensional Objects From XML Templates
- Saving Object Definitions to XML Templates
- Creating Analytic Workspaces from EIF Files
- Saving Analytic Workspaces to EIF Files

- Creating Dimensional Objects From XML Templates
- Saving Object Definitions to XML Templates
- Creating Analytic Workspaces from EIF Files
- Saving Analytic Workspaces to EIF Files

Copyright © 2003, 2025, Oracle and/or its affiliates.

> **See Also**
>
> - "Mapping Cubes" for information on saving the SQL statements for a mapping.

3.9.1 Creating Dimensional Objects From XML Templates

You can create all or part of an analytic workspace from a template.

To create dimensional objects from a template:

1. In the navigation tree, right-click Analytic Workspaces, Dimensions, Cubes, or Measure Folders.
2. Select **Create *Object* from Template** to display the Create *Object* from Template dialog box.
3. Select the schema in which to create the objects and click **OK**.
4. Complete the Create *Object* from Template dialog box.

   To overwrite the metadata for an existing object select **Modify Existing Objects** on the Options tab.

> **See Also**
>
> - "About XML Templates"

3.9.2 Saving Object Definitions to XML Templates

You can save the XML descriptions of all the objects in an analytic workspace, or just selected objects, and re-create them later in the same database or in a database on another computer or platform.

To save object definitions in an XML template:

1. In the navigation tree, right-click an analytic workspace, dimension, cube, or measure folder.
2. Select **Save *Object* to Template** to display the Save *Object* to Template dialog box.
3. Select **Save to File** or **Save to Table**.
4. Verify the selection of objects in the Object Selection tab.
5. To modify the use of the schema name in the template, use the Options tab.
6. Complete the remaining fields to identify the name and location of the saved template. You can overwrite an existing template.

> **See Also**
>
> - "About XML Templates"

Copyright © 2003, 2025, Oracle and/or its affiliates.

3.9.3 Creating Analytic Workspaces from EIF Files

EIF files are specially formatted files for transferring dimensional objects and data.

To create an analytic workspace from an EIF file:

1. In the navigation tree, right-click Analytic Workspaces and select **Create Analytic Workspace From EIF File**.

   The Create Analytic Workspace From EIF File dialog box appears.

2. Specify the directory that contains the EIF file and the name of the file, a name for the new analytic workspace and the tablespace for it, and then click **OK**.

> **See Also**
>
> - "About EIF Files"

3.9.4 Saving Analytic Workspaces to EIF Files

You can save, or export, an analytic workspace to an EIF file.

To save analytic workspace objects to an EIF file:

1. In the navigation tree, right-click the analytic workspace.
2. Select **Export Analytic Workspace *Object*To EIF File** to display the Export Analytic Workspace *Object* to EIF File dialog box.
3. Specify the directory and file name for the EIF file, then click **OK**.

> **See Also**
>
> - "About EIF Files"

3.10 Copying and Pasting Dimensional Objects

You can copy a dimensional object and paste it in an appropriate location. The analytic workspace objects that you can copy are the following.

- Analytic workspace
- Dimension
- Cube
- Measure
- Calculated measure

Copyright © 2003, 2025, Oracle and/or its affiliates.

To copy a dimensional object:

- In the navigation tree, right-click an analytic workspace, dimension, cubes, measure, or calculated measure.
- Select **Copy**.

To paste a dimensional object:

- In the navigation tree, right-click Analytic Workspaces, Dimensions, Cubes, Measures, or Calculated Measures.
- Select **Paste**.

Copyright © 2003, 2025, Oracle and/or its affiliates.

4

Querying Dimensional Objects

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.

The OLAP option automatically generates a set of relational views on cubes, dimensions, and hierarchies. SQL applications query these views to display the information-rich contents of these objects to analysts and decision makers. You can also create custom views that follow the structure expected by your applications, using the system-generated views like base tables.

In this chapter, you learn the basic methods for querying dimensional objects in SQL. It contains the following topics:

- Exploring the OLAP Views
- Creating Basic Queries
- Creating Hierarchical Queries
- Using Calculations in Queries
- Using Attributes for Aggregation
- Joining Cubes to Tables and Views
- Viewing Execution Plans
- Querying the Data Dictionary

- Exploring the OLAP Views
- Creating Basic Queries
- Creating Hierarchical Queries
- Using Calculations in Queries
- Using Attributes for Aggregation
- Joining Cubes to Tables and Views
- Viewing Execution Plans
- Querying the Data Dictionary

See Also

- "Developing Reports and Dashboards Using SQL Tools and Application Builders" for a sample dashboard created using Oracle Application Express
- "Overview of the Dimensional Data Model" for a discussion of cubes, dimensions, and hierarchies

Copyright © 2003, 2025, Oracle and/or its affiliates.

4.1 Exploring the OLAP Views

The system-generated views are created in the same schema as the analytic workspace. Oracle OLAP provides three types of views:

- Cube views
- Dimension views
- Hierarchy views

These views are related in the same way as fact and dimension tables are in a star schema. Cube views serve the same function as fact tables, and hierarchy views and dimension views serve the same function as dimension tables. Typical queries join a cube view with either a hierarchy view or a dimension view.

- Cube Views
- Dimension and Hierarchy Views

4.1.1 Cube Views

Each cube has a cube view that presents the data for all the measures and calculated measures in the cube. You can use a cube view like a fact table in a star or snowflake schema. However, the cube view contains all the summary data in addition to the detail level data.

- Discovering the Names of the Cube Views
- Discovering the Columns of a Cube View
- Displaying the Contents of a Cube View

4.1.1.1 Discovering the Names of the Cube Views

The default name for a cube view is *cube*_VIEW. To find the view for UNITS_CUBE in your schema, you might issue a query like this one:

```
SELECT view_name FROM user_views WHERE view_name LIKE 'UNITS_CUBE%';

VIEW_NAME
------------------------------
UNITS_CUBE_VIEW
```

The next query returns the names of all the cube views in your schema from USER_CUBE_VIEWS:

```
SELECT view_name FROM user_cube_views;

VIEW_NAME
------------------------------
UNITS_CUBE_VIEW
PRICE_CUBE_VIEW
```

4.1.1.2 Discovering the Columns of a Cube View

Like a fact table, a cube view contains a column for each measure, calculated measure, and dimension in the cube. In the following example, UNITS_CUBE_VIEW has columns for the SALES, UNITS, and COST measures, for several calculated measures on SALES, and for the TIME, CUSTOMER, PRODUCT, and CHANNEL dimensions.

Copyright © 2003, 2025, Oracle and/or its affiliates.

```
DESCRIBE units_cube_view
Name                                      Null?    Type
----------------------------------------- -------- ----------------------------
SALES                                              NUMBER
UNITS                                              NUMBER
COST                                               NUMBER
SALES_PP                                           NUMBER
SALES_CHG_PP                                       NUMBER
SALES_PCTCHG_PP                                    NUMBER
SALES_PROD_SHARE_PARENT                            NUMBER
SALES_PROD_SHARE_TOTAL                             NUMBER
SALES_PROD_RANK_PARENT_PP                          NUMBER
TIME                                               VARCHAR2(100)
CUSTOMER                                           VARCHAR2(100)
PRODUCT                                            VARCHAR2(100)
CHANNEL                                            VARCHAR2(100)
```

The USER_CUBE_VIEW_COLUMNS data dictionary view describes the columns of a cube view, as shown by the following query.

```
SELECT column_name, column_type FROM user_cube_view_columns
     WHERE view_name = 'UNITS_CUBE_VIEW';

COLUMN_NAME                    COLUMN_TYPE
------------------------------ --------------
SALES                          MEASURE
UNITS                          MEASURE
COST                           MEASURE
SALES_PP                       MEASURE
SALES_CHG_PP                   MEASURE
SALES_PCTCHG_PP                MEASURE
SALES_PROD_SHARE_PARENT        MEASURE
SALES_PROD_SHARE_TOTAL         MEASURE
SALES_PROD_RANK_PARENT_PP      MEASURE
TIME                           KEY
CUSTOMER                       KEY
PRODUCT                        KEY
CHANNEL                        KEY

13 rows selected.
```

4.1.1.3 Displaying the Contents of a Cube View

You can display the contents of a cube view quickly with a query like this one. All levels of the data are contained in the cube, from the detail level to the top.

```
SELECT sales, units, time, customer, product, channel
     FROM units_cube_view WHERE ROWNUM < 15;

     SALES      UNITS TIME       CUSTOMER   PRODUCT    CHANNEL
---------- ---------- ---------- ---------- ---------- --------
1120292752    4000968 TOTAL      TOTAL      TOTAL      TOTAL
 134109248     330425 CY1999     TOTAL      TOTAL      TOTAL
 130276514     534069 CY2003     TOTAL      TOTAL      TOTAL
 100870877     253816 CY1998     TOTAL      TOTAL      TOTAL
 136986572     565718 CY2005     TOTAL      TOTAL      TOTAL
 140138317     584929 CY2006     TOTAL      TOTAL      TOTAL
 144290686     587419 CY2004     TOTAL      TOTAL      TOTAL
 124173522     364233 CY2000     TOTAL      TOTAL      TOTAL
  92515295     364965 CY2002     TOTAL      TOTAL      TOTAL
 116931722     415394 CY2001     TOTAL      TOTAL      TOTAL
31522409.5      88484 CY2000.Q1  TOTAL      TOTAL      TOTAL
```


Copyright © 2003, 2025, Oracle and/or its affiliates.

```
27798426.6      97346 CY2001.Q2  TOTAL      TOTAL      TOTAL
29691668.2     105704 CY2001.Q3  TOTAL      TOTAL      TOTAL
32617248.6     138953 CY2005.Q3  TOTAL      TOTAL      TOTAL

14 rows selected.
```

4.1.2 Dimension and Hierarchy Views

Each dimension has one dimension view plus a hierarchy view for each hierarchy associated with the dimension. For example, a Time dimension might have these three views:

- Time dimension view
- Calendar hierarchy view
- Fiscal hierarchy view

You can use dimension views and hierarchy views like dimension tables in a star schema.

- Discovering the Names of Dimension and Hierarchy Views
- Discovering the Columns of a Dimension View
- Displaying the Contents of a Dimension View
- Discovering the Columns of a Hierarchy View
- Displaying the Contents of a Hierarchy View

4.1.2.1 Discovering the Names of Dimension and Hierarchy Views

USER_CUBE_DIM_VIEWS identifies the dimension views for all dimensions. The default name for a dimension view is *dimension*_VIEW.

```
SELECT * FROM user_cube_dim_views;

DIMENSION_NAME                 VIEW_NAME
------------------------------ ------------------------------
PRODUCT                        PRODUCT_VIEW
CUSTOMER                       CUSTOMER_VIEW
CHANNEL                        CHANNEL_VIEW
TIME                           TIME_VIEW
```

USER_CUBE_HIER_VIEWS identifies the hierarchy views for all the dimensions. For a hierarchy view, the default name is *dimension_hierarchy*_VIEW. The following query returns the dimension, hierarchy, and view names.

```
SELECT * FROM user_cube_hier_views ORDER BY dimension_name;

DIMENSION_NAME  HIERARCHY_NAME  VIEW_NAME
--------------- --------------- ------------------------------
CHANNEL         PRIMARY         CHANNEL_PRIMARY_VIEW
CUSTOMER        MARKET          CUSTOMER_MARKET_VIEW
CUSTOMER        SHIPMENTS       CUSTOMER_SHIPMENTS_VIEW
PRODUCT         PRIMARY         PRODUCT_PRIMARY_VIEW
TIME            FISCAL          TIME_FISCAL_VIEW
TIME            CALENDAR        TIME_CALENDAR_VIEW
```

4.1.2.2 Discovering the Columns of a Dimension View

Like a dimension table, a dimension view contains a key column, level name, level keys for every level of every hierarchy associated with the dimension, and attribute columns. In the

Copyright © 2003, 2025, Oracle and/or its affiliates.

following example, TIME_VIEW has a column for the dimension keys, the level name, and the dimension attributes.

```
DESCRIBE time_view
Name                                      Null?    Type
----------------------------------------- -------- ----------------------------
DIM_KEY                                            VARCHAR2(100)
LEVEL_NAME                                         VARCHAR2(30)
DIM_ORDER                                          NUMBER
END_DATE                                           DATE
LONG_DESCRIPTION                                   VARCHAR2(100)
SHORT_DESCRIPTION                                  VARCHAR2(100)
TIME_SPAN                                          NUMBER
```

USER_CUBE_DIM_VIEW_COLUMNS describes the information in each column, as shown in this query.

```
SELECT column_name, column_type FROM user_cube_dim_view_columns
     WHERE view_name ='TIME_VIEW';

COLUMN_NAME                    COLUMN_TYPE
------------------------------ --------------------
DIM_KEY                        KEY
LEVEL_NAME                     LEVEL_NAME
DIM_ORDER                      DIM_ORDER
END_DATE                       ATTRIBUTE
TIME_SPAN                      ATTRIBUTE
LONG_DESCRIPTION               ATTRIBUTE
SHORT_DESCRIPTION              ATTRIBUTE
```

4.1.2.3 Displaying the Contents of a Dimension View

The following query displays the level and attributes of each dimension key.

```
SELECT dim_key, level_name, long_description description, time_span, end_date
     FROM time_view WHERE dim_key LIKE '%2005%';

DIM_KEY      LEVEL_NAME           DESCRIPTION   TIME_SPAN END_DATE
------------ -------------------- ------------ ---------- ---------
CY2005       CALENDAR_YEAR        2005                365 31-DEC-05
CY2005.Q2    CALENDAR_QUARTER     Q2.05                91 30-JUN-05
CY2005.Q4    CALENDAR_QUARTER     Q4.05                92 31-DEC-05
CY2005.Q3    CALENDAR_QUARTER     Q3.05                92 30-SEP-05
CY2005.Q1    CALENDAR_QUARTER     Q1.05                90 31-MAR-05
2005.01      MONTH                JAN-05               31 31-JAN-05
2005.05      MONTH                MAY-05               31 31-MAY-05
2005.07      MONTH                JUL-05               31 31-JUL-05
2005.03      MONTH                MAR-05               31 31-MAR-05
2005.04      MONTH                APR-05               30 30-APR-05
2005.08      MONTH                AUG-05               31 31-AUG-05
2005.09      MONTH                SEP-05               30 30-SEP-05
2005.02      MONTH                FEB-05               28 28-FEB-05
2005.11      MONTH                NOV-05               30 30-NOV-05
2005.06      MONTH                JUN-05               30 30-JUN-05
2005.10      MONTH                OCT-05               31 31-OCT-05
2005.12      MONTH                DEC-05               31 31-DEC-05
FY2005       FISCAL_YEAR          FY2005              365 30-JUN-05
FY2005.Q4    FISCAL_QUARTER       Q4 FY-05             91 30-JUN-05
FY2005.Q1    FISCAL_QUARTER       Q1 FY-05             92 30-SEP-04
FY2005.Q2    FISCAL_QUARTER       Q2 FY-05             92 31-DEC-04
FY2005.Q3    FISCAL_QUARTER       Q3 FY-05             90 31-MAR-05
```


Copyright © 2003, 2025, Oracle and/or its affiliates.

```
22 rows selected.
```

4.1.2.4 Discovering the Columns of a Hierarchy View

Like the dimension views, the hierarchy views also contain columns for the dimension key, level name, and level keys. However, all of the rows and columns are associated with the dimension keys that belong to the hierarchy.

```
DESCRIBE time_calendar_view
Name                                      Null?    Type
----------------------------------------- -------- ----------------------------
DIM_KEY                                            VARCHAR2(100)
LEVEL_NAME                                         VARCHAR2(30)
DIM_ORDER                                          NUMBER
HIER_ORDER                                         NUMBER
LONG_DESCRIPTION                                   VARCHAR2(100)
SHORT_DESCRIPTION                                  VARCHAR2(100)
END_DATE                                           DATE
TIME_SPAN                                          NUMBER
PARENT                                             VARCHAR2(100)
TOTAL                                              VARCHAR2(100)
CALENDAR_YEAR                                      VARCHAR2(100)
CALENDAR_QUARTER                                   VARCHAR2(100)
MONTH                                              VARCHAR2(100)
```

4.1.2.5 Displaying the Contents of a Hierarchy View

The following query displays the dimension keys, parent key, and the full ancestry for calendar year 2005.

```
SELECT dim_key, long_description description, parent, calendar_year year,
       calendar_quarter quarter, month FROM time_calendar_view
       WHERE calendar_year='CY2005'
       ORDER BY level_name, end_date;

DIM_KEY      DESCRIPTION  PARENT       YEAR         QUARTER      MONTH
------------ ------------ ------------ ------------ ------------ ------------
CY2005.Q1    Q1.05        CY2005       CY2005       CY2005.Q1
CY2005.Q2    Q2.05        CY2005       CY2005       CY2005.Q2
CY2005.Q3    Q3.05        CY2005       CY2005       CY2005.Q3
CY2005.Q4    Q4.05        CY2005       CY2005       CY2005.Q4
CY2005       2005         TOTAL        CY2005
2005.01      JAN-05       CY2005.Q1    CY2005       CY2005.Q1    2005.01
2005.02      FEB-05       CY2005.Q1    CY2005       CY2005.Q1    2005.02
2005.03      MAR-05       CY2005.Q1    CY2005       CY2005.Q1    2005.03
2005.04      APR-05       CY2005.Q2    CY2005       CY2005.Q2    2005.04
2005.05      MAY-05       CY2005.Q2    CY2005       CY2005.Q2    2005.05
2005.06      JUN-05       CY2005.Q2    CY2005       CY2005.Q2    2005.06
2005.07      JUL-05       CY2005.Q3    CY2005       CY2005.Q3    2005.07
2005.08      AUG-05       CY2005.Q3    CY2005       CY2005.Q3    2005.08
2005.09      SEP-05       CY2005.Q3    CY2005       CY2005.Q3    2005.09
2005.10      OCT-05       CY2005.Q4    CY2005       CY2005.Q4    2005.10
2005.11      NOV-05       CY2005.Q4    CY2005       CY2005.Q4    2005.11
2005.12      DEC-05       CY2005.Q4    CY2005       CY2005.Q4    2005.12

17 rows selected.
```


Copyright © 2003, 2025, Oracle and/or its affiliates.

4.2 Creating Basic Queries

Querying a cube is similar to querying a star schema. In a star schema, you join a fact table to a dimension table. The fact table provides the numeric business measures, and the dimension table provides descriptive attributes that give meaning to the data. Similarly, you join a cube view with either a dimension view or a hierarchy view to provide fully identified and meaningful data to your users.

For dimensions with no hierarchies, use the dimension views in your queries. For dimensions with hierarchies, use the hierarchy views, because they contain more information than the dimension views.

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 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
- Allowing the Cube to Aggregate the Data
- Query Processing

4.2.1 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 dimension or hierarchy views:

```
SELECT DISTINCT level_name FROM time_calendar_view;

LEVEL_NAME
------------------------------
CALENDAR_YEAR
CALENDAR_QUARTER
MONTH
TOTAL
```

Several data dictionary views list the names of the levels. The following example queries `USER_CUBE_HIER_LEVELS`.

```
SELECT level_name FROM user_cube_hier_levels
     WHERE dimension_name = 'TIME' AND hierarchy_name ='CALENDAR';

LEVEL_NAME
--------------------
TOTAL
CALENDAR_YEAR
CALENDAR_QUARTER
MONTH
```


Copyright © 2003, 2025, Oracle and/or its affiliates.

Example 4-1 Displaying Aggregates at All Levels of Time

To see the importance of applying a filter to every dimension, consider the query in this example, which has no filter on the time dimension.

```
/* Select key descriptions and facts */
SELECT t.long_description time,
     ROUND(f.sales) sales
/* From dimension views and cube view */
  FROM time_calendar_view t,
     product_primary_view p,
     customer_shipments_view cu,
     channel_primary_view ch,
     units_cube_view f
/* No filter on Time */
  WHERE p.level_name = 'TOTAL'
     AND cu.level_name = 'TOTAL'
     AND ch.level_name = 'TOTAL'
/* Join dimension views to cube view */
     AND t.dim_key = f.time
     AND p.dim_key = f.product
     AND cu.dim_key = f.customer
     AND ch.dim_key = f.channel
  ORDER BY t.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 negatively impact performance.

```
TIME            SALES
---------- ----------
JAN-98        8338545
FEB-98        7972132
Q1.98        24538588
MAR-98        8227911
APR-98        8470315
MAY-98        8160573
JUN-98        8362386
Q2.98        24993273
JUL-98        8296226
AUG-98        8377587
SEP-98        8406728
Q3.98        25080541
OCT-98        8316169
NOV-98        8984156
Q4.98        26258474
1998        100870877
                .
                .
                .
```

Example 4-2 Basic Cube View Query

Now consider the results when a filter restricts Time to yearly data. This example shows a basic query. It selects 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.

```
/* Select key descriptions and facts */
SELECT t.long_description time,
     ROUND(f.sales) sales
/* From dimension views and cube view */
  FROM time_calendar_view t,
```


Copyright © 2003, 2025, Oracle and/or its affiliates.

```
     product_primary_view p,
     customer_shipments_view cu,
     channel_primary_view ch,
     units_cube_view f
/* Create level filters */
  WHERE t.level_name = 'CALENDAR_YEAR'
     AND p.level_name = 'TOTAL'
     AND cu.level_name = 'TOTAL'
     AND ch.level_name = 'TOTAL'
/* Join dimension views to cube view */
     AND t.dim_key = f.time
     AND p.dim_key = f.product
     AND cu.dim_key = f.customer
     AND ch.dim_key = f.channel
  ORDER BY t.end_date;
```

The example 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      116931722
2002       92515295
2003      130276514
2004      144290686
2005      136986572
2006      140138317
```

Example 4-3 Selecting Data with Attribute Filters

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

```
/* Select key descriptions and facts */
SELECT t.long_description time,
     p.long_description product,
     cu.long_description customer,
     ch.long_description channel,
     ROUND(f.sales) sales
/* From dimension views and cube view */
FROM time_calendar_view t,
     product_primary_view p,
     customer_shipments_view cu,
     channel_primary_view ch,
     units_cube_view f
/* Create attribute filters */
WHERE t.long_description in ('2005', '2006')
    AND p.package = 'Laptop Value Pack'
    AND cu.long_description LIKE '%Boston%'
    AND ch.long_description = 'Internet'
/* Join dimension views to cube view */
    AND t.dim_key = f.time
    AND p.dim_key = f.product
    AND cu.dim_key = f.customer
    AND ch.dim_key = f.channel
ORDER BY time, customer;
```


Copyright © 2003, 2025, Oracle and/or its affiliates.

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

```
TIME   PRODUCT                        CUSTOMER             CHANNEL       SALES
------ ------------------------------ -------------------- -------- ----------
2005   Laptop carrying case           KOSH Entrpr Boston    Internet       5936
2005   56Kbps V.92 Type II Fax/Modem  KOSH Entrpr Boston    Internet      45285
2005   Internal 48X CD-ROM            KOSH Entrpr Boston    Internet       2828
2005   Standard Mouse                 KOSH Entrpr Boston    Internet        638
2005   Envoy Standard                 Warren Systems Boston Internet      19359
2005   Laptop carrying case           Warren Systems Boston Internet      13434
2005   Standard Mouse                 Warren Systems Boston Internet        130
2006   Standard Mouse                 KOSH Entrpr Boston    Internet        555
2006   Laptop carrying case           KOSH Entrpr Boston    Internet       6357
2006   56Kbps V.92 Type II Fax/Modem  KOSH Entrpr Boston    Internet      38042
2006   Internal 48X CD-ROM            KOSH Entrpr Boston    Internet       3343
2006   Envoy Standard                 Warren Systems Boston Internet      24198
2006   Laptop carrying case           Warren Systems Boston Internet      13153
2006   Standard Mouse                 Warren Systems Boston Internet         83

14 rows selected.
```

4.2.2 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 selects the aggregate data. It does not calculate the values.

The following is a basic query against a fact table:

```
/* Querying a fact table */
SELECT t.calendar_year_dsc time,
     SUM(f.sales) sales
  FROM time_dim t, units_fact f
  WHERE t.calendar_year_dsc IN ('2005', '2006')
     AND t.month_id = f.month_id
  GROUP BY t.calendar_year_dsc;
```

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

```
/* Querying a cube */
SELECT t.long_description time, f.sales sales
  FROM time_calendar_view t,
     product_primary_view p,
     customer_shipments_view cu,
     channel_primary_view ch,
     units_cube_view f
/* Apply filters to every dimension */
  WHERE t.long_description IN ('2005', '2006')
     AND p.level_name = 'TOTAL'
     AND cu.level_name = 'TOTAL'
     AND ch.level_name = 'TOTAL'
/* Join dimension views to cube view */
     AND t.dim_key = f.TIME
     AND p.dim_key = f.product
     AND cu.dim_key = f.customer
     AND ch.dim_key = f.channel
  ORDER BY time;
```

Both queries return these results:

```
TIME       SALES
----- ----------
```


Copyright © 2003, 2025, Oracle and/or its affiliates.

```
2005   136986572
2006   140138317
```

The query against the cube does not compute the aggregate values with a `SUM` operator and `GROUP BY` clause. Because the aggregates exist 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.

4.2.3 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.

4.3 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 hierarchy 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_NAME` 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.

Copyright © 2003, 2025, Oracle and/or its affiliates.

- Drilling Down to Children
- Drilling Up to Parents
- Drilling Down to Descendants
- Drilling Up to Ancestors

> **See Also**
>
> Developing Reports and Dashboards about using bind variables to support drilling

4.3.1 Drilling Down to Children

You can use the PARENT column of a hierarchy view to select only the children of a particular value. The following WHERE clause selects the children of calendar year 2005.

```
/* Select children of calendar year 2005 */
WHERE  t.parent = 'CY2005'
   AND p.dim_key = 'TOTAL'
   AND cu.dim_key = 'TOTAL'
   AND ch.dim_key = 'TOTAL'
```

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
-------- ----------
Q1.05      31381338
Q2.05      37642741
Q3.05      32617249
Q4.05      35345244
```

4.3.2 Drilling Up to Parents

The PARENT column of a hierarchy view identifies the parent of each dimension key. Columns of level keys identify the full heritage. The following WHERE clause selects the parent of a Time key based on its LONG_DESCRIPTION attribute.

```
/* Select the parent of a Time key*/
WHERE t.dim_key =
   (SELECT DISTINCT parent
      FROM time_calendar_view
      WHERE long_description='JAN-05')
   AND p.dim_key= 'TOTAL'
   AND cu.dim_key = 'TOTAL'
   AND ch.dim_key = 'TOTAL'
```

The query drills up from Month to Quarter. The parent of month JAN-05 is the quarter Q1-05 in the Calendar hierarchy.

```
TIME          SALES
-------- ----------
Q1.05      31381338
```


Copyright © 2003, 2025, Oracle and/or its affiliates.

4.3.3 Drilling Down to Descendants

The following WHERE clause selects the descendants of calendar year 2005 by selecting the rows with a LEVEL_NAME of MONTH and a CALENDAR_YEAR of CY2005.

```
/* Select Time level and ancestor */
WHERE t.level_name = 'MONTH'
    AND t.calendar_year = 'CY2005'
    AND p.dim_key = 'TOTAL'
    AND cu.dim_key = 'TOTAL'
    AND ch.dim_key = 'TOTAL'
```

The query drills down two levels, from year to quarter to month. The 12 months Jan-05 to Dec-05 are the descendants of year 2005 in the Calendar hierarchy.

```
TIME          SALES
-------- ----------
JAN-05     12093518
FEB-05     10103162
MAR-05      9184658
APR-05      9185964
MAY-05     11640216
JUN-05     16816561
JUL-05     11110903
AUG-05      9475807
SEP-05     12030538
OCT-05     11135032
NOV-05     11067754
DEC-05     13142459
```

4.3.4 Drilling Up to Ancestors

The hierarchy views provide the full ancestry of each dimension key, as shown in "Displaying the Contents of a Hierarchy View". The following WHERE clause uses the CALENDAR_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 t.dim_key =
   (SELECT calendar_year
      FROM time_calendar_view
      WHERE long_description = 'JAN-05')
   AND p.dim_key = 'TOTAL'
   AND cu.dim_key = 'TOTAL'
   AND ch.dim_key = 'TOTAL'
```

The query drills up two levels from month to quarter to year. The ancestor of month Jan-05 is the year 2005 in the Calendar hierarchy.

```
TIME          SALES
-------- ----------
2005      136986572
```

4.4 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.

Copyright © 2003, 2025, Oracle and/or its affiliates.

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 must ensure only 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 4-4 Calculating Average Sales Across Customers

This example shows a query that answers the question, What was the average sales of Sentinel Standard computers to Government customers for the third quarter of fiscal year 2005. `UNITS_CUBE` is summed over all dimensions, so that `FY2005.Q3` 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.

```
SELECT customer, ROUND(MIN(sales)) minimum, ROUND(MAX(sales)) maximum,
   ROUND(AVG(sales)) average
FROM
   (SELECT cu.long_description customer,
       t.month_long_description time
       f.sales sales
   FROM time_fiscal_view t,
     product_primary_view p,
     customer_market_view cu,
     channel_primary_view ch,
     units_cube_view f
   WHERE t.parent = 'FY2005.Q3'
     AND p.dim_key = 'SENT STD'
     AND cu.parent = 'GOV'
     AND ch.level_name = 'TOTAL'
     AND t.dim_key = f.time
     AND p.dim_key = f.product
     AND cu.dim_key = f.customer
     AND ch.dim_key = f.channel
   )
GROUP BY customer
ORDER BY customer;
```

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

```
CUSTOMER                               TIME          SALES
-------------------------------------- -------- ----------
Dept. of Labor                         JAN-05      1553.26
Dept. of Labor                         MAR-05       1555.6
Ministry of Intl Trade                 JAN-05      1553.26
Ministry of Intl Trade                 FEB-05      1554.56
Ministry of Intl Trade                 MAR-05       1555.6
Royal Air Force                        JAN-05      1553.26
Royal Air Force                        FEB-05      6218.23
UK Environmental Department            JAN-05      4659.78
UK Environmental Department            FEB-05      3109.12
```

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

```
CUSTOMER                         MINIMUM    MAXIMUM    AVERAGE
------------------------------ ---------- ---------- ----------
Dept. of Labor                       1553       1556       1554
Ministry of Intl Trade               1553       1556       1554
Royal Air Force                      1553       6218       3886
UK Environmental Department          3109       4660       3884
```


Copyright © 2003, 2025, Oracle and/or its affiliates.

4.5 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
- Aggregating Calculated Measures Over Attributes

4.5.1 Aggregating Measures Over Attributes

Example 4-5 shows a query that aggregates over an attribute named Package. It returns these results:

```
TIME   PACKAGE                 SALES
------ ------------------ ----------
2005   All                1809157.64
2005   Multimedia         18083256.3
2005   Executive            19836977
2005   Laptop Value Pack  9547494.81
```

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 4-5 Aggregating Over an Attribute

```
SELECT t.long_description time,
     p.package package,
     SUM(f.sales) sales
  FROM time_calendar_view t,
     product_primary_view p,
     customer_shipments_view cu,
     channel_primary_view ch,
     units_cube_view f
/* Select Product by level and attribute */
  WHERE p.level_name = 'ITEM'
     AND p.package IS NOT NULL
```


Copyright © 2003, 2025, Oracle and/or its affiliates.

```
     AND t.long_description = '2005'
     AND cu.level_name = 'TOTAL'
     AND ch.level_name = 'TOTAL'
/* Join dimensions and cube */
     AND t.dim_key = f.time
     AND p.dim_key = f.product
     AND cu.dim_key = f.customer
     AND ch.dim_key = f.channel
  GROUP BY t.long_description, p.package;
```

4.5.2 Aggregating Calculated Measures Over Attributes

Before using the technique described in "Aggregating Measures Over Attributes", ensure 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 4-6 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_PP:

```
TIME   PACKAGE                SALES PRIOR_PERIOD PCT_CHG_PP
------ ------------------ ---------- ------------ ----------
2005   All                1809157.64   1853928.06 -.02414895
2006   All                1720399.03   1809157.64 -.04906074
2005   Executive            19836977   20603879.8 -.03722128
2006   Executive          19580638.4     19836977 -.01292226
2005   Laptop Value Pack  9547494.81   10047298.6 -.04974509
2006   Laptop Value Pack  9091450.58   9547494.81 -.04776585
2005   Multimedia         18083256.3   19607675.5 -.07774604
2006   Multimedia         18328678.7   18083256.3 .013571806

8 rows selected.
```

Example 4-6 Querying Over Attributes Using Calculated Measures

```
/* Calculate Percent Change */
SELECT TIME, package, sales, prior_period,
     ((sales - prior_period) / prior_period) pct_chg_pp
FROM
/* Fetch data from the cube and aggregate over Package */
     (SELECT t.long_description time,
          p.package package,
          SUM(f.sales) sales,
          SUM(f.sales_pp) prior_period
       FROM time_calendar_view t,
          product_primary_view p,
          customer_shipments_view cu,
          channel_primary_view ch,
```


Copyright © 2003, 2025, Oracle and/or its affiliates.

```
          units_cube_view f
/* Create filters */
       WHERE p.level_name = 'ITEM'
          AND p.package IS NOT NULL
          AND t.long_description IN ('2005', '2006')
          AND cu.level_name = 'TOTAL'
          AND ch.level_name = 'TOTAL'
/* Join dimension views to cube view */
          AND t.dim_key = f.time
          AND p.dim_key = f.product
          AND cu.dim_key = f.customer
          AND ch.dim_key = f.channel
       GROUP BY t.long_description, p.package
       ORDER BY p.package);
```

4.6 Joining Cubes to Tables and Views

You can join cubes to other cubes and to relational objects such as:

- Tables
- Views including external tables and PL/SQL table functions
- Other row source types, like other joins

Typically, you do not need a fully aggregated cube when joining it to a table or view, and a `CUBE JOIN` operation limits the number of fetched values to improve performance automatically. The cube must be on the right side of the equation. If the query does not support `CUBE JOIN`, then the more expensive `HASH JOIN`, `MERGE JOIN`, or `NESTED LOOPS` are commonly used.

You can use hints in the query to influence the use of `CUBE JOIN`:

- `USE_CUBE` forces a `CUBE JOIN` if possible.
- `NO_USE_CUBE` prevents a `CUBE JOIN`.

See "Viewing Execution Plans" for more information about `CUBE JOIN`.

Example 4-7 joins a table that contains French descriptions of the Customer dimension to a cube that supports only English. The query returns these results:

```
CUSTOMER                                       SALES
---------------------------------------- ------------
La Marine des USA Washington                    600.34
Monolith Motor Co. Chattanooga                17946.51
Piedmont, Inc. San Jose                       24874.41
Ministere du Commerce Int. Nagano             27595.97
Depart. des commun. - Stuttgart               30706.10
Min. Env. Brit. Londres                       38125.77
Departement de travail Nouvelle-Orleans       42507.50
Ministere des Finances Sorbonne               43607.58
Monolith Motor Co. Knoxville                  50874.53
Serv. des USA de recherche Wyo                54497.19
Depart. des commun. - Bonn                    58944.97
     .
     .
     .
```

Example 4-7 Joining a Cube and a Table

```
SELECT cu.ship_to_dsc_french customer,
     f.sales sales
FROM time_calendar_view t,
```


Copyright © 2003, 2025, Oracle and/or its affiliates.

```
    product_primary_view p,
    customer_dim cu,
    channel_primary_view ch,
    units_cube_view f
WHERE t.dim_key = 'CY2006'
   AND p.level_name = 'TOTAL'
   AND ch.level_name = 'TOTAL'
   AND t.dim_key = f.TIME
   AND p.dim_key = f.product
   AND cu.ship_to_id = f.customer
   AND ch.dim_key = f.channel
ORDER BY f.sales;
```

4.7 Viewing Execution Plans

You can generate and view execution plans for queries against cubes and dimensions the same as for those against relational tables.

The SQL `EXPLAIN PLAN` command creates a table with the content of the explain plan. The default table name is `PLAN_TABLE`.

- Generating Execution Plans
- Types of Execution Plans

> **See Also**
>
> *Oracle Database SQL Tuning Guide* for a complete discussion of execution plans

4.7.1 Generating Execution Plans

The following command creates an execution plan for a basic query on a cube:

```
EXPLAIN PLAN FOR
  SELECT t.long_description time,
     p.long_description product,
     cu.long_description customer,
     ch.long_description channel,
     f.sales sales
  FROM time_calendar_view t,
     product_primary_view p,
     customer_shipments_view cu,
     channel_primary_view ch,
     units_cube_view f
  WHERE t.level_name = 'CALENDAR_YEAR'
     AND p.level_name = 'TOTAL'
     AND cu.level_name = 'TOTAL'
     AND ch.level_name = 'TOTAL'
     AND t.dim_key = f.TIME
     AND p.dim_key = f.product
     AND cu.dim_key = f.customer
     AND ch.dim_key = f.channel
  ORDER BY t.end_date;
```


Copyright © 2003, 2025, Oracle and/or its affiliates.

Example 4-8 Execution Plan for a Cube Query

The DISPLAY table function of the DBMS_XPLAN PL/SQL package formats and displays information from an execution plan, as shown in this example.

```
SQL> SELECT plan_table_output FROM TABLE(dbms_xplan.display());

PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------------------
Plan hash value: 1667678335

-----------------------------------------------------------------------------------------------
| Id  | Operation                        | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                 |            |     1 |   100 |   104   (3)| 00:00:02 |
|   1 |  SORT ORDER BY                   |            |     1 |   100 |   104   (3)| 00:00:02 |
|   2 |   JOINED CUBE SCAN PARTIAL OUTER|            |       |       |            |          |
|   3 |    CUBE ACCESS                   | UNITS_CUBE |       |       |            |          |
|   4 |    CUBE ACCESS                   | CHANNEL    |       |       |            |          |
|   5 |    CUBE ACCESS                   | CUSTOMER   |       |       |            |          |
|   6 |    CUBE ACCESS                   | PRODUCT    |       |       |            |          |
|*  7 |    CUBE ACCESS                   | TIME       |     1 |   100 |   103   (2)| 00:00:02 |
-----------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   7 - filter(SYS_OP_ATG(VALUE(KOKBF$),12,13,2)='CALENDAR_YEAR' AND
              SYS_OP_ATG(VALUE(KOKBF$),43,44,2)='TOTAL' AND
              SYS_OP_ATG(VALUE(KOKBF$),33,34,2)='TOTAL' AND
              SYS_OP_ATG(VALUE(KOKBF$),23,24,2)='TOTAL')

22 rows selected.
```

Example 4-9 Execution Plan for a Cube Join

This example shows an execution plan for a query that joins a cube and a table. See "Joining Cubes to Tables and Views" for the query.

```
PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------------------------
Plan hash value: 3634608218
-------------------------------------------------------------------------------------------------
| Id  | Operation                        |Name        |Rows | Bytes |TempSpc|Cost (%CPU)| Time    |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                 |            |1464 |   128K|       | 1524  (94)| 00:00:19|
|   1 |  SORT ORDER BY                   |            |1464 |   128K|   152K| 1524  (94)| 00:00:19|
|*  2 |   CUBE JOIN                      |            |1464 |   128K|       | 1422 (100)| 00:00:18|
|   3 |    TABLE ACCESS FULL             |CUSTOMER_DIM|  61 |  2379 |       |    4   (0)| 00:00:01|
|   4 |    JOINED CUBE SCAN PARTIAL OUTER|            |     |       |       |           |         |
|   5 |     CUBE ACCESS                  |UNITS_CUBE  |     |       |       |           |         |
|   6 |     CUBE ACCESS                  |CHANNEL     |     |       |       |           |         |
|   7 |     CUBE ACCESS                  |PRODUCT     |     |       |       |           |         |
|*  8 |     CUBE ACCESS                  |TIME        |2520 |   125K|       | 1417 (100)| 00:00:18|
-------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
   2 - access("CU"."SHIP_TO_ID"=SYS_OP_ATG(VALUE(KOKBF$),76,77,2))
   8 - filter(SYS_OP_ATG(VALUE(KOKBF$),32,33,2)='CY2006' AND
              SYS_OP_ATG(VALUE(KOKBF$),85,86,2)='TOTAL' AND
              SYS_OP_ATG(VALUE(KOKBF$),65,66,2)='TOTAL')
```


Copyright © 2003, 2025, Oracle and/or its affiliates.

```
22 rows selected.
```

4.7.2 Types of Execution Plans

Table 4-1 describes the types of execution plans for cubes.

Table 4-1 Descriptions of Execution Plans for Cubes and Dimensions

Operation	Option	Description
CUBE JOIN	--	Joins a table or view on the left and a cube on the right.
CUBE JOIN	ANTI	Uses an antijoin for a table or view on the left and a cube on the right.

CUBE JOIN	ANTI SNA	Uses an antijoin (Single-sided Null Aware) for a table or view on the left and a cube on the right. The join column on the right (cube side) is NOT NULL. For example:
		`SELECT cols FROM table`
		`WHERE table.c1 NOT IN`
		`(SELECT col FROM cube`
		`WHERE cube.col IS NOT NULL)`

CUBE JOIN	OUTER	Uses an outer join for a table or view on the left and a cube on the right.
CUBE JOIN	RIGHT SEMI	Uses a right semijoin for a table or view on the left and a cube on the right.
CUBE SCAN	--	Uses inner joins for all cube access.
CUBE SCAN	PARTIAL OUTER	Uses an outer join for least one dimension, and inner joins for the other dimensions.
CUBE SCAN	OUTER	Uses outer joins for all cube access.

> **See Also**
>
> *Oracle Database SQL Language Reference* for descriptions of these join types.

4.8 Querying the Data Dictionary

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 data dictionary.

Among the static views of the database data dictionary are those that provide information about dimensional objects. All OLAP metadata is stored in the data dictionary. A few of the data dictionary views were introduced previously in this chapter.

Table 4-2 provides brief descriptions of the ALL views. There are corresponding DBA and USER views.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Table 4-2 Static Data Dictionary Views for OLAP

View	Description
ALL_CUBE_ATTR_VISIBILITY	Describes the visibility of the attributes for cube dimensions.
ALL_CUBE_ATTRIBUTES	Describes the attributes for cube dimensions.
ALL_CUBE_BUILD_PROCESSES	Describes the cube build processes and maintenance scripts.
ALL_CUBE_CALCULATED_MEMBERS	Describes the calculated members (keys) for cube dimensions.
ALL_CUBE_DIM_LEVELS	Describes the cube dimension levels.
ALL_CUBE_DIM_MODELS	Describes the models for cube dimensions.
ALL_CUBE_DIM_VIEW_COLUMNS	Describes the columns of the system-generated relational views of cube dimensions.
ALL_CUBE_DIM_VIEWS	Describes the system-generated relational views of OLAP dimensions.
ALL_CUBE_DIMENSIONALITY	Describes the dimension order of the OLAP cubes.
ALL_CUBE_DIMENSIONS	Describes the cube dimensions.
ALL_CUBE_HIER_LEVELS	Describes the hierarchy levels for cube dimensions.
ALL_CUBE_HIER_VIEW_COLUMNS	Describes the columns of relational hierarchy views of cube dimensions.
ALL_CUBE_HIER_VIEWS	Describes the hierarchies for cube dimensions.
ALL_CUBE_HIERARCHIES	Describes the OLAP dimension hierarchies.
ALL_CUBE_MEASURES	Describes the measures in the OLAP cubes.
ALL_CUBE_VIEW_COLUMNS	Describes the columns of the relational views of OLAP cubes.
ALL_CUBE_VIEWS	Describes the system-generated relational views of OLAP cubes.
ALL_CUBES	Describes the OLAP cubes.
ALL_MEASURE_FOLDER_CONTENTS	Describes the contents of OLAP measure folders.
ALL_MEASURE_FOLDERS	Describes the OLAP measure folders.

See Also

Oracle Database Reference for full descriptions of data dictionary views.

Copyright © 2003, 2025, Oracle and/or its affiliates.

5

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 create calculated measures using templates and free-form calculations.

This chapter contains the following topics:

- What Is a Calculated Measure?
- Functions for Defining Calculations
- Creating Calculated Measures
- Using Calculation Templates
- Creating User-Defined Expressions
- Creating Calculated Measures Using the OLAP DML

- What Is a Calculated Measure?
- Functions for Defining Calculations
- Creating Calculated Measures
- Using Calculation Templates
- Creating User-Defined Expressions
- Creating Calculated Measures Using the OLAP DML

5.1 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 using the templates in Analytic Workspace Manager.

As soon as you create a calculated measure, it appears as a column in a cube view. 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 by SQL applications.

5.2 Functions for Defining Calculations

The library of functions for defining calculated measures contains these basic categories:

- Arithmetic Operators: Perform calculations on the values of two measures.
- Analytic Functions: Perform calculations on an ordered series or a range of values in a single measure or column.

Copyright © 2003, 2025, Oracle and/or its affiliates.

- Single-Row Functions: Perform calculations on a value in a single row.
- Arithmetic Operators
- Analytic Functions
- Single-Row Functions

5.2.1 Arithmetic Operators

You can perform the following arithmetic operations using two measures. The calculations in the cube are performed on a cell-by-cell basis at all levels of the dimension hierarchies.

- **Addition**: Adds the values of two measures.
- **Subtraction**: Subtracts the values of one measure from the values of another measure.
- **Multiplication**: Multiplies the values of two measures.
- **Division** or **Ratio**: Divides the values of one measure by the values of another measure.
- **Percent Difference**: Calculates the percent difference between the values of two measures.

The arithmetic operations are available in Analytic Workspace Manager as templates. as described in "Using Calculation Templates".

5.2.2 Analytic Functions

The analytic functions provide the most powerful computations and fuel the most useful queries for business intelligence and similar applications. They include a variety of rank, share, time series, and other single-column functions. The analytic functions enable analysts and decision makers to make comparisons and identify trends.

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 cumulative calculations. Because the knowledge is stored with the dimension, you do not need to specify these relationships when creating a calculated measure.

The analytic functions are available in Analytic Workspace Manager as templates. They are described in "Using Calculation Templates".

5.2.3 Single-Row Functions

Oracle OLAP supports most of the SQL single-row functions including:

- Numeric functions such as `ABS`, `CEIL`, `FLOOR`, `POWER`, `ROUND`, and `TRUNC`.
- Character functions such as `CONCAT`, `LPAD`, `RPAD`, `LTRIM`, `RTRIM`, `REPLACE`, and `SUBSTR`.
- Datetime functions such as `CURRENT_DAY`, `MONTHS_BETWEEN`, `NEXT_DAY`, and `SYSTIMESTAMP`.
- Comparison functions `GREATEST` and `LEAST`.
- Conversion functions such as `TO_CHAR`, `TO_DATE`, `TO_NUMBER`, and `TO_TIMESTAMP`.

You can use these functions to manipulate the data values in a measure, typically as part of a more complex calculation. These functions are not available as templates, but you can use them in free-form calculations, as described in "Creating User-Defined Expressions".

Copyright © 2003, 2025, Oracle and/or its affiliates.

5.3 Creating Calculated Measures

Analytic Workspace Manager provides easy-to-use templates for creating calculated measures. You can create them in the same cube with the source measures, or you can create them in a separate cube.

Calculated measures are available for querying as additional columns in a cube view (such as `UNITS_CUBE_VIEW`). They are not available through cube materialized views (such as `CB$UNITS_CUBE`).

The calculated measure generator quickly generates the standard calculated measures for one or more measures of a cube, including rank, share, prior and future periods, period-to-date, parallel period, moving aggregates, and cumulative aggregates. The generator uses naming rules for formulating the names and descriptions. You can customize these rules on the Naming Rules tab.

You can also create individual calculated measures, including user-defined expressions in the OLAP expression syntax or the OLAP DML.

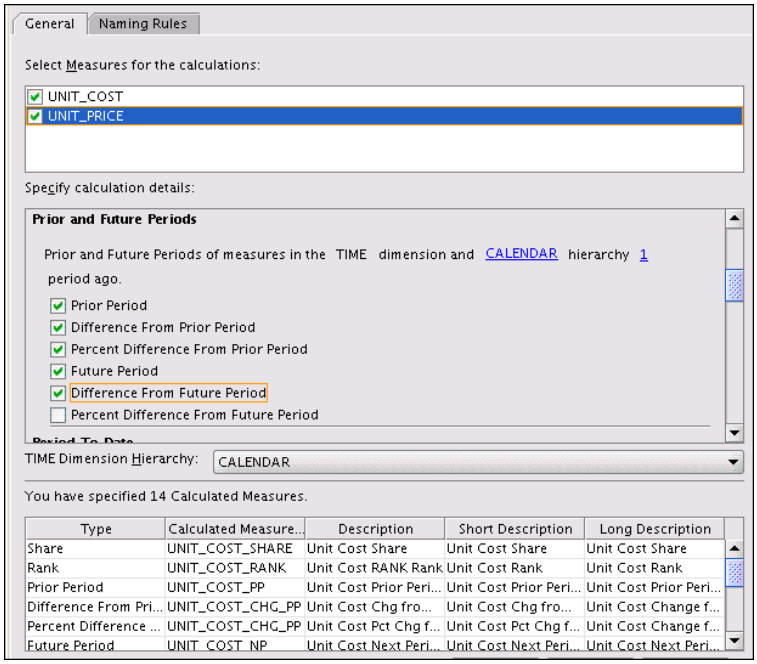
To create multiple calculated measures:

1. In the navigation tree, right-click a cube and select **Generate Calculated Measures**.
2. On the Calculations tab, select the measures on which to base the calculated measures.
3. Scroll down the Calculation Details and select each type of calculated measure you want to create for this selection of measures. Modify the calculations as desired by altering the templates.
4. Select the Time dimension to use for time series calculations.
5. Review the list of calculated measures. You can change the generated names by using the Naming Rules tab.
6. Click **Generate Calculations** to create the calculated measures.
7. Repeat this procedure if you want to generate variations of the same basic types of calculations, such as another Share calculation for the same measure but on a different dimension. Change the naming rules to generate new, unique names.

Figure 5-1 shows the Generate Calculated Measures dialog box.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Figure 5-1 Generating Multiple Calculated Measures



To create a single calculated measure:

1. In the navigation tree, expand a cube folder.
2. Right-click Calculated Measures, then select **Create Calculated Measure** from the context menu.

   In the Create Calculated Measure dialog box, Enter a descriptive name.
3. Select a calculation type.

   Your choice of an arithmetic or analytic function dynamically changes the Calculation template.
4. Modify the calculation template.
5. Click **Create**.

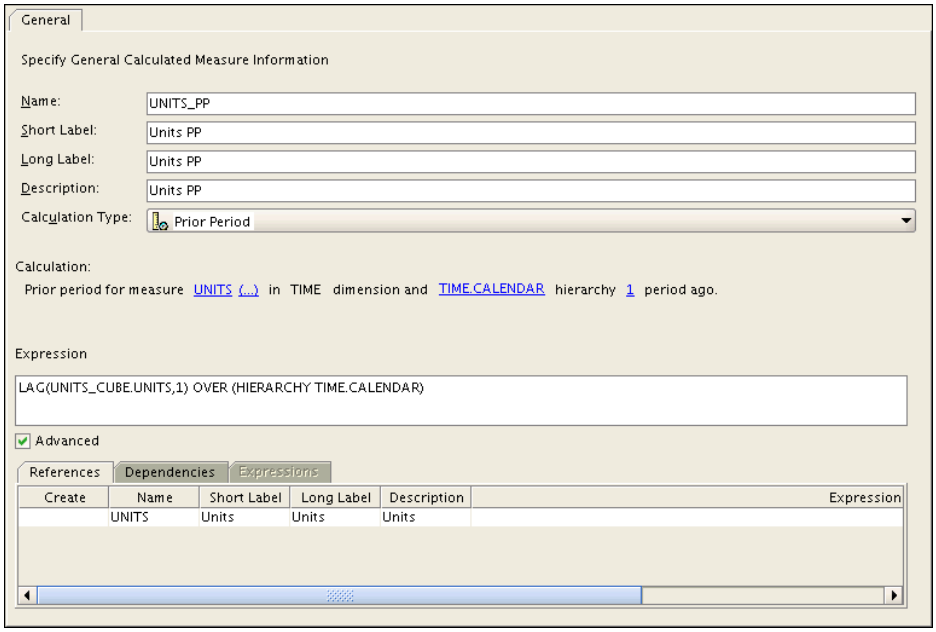
   The calculated measure appears in the navigation tree in the Calculated Measures folder.
6. Select the **Advanced** option to display the References, Dependencies, and Expressions tabs. The tabs have the following information:

   - The References tab has a table that lists the measures that Analytic Workspace Manager references as it performs the calculations specified by this calculated measure. If the Enable SQL Expressions option is selected for the cube, then the table has a check mark in the Create column for any additional calculated measure that Analytic Workspace automatically creates.
   - The Dependencies tab has a table that lists the other calculated measures that depend on this calculated measure. Analytic Workspace Manager uses this calculated measure as it performs the calculations for the measures in this table.
   - The Expressions tab has a table that lists the expressions used by the calculated measure. This tab appears only if the Enable SQL Expressions option is selected for the cube.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Figure 5-2 displays the Create Calculated Measure dialog box.

Figure 5-2 Creating a Calculated Measure



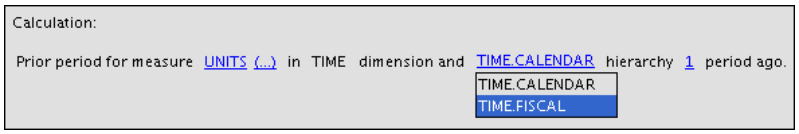
- Modifying a Template
- Choosing a Range of Time Periods

5.3.1 Modifying a Template

The calculation that you selected is presented as template, which is a description of the calculation with variable parts that enable you to customize it.

Figure 5-3 shows the template for calculating the prior period. You can view the choice lists by clicking the links.

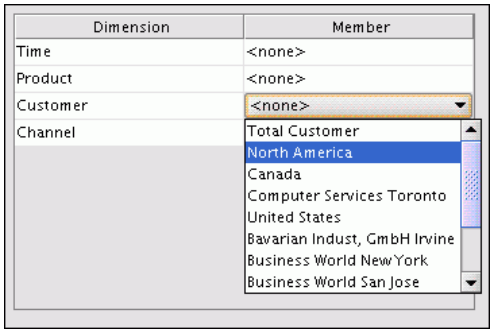
Figure 5-3 Changing the Variable Parts of a Calculation



You can include all values of a measure in a calculation, or, for some types of calculations, you can filter the measure to include only a selection of values. To limit one or more dimensions to a single dimension member, click the ellipses (. . .) next to the measure. The Qualify Measure dialog box appears, as shown in Figure 5-4.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Figure 5-4 Limiting a Dimension to a Single Member



5.3.2 Choosing a Range of Time Periods

Many calculations are performed over time periods at the same level in the hierarchy. In some types of calculations, you can control the range of time periods that are used in the same calculation. For example, you might want to calculate a running total of months for each fiscal year, not a running total that begins with the first month stored in the cube.

You can use the following methods for identifying the range of time periods to calculate together:

- **Level**: Calculates all time periods at the same level, so that all months in the cube are included in one calculation, all quarters are included in another calculation, and so forth.
- **Parent**: Calculates all time periods with the same parent, so that all months in Q1-07 are included in one calculation, all months in Q2-07 are included in another calculation, and so forth.
- **Ancestor at level**: Calculates all time periods with the same ancestor at a specified level. For example, if the specified level is Year in a Year-Quarter-Month hierarchy, then Q1-06 to Q4-06 are included in one calculation, Q1-07 to Q4-07 are included in another calculation, Jan-06 to Dec-06 are included in a third calculation, and so forth. Any levels higher in the hierarchy are not calculated.
- **Gregorian periods**: The Gregorian periods -- Year, Quarter, Month, and Week -- impose the Gregorian calendar onto the selected hierarchy. This can be useful for analyzing data that uses nonstandard calendar hierarchies. For example, if you use Gregorian Year on a fiscal hierarchy that begins July 1 and ends June 30, then the last half of one fiscal year and the first half of the next fiscal year are calculated together. Time periods higher in the hierarchy than the specified Gregorian period are not calculated.

5.4 Using Calculation Templates

Analytic Workspace Manager provides templates for all of the calculations typically in demand for business intelligence applications. The following topics describe the types of calculations available as calculation templates in Analytic Workspace Manager.

- Arithmetic Calculations
- Index
- Prior and Future Periods
- Period to Date
- Share

Copyright © 2003, 2025, Oracle and/or its affiliates.

- Rank
- Parallel Period
- Moving Calculations
- Cumulative Calculations
- Nested Calculations
- Arithmetic Calculations
- Index
- Prior and Future Periods
- Period to Date
- Share
- Rank
- Parallel Period
- Moving Calculations
- Cumulative Calculations
- Nested Calculations

5.4.1 Arithmetic Calculations

Basic mathematical operations enable you to perform cell-by-cell calculations on two measures, as described in "Arithmetic Operators".

Arithmetic Example

This template defines a calculated measure for the Global Price Cube using Percent Difference:

```
Percent difference between measure UNIT_PRICE and measure UNIT_COST.
```

A query against this calculated measure returns results like these. The PCT_CHG column shows the percent change between PRICE and COST, which is calculated as PRICE-COST/COST.

```
PRODUCT                   PRICE       COST   PCT_DIFF
-------------------- ---------- ---------- ----------
Envoy Ambassador           2892       2664        .09
Envoy Executive            2803       2644        .06
Envoy Standard             1662       1737       -.04
Sentinel Financial         1755       1658        .06
Sentinel Multimedia        1770       1813       -.02
Sentinel Standard          1552       1410         .1
```

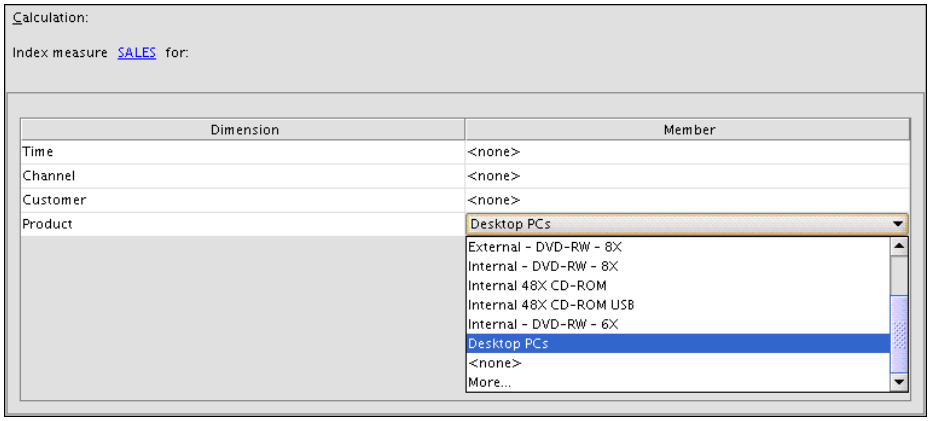
5.4.2 Index

An index is a mathematical operation calculated on a single measure. An index calculates the percentage difference between the values of a measure and a selected value that serves as a base number.

An index does not use a calculation template. Instead, it provides a list of dimension members for each dimension of the cube, from which you can choose one to use as an index, as shown in Figure 5-5.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Figure 5-5 Calculating a Product Index



Index Example

This example creates an index on the Product dimension using `Desktop PCs` as the index.

```
PRODUCT                   SALES PROD_INDEX
-------------------- ---------- ----------
Desktop PCs            76682955        100
Portable PCs           18072328         24
CD/DVD                 17302122         23
Modems/Fax              5565552          7
Memory                  5347292          7
Monitors                3926632          5
```

5.4.3 Prior and Future Periods

Oracle OLAP provides several calculations for prior or future time periods:

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

When creating a calculation for prior or future time periods, you choose the measure, the time dimension, the hierarchy, and the number of periods from the current period.

Prior Period Example

This template defines a calculated measure using Prior Period:

```
Prior period for measure SALES in TIME dimension and TIME.CALENDAR hierarchy 1 period
ago.
```


Copyright © 2003, 2025, Oracle and/or its affiliates.

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

```
TIME     TIME_LEVEL               SALES PRIOR_PERIOD
-------- ------------------- ---------- ------------
2005     CALENDAR_YEAR         136986572    144290686
2006     CALENDAR_YEAR         140138317    136986572
Q1.05    CALENDAR_QUARTER       31381338     41988687
Q2.05    CALENDAR_QUARTER       37642741     31381338
Q3.05    CALENDAR_QUARTER       32617249     37642741
Q4.05    CALENDAR_QUARTER       35345244     32617249
Q1.06    CALENDAR_QUARTER       36154815     35345244
Q2.06    CALENDAR_QUARTER       36815657     36154815
Q3.06    CALENDAR_QUARTER       32318935     36815657
Q4.06    CALENDAR_QUARTER       34848911     32318935
```

5.4.4 Period to Date

Period-to-date functions perform a calculation over time periods with the same parent up to the current period. These functions calculate period-to-date:

- **Period to Date**: Calculates the values up to the current time period.
- **Period to Date Period Ago**: Calculates the data values up to a prior time period.
- **Difference From Period to Date Period Ago**: Calculates the difference in data values up to the current time period compared to the same calculation up to a prior period.
- **Percent Difference From Period To Date Period Ago**: Calculates the percent difference in data values up to the current time period compared to the same calculation up to a prior period.

When creating a period-to-date calculation, you can choose from these aggregation methods:

- Sum
- Average
- Maximum
- Minimum

You also choose the measure, the time dimension, and the hierarchy.

Period to Date Example

This template defines a calculated measure using Period to Date.

```
Gregorian Year to date for SALES in the TIME dimension and TIME.CALENDAR hierarchy.
Aggregate using MINIMUM from the beginning of the period.
```

These are the results of a query against the calculated measure. The MIN_TO_DATE column displays the current minimum SALES value within the current level and year.

```
TIME     TIME_LEVEL               SALES MIN_TO_DATE
-------- ------------------- ---------- -----------
Q1.06    CALENDAR_QUARTER       36154815    36154815
Q2.06    CALENDAR_QUARTER       36815657    36154815
Q3.06    CALENDAR_QUARTER       32318935    32318935
Q4.06    CALENDAR_QUARTER       34848911    32318935
JAN-06   MONTH                  13119235    13119235
FEB-06   MONTH                  11441738    11441738
MAR-06   MONTH                  11593842    11441738
APR-06   MONTH                  11356940    11356940
```


Copyright © 2003, 2025, Oracle and/or its affiliates.

```
MAY-06   MONTH                13820218    11356940
JUN-06   MONTH                11638499    11356940
JUL-06   MONTH                 9417316     9417316
AUG-06   MONTH                11596052     9417316
SEP-06   MONTH                11305567     9417316
OCT-06   MONTH                11780401     9417316
NOV-06   MONTH                10653184     9417316
DEC-06   MONTH                12415325     9417316
```

5.4.5 Share

Share calculates the ratio of a measure's value for the current dimension member to the value for a related member of the same dimension. You can choose whether the related member is:

- **Top of hierarchy**: Calculates the ratio of each member to the total.
- **Member's parent**: Calculates the ratio of each member to its parent.
- **Member's ancestor at level**: Calculates the ratio of each member to its ancestor, that is, a member at a specified level higher in the hierarchy.

When creating a share calculation, you can choose the measure, dimension, and hierarchy. You also have the option of multiplying the results by 100 to get percentages instead of fractions.

Share Example

This template defines a calculated measure using `SHARE`:

```
Share of measure SALES in PRODUCT.PRIMARY hierarchy of the PRODUCT dimension as a ratio
of top of hierarchy.
```

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

```
PRODUCT              PROD_LEVEL           SALES TOTAL_SHARE
-------------------- -------------- ---------- -----------
Total Product        TOTAL           144290686         100
Hardware             CLASS           130145388          90
Desktop PCs          FAMILY           78770152          55
Portable PCs         FAMILY           19066575          13
CD/DVD               FAMILY           16559860          11
Software/Other       CLASS            14145298          10
Accessories          FAMILY            6475353           4
Operating Systems    FAMILY            5738775           4
Memory               FAMILY            5430466           4
Modems/Fax           FAMILY            5844185           4
Monitors             FAMILY            4474150           3
Documentation        FAMILY            1931170           1
```

5.4.6 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 can choose a method for handling identical values:

- **Rank**: Assigns the same rank to identical values, so there may be fewer ranks than there are members. For example, it may return `1`, `2`, `3`, `3`, `4` for a series of five dimension members.

Copyright © 2003, 2025, Oracle and/or its affiliates.

- **Dense Rank**: Assigns the same minimum rank to identical values. For example, it may return 1, 2, 3, 3, 5 for a series of five dimension members.
- **Average Rank**: Assigns the same average rank to identical values. For example, it may return 1, 2, 3.5, 3.5, 5 for a series of five dimension members.

You can also choose the group in which the dimension members are ranked:

- **Member's level**: Ranks members at the same level.
- **Member's parent**: Ranks members with the same parent.
- **Member's ancestor at level**: Ranks members with the same ancestor at a specified level higher in the hierarchy.

Rank Example

This template defines a calculated measure using Rank:

```
Rank members of the PRODUCT dimension and PRODUCT.PRIMARY hierarchy based on measure
SALES. Calculate rank using RANK method with member's parent in order lowest to highest.
Rank NA (null) values nulls last.
```

These are the results of a query against the calculated measure in which the products are ordered by RANK:

```
PRODUCT                    SALES       RANK
-------------------- ---------- ----------
Monitors                 4474150          1
Memory                   5430466          2
Modems/Fax               5844185          3
CD/DVD                  16559860          4
Portable PCs            19066575          5
Desktop PCs             78770152          6
```

5.4.7 Parallel Period

Parallel periods are at the same level as the current time period, but have different parents in an earlier period. For example, you may want to compare current sales with sales for the prior year at the quarter and month levels.

Oracle OLAP provides several functions for parallel periods:

- **Parallel Period**: Calculates the value of the parallel period.
- **Difference From Parallel Period**: Calculates the difference in values between the current period and the parallel period.
- **Percent Difference From Parallel Period**: Calculates the percent difference in values between the current period and the parallel period.

To identify the parallel period, you specify a level and the number of periods before the current period. You can also decide what happens when two periods do not exactly match, such as comparing daily sales for February (28 days) with January (31 days).

You also choose the measure, the time dimension, and the hierarchy.

Parallel Period Example

This template defines a calculated measure using Parallel Period.

```
Parallel period for SALES in the TIME dimension and TIME.CALENDAR hierarchy 1
TIME.CALENDAR.QUARTER ago based on position from beginning to ending of period.
```


Copyright © 2003, 2025, Oracle and/or its affiliates.

These are the results of a query against the calculated measure, which lists the months for two calendar quarters. The parallel month has the same position within the previous quarter. The prior period for JUL-06 is APR-06, for AUG-06 is MAY-06, and for SEP-06 is JUN-06.

```
TIME     PARENT          SALES   LAST_QTR
-------- ---------- ---------- ----------
APR-06   CY2006.Q2    11356940   13119235
MAY-06   CY2006.Q2    13820218   11441738
JUN-06   CY2006.Q2    11638499   11593842
JUL-06   CY2006.Q3     9417316   11356940
AUG-06   CY2006.Q3    11596052   13820218
SEP-06   CY2006.Q3    11305567   11638499
```

5.4.8 Moving Calculations

Moving calculations are performed over the time periods surrounding the current period. Oracle OLAP provides several aggregation methods 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 can choose the measure, the time dimension, and the hierarchy. You can also select the range, as described in "Choosing a Range of Time Periods", and the number of time periods before and after the current period to include in the calculation.

Moving Calculation Example

This template defines a calculated measure using Moving Minimum.

```
Moving minimum of SALES in the TIME dimension and TIME.CALENDAR hierarchy. Include 1
preceding and 1 following members within level.
```

These are the results of a query against the calculated measure, which displays values for the descendants of calendar year 2004. Each value of Minimum Sales is the smallest among the current value and the values immediately before and after it. The calculation is performed over all members of a level in the cube.

```
TIME     TIME_LEVEL                SALES  MIN_SALES
-------- ------------------- ---------- ----------
Q1.04    CALENDAR_QUARTER       32977874   32977874
Q2.04    CALENDAR_QUARTER       35797921   32977874
Q3.04    CALENDAR_QUARTER       33526203   33526203
Q4.04    CALENDAR_QUARTER       41988687   31381338
JAN-04   MONTH                  11477898   10982016
FEB-04   MONTH                  10982016   10517960
MAR-04   MONTH                  10517960   10517960
APR-04   MONTH                  11032057   10517960
MAY-04   MONTH                  11432616   11032057
JUN-04   MONTH                  13333248   11432616
JUL-04   MONTH                  12070352   11108893
AUG-04   MONTH                  11108893   10346958
SEP-04   MONTH                  10346958   10346958
OCT-04   MONTH                  14358605   10346958
```


Copyright © 2003, 2025, Oracle and/or its affiliates.

```
NOV-04   MONTH                12757560   12757560
DEC-04   MONTH                14872522   12093518
```

5.4.9 Cumulative Calculations

Cumulative calculations start with the first time period and calculate up to the current member, or start with the last time period and calculate back to the current member. Oracle OLAP provides several aggregation methods for cumulative calculations:

- **Cumulative Average**: Calculates a running average across time periods.
- **Cumulative Maximum**: Calculates the maximum value across time periods.
- **Cumulative Minimum**: Calculates the minimum value across time periods.
- **Cumulative Total**: Calculates a running total across time periods.

You can choose the measure, the time dimension, and the hierarchy. You can also select the range, as described in "Choosing a Range of Time Periods", and whether you want to start the calculation with the first period and calculate forward, or start with the last period and calculate back.

Cumulative Calculation Example

This template defines a calculated measure using Cumulative Minimum.

```
Cumulative minimum of SALES in the TIME dimension and TIME.CALENDAR hierarchy within
ancestor at level TIME.CALENDAR_YEAR. Total from beginning to current member.
```

These are the results of a query against the calculated measure, which displays values for the descendants of calendar year 2004. The minimum value for quarters begins with Q1-04 and ends with Q4-04, and for months begins with Jan-04 and ends with Dec-04.

```
TIME     TIME_LEVEL                SALES  MIN_SALES
-------- ------------------- ---------- ----------
Q1.04    CALENDAR_QUARTER       32977874   32977874
Q2.04    CALENDAR_QUARTER       35797921   32977874
Q3.04    CALENDAR_QUARTER       33526203   32977874
Q4.04    CALENDAR_QUARTER       41988687   32977874
JAN-04   MONTH                  11477898   11477898
FEB-04   MONTH                  10982016   10982016
MAR-04   MONTH                  10517960   10517960
APR-04   MONTH                  11032057   10517960
MAY-04   MONTH                  11432616   10517960
JUN-04   MONTH                  13333248   10517960
JUL-04   MONTH                  12070352   10517960
AUG-04   MONTH                  11108893   10517960
SEP-04   MONTH                  10346958   10346958
OCT-04   MONTH                  14358605   10346958
NOV-04   MONTH                  12757560   10346958
DEC-04   MONTH                  14872522   10346958
```

5.4.10 Nested Calculations

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

For example, Analytic Workspace Manager has templates for Moving Average and for Difference From Prior Period. You can create a calculated measure that calculates a moving average, then calculate the difference between the current and the previous moving averages.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Nested Calculations Example

This template creates a moving average for Units named UNITS_MOVING_AVG:

```
Moving average of UNITS in the TIME dimension and TIME.CALENDAR hierarchy. Include 1
preceding and 1 following members within level.
```

The next template creates a Difference From Prior Period calculation from UNITS_MOVING_AVG.

```
Difference from prior period for UNITS_MOVING_AVG in TIME dimension and TIME.CALENDAR
hierarchy 1 period ago.
```

These are the results of a query against the Units measure and the two calculated measures. The MOVING_AVG column shows the moving average, and the DIFF column shows the difference between the current moving average and the prior period's.

```
TIME     TIME_LEVEL               UNITS MOVING_AVG       DIFF
-------- -------------------- ---------- ---------- ----------
JAN-06   MONTH                     47776      48520         66
FEB-06   MONTH                     47695      48940        419
MAR-06   MONTH                     51348      48683       -257
APR-06   MONTH                     47005      50387       1705
MAY-06   MONTH                     52809      48411      -1976
JUN-06   MONTH                     45419      48872        461
JUL-06   MONTH                     48388      47546      -1326
AUG-06   MONTH                     48830      47857        312
SEP-06   MONTH                     46354      47532       -326
OCT-06   MONTH                     47411      46869       -663
NOV-06   MONTH                     46842      49768       2899
DEC-06   MONTH                     55052      50947       1179
2006     CALENDAR_YEAR            584929     575324      -4032
Q1.06    CALENDAR_QUARTER         146819     145705       2093
Q2.06    CALENDAR_QUARTER         145233     145208       -497
Q3.06    CALENDAR_QUARTER         143572     146037        829
Q4.06    CALENDAR_QUARTER         149305     146439        402
```

5.5 Creating User-Defined Expressions

Among the calculation types is a user-defined expression. Typically, you create calculations using the OLAP expression syntax, which includes the analytic functions, arithmetic operators, and single-row functions described in this chapter. The OLAP syntax is an extension of the SQL syntax. If you have used SQL analytic functions or single-row functions, then this syntax is familiar to you.

- Using the OLAP Expression Syntax
- Expression Syntax Example Using an Arithmetic Operator
- Free-Form Calculation Example Using an Analytic Function
- Expression Syntax Analytic Functions

> **See Also**
>
> For user-defined OLAP DML expressions, see "Creating Calculated Measures Using the OLAP DML".

Copyright © 2003, 2025, Oracle and/or its affiliates.

5.5.1 Using the OLAP Expression Syntax

The easiest way to formulate an expression in the OLAP expression syntax is to let Analytic Workspace Manager do the work for you. You can use the templates to create a similar calculation, and cut-and-paste the syntax as the basis for a new calculation.

To create a user-defined expression in the OLAP expression syntax:

1. Open the Create Calculated Measure dialog box.
2. Select the calculation type that most closely matches the one you want to define.
3. Modify the template as desired.
4. Cut-and-paste the calculation from the Calculation box into a text editor.
5. Repeat these steps if your calculation uses two or more functions.
6. Modify the calculation as desired in the text editor. You can combine numeric operators, analytic functions, and single-row functions in a single calculation.
7. From the Calculation Types list, select **OLAP Expression Syntax**.
8. Cut-and-paste the calculation from the text editor into the Calculation box.
9. Click **Create**.

> **See Also**
>
> Analytic Workspace Manager Help for detailed information about the OLAP expression syntax.

5.5.2 Expression Syntax Example Using an Arithmetic Operator

This template for Multiplication generates a calculation using Units Sold and Unit Cost.

```
Multiply measure UNITS by measure UNIT_COST.
```

The template generates this calculation using the multiplication operator (*). It appears in the Calculation box. Notice that UNITS is in the Units Cube and UNIT_COST is in the Price Cube.

```
UNITS_CUBE.UNITS * PRICE_CUBE.UNIT_COST
```

The syntax of this calculation is so simple that you only need the template to obtain the qualified name of the measure.

Following is a free-form calculation that calculates a 2% increase in units sold:

```
UNITS_CUBE.UNITS * 1.02
```

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

```
PRODUCT                  UNITS     TARGET
-------------------- ---------- ----------
Envoy Ambassador           2116       2158
Envoy Executive            2481       2531
Envoy Standard             3300       3366
Sentinel Financial        30513      31123
```


Copyright © 2003, 2025, Oracle and/or its affiliates.

```
Sentinel Multimedia        7948       8107
Sentinel Standard          7302       7448
```

5.5.3 Free-Form Calculation Example Using an Analytic Function

This template for Cumulative Average generates a calculation for the average number of units sold:

```
Cumulative average of UNITS in the TIME dimension and TIME.CALENDAR hierarchy within
level. Total from beginning to following member.
```

The template generates this calculation using the AVG function.

```
AVG(UNITS_CUBE.UNITS) OVER HIERARCHY (TIME.CALENDAR BETWEEN UNBOUNDED PRECEDING AND
UNBOUNDED FOLLOWING WITHIN LEVEL)
```

Following is a free-form calculation that computes the percent difference between current units sold and the cumulative average. It uses the AVG function and the subtraction (-), division (/) and multiplication (*) operators.

```
((UNITS_CUBE.UNITS - AVG(UNITS_CUBE.UNITS) OVER HIERARCHY (TIME.CALENDAR BETWEEN
UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING WITHIN LEVEL)) / AVG(UNITS_CUBE.UNITS) OVER
HIERARCHY (TIME.CALENDAR BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING WITHIN
LEVEL)) * 100
```

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

```
TIME          UNITS    CUM_AVG   PCT_DIFF
-------- ---------- ---------- ----------
Q1.06        146819     107965         36
Q2.06        145233     109062         33
Q3.06        143572     110048         30
Q4.06        149305     111138         34
```

You could also create this calculation using templates:

1. Calculate the cumulative average of UNITS with the Cumulative Average template.
2. Calculate the percent difference between current UNITS and the cumulative average with the Percent Difference template.

5.5.4 Expression Syntax Analytic Functions

Table 5-1 describes the analytic functions that you can use to create free-form calculations using the OLAP expression syntax. For the syntax of these functions, refer to Analytic Workspace Manager Help.

Table 5-1 OLAP Expression Syntax Analytic Functions

Function	Description
AVERAGE_RANK	Orders the members of a dimension based on the values of an expression. The function returns the sequence numbers of the dimension members, and assigns the same average rank to identical values.
AVG	Returns the average of a selection of values calculated over time.
COUNT	Tallies the number of data values identified by a selection of dimension members.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Table 5-1 (Cont.) OLAP Expression Syntax Analytic Functions

Function	Description
DENSE_RANK	Orders dimension members based on the values of an expression. The function returns the sequence numbers of the dimension members, and assigns the same minimum rank to identical values.
HIER_ANCESTOR	Returns an ancestor at a particular level of a hierarchy for either all members in the hierarchy or a particular member.
HIER_CHILD_COUNT	Returns the number of children of either all dimension members in a hierarchy or a particular member.
HIER_DEPTH	Returns a number representing the level depth of either all members of a hierarchy or a particular member, where 0 is the top level.
HIER_LEVEL	Returns the level of either all members of a hierarchy or a particular member.
HIER_PARENT	Returns the parent of either all dimension members in a hierarchy or a particular member.
HIER_TOP	Returns the topmost ancestor of either all members of a hierarchy or a particular member.
LAG	Returns the value of an expression at a specified number of time periods before the current period.
LAG_VARIANCE	Returns the difference between values for the current time period and a prior period.
LAG_VARIANCE_PERCENT	Returns the percent different between values for the current time period and a prior period.
LEAD	Returns the value of an expression at a specified number of time periods after the current period.
LEAD_VARIANCE	Returns the difference between values for the current time period and a future period.
LEAD_VARIANCE_PERCENT	Returns the percent different between values for the current time period and a future period.
MAX	Returns the largest of a selection of data values calculated over a particular dimension.
MIN	Returns the smallest of a selection of data values calculated over a particular dimension.
OLAP_DML_EXPRESSION	Runs an OLAP DML language expression.
RANK	Orders the members of a dimension based on the values of an expression. The function returns the sequence numbers of the dimension members, and assigns the same rank to identical values.
ROW_NUMBER	Orders the members of a dimension based on the values of an expression. The function returns the sequence numbers of the dimension members, and assigns a unique and arbitrary rank to identical values.
SHARE	Calculates the ratio of an expression's value for the current dimension member to the value for a related member of the same dimension.
SUM	Returns the total of a selection of values calculated over a particular dimension.

Copyright © 2003, 2025, Oracle and/or its affiliates.

5.6 Creating Calculated Measures Using the OLAP DML

The most advanced business calculations, such as forecasts, models, and allocations, are available through the OLAP DML. The OLAP DML is the internal data definition and manipulation language for analytic workspaces. Its primary data structures are dimensions, variables, formulas, and valuesets. These dimensional objects in an analytic workspace support the high-level dimensional objects in the database, such as cubes, cube dimensions, measures, attributes, and hierarchies.

Several commands in the OLAP DML support dimensional database objects such as cubes, levels, and hierarchies. You can use these commands, as well as the other functions, operators, and so forth in the language.

> **See Also**
>
> "Cube-Aware OLAP DML Statements" in the *Oracle OLAP DML Reference*

The OLAP DML is a mature language that was developed specifically for creating and managing dimensional objects and for manipulating dimensional data. Although programming in the OLAP DML requires significant skill, the language offers more power and flexibility than any other language.

- Selecting an OLAP DML Calculation Type
- OLAP DML Expression Examples
- OLAP DML Function Example

5.6.1 Selecting an OLAP DML Calculation Type

Analytic Workspace Manager supports two types of user-defined expressions using the OLAP DML:

- **OLAP DML Expression**: Calculates an OLAP DML expression. Choose this calculation type to run an existing program, a built-in function, or a single expression. The expression is stored as the EQ statement of a formula in the analytic workspace.
- **OLAP DML Function**: Runs an OLAP DML program entered in the Program Body field that returns values. Choose this calculation type to develop a new program in the OLAP DML. The name of the program is stored in the EQ statement of a formula in the analytic workspace.

To create an OLAP DML Expression:

1. Open the Create Calculated Measure dialog box.
2. From the Calculation Types list, select **OLAP DML Expression**.
3. For Data Type, select the data type of the return value.
4. Enter the expression in the OLAP DML field.
5. Click **Compile Expression** to check for syntax errors and to save a compiled version of the expression.
6. Click **Create** to create the calculated measure.

Copyright © 2003, 2025, Oracle and/or its affiliates.

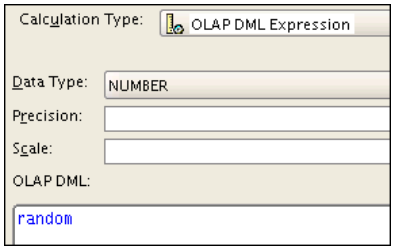
To create an OLAP DML Function:

1. Open the Create Calculated Measure dialog box.
2. From the Calculation Types list, select **OLAP DML Function.**
3. For Data Type, select the data type of the return value.
4. Enter a name for the function.
5. Enter the program in the Program Body field. Omit the `DEFINE`, `PROGRAM`, and `END` commands, because they are generated automatically.
6. Click **Compile Expression** to check for syntax errors and to save a compiled version of the program.
7. Click **Create** to create the calculated measure.

5.6.2 OLAP DML Expression Examples

The OLAP DML has many built-in functions. This example creates a calculated measure using the `RANDOM` function. Figure 5-6 shows the definition of this simple calculation. The calculated measure generates values in the default range of 0 to 1.

Figure 5-6 Using an OLAP DML Expression



The next example uses an arithmetic operator to calculate a 2% increase in units sold. This example of the OLAP DML is identical to the example in "Expression Syntax Example Using an Arithmetic Operator". However, note the difference in naming convention for the measure.

```
units_cube_units * 1.02
```

These are the results of a query against the two calculated measures created as OLAP DML expressions:

```
PRODUCT                   UNITS     TARGET     RANDOM
-------------------- ---------- ---------- ----------
Envoy Ambassador           2116       2158      .6467
Envoy Executive            2481       2531      .0773
Envoy Standard             3300       3366      .2349
Sentinel Financial        30513      31123      .6027
Sentinel Multimedia        7948       8107      .6494
Sentinel Standard          7302       7448      .5912
```

5.6.3 OLAP DML Function Example

An OLAP DML program that returns a value is also function.

Copyright © 2003, 2025, Oracle and/or its affiliates.

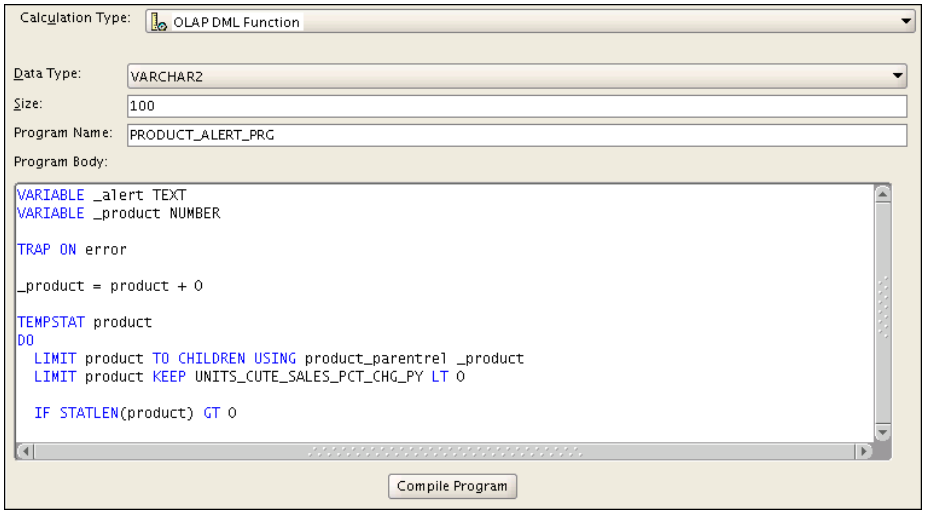
Example 5-1 OLAP DML Function

The program in this example returns the value ALERT when current sales are less than the previous year's. The actual calculation is performed by another calculated measure, UNITS_CUBE_SALES_PCT_CHG_PY, which is the percent change from the prior year for Sales. If sales are greater, then the program returns OKAY.

```
VARIABLE _alert   TEXT
VARIABLE _product NUMBER

TRAP ON error

_product = product + 0

TEMPSTAT product
DO
  LIMIT product TO CHILDREN USING product_parentrel _product
  LIMIT product KEEP UNITS_CUBE_SALES_PCT_CHG_PY LT 0

  IF STATLEN(product) GT 0
    THEN _alert = 'ALERT'
    ELSE _alert = 'OKAY'

DOEND

RETURN _alert

error:
RETURN 'ERROR'
```

This figure shows the definition of the program as a calculated measure.

Figure 5-7 Using an OLAP DML Function



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

```
CHANNEL         TIME         PCTCHG STATUS
--------------- -------- ---------- ------
Catalog         Q1.06            -1 ALERT
Catalog         Q2.06            -1 ALERT
Catalog         Q3.06            -3 ALERT
Catalog         Q4.06            -7 ALERT
```


Copyright © 2003, 2025, Oracle and/or its affiliates.

```
Direct Sales    Q1.06           -3 ALERT
Direct Sales    Q2.06           -1 ALERT
Direct Sales    Q3.06           10 OKAY
Direct Sales    Q4.06           -4 ALERT
Internet        Q1.06           29 OKAY
Internet        Q2.06            3 ALERT
Internet        Q3.06            0 ALERT
Internet        Q4.06           16 OKAY
```


Copyright © 2003, 2025, Oracle and/or its affiliates.

6

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 OLAP Applications
- Developing a Report Using BI Publisher
- Developing a Dashboard Using Application Express

- Developing OLAP Applications
- Developing a Report Using BI Publisher
- Developing a Dashboard Using Application Express

See Also

Querying Dimensional Objects

6.1 Developing OLAP Applications

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. You will 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.

Copyright © 2003, 2025, Oracle and/or its affiliates.

When the variable sets the value of the PARENT column of the hierarchy views, users can drill on a parent to view its children.

Example 6-1 shows a basic SQL query against UNITS_CUBE_VIEW 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_PCTCHG_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 level.

Example 6-1 SQL Query Against the Sales Cube

```
SELECT p.long_description "Product",
     cu.long_description "Customer",
     t.long_description "Time",
     ROUND(f.sales) "Sales",
     ROUND(f.sales_pp) "Prior Period",
     ROUND(f.sales_chg_pp) "Change",
     ROUND(f.sales_pctchg_pp * 100) "Percent Change"
/* From dimension views and cube view */
FROM product_primary_view p,
     customer_shipments_view cu,
     time_calendar_view t,
     channel_primary_view ch,
     units_cube_view f
/* Use parent columns to implement drilling */
WHERE p.parent = 'TOTAL'
    AND cu.parent = 'TOTAL'
    AND t.parent = 'CY2006'
    AND ch.level_name = 'TOTAL'
/* Join dimension views to cube view */
    AND p.dim_key = f.product
    AND cu.dim_key = f.customer
    AND t.dim_key = f.time
    AND ch.dim_key = f.channel
ORDER BY product, customer, t.end_date;

Product         Customer        Time            Sales Prior Period     Change Percent Change
--------------- --------------- ---------- ---------- ------------ ---------- --------------
Hardware        North America   Q1.06        16002175     14493426    1508749             10
Hardware        North America   Q2.06        16032643     16002175      30469              0
Hardware        North America   Q3.06        15698208     16032643    -334436             -2
Hardware        North America   Q4.06        15958791     15698208     260583              2
Hardware        Asia Pacific    Q1.06        13416447     14273900    -857453             -6
Hardware        Asia Pacific    Q2.06        14306431     13416447     889984              7
                 .
                 .
                 .
Software/Other  Asia Pacific    Q4.06          652300       647019       5281              1
Software/Other  Europe          Q1.06          737523       634293     103230             16
Software/Other  Europe          Q2.06          678391       737523     -59132             -8
Software/Other  Europe          Q3.06          499008       678391    -179383            -26
Software/Other  Europe          Q4.06          710796       499008     211788             42

24 rows selected.
```


Copyright © 2003, 2025, Oracle and/or its affiliates.

6.2 Developing a Report Using BI Publisher

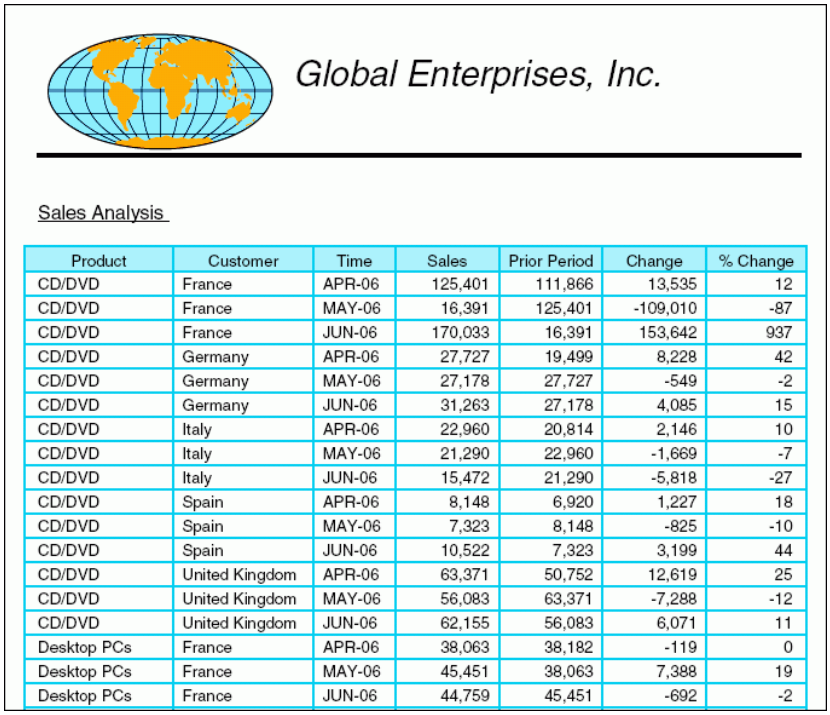
BI Publisher is an efficient, scalable reporting solution for generating and delivering information through a variety of distribution methods. It reduces the high costs associated with the development and maintenance of business documents, while increasing the efficiency of reports management. BI Publisher 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

Example 6-1 shows a report in PDF format based on the query shown in Example 6-1. 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 Q2-06. This chapter explains how you can create a report like this one using drillable dimensions.

Figure 6-1 Sales Report in BI Publisher


Global Enterprises, Inc.

Sales Analysis

Product	Customer	Time	Sales	Prior Period	Change	% Change
CD/DVD	France	APR-06	125,401	111,866	13,535	12
CD/DVD	France	MAY-06	16,391	125,401	-109,010	-87
CD/DVD	France	JUN-06	170,033	16,391	153,642	937
CD/DVD	Germany	APR-06	27,727	19,499	8,228	42
CD/DVD	Germany	MAY-06	27,178	27,727	-549	-2
CD/DVD	Germany	JUN-06	31,263	27,178	4,085	15
CD/DVD	Italy	APR-06	22,960	20,814	2,146	10
CD/DVD	Italy	MAY-06	21,290	22,960	-1,669	-7
CD/DVD	Italy	JUN-06	15,472	21,290	-5,818	-27
CD/DVD	Spain	APR-06	8,148	6,920	1,227	18
CD/DVD	Spain	MAY-06	7,323	8,148	-825	-10
CD/DVD	Spain	JUN-06	10,522	7,323	3,199	44
CD/DVD	United Kingdom	APR-06	63,371	50,752	12,619	25
CD/DVD	United Kingdom	MAY-06	56,083	63,371	-7,288	-12
CD/DVD	United Kingdom	JUN-06	62,155	56,083	6,071	11
Desktop PCs	France	APR-06	38,063	38,182	-119	0
Desktop PCs	France	MAY-06	45,451	38,063	7,388	19
Desktop PCs	France	JUN-06	44,759	45,451	-692	-2

- Creating an OLAP Report in BI Publisher
- Creating a Template in Microsoft Word
- Generating a Formatted Report
- Adding Dimension Choice Lists in BI Publisher

Copyright © 2003, 2025, Oracle and/or its affiliates.

6.2.1 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 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 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 folder to open it.
5. Create a 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 report appears in the folder, as shown in Figure 6-2.

Figure 6-2 Creating a Report in BI Publisher

Copyright © 2003, 2025, Oracle and/or its affiliates.

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 6-1. Do not use a semicolon.

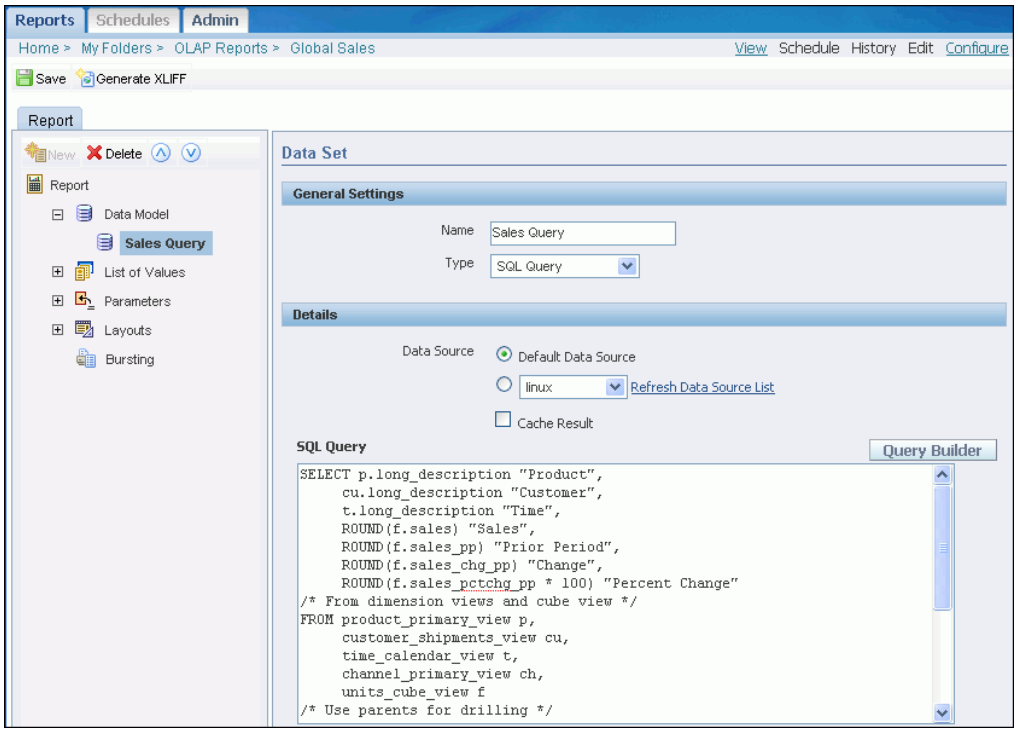
5. Click **Save**.

6. Click **View**.

   BI Publisher checks the report definition for errors. If there are none, then it generates the XML for the report.

Figure 6-3 shows the Report Editor with the Data Set page displayed.

Figure 6-3 Creating a Data Model in the BI Publisher Report Editor



6.2.2 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:

Copyright © 2003, 2025, Oracle and/or its affiliates.

- 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 6-2 XML for a SQL Query

This example 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.

```
<ROW>
<Product>Hardware</Product>
<Customer>North America</Customer>
<Time>Q1.06</Time>
<Sales>16002175</Sales>
<Prior_Period>14493426</Prior_Period>
<Change>1508749</Change>
<Percent_Change>10</Percent_Change>
</ROW>
```

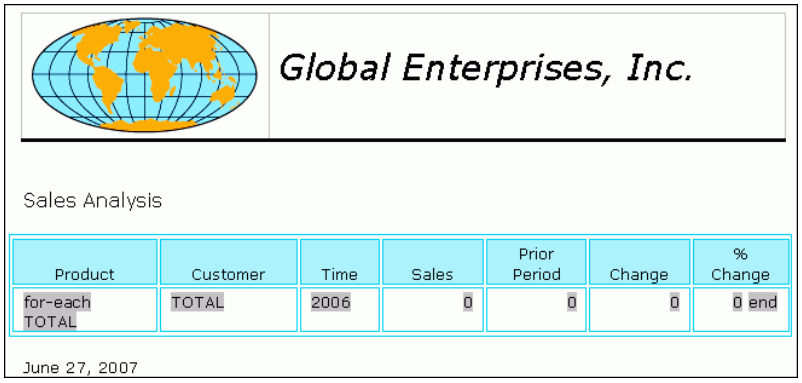
Figure 6-4 shows the Word document that is used as the template for the sample report. It contains these elements:

- 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. The first and last columns contain two fields. The first and last 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.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Figure 6-4 Sample Report Template Created in Word for BI Publisher



The following procedure defines the template manually. Alternatively, you can use a Word plug-in called Oracle BI Publisher Desktop. On the BI Publisher My Folders page, click **Template Builder** to download the plug-in.

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 6-4 is very simple. You can use much more elaborate formatting if you want, including nested columns and tables.
4. From the View menu, select **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. Select 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:

Copyright © 2003, 2025, Oracle and/or its affiliates.

```
<?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.

6.2.3 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.

To 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 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**.

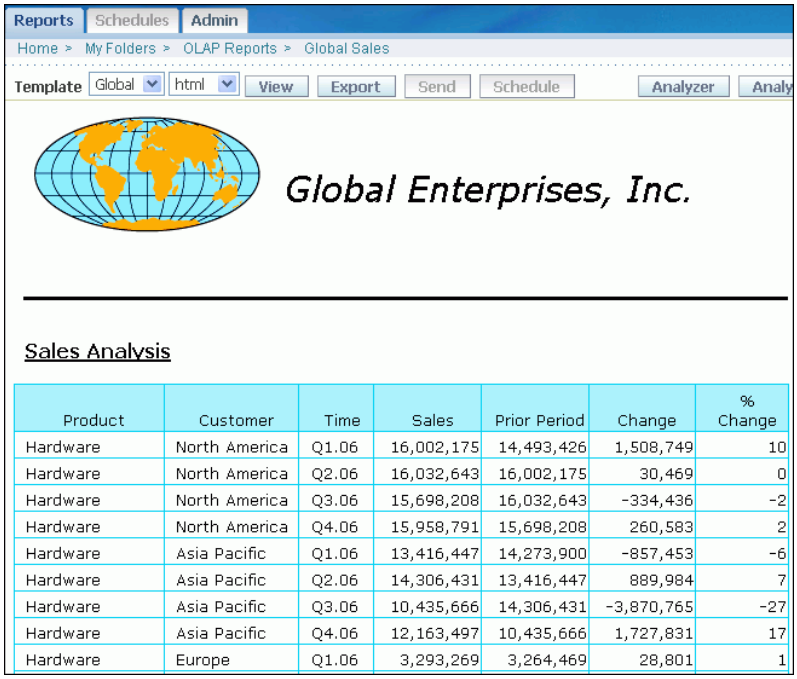
   The report appears.
10. To change the format, select a format from the list and click **View**.

    To see the XML, select **Data**.

Figure 6-5 shows the report in HTML format.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Figure 6-5 BI Publisher Report Displayed in HTML Format



Product	Customer	Time	Sales	Prior Period	Change	% Change
Hardware	North America	Q1.06	16,002,175	14,493,426	1,508,749	10
Hardware	North America	Q2.06	16,032,643	16,002,175	30,469	0
Hardware	North America	Q3.06	15,698,208	16,032,643	-334,436	-2
Hardware	North America	Q4.06	15,958,791	15,698,208	260,583	2
Hardware	Asia Pacific	Q1.06	13,416,447	14,273,900	-857,453	-6
Hardware	Asia Pacific	Q2.06	14,306,431	13,416,447	889,984	7
Hardware	Asia Pacific	Q3.06	10,435,666	14,306,431	-3,870,765	-27
Hardware	Asia Pacific	Q4.06	12,163,497	10,435,666	1,727,831	17
Hardware	Europe	Q1.06	3,293,269	3,264,469	28,801	1

6.2.4 Adding Dimension Choice Lists in BI Publisher

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.

These steps are described in the following topics:

- Creating a List of Values for a BI Publisher Report
- Creating a Menu
- Editing the Query in BI Publisher

- Creating a List of Values for a BI Publisher Report
- Creating a Menu
- Editing the Query in BI Publisher

6.2.4.1 Creating a List of Values for a BI Publisher Report

For a list of values, use a SQL query that selects the dimension keys to display. Include the `LONG_DESCRIPTION` and `DIM_KEY` columns from the hierarchy view. This example creates a list for the Product Primary hierarchy:

```
SELECT long_description, dim_key
     FROM product_primary_view
     WHERE parent = 'TOTAL'
     OR dim_key = 'TOTAL'
```


Copyright © 2003, 2025, Oracle and/or its affiliates.

```
     ORDER BY level_name, long_description

LONG_DESCRIPTION     DIM_KEY
-------------------- ------------
Hardware             HRD
Software/Other       SFT
Total Product        TOTAL
```

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 the dimension hierarchy view, as shown previously.
4. Click **Save**.

Repeat these steps for the other dimensions. This example uses lists for Product, Customer, and Time.

6.2.4.2 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 will 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 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 creates menus for Product, Customer, and Time.

Copyright © 2003, 2025, Oracle and/or its affiliates.

6.2.4.3 Editing the Query in BI Publisher

To activate the menus, you change the `WHERE` clause in the query for the report to use 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  p.parent  = :product
   AND cu.parent = :customer
   AND t.parent  = :time
   AND ch.level_name = 'TOTAL'
```

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.

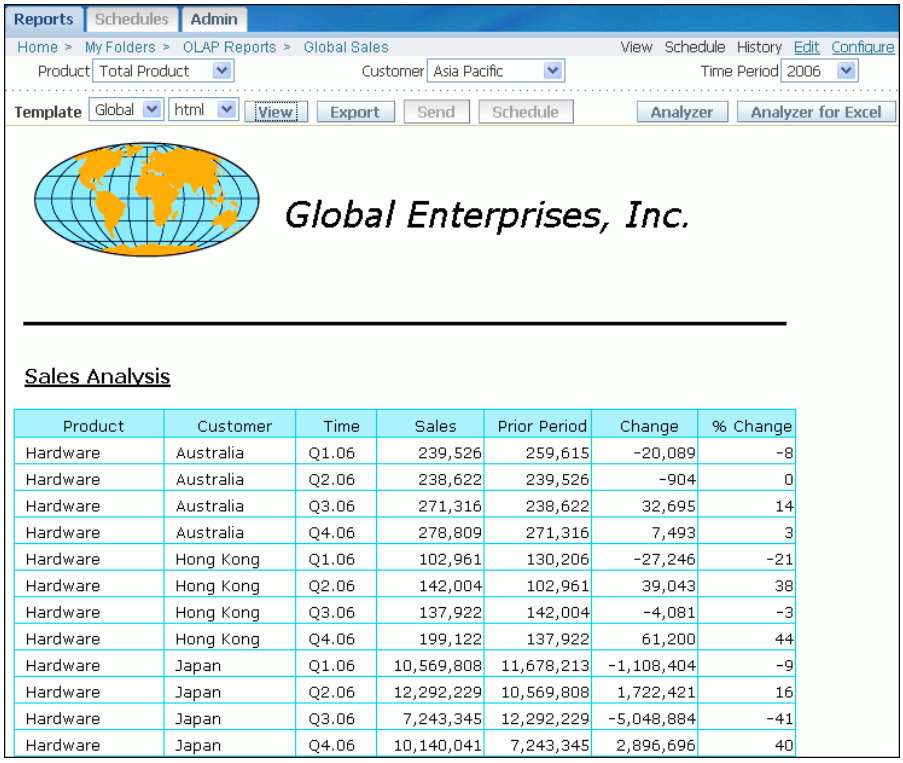
3. Click **Save**.

Figure 6-6 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 months in Q3.06, the Hardware products, and the countries in Europe. To see a different selection of data, you 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 one shown in Figure 6-1.

You can continue working on this report, adding charts and other tables.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Figure 6-6 Sales Report With Choice Lists in BI Publisher



Product	Customer	Time	Sales	Prior Period	Change	% Change
Hardware	Australia	Q1.06	239,526	259,615	-20,089	-8
Hardware	Australia	Q2.06	238,622	239,526	-904	0
Hardware	Australia	Q3.06	271,316	238,622	32,695	14
Hardware	Australia	Q4.06	278,809	271,316	7,493	3
Hardware	Hong Kong	Q1.06	102,961	130,206	-27,246	-21
Hardware	Hong Kong	Q2.06	142,004	102,961	39,043	38
Hardware	Hong Kong	Q3.06	137,922	142,004	-4,081	-3
Hardware	Hong Kong	Q4.06	199,122	137,922	61,200	44
Hardware	Japan	Q1.06	10,569,808	11,678,213	-1,108,404	-9
Hardware	Japan	Q2.06	12,292,229	10,569,808	1,722,421	16
Hardware	Japan	Q3.06	7,243,345	12,292,229	-5,048,884	-41
Hardware	Japan	Q4.06	10,140,041	7,243,345	2,896,696	40

6.3 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.

Overview shows a sophisticated dashboard that extracts analytic data from cubes and presents it in a variety of graphs and reports. You can easily create dashboards from your cubes that display the rich analytical content generated by Oracle OLAP.

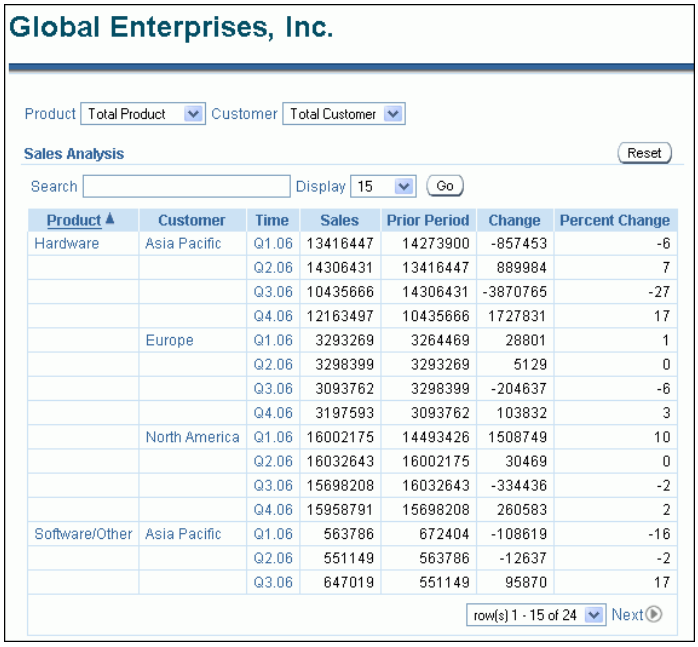
If you have not used Application Express, you can download the software, tutorials, and full documentation from the Oracle Technology Network at

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

Figure 6-7 shows a crosstab with display lists for Product and Customer, and links in all three dimension columns. Choosing a different 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.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Figure 6-7 Drillable Dimensions in Application Express



- Creating an OLAP Application in Application Express
- Adding Dimension Choice Lists in Application Express
- Drilling on Dimension Columns

6.3.1 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:

Copyright © 2003, 2025, Oracle and/or its affiliates.

a. For Page Source, select **SQL Query**.

b. For Title, enter a name such as `Sales Analysis`.

   This title appears on the page.

c. For Query, enter a SQL `SELECT` statement for your cube, like the one shown in Example 6-1. 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).

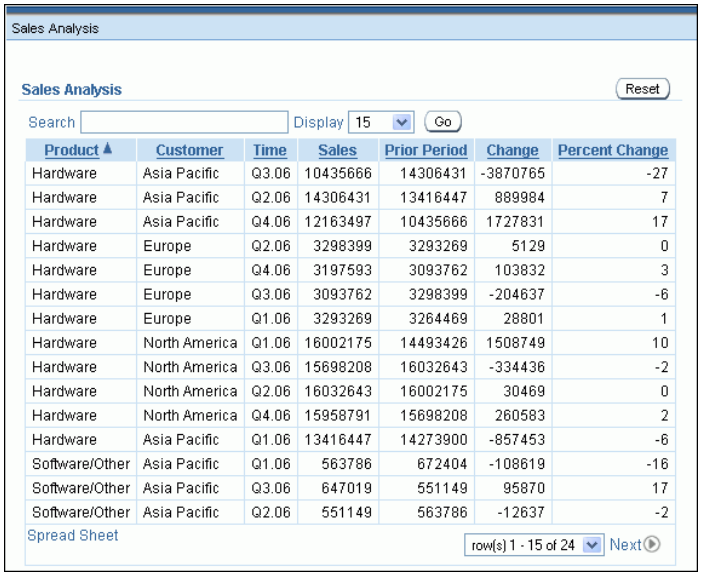
8. On the Confirm page, click **Create**.

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 6-8 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 want.

Figure 6-8 Basic Sales Report in Application Express



Copyright © 2003, 2025, Oracle and/or its affiliates.

6.3.2 Adding Dimension Choice Lists in Application Express

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 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.

In Application Express, the Page Definition is where you can create and edit pages, including adding and modifying graphical items. The items are organized in three columns: Page Rendering, Page Processing, and Shared Components.

To open a Page Definition:

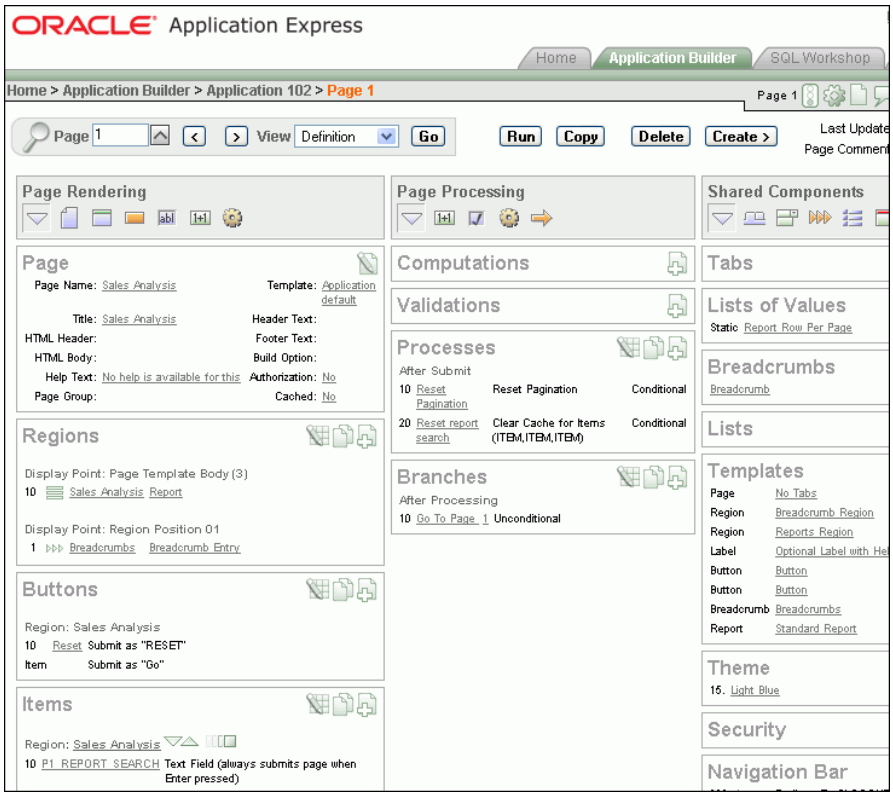
- After running the application, click the **Edit Page** link at the bottom of the page.

  or

- On the Application home page, click the icon for the page where the report is defined.

Figure 6-9 Application Express Page Definition

This figure shows an area of the Page Definition.



Copyright © 2003, 2025, Oracle and/or its affiliates.

The steps in implementing a choice list are described in the following topics:

- Creating a Region
- Creating a List of Values in Application Express
- Creating the Choice List
- Editing the Query in Application Express

- Creating a Region
- Creating a List of Values in Application Express
- Creating the Choice List
- Editing the Query in Application Express

6.3.2.1 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 name is `lov_region`, the template is **No Template**, and the location is **Page Template Body (1 items below template content)**. The name can be displayed on the rendered page, but it is hidden in this example.
4. Click **Create Region**.

   The region appears on the Page Definition under Regions.

6.3.2.2 Creating a List of Values in Application Express

For a list of values, use a SQL query like the one shown here. Include the `LONG_DESCRIPTION` and `DIM_KEY` columns from the hierarchy view. This query creates a list for the Customer Shipments hierarchy:

```
SELECT long_description, dim_key
     FROM customer_shipments_view
     WHERE parent = 'TOTAL'
     OR dim_key= 'TOTAL'
     ORDER BY level_name, long_description;

LONG_DESCRIPTION     DIM_KEY
-------------------- ----------
Asia Pacific         APAC
Europe               EMEA
North America        AMER
Total Customer       TOTAL
```

To create a List of Values:

1. On the Page Definition under List of Values, click the **Create** icon.

   The Create List of Values wizard opens.

Copyright © 2003, 2025, Oracle and/or its affiliates.

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 list of values (LOV) appears in the Page Definition under List of Values.

For additional lists of values (LOVs), repeat these steps. This example creates LOVs for the Product and Customer dimensions.

6.3.2.3 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.
   - Select 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`.
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 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**.

Copyright © 2003, 2025, Oracle and/or its affiliates.

4. On the Branch Conditions page, accept the default settings to create an unconditional branch.
5. Click **Apply Changes**.

   The Edit Branch page closes, and you return to the Page Definition. The unconditional branch is listed under Branches.

6.3.2.4 Editing the Query in Application Express

This is the format for the dynamic conditions in the `WHERE` clause:

```
parent_column = NVL(:bind_variable, 'top dim_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.

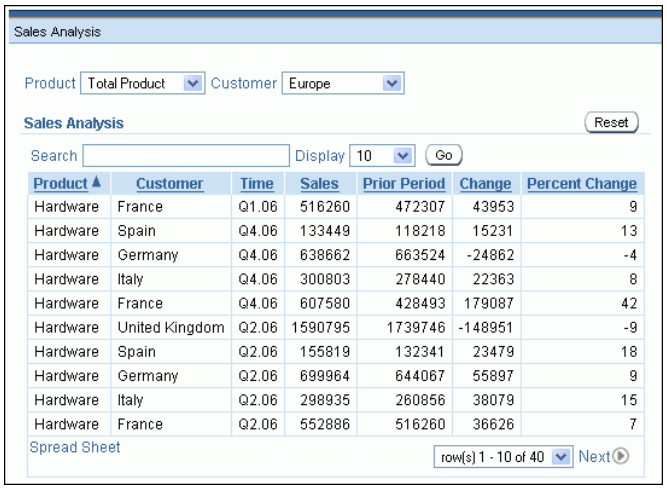
   The Edit Region page opens.
3. Under Source, modify the query:
   - Change the `WHERE` clause to use the bind variables.
   - Delete the outer `SELECT` added by Application Express.
4. Click **Apply Changes**.

For this example, the `WHERE` clause now looks like this:

```
WHERE p.parent = NVL(:P1_PRODUCT, 'TOTAL')
    AND cu.parent = NVL(:P1_CUSTOMER, 'TOTAL')
    AND t.parent = 'CY2006'
    AND ch.level_name = 'TOTAL'
```

Figure 6-10 shows the modified page with choice lists for Product and Customer.

Figure 6-10 Dashboard With Choice Lists for Drilling



Copyright © 2003, 2025, Oracle and/or its affiliates.

6.3.3 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. See Creating Hidden Items.
- Edit the query to use the bind variables. See Editing the Query to Use Bind Variables.
- Add links to the dimension columns of the crosstab. See Adding Links to the Dimension Columns.

The example in these topics adds drilling to all displayed dimensions.

- Creating Hidden Items
- Editing the Query to Use Bind Variables
- Adding Links to the Dimension Columns

6.3.3.1 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 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.
   - Select the region where the report is defined.
5. On the Source page, enter the dimension key at the top of the hierarchy.

   `TOTAL` is the top of all hierarchies in the Global schema. For this example, Time is set to `CY2006` to restrict the selection to one year.
6. Click **Create Item**.
7. Repeat these steps for any other dimensions that support drilling only on the column links.

   For this example, a hidden item is defined for Time.

6.3.3.2 Editing the Query to Use Bind Variables

To add column links to a report, you must change two areas of the `SELECT` statement:

Copyright © 2003, 2025, Oracle and/or its affiliates.

- 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 `DIM_KEY` and `PARENT` columns in the hierarchy views 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".

Example 6-3 shows the modified sample query.

Example 6-3 Revised Query for Column Links in Application Express

```
SELECT p.long_description "Product",
     cu.long_description "Customer",
     t.long_description "Time",
     ROUND(f.sales) "Sales",
     ROUND(f.sales_pp) "Prior Period",
     ROUND(f.sales_chg_pp) "Change",
     ROUND(f.sales_pctchg_pp * 100) "Percent Change",
/* Add DIM_KEY and PARENT columns for column links */
     p.dim_key product_key,
     p.parent product_parent,
     cu.dim_key customer_key,
     cu.parent customer_parent,
     t.dim_key time_key,
     t.parent time_parent
/* From dimension views and cube view */
FROM product_primary_view p,
     customer_shipments_view cu,
     time_calendar_view t,
     channel_primary_view ch,
     units_cube_view f
/* Use parent columns and bind variables for drilling */
WHERE p.parent = NVL(:P1_PRODUCT, 'TOTAL')
    AND cu.parent = NVL(:P1_CUSTOMER, 'TOTAL')
    AND t.parent = NVL(:P1_TIME, 'CY2006')
    AND ch.level_name = 'TOTAL'
/* Join dimension views to cube view */
    AND p.dim_key = f.product
    AND cu.dim_key = f.customer
    AND t.dim_key = f.time
    AND ch.dim_key = f.channel
```

6.3.3.3 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 to hide, such as the `DIM_KEY` and `PARENT` columns.

Copyright © 2003, 2025, Oracle and/or its affiliates.

- 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: Select 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 `DIM_KEY` column for the dimension being defined or the `PARENT` column for the other dimensions.

   Figure 6-11 shows the link definition for the Time dimension.

6. Click **Apply Changes**.

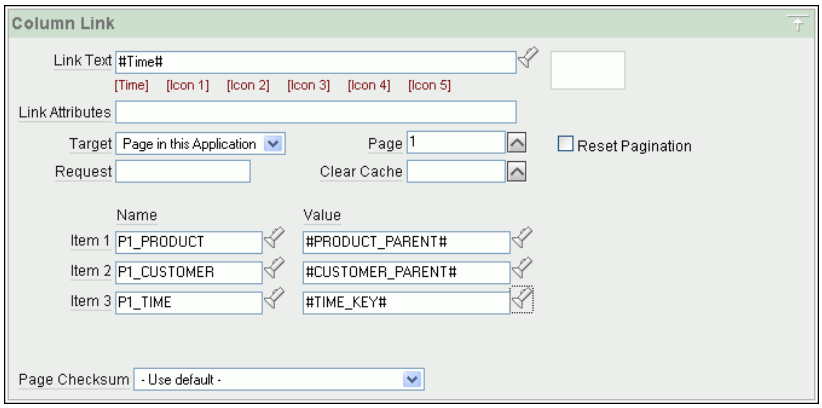
   The Column Attributes page closes, and you return to the Report Attributes page.

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 6-11 Definition of the Time Link



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**.

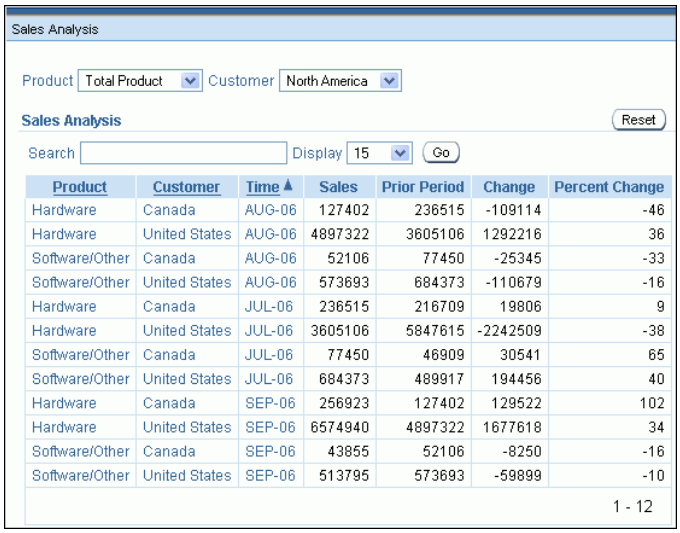
   The Edit Branch page closes, and you return to the Page Definition.

6. Click **Run** to display the page.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Figure 6-12 shows the finished page displaying months in Q3.06. 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 6-12 Sales Analysis Report With Column Links in Application Express



Copyright © 2003, 2025, Oracle and/or its affiliates.

7

Administering Oracle OLAP

Because Oracle OLAP is contained in the database and its resources are managed using the same tools, the management tasks of Oracle OLAP and the database converge. Nonetheless, you should address tasks such as database tuning in the specific context of data warehousing.

This chapter contains the following topics:

- Setting Database Initialization Parameters
- Storage Management
- Dictionary Views and System Tables
- Partitioned Cubes and Parallelism
- Analyzing Cubes and Dimensions
- Monitoring Analytic Workspaces
- About Backing Up and Recovering Analytic Workspaces
- About Copying Analytic Workspaces
- Cube Materialized Views

- Setting Database Initialization Parameters
- Storage Management
- Dictionary Views and System Tables
- Partitioned Cubes and Parallelism
- Analyzing Cubes and Dimensions
- Monitoring Analytic Workspaces
- About Backing Up and Recovering Analytic Workspaces
- About Copying Analytic Workspaces

- About Saving Dimensional Object Definitions
- Cube Materialized Views

7.1 Setting Database Initialization Parameters

Table 7-1 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

Copyright © 2003, 2025, Oracle and/or its affiliates.

Table 7-1 Initial Settings for Database Parameters

Parameter	Default Value	Recommended Setting	Description
JOB_QUEUE_PROCESSES	1000	If you reduce this value to limit the maximum number of job slaves running on an instance, then calculate the following number of processes for use by OLAP: 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, as described in "Parallelism"
PARALLEL_DEGREE_POLICY	MANUAL	AUTO or LIMITED	Controls how the degree of parallelism is determined When set to AUTO or LIMITED, Oracle determines whether a SQL statement runs in parallel and, if so, the degree of parallelism used
SESSIONS	Derived	2.5 * maximum number of simultaneous OLAP users	Provides sufficient background processes for each user
UNDO_MANAGEMENT	AUTO (MANUAL in 10*g*)	AUTO	Specifies use of an undo tablespace
UNDO_TABLESPACE	Derived	Name of the undo tablespace, which must be defined previously	Identifies the undo tablespace defined for OLAP use, as shown in "Creating an Undo Tablespace"

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, as described in Table 7-1.
3. Stop and restart the database.

   On Windows, use the Services utility to stop and restart OracleService.

   On Linux, use commands like 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
```

7.2 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

Copyright © 2003, 2025, Oracle and/or its affiliates.

- Creating Permanent Tablespaces for OLAP Use
- Creating Temporary Tablespaces for OLAP Use
- Spreading Data Across Storage Resources

7.2.1 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".

```
UNDO_TABLESPACE=tablespace
UNDO_MANAGEMENT=AUTO
```

7.2.2 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;
```

7.2.3 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;
```

7.2.4 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.

Copyright © 2003, 2025, Oracle and/or its affiliates.

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;
```

7.3 Dictionary Views and System Tables

Oracle Database data dictionary views and system tables contain extensive information about analytic workspaces.

- Static Data Dictionary Views
- System Tables
- Analytic Workspace Tables
- Maintenance Logs

7.3.1 Static Data Dictionary Views

Among the static views of the database data dictionary are several that provide information about analytic workspaces. Table 7-2 provides brief descriptions of them. All data dictionary views have corresponding DBA and USER views.

Table 7-2 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.

Copyright © 2003, 2025, Oracle and/or its affiliates.

> **See Also**
>
> - "Querying the Data Dictionary" for a list of data dictionary views that describe OLAP dimensional objects
> - *Oracle Database Reference* for full descriptions of all data dictionary views

7.3.2 System Tables

The SYS user owns several tables associated with analytic workspaces.

> **Note**
>
> 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 7-3 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.
AW_TRACK$	Stores tracking data about access to aggregate cells. Not currently used.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Table 7-3 (Cont.) OLAP Tables Owned By SYS

Table	Description
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$.

7.3.3 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 `FINANCIALS` and the other named `MARKETING`, then these tables are created in the `GLOBAL` schema:

```
AW$FINANCIALS
AW$MARKETING
```

The tables store all of the object definitions and data.

7.3.4 Maintenance Logs

The first time you load data into a cube or dimension using Analytic Workspace Manager, it creates several logs. These logs are stored in tables in the same schema as the analytic workspace:

- **Cube Build Log**: Contains information about what happened during a build. Use this log to determine whether the build produced the results you were expecting, and if not, why not. The log is continually updated whenever a cube or dimension is refreshed, whether by Analytic Workspace Manager, the database materialized view refresh subsystem, or a PL/SQL procedure. You can query the log at any time to evaluate the progress of the build and to estimate the time to completion. The default table name is `CUBE_BUILD_LOG`.
- **Cube Dimension Compile Log**: Contains errors that occur during the validation of the dimension hierarchies when OLAP is aggregating a cube. The default table name is `CUBE_DIMENSION_COMPILE`.
- **Cube Operations Log**: Contains messages and debugging information for all OLAP engine events. The default table name is `CUBE_OPERATIONS_LOG`.
- **Cube Rejected Records Log**: Identifies any records that were rejected because they did not meet the expected format. The default table name is `CUBE_REJECTED_RECORDS`.

These logs enable you to track the progress of long running processes, then use the results to profile performance characteristics. They provide information to help you diagnose and remedy problems that may occur during development and maintenance of a cube. They also help diagnose performance problems in querying cubes.

You can also run the $ORACLE_HOME/olap/admin/utlolaplog.sql script to create the build log with some useful views.

Copyright © 2003, 2025, Oracle and/or its affiliates.

The Maintenance Wizard in Analytic Workspace Manager displays the relevant rows from these tables during every build on the Maintenance Log page. You can query the tables directly in any SQL interface.

> **See Also**
>
> `DBMS_CUBE_LOG` in *Oracle Database PL/SQL Packages and Types Reference*

7.4 Partitioned Cubes and Parallelism

Cubes are often partitioned to improve build and maintenance times. For information about creating a partitioned cube, refer to "Partitioning a Cube". Partitioning and parallelism are discussed in the following topics:

- Querying Metadata for Cube Partitioning
- Creating and Dropping Partitions
- Parallelism

- Querying Metadata for Cube Partitioning
- Creating and Dropping Partitions
- Parallelism

7.4.1 Querying Metadata for Cube Partitioning

To discover the current partitioning, query the `ALL_CUBES` data dictionary view. The `PARTITION_DIMENSION_NAME`, `PARTITION_HIERARCHY_NAME`, and `PARTITION_LEVEL_NAME` columns display partitioning information. For example, the following query shows that the Units Cube is partitioned on the Time dimension, the Calendar hierarchy, and the Calendar Year level.

```
SELECT partition_dimension_name, partition_hierarchy_name,
     partition_level_name FROM all_cubes
     WHERE owner='GLOBAL' AND cube_name='UNITS_CUBE';

PARTITION_DIMENSION_NAME  PARTITION_HIERARCHY_NAME  PARTITION_LEVEL_NAME
------------------------- ------------------------- --------------------
TIME                      CALENDAR                  CALENDAR_YEAR
```

7.4.2 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 partition is added to the

Copyright © 2003, 2025, Oracle and/or its affiliates.

cube. Whenever a Calendar Quarter value is deleted from the Time dimension, the corresponding empty partition is deleted from the cube.

7.4.3 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 Partitioning 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. If you have `SYS` privileges, 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, or on the Materialized View tab of the cube.
- The number of processes allocated to SQL to fetch rows from the relational source tables. When `PARALLEL_DEGREE_POLICY` is set to `AUTO` or `LIMITED`, the database can allocate additional processes for executing SQL statements.

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 when `PARALLEL_DEGREE_POLICY` is set to its default value of `MANUAL`:

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.

When `PARALLEL_DEGREE_POLICY` is set to `AUTO` or `LIMITED`, Oracle Database may allocate more than the designated processes.

Example 7-1 Build Log for Global Units Cube

This example shows excerpts from `CUBE_BUILD_LOG` for a build of the Units cube and its dimensions. Partitioning on the Calendar Year level of the Time dimension created 10 bottom partitions for 1998 to 2007. `JOB_QUEUE_PROCESSES` is set to 2 and the parallelism option is set to 2 for the build also. The log shows that Oracle Database processed the Global in this way:

1. Processed the four dimensions serially.
2. Processed each partition of the Units cube

Copyright © 2003, 2025, Oracle and/or its affiliates.

```
SLAVE_NUMBER STATUS     COMMAND              BUILD_OBJECT    PARTITION
------------ ---------- -------------------- --------------- ---------------
           0 STARTED    BUILD
           0 STARTED    ATTACH AW RW WAIT
           0 COMPLETED  ATTACH AW RW WAIT
           0 STARTED    FREEZE
           0 COMPLETED  FREEZE
           0 STARTED    LOAD NO SYNCH        TIME
           0 SQL        LOAD NO SYNCH        TIME
               .
               .
               .
           0 SQL        LOAD NO SYNCH        PRODUCT
           0 SQL        LOAD NO SYNCH        PRODUCT
           0 COMPLETED  LOAD NO SYNCH        PRODUCT
           0 STARTED    COMPILE              PRODUCT
           0 COMPLETED  COMPILE              PRODUCT
           0 STARTED    COMPILE AGGMAP       UNITS_CUBE
           0 COMPLETED  COMPILE AGGMAP       UNITS_CUBE
           0 STARTED    COMPILE AGGMAP       PRICE_CUBE
           0 COMPLETED  COMPILE AGGMAP       PRICE_CUBE
           0 STARTED    UPDATE/COMMIT        PRODUCT
           0 COMPLETED  UPDATE/COMMIT        PRODUCT
           0 STARTED    UPDATE/COMMIT
           0 COMPLETED  UPDATE/COMMIT
           0 STARTED    REATTACH AW MULTI TH
                        AW

           0 COMPLETED  REATTACH AW MULTI TH
                        AW

           0 STARTED    SLAVE                UNITS_CUBE      P10:CY2007
           0 STARTED    SLAVE                UNITS_CUBE      P9:CY2006
           1 STARTED    BUILD                                P10:CY2007
           1 STARTED    ATTACH AW MULTI THAW UNITS_CUBE      P10:CY2007
           1 COMPLETED  ATTACH AW MULTI THAW UNITS_CUBE      P10:CY2007
           1 STARTED    ACQUIRE              UNITS_CUBE      P10:CY2007
           1 COMPLETED  ACQUIRE              UNITS_CUBE      P10:CY2007
           1 STARTED    LOAD                 UNITS_CUBE      P10:CY2007
           1 SQL        LOAD                 UNITS_CUBE      P10:CY2007
           1 COMPLETED  LOAD                 UNITS_CUBE      P10:CY2007
           1 STARTED    UPDATE/COMMIT        UNITS_CUBE      P10:CY2007
           1 COMPLETED  UPDATE/COMMIT        UNITS_CUBE      P10:CY2007
               .
               .
               .
          10 STARTED    BUILD                                P1:CY1998
          10 STARTED    ATTACH AW MULTI THAW UNITS_CUBE      P1:CY1998
          10 COMPLETED  ATTACH AW MULTI THAW UNITS_CUBE      P1:CY1998
          10 STARTED    ACQUIRE              UNITS_CUBE      P1:CY1998
          10 COMPLETED  ACQUIRE              UNITS_CUBE      P1:CY1998
          10 STARTED    LOAD                 UNITS_CUBE      P1:CY1998
          10 SQL        LOAD                 UNITS_CUBE      P1:CY1998
          10 COMPLETED  LOAD                 UNITS_CUBE      P1:CY1998
          10 STARTED    SOLVE                UNITS_CUBE      P1:CY1998
          10 COMPLETED  SOLVE                UNITS_CUBE      P1:CY1998
          10 STARTED    UPDATE/COMMIT        UNITS_CUBE      P1:CY1998
          10 COMPLETED  UPDATE/COMMIT        UNITS_CUBE      P1:CY1998
          10 STARTED    DETACH AW            UNITS_CUBE      P1:CY1998
          10 COMPLETED  DETACH AW            UNITS_CUBE      P1:CY1998
          10 COMPLETED  BUILD                                P1:CY1998
           0 COMPLETED  SLAVE                UNITS_CUBE      P1:CY1998
```


Copyright © 2003, 2025, Oracle and/or its affiliates.

```
 0 STARTED    REATTACH AW MULTI TH
              AW

 0 COMPLETED  REATTACH AW MULTI TH
              AW

 0 STARTED    SLAVE                UNITS_CUBE      P0
11 STARTED    BUILD                                P0
11 STARTED    ATTACH AW MULTI THAW UNITS_CUBE      P0
11 COMPLETED  ATTACH AW MULTI THAW UNITS_CUBE      P0
11 STARTED    ACQUIRE              UNITS_CUBE      P0
11 COMPLETED  ACQUIRE              UNITS_CUBE      P0
11 STARTED    LOAD                 UNITS_CUBE      P0
11 COMPLETED  LOAD                 UNITS_CUBE      P0
11 STARTED    SOLVE                UNITS_CUBE      P0
11 COMPLETED  SOLVE                UNITS_CUBE      P0
11 STARTED    UPDATE/COMMIT        UNITS_CUBE      P0
11 COMPLETED  UPDATE/COMMIT        UNITS_CUBE      P0
11 STARTED    DETACH AW            UNITS_CUBE      P0
11 COMPLETED  DETACH AW            UNITS_CUBE      P0
11 COMPLETED  BUILD                                P0
 0 COMPLETED  SLAVE                UNITS_CUBE      P0
 0 STARTED    REATTACH AW RW WAIT
 0 COMPLETED  REATTACH AW RW WAIT
 0 STARTED    ANALYZE              UNITS_CUBE
 0 COMPLETED  ANALYZE              UNITS_CUBE
 0 STARTED    THAW
 0 COMPLETED  THAW
 0 STARTED    DETACH AW
 0 COMPLETED  DETACH AW
 0 COMPLETED  BUILD

268 rows selected.
```

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 Real Application Clusters (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.

7.5 Analyzing Cubes and Dimensions

If your application runs queries directly against a single cube, you do not need to generate optimizer statistics for the cube. These queries are automatically optimized within the analytic workspace.

Optimizer statistics are used to create execution plans for queries that join two cube views or join a cube view to a table or a view of a table. They are also used for cost-based rewrite to cube materialized views. You must generate the statistics only for these types of queries.

To generate optimizer statistics, use the `DBMS_AW_STATS` PL/SQL package. You can run this package in Analytic Workspace Manager as part of a cube script, in SQL*Plus, or in any other SQL interface. Generating the statistics does not have a significant performance cost.

`DBMS_AW_STATS` has the following syntax:

Copyright © 2003, 2025, Oracle and/or its affiliates.

```
DBMS_AW_STATS.ANALYZE
     (object       IN VARCHAR2);
```

The argument can be either a cube or a dimension. Example 7-2 shows a sample script for generating statistics on the Units cube and its dimensions.

Example 7-2 Generating Statistics for the Units Cube

```
BEGIN
     DBMS_AW_STATS.ANALYZE('units_cube');
     DBMS_AW_STATS.ANALYZE('time');
     DBMS_AW_STATS.ANALYZE('customer');
     DBMS_AW_STATS.ANALYZE('product');
     DBMS_AW_STATS.ANALYZE('channel');
END;
/
```

Although you cannot view the statistics directly, you can examine the execution plans, as described in "Viewing Execution Plans".

> **See Also**
>
> *Oracle Database SQL Tuning Guide*

7.6 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 Cloud Control (Cloud Control) is a general database management and administration tool. In addition to facilitating basic tasks like adding users and modifying datafiles, Cloud 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
- Basic Queries for Monitoring the OLAP Option
- OLAP DBA Scripts
- Scripts for Monitoring Performance
- Monitoring Disk Space

Copyright © 2003, 2025, Oracle and/or its affiliates.

7.6.1 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 7-4 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 7-4 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$AW_SESSION_INFO	Collects information about each active session.
V$AW_OLAP	Collects information about the status of active analytic workspaces.

Table 7-5 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 7-5 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$PGASTAT	Provides PGA memory usage statistics and statistics about the automatic PGA memory manager when PGA_AGGREGATE_TARGET is set.
V$ROWCACHE	Displays statistics for data dictionary activity. Each row contains statistics for one data dictionary cache.
V$SYSSTAT	Lists system statistics.

7.6.2 Basic Queries for Monitoring the OLAP Option

The following queries extract OLAP information from the data dictionary. You must have a privileged account to query the DBA views.

Copyright © 2003, 2025, Oracle and/or its affiliates.

More complex queries are provided in a script that you can download from the Oracle OLAP website on the Oracle Technology Network. For descriptions of these scripts and download instructions, refer to "OLAP DBA Scripts".

- Is the OLAP Option Installed in the Database?
- What Analytic Workspaces Are in the Database?
- How Big Is the Analytic Workspace?
- When Were the Analytic Workspaces Created?

7.6.2.1 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  12.1.0.1.0                     VALID
Oracle OLAP API          12.1.0.1.0                     VALID
OLAP Catalog             12.1.0.1.0                     VALID
```

7.6.2.2 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                        12.0
GLOBAL     GLOBAL                         12.0
SYS        AWCREATE                       12.0
SH         SH                             12.0
SYS        AWMD                           12.0
SYS        AWXML                          12.0
SYS        AWREPORT                       12.0
SYS        AWCREATE10G                    12.0
```

> **See Also**
>
> "System Tables" for descriptions of the analytic workspaces owned by SYS.

7.6.2.3 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
```


Copyright © 2003, 2025, Oracle and/or its affiliates.

```
---------- ----------
         0  191776956
```

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
----------
    187282
```

7.6.2.4 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$GLOBAL       20-SEP-12 VALID
SYS        AW$AWCREATE     20-SEP-12 VALID
SYS        AW$AWCREATE10G  20-SEP-12 VALID
SYS        AW$AWMD         20-SEP-12 VALID
SYS        AW$AWREPORT     20-SEP-12 VALID
SYS        AW$AWXML        20-SEP-12 VALID
SYS        AW$EXPRESS      20-SEP-12 VALID

7 rows selected.
```

7.6.3 OLAP DBA Scripts

You can download a file that contains several SQL scripts from the Oracle OLAP website 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 7-6 describes these scripts. For more information, refer to the README file provided with the scripts.

Table 7-6 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_size	Displays the amount of disk space used by each analytic workspace.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Table 7-6 (Cont.) OLAP DBA Scripts

SQL Script	Description
aw_tablespaces	Provides extensive information about the tablespaces used by analytic workspaces.
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_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.

7.6.4 Scripts for Monitoring Performance

Several of the scripts listed in "OLAP DBA Scripts" 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 7-3 shows the information returned by the session_resources script. It lists the use of resources such as cursors, PGA, and UGA.

Example 7-3 Querying Session Resources

```
@session_resources

USERNAME             NAME                                VALUE
-------------------- ------------------------------ ----------
GLOBAL:86            opened cursors cumulative              621
                     opened cursors current                  18
                     session cursor cache count              50
                     session cursor cache hits              432
                     session pga memory                 5356368
                     session pga memory max            10468176
                     session stored procedure space           0
                     session uga memory                 4230692
                     session uga memory max             7049780

9 rows selected.
```

7.6.5 Monitoring Disk Space

Several of the scripts listed in "OLAP DBA Scripts" provide detailed information about the use of disk space by analytic workspaces. Example 7-4 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.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Example 7-4 Querying the Use of Disk Space By Analytic Workspaces

```
@aw_size

Analytic Workspace                          On Disk MB Tablespace
------------------------------------- --------------- --------------------
GLOBAL.GLOBAL                                  249.31 GLOBAL
SYS.AWCREATE                                     7.81 SYSAUX
SYS.AWCREATE10G                                  1.63 SYSAUX
SYS.AWMD                                         8.00 SYSAUX
SYS.AWREPORT                                     1.63 SYSAUX
SYS.AWXML                                       18.00 SYSAUX
SYS.EXPRESS                                      2.25 SYSAUX
                                       ---------------
Total Disk:                                    288.63

7 rows selected.
```

7.7 About Backing Up and Recovering Analytic Workspaces

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 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.

7.8 About Copying Analytic Workspaces

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.

  > **Tip**
  >
  > Verify that the target schema of an import has the OLAP_XS_ADMIN privilege. Otherwise, the analytic workspace will not be created with the necessary permissions.

- **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.

Copyright © 2003, 2025, Oracle and/or its affiliates.

The owner of an analytic workspace can 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

- "Saving and Re-Creating Dimensional Objects with Object Definitions" for information about XML templates
- *Oracle Database Utilities* for information about Oracle Data Pump and the `expdp`/`impdp` commands

7.9 About Saving Dimensional Object Definitions

You can save object definitions in an external file for transferring them to another database or saving a backup copy. You can also save objects definitions to a table to make them available in the Oracle Database. You can save the definitions either in an XML template or in an EIF file. Both files are platform independent.

- About XML Templates
- About EIF Files

7.9.1 About XML Templates

Templates are XML documents that describe dimensional objects. You can save the XML descriptions of all the objects in an analytic workspace or just selected objects, and re-create them later in the same database or in a database on another computer or platform. You can use templates to back up your work while developing a dimensional model of your data or to distribute the design to other users.

You can save the XML definitions of the following types of objects:

- **Analytic workspace**: Saves all dimensional objects and all user-defined OLAP DML programs and objects.
- **Dimension**: Saves the dimension and its levels, hierarchies, attributes, and mappings.
- **Cube**: Saves the cube and its measures, calculated measures, dimensions, mappings, and all user-defined OLAP DML programs and objects associated with the cube.
- **Measure Folder**: Saves a list of the measures in the measure folder. It does not save the objects.

Templates store metadata, not data. You can store templates in a small text file or in a database table. When re-creating objects from a template, you must have access to the source data.

See Also

- "Saving and Re-Creating Dimensional Objects with Object Definitions"

Copyright © 2003, 2025, Oracle and/or its affiliates.

7.9.2 About EIF Files

You can export objects in an analytic workspace to an EIF file. EIF files are specially formatted files for copying analytic workspaces. You can use EIF files to:

- Backup individual analytic workspaces
- Copy an analytic workspace to another database

EIF files are upwardly compatible among releases of Oracle Database. An EIF file saves the definitions of OLAP DML objects and optionally saves the data also. When you create an EIF file, you can save only the data that you have permission to access.

EIF files do not save object security rules.

You can export and import EIF files for analytic workspaces. You can use EIF files at a more granular level, such as saving just your custom programs, using the OLAP DML.

See Also

- "Saving and Re-Creating Dimensional Objects with Object Definitions"

7.10 Cube Materialized Views

A cube materialized view is an Oracle OLAP cube that has been enhanced with the capabilities of a materialized view at build time. Cube materialized views are discussed in the following topics:

- Acquiring Information From the Data Dictionary
- Initiating a Data Refresh
- Refresh Methods
- Using Query Rewrite
- Acquiring Additional Information About Cube Materialized Views

- Acquiring Information From the Data Dictionary
- Initiating a Data Refresh
- Refresh Methods
- Using Query Rewrite
- Acquiring Additional Information About Cube Materialized Views

See Also

"Adding Materialized View Capability to a Cube"

Copyright © 2003, 2025, Oracle and/or its affiliates.

7.10.1 Acquiring Information From the Data Dictionary

The data dictionary contains numerous static views that provide information about materialized views. They list cube materialized views along with all other materialized views.

- Identifying Cube Materialized Views
- Identifying the Refresh Logs

> **See Also**
>
> *Oracle Database Reference* for complete descriptions of the data dictionary views

7.10.1.1 Identifying Cube Materialized Views

`USER_MVIEWS` contains a row for each materialized view owned by the current user. The following query lists the materialized views owned by the `GLOBAL` user. The `CB$` prefix identifies a cube materialized view.

```
SELECT mview_name, refresh_mode "MODE", refresh_method "METHOD",
     last_refresh_date "DATE", staleness FROM user_mviews;

MVIEW_NAME                MODE     METHOD   DATE            STALENESS
------------------------- -------- -------- --------------- ----------
CB$CUSTOMER_MARKET        DEMAND   COMPLETE 20-SEP-12       UNKNOWN
CB$CHANNEL_PRIMARY        DEMAND   COMPLETE 20-SEP-12       UNKNOWN
CB$CUSTOMER_SHIPMENTS     DEMAND   COMPLETE 20-SEP-12       UNKNOWN
CB$PRODUCT_PRIMARY        DEMAND   COMPLETE 20-SEP-12       UNKNOWN
CB$TIME_CALENDAR          DEMAND   COMPLETE 20-SEP-12       UNKNOWN
CB$TIME_FISCAL            DEMAND   COMPLETE 20-SEP-12       UNKNOWN
CB$UNITS_CUBE             DEMAND   FORCE    20-SEP-12       UNKNOWN

7 rows selected.
```

The example shows the cube materialized views defined by Analytic Workspace Manager: One for each dimension hierarchy and one for each cube.

7.10.1.2 Identifying the Refresh Logs

Oracle Database can maintain a set of logs on the master tables for the cube materialized views. These logs support incremental (fast) refresh of the cube. The script generated by the Relational Schema Advisor creates a log for each fact and dimension table to record any changes to the data. The following query lists the materialized view logs owned by the `GLOBAL` user:

```
SELECT master, log_table FROM user_mview_logs;

MASTER                         LOG_TABLE
------------------------------ ------------------------------
CHANNEL_DIM                    MLOG$_CHANNEL_DIM
CUSTOMER_DIM                   MLOG$_CUSTOMER_DIM
PRODUCT_DIM                    MLOG$_PRODUCT_DIM
TIME_DIM                       MLOG$_TIME_DIM
UNITS_FACT                     MLOG$_UNITS_FACT
```


Copyright © 2003, 2025, Oracle and/or its affiliates.

7.10.2 Initiating a Data Refresh

You can initiate a data refresh of a cube materialized view in several different ways using Analytic Workspace Manager or a PL/SQL package:

- **Automatic Refresh**: On the Materialized View tab for a cube, you can create a regular schedule for the materialized view refresh subsystem, as described in "Adding Materialized View Capability to a Cube".
- **Maintenance Wizard**: The Maintenance Wizard is available for refreshing all cubes and dimensions, including cube materialized views.
- **`DBMS_CUBE`**: The `DBMS_CUBE` PL/SQL package is available for refreshing all cubes, cube dimensions, and cube materialized views.
- **`DBMS_MVIEW`**: The `DBMS_MVIEW` PL/SQL package contains several procedures for use with cube materialized views.
- Using DBMS_CUBE
- Using DBMS_MVIEW

7.10.2.1 Using DBMS_CUBE

You can use `DBMS_CUBE` to create and populate an analytic workspace or to maintain any cube, including cube materialized views.

The following command initiates a complete refresh of `UNITS_CUBE`, which is enabled as a cube materialized view. It automatically refreshes any stale dimensions before refreshing the cube.

```
EXECUTE dbms_cube.build('GLOBAL.UNITS_CUBE');
```

You can determine the refresh method from `USER_MVIEWS`, as shown in "Identifying Cube Materialized Views".

7.10.2.2 Using DBMS_MVIEW

You can use `DBMS_MVIEW` to refresh all types of materialized views. These refresh procedures can be used with cube materialized views:

- `REFRESH` refreshes a list of one or more materialized views.
- `REFRESH_ALL_MVIEWS` refreshes all materialized views that meet certain criteria.
- `REFRESH_DEPENDENT` refreshes all materialized views that depend on a particular master table and meet certain criteria.

Dimensions must be refreshed before the cube. An error is raised during refresh of a cube materialized view if any of its associated dimension materialized views are stale. The procedures in `DBMS_MVIEW` can refresh multiple materialized views in one call, but they do not guarantee the refresh order. To control the refresh order, call `DBMS_MVIEW.REFRESH` for the cube materialized view separately from its dimension materialized views.

The following command initiates a refresh of the materialized view for the `CHANNEL_PRIMARY` hierarchy.

```
EXECUTE dbms_mview.refresh('CB$CHANNEL_PRIMARY', 'C');
```


Copyright © 2003, 2025, Oracle and/or its affiliates.

7.10.3 Refresh Methods

In Analytic Workspace Manager, you can specify the COMPLETE, FAST, or FORCE methods for refreshing a cube. Two additional methods, FAST_PCT and FAST_SOLVE, are invoked by the materialized view subsystem. They are not separate choices.

- Refresh Method Descriptions
- Fast Solve Refreshes

7.10.3.1 Refresh Method Descriptions

Table 7-7 describes the refresh methods that are supported on cube materialized views.

Table 7-7 Refresh Methods For Cube Materialized Views

Refresh Method	Description
COMPLETE	Deletes and recreates the cube. This option supports arbitrarily complex mappings from the source tables to the cube.
FAST	Loads and re-aggregates only changed values, based on the materialized view logs or, after direct path loading, on the ALL_SUMDELTA data dictionary view. The source for the refresh is the incremental differences that have been captured in the materialized view logs, rather than the original mapped sources. These differences are used to incrementally rebuild the cube. Only cells that are affected by the changed values are re-aggregated. This option supports only simple mappings for cube materialized views, that is, where no expressions (other than *table.column*), views, or aggregations occur in the query defining the mapping. The materialized view subsystem determines whether to perform a FAST or a FAST_PCT refresh. See *Oracle Database Data Warehousing Guide* for information about the methodology.
FAST_PCT	Loads and re-aggregates data only from changed partitions. This method works best when the source table and the cube are partitioned on the same dimension. FAST_PCT does not use change logs. The materialized view subsystem determines whether to perform a FAST or a FAST_PCT refresh. See for information about the methodology.
FAST_SOLVE	Loads and re-aggregates only changed values, based on the original mapped data source. FAST_SOLVE is a type of refresh only for cube materialized views. It incrementally re-aggregates the cube even when the refresh source is the original mapped source instead of the materialized view logs. The aggregation subsystem identifies the differences and then incrementally re-aggregates the cube. This option is supported for arbitrarily complex mappings from the source tables to the cube. To discover whether a FAST_SOLVE refresh has occurred, review the CUBE_BUILD_LOG table as shown in "Fast Solve Refreshes". Or review the LAST_REFRESH_TYPE column of ALL_MVIEWS; a FAST_SOLVE refresh appears as FAST_CS.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Table 7-7 (Cont.) Refresh Methods For Cube Materialized Views

Refresh Method	Description
FORCE	Loads and re-aggregates values using the best method possible. When a COMPLETE refresh is not necessary, the materialized view system first attempts a FAST refresh. If it cannot FAST refresh a cube materialized view, it performs a FAST_SOLVE refresh.

7.10.3.2 Fast Solve Refreshes

The build log lists the CLEAR LEAVES command when the FAST SOLVE method was used. Example 7-5 shows the rows of CUBE_BUILD_LOG concerned with building UNITS_CUBE.

> **See Also**
>
> "Maintenance Logs"

Example 7-5 Identifying a FAST SOLVE Refresh

```
SELECT build_object, status, command FROM cube_build_log
     WHERE build_object='UNITS_CUBE'
     AND build_id=8;

BUILD_OBJECT STATUS     COMMAND
------------ ---------- ------------------------
UNITS_CUBE   STARTED    COMPILE AGGMAP
UNITS_CUBE   COMPLETED  COMPILE AGGMAP
UNITS_CUBE   STARTED    UPDATE
UNITS_CUBE   COMPLETED  UPDATE
UNITS_CUBE   STARTED    CLEAR LEAVES
UNITS_CUBE   COMPLETED  CLEAR LEAVES
UNITS_CUBE   STARTED    LOAD
UNITS_CUBE   COMPLETED  LOAD
UNITS_CUBE   STARTED    SOLVE
UNITS_CUBE   COMPLETED  SOLVE
UNITS_CUBE   STARTED    UPDATE
UNITS_CUBE   COMPLETED  UPDATE
UNITS_CUBE   STARTED    ANALYZE
UNITS_CUBE   COMPLETED  ANALYZE

14 rows selected.
```

7.10.4 Using Query Rewrite

Query rewrite changes a query to select data from the materialized views instead of calculating the result set from the master tables. The transformation is fully transparent to the client, and requires no mention of the materialized views in the SQL statement. In the case of cube materialized views, the query is written against the tables or views of a star or snowflake schema, and it is transformed into a query against a cube materialized view. This transformation can result in significant improvements in run-time performance.

Query rewrite requires optimizer statistics on the cubes and dimensions. You can discover whether a query is rewritten by generating and examining its execution plan.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Oracle Database uses two initialization parameters to control query rewrite:

- `QUERY_REWRITE_ENABLED`: Enables or disables query rewrite globally for the database.
- `QUERY_REWRITE_INTEGRITY`: Determines the degree to which query rewrite monitors the consistency of materialized views with the source data. The `trusted` or `stale tolerated` settings are recommended when using rewrite to cube materialized views.

Administration of cube materialized views is the same as any other materialized view except that the cube materialized views must be in the same schema as the analytic workspace. Users require the `GLOBAL QUERY REWRITE` privilege to have rewrite to materialized views that are in schemas other than their own. However, the owner can access the materialized views from any schema without additional privileges.

See Also

- "Analyzing Cubes and Dimensions" for information about optimizer statistics
- Viewing Execution Plans for information about execution plans
- *Oracle Database Reference* for complete descriptions of the initialization parameters

7.10.5 Acquiring Additional Information About Cube Materialized Views

Oracle Database has numerous PL/SQL packages for managing materialized views. Cube materialized views are optimized to provide the best performance, so you have no need to use most of these packages. Few design decisions remain for you to make. For this reason, the `TUNE_MVIEW` procedure of `DBMS_ADVISOR` is disabled for cube materialized views.

However, there are a few packages that you may find useful, as shown in Table 7-8.

Table 7-8 PL/SQL Packages for Cube Materialized Views

Package	Description
`DBMS_METADATA`	Returns the metadata for an object.
`DBMS_MVIEW`	Executes data refreshes. See "Initiating a Data Refresh". You can use the `EXPLAIN_REWRITE` and `EXPLAIN_MVIEW` procedures to obtain information about cube materialized views. `EXPLAIN_MVIEW` is particular useful for evaluating and explaining the `FAST` refresh capabilities of a cube.
`DBMS_XPLAN`	Displays an execution plan. See "Viewing Execution Plans".

Copyright © 2003, 2025, Oracle and/or its affiliates.

8
Security

Oracle OLAP secures your data using the standard security mechanisms of Oracle Database.

This chapter contains the following topics:

- Security of Multidimensional Data in Oracle Database
- Setting Object Security
- Creating Data Security Policies on Dimensions and Cubes
- Creating OLAP Data Security Roles

- Security of Multidimensional Data in Oracle Database
- Setting Object Security
- Creating Data Security Policies on Dimensions and Cubes
- Creating OLAP Data Security Roles

8.1 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
- Types of Security
- About the Privileges
- Layered Security

8.1.1 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

Copyright © 2003, 2025, Oracle and/or its affiliates.

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.

8.1.2 Types of Security

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.

Oracle OLAP provides two types of security: Object security and data security.

- **Object security** provides access to dimensional objects. You must set object security before other users can access them. Object security is implemented using SQL `GRANT` and `REVOKE`.
- **Data security** provides fine-grained control of the data on a cellular level. This type of security is optional. You must define data security policies only when you want to restrict access to specific areas of a cube. Data security is implemented using Oracle Real Application Security.

> **Note**
>
> Only the owner of a schema can create data security policies and OLAP data security roles. The data security policies and OLAP data security roles apply only to objects in the schema.

You can administer both data security and object security in Analytic Workspace Manager. For object security, you also have the option of using SQL `GRANT` and `REVOKE`.

8.1.3 About the Privileges

Using both object security and data security, you can grant and revoke the following privileges:

- **Alter**: Change the definition of a cube or dimension. Users need this privilege to create and modify a dimensional model.
- **Delete**: Remove old dimension members. Users need this privilege to refresh a dimension.
- **Insert**: Add new dimension members. Users need this privilege to refresh a dimension.
- **Select**: Query the cube or dimension. Users need this privilege to query a view of the cube or dimension or to use the `CUBE_TABLE` function. `CUBE_TABLE` is a SQL function that returns the values of a dimensional object.
- **Update**: Change the data values of a cube or the name of a dimension member. Users need this privilege to refresh a dimension or cube.

Users exercise these privileges either by using Analytic Workspace Manager to create and administer dimensional objects, or by using SQL to query them. They do not issue commands such as SQL `INSERT` and `UPDATE` directly on the cubes and dimensions.

8.1.4 Layered Security

For dimensional objects, you can manage security at these levels:

- Dimension member

Copyright © 2003, 2025, Oracle and/or its affiliates.

- Dimension
- Cube
- Analytic workspace
- View
- Materialized view

The privileges are layered so that, for example, a user with `SELECT` data security on Software products must also have `SELECT` object security on the `PRODUCT` dimension and the Global analytic workspace. Users also need `SELECT` privileges on the views of the dimensional objects.

You administer security on views and materialized views for dimensional objects the same way as for any other views and materialized views in the database.

8.2 Setting Object Security

You can use either SQL or Analytic Workspace Manager to set object security. The results are identical. These following topics describe these methods.

- Using SQL to Set Object Security
- Using Analytic Workspace Manager to Set Object Security
- Using SQL to Set Object Security
- Using Analytic Workspace Manager to Set Object Security

8.2.1 Using SQL to Set Object Security

You can set and revoke object privileges on dimensional objects using the SQL `GRANT` and `REVOKE` commands.

- Setting Object Security on an Analytic Workspace
- Setting Object Security on Dimensions
- Setting Object Security on Cubes

8.2.1.1 Setting Object Security on an Analytic Workspace

Object privileges on an analytic workspace simply open the container. You must grant object privileges on the cubes and dimensions for users to be able to access them. The table name is the same as the analytic workspace name, with the addition of an `AW$` prefix.

The following command enables Scott to attach the Global analytic workspace, `AW$GLOBAL`, to a session:

```
GRANT SELECT ON aw$global TO scott;
```

8.2.1.2 Setting Object Security on Dimensions

You can grant privileges on individual dimensions to enable users to query the dimension members and attributes. For users to query a cube, they must have privileges on every dimension of the cube.

The privileges apply to the entire dimension. However, you can set fine-grained access on a dimension to restrict the privileges, as described in "Creating Data Security Policies on Dimensions and Cubes".

Copyright © 2003, 2025, Oracle and/or its affiliates.

Example 8-1 shows the SQL commands that enable Scott to query the Product dimension. They give Scott SELECT privileges on the Product dimension, on the Global analytic workspace, and on the Product view.

Example 8-1 Privileges to Query the Product Dimension

```
GRANT SELECT ON product TO scott;
GRANT SELECT ON aw$global TO scott;
GRANT SELECT ON product_view TO scott;
```

8.2.1.3 Setting Object Security on Cubes

Privileges on cubes enable users to access business measures and perform analysis. You must also grant privileges on each of the dimensions of the cube.

The privileges apply to the entire cube. However, you can create a data security policy on the cube or on its dimensions to restrict the privileges, as described in "Creating Data Security Policies on Dimensions and Cubes".

Example 8-2 Privileges to Query the Units Cube

This example shows the SQL commands that enable Scott to query the Units cube. They give Scott SELECT privileges on the Global analytic workspace, the cube, and all of its dimensions. Scott also gets privileges on the dimension views so that he can query the dimension attributes for formatted reports.

```
/* Grant privileges on the analytic workspace */
GRANT SELECT ON global.aw$global TO scott;

/* Grant privileges on the cube */
GRANT SELECT ON global.units_cube TO scott;

/* Grant privileges on the dimensions */
GRANT SELECT ON global.channel TO scott;
GRANT SELECT ON global.customer TO scott;
GRANT SELECT ON global.product TO scott;
GRANT SELECT ON global.time TO scott;

/* Grant privileges on the cube, dimension, and hierarchy views */
GRANT SELECT ON global.units_cube_view TO scott;
GRANT SELECT ON global.channel_view TO scott;
GRANT SELECT ON global.channel_primary_view TO scott;
GRANT SELECT ON global.customer_view TO scott;
GRANT SELECT ON global.customer_shipments_view TO scott;
GRANT SELECT ON global.customer_segments_view TO scott;
GRANT SELECT ON global.product_view TO scott;
GRANT SELECT ON global.product_primary_view TO scott;
GRANT SELECT ON global.time_view TO scott;
GRANT SELECT ON global.time_calendar_view TO scott;
GRANT SELECT ON global.time_fiscal_view TO scott;
```

Example 8-3 Privileges to Use Cube Materialized Views for Query Rewrite

This example shows the SQL commands that give SCOTT the privileges to query the relational tables for the detail level data and to use query rewrite to obtain summary data from the Units cube.

```
/* Grant privileges on materialized views using query rewrite */
GRANT GLOBAL QUERY REWRITE TO scott;

/* Grant privileges on the relational source tables */
GRANT SELECT ON global.channel_dim TO scott;
```


Copyright © 2003, 2025, Oracle and/or its affiliates.

```
GRANT SELECT ON global.customer_dim TO scott;
GRANT SELECT ON global.product_dim TO scott;
GRANT SELECT ON global.time_dim TO scott;
GRANT SELECT ON global.units_fact TO scott;

/* Grant privileges on the analytic workspace */
GRANT SELECT ON global.aw$global TO scott;

/* Grant privileges on the cube */
GRANT SELECT ON global.units_cube TO scott;

/* Grant privileges on the dimensions */
GRANT SELECT ON global.channel TO scott;
GRANT SELECT ON global.customer TO scott;
GRANT SELECT ON global.product TO scott;
GRANT SELECT ON global.time TO scott;
```

Example 8-4 Privileges to Modify and Refresh GLOBAL

This example shows the SQL commands that give `SCOTT` the privileges to modify and update all dimensional objects in `GLOBAL` using Analytic Workspace Manager.

> **Note**
>
> The `GRANT ALL` commands encompass more privileges than those discussed in Security. Be sure to review the list of privileges before using `GRANT ALL`.

```
/* Grant privilege to use Analytic Workspace Manager */
GRANT OLAP_USER TO scott;

/* Grant privileges on the analytic workspace */
GRANT ALL ON global.aw$global TO scott;

/* Grant privileges on the cubes */
GRANT ALL ON global.units_cube TO scott;
GRANT ALL ON global.price_cost_cube TO scott;

/* Grant privileges on the dimensions */
GRANT ALL ON global.channel TO scott;
GRANT ALL ON global.customer TO scott;
GRANT ALL ON global.product TO scott;
GRANT ALL ON global.time TO scott;
```

8.2.2 Using Analytic Workspace Manager to Set Object Security

Analytic Workspace Manager provides a graphical interface for setting object security. It also displays the SQL commands, so that you can cut-and-paste them into a script.

- Setting Object Security on an Analytic Workspace
- Setting Object Security on Dimensions
- Setting Object Security on Cubes

8.2.2.1 Setting Object Security on an Analytic Workspace

Take these steps to set object security on an analytic workspace in Analytic Workspace Manager:

Copyright © 2003, 2025, Oracle and/or its affiliates.

1. In the navigation tree, right-click the analytic workspace and select **Set Analytic Workspace Object Security**.

   The Set Analytic Workspace Object Security dialog box appears.

2. Complete the dialog box, then click **OK**.

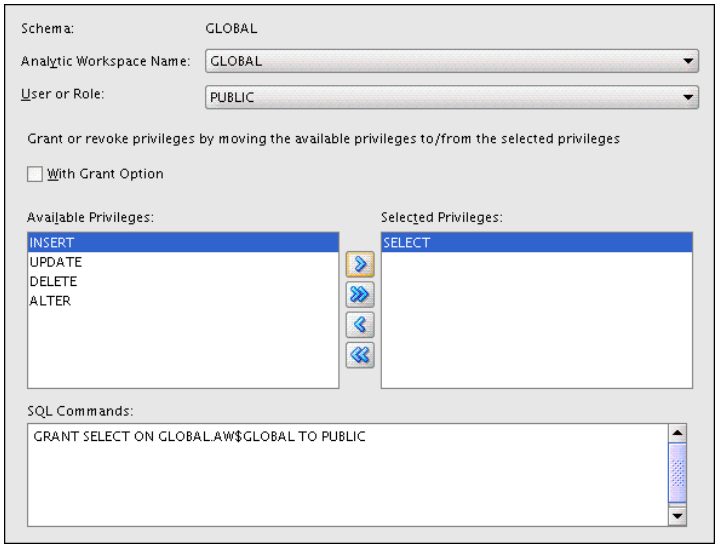
   Click **Help** for specific information about the choices.

3. Grant privileges on one or more cubes and their dimensions.

   Privileges on the analytic workspace do not automatically extend to the cubes and dimensions contained in the analytic workspace.

Figure 8-1 shows the `SELECT` privilege on `GLOBAL` granted to `PUBLIC`.

Figure 8-1 Setting Object Security on GLOBAL



8.2.2.2 Setting Object Security on Dimensions

Take these steps to set object security on dimensions in Analytic Workspace Manager:

1. In the navigation tree, right-click any dimension and select **Set Dimension Object Security**.

   The Set Dimension Object Security dialog box appears.

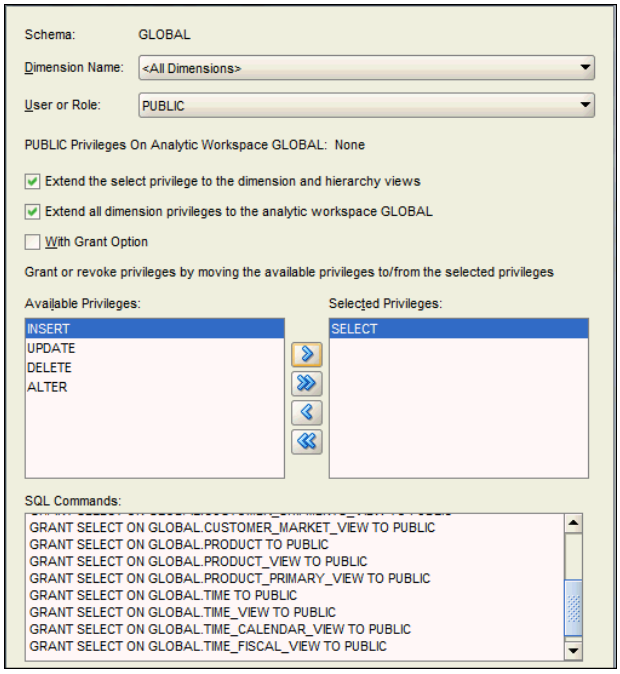
2. Complete the dialog box, then click **OK**.

   You can set privileges on all of the dimensions simultaneously. You can extend the privileges to the dimension and hierarchy views and to the analytic workspace. Click **Help** for specific information about the choices.

Figure 8-2 shows the `SELECT` privilege on all dimensions granted to `PUBLIC`.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Figure 8-2 Setting Object Security on Dimensions



8.2.2.3 Setting Object Security on Cubes

Before setting object security on a cube, set object security on the dimensions that the cube uses. Take these steps to set object security on cubes in Analytic Workspace Manager:

1. In the navigation tree, right-click any cube and select **Set Cube Object Security**.

   The Set Cube Object Security dialog box appears.

2. Complete the dialog box, then click **OK**.

   You can set privileges on all of the cubes simultaneously. You can extend the privileges to the cube views and to the analytic workspace. Click **Help** for specific information about the choices.

8.3 Creating Data Security Policies on Dimensions and Cubes

Data security policies enable you to grant database users and roles privileges on a selection of dimension members. For example, you might restrict district sales managers to the data for just their own districts instead of all geographic areas. You can create a data security policy on dimensions, cubes, or both:

- Only the owner of a schema can create data security policies for dimensions and cubes in the schema.
- When you create a data security policy on a dimension, the policy extends to all cubes with that dimension. You do not need to re-create the policy for each cube.
- When you create a data security policy on a cube, you select the members for each dimension of the cube. The policy only applies to that cube.
- When you create data security policies on both dimensions and cubes, users have privileges on the most narrowly defined portion of the data, where the policies overlap.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Granting Data Privileges

You can apply a policy to one or more database users and roles. You can also apply a policy to an OLAP data security role. An OLAP data security role is a group of database users and roles that you can manage in Analytic Workspace Manager just for use in security policies. You create OLAP data security roles and data security policies in Analytic Workspace Manager.

Selecting Data By Criteria

When defining a data security policy, you can select specific dimension members or those that meet certain criteria based on the dimension hierarchy. By using criteria instead of hard-coding specific dimension members, the selection remains valid after a data refresh. You do not need to modify the selection after adding members. For example, a security policy that grants `SELECT` privileges to all Hardware products remains valid when old products are rolled off and new products are added to the `PRODUCT` dimension.

> **Note**
>
> You must have the `OLAP_XS_ADMIN` role to manage data security policies in Analytic Workspace Manager.

To create a data security policy:

1. Expand the folder for a dimension or a cube.
2. Right-click Data Security and select **Create Data Security Policy**.

   The Create Data Security Policy dialog box appears.
3. On the General tab, enter a descriptive name in the Data Security Policy Name field.
4. *Optional*: Enter a description in the Description field.
5. For a dimension, select the method you want to use to select the viewable dimension members, either **Member Selection** or **OLAP DML Expression**. The related tab becomes active.

   For a cube, the method is Member Selection.
6. Click **Add Users or Roles**.

   The Add Users or Roles dialog box appears.
7. Select the database users and roles and the OLAP data security role to use this policy. Then click **OK** to close the dialog box.

   The selected database users and roles and OLAP data security role are now listed in the table on the General tab.
8. Select the permissions you want to grant to each user or role. You cannot assign permissions to the OLAP data security role because the permissions are part of its definition.
9. For a cube, complete the Member Selection tab.

   For a dimension, complete the Member Selection tab or the OLAP DML Expression tab, depending on the previously selected method.
10. Click **Create** to save the data security policy.

Copyright © 2003, 2025, Oracle and/or its affiliates.

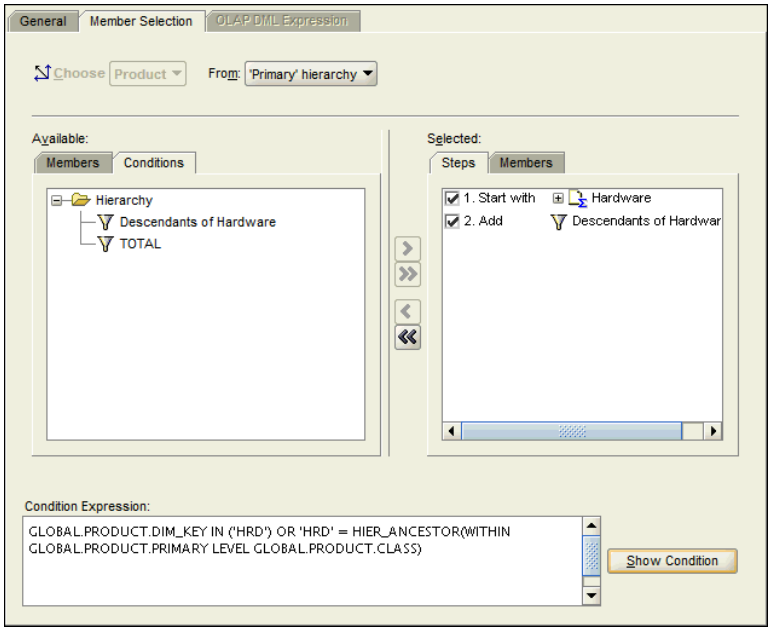
The data security policy appears in the navigation tree in the Data Security folder for the dimension or cube.

11. Grant these users and roles object privileges on the dimension or cube, and on the analytic workspace.

> **See Also**
>
> - "Setting Object Security on an Analytic Workspace"
> - "Setting Object Security on Dimensions "
> - "Setting Object Security on Cubes "
> - "Creating OLAP Data Security Roles"

Figure 8-3 shows the Member Selection tab of the data security policy for PRODUCT. Users who have privileges on the PRODUCT dimension based on this policy have access to all Hardware products. They do not have access to Software products or Total Product.

Figure 8-3 Restricting Product to Hardware and Descendants



Disabling and Enabling Data Security

When you create a data security policy, Oracle OLAP enables data security for the dimension or cube. You can disable data security for a dimension or a cube. You can then enable data security for the object again.

To disable or enable data security:

1. Expand the folder for a dimension or a cube.
2. Right-click Data Security and select **Disable Data for *Object*** or **Enable Data for *Object***.

   The Disable Confirmation dialog box or the Enable Confirmation dialog box appears.

Copyright © 2003, 2025, Oracle and/or its affiliates.

3. Click **Yes**.

8.4 Creating OLAP Data Security Roles

You can create OLAP data security roles to manage a group of users to whom you want to assign the same data access permissions. You can then use the data security role when managing your data security policies, instead of defining the privileges of each individual user. OLAP data security roles are like database roles except they only function within the context of OLAP data security, and they can be created by a user with less powerful database privileges. Only the owner of a schema can create data security roles in the schema.

> **Note**
>
> You must have the `OLAP_XS_ADMIN` privilege to manage data security policies in Analytic Workspace Manager.

To create an OLAP data security role:

1. In the navigation tree, right-click Data Security Roles and then select **Create Data Security Role**.

   The Create Data Security Role dialog box appears.
2. On the General tab, enter a descriptive name in the Data Security Role Name field.
3. *Optional*: Enter a description in the Description field.
4. Click **Add Users or Roles**.

   The Add Users or Roles dialog box appears.
5. Select the users and roles that you want to include in this OLAP data security role. Then click **OK** to close this dialog box.

   The selected users and roles are now listed in the table on the General tab.
6. Select the permissions you want to grant to each user or role.
7. Click **Create** to save the OLAP data security role.

   The new OLAP data security role appears in the navigation tree in the Data Security Roles folder.

> **See Also**
>
> - "Creating Data Security Policies on Dimensions and Cubes"

Copyright © 2003, 2025, Oracle and/or its affiliates.

9

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 can return 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 Operators
- When Does Aggregation Order Matter?
- Example: Aggregating the Units Cube

- What Is Aggregation?
- Aggregation Operators
- When Does Aggregation Order Matter?
- Example: Aggregating the Units Cube

9.1 What Is Aggregation?

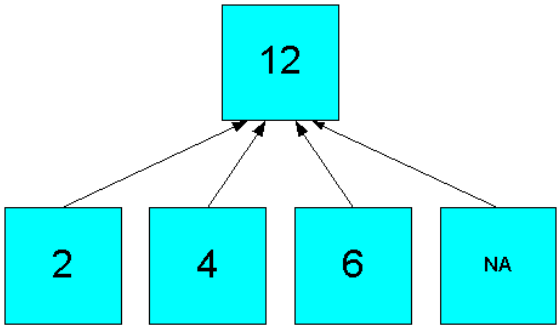
Aggregation is the process of consolidating multiple values into a single value. For example, data can be collected on a daily basis and aggregated into a value for the week, the weekly data can be aggregated into a value for the month, and so on. Aggregation allows patterns in the data to emerge, and 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" .

Figure 9-1 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.

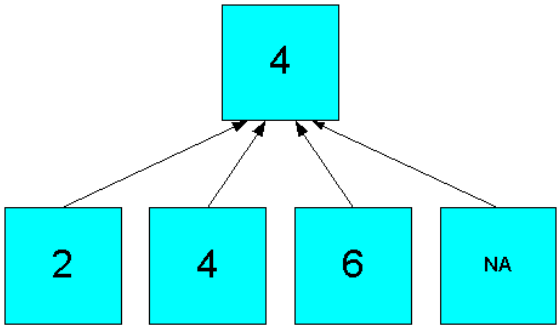
Copyright © 2003, 2025, Oracle and/or its affiliates.

Figure 9-1 Summary Aggregation in a Simple Hierarchy



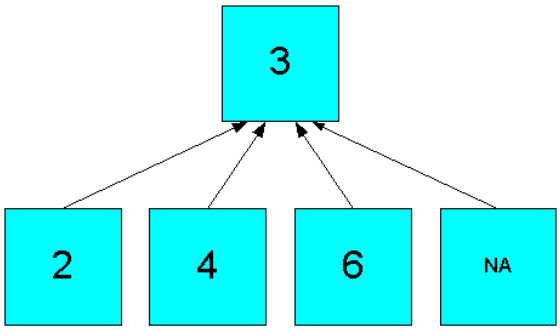
The Average operator calculates the average of all real data, producing an aggregate value of ((2 + 4 + 6)/3)=4, as shown in Figure 9-2.

Figure 9-2 Average Aggregation in a Simple Hierarchy



The hierarchical operators include null values in the count of cells. In Figure 9-3, the Hierarchical Average operator produces an aggregate value of ((2 + 4 + 6 +NA)/4)=3.

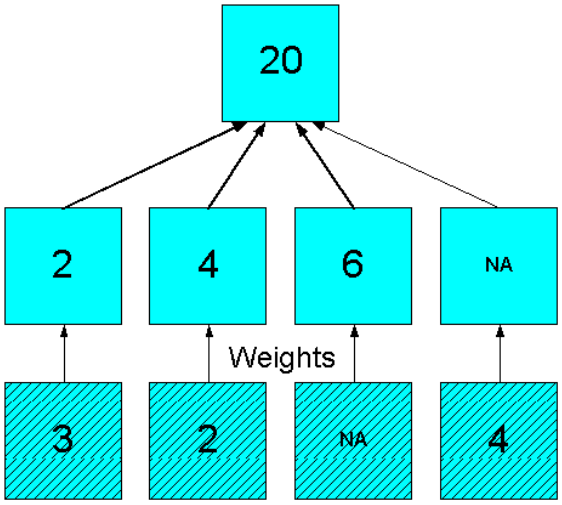
Figure 9-3 Hierarchical Average Aggregation in a Simple Hierarchy



The weighted operators use the values in another measure to generate weighted values before performing the aggregation. Figure 9-4 shows how the simple sum of 12 in Figure 9-1 changes to 20 by using weights ((3*2) + (2*4) + (NA*6) +(4*NA)).

Copyright © 2003, 2025, Oracle and/or its affiliates.

Figure 9-4 Weighted Sum Aggregation in a Simple Hierarchy



9.2 Aggregation Operators

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

- Basic Operators
- Scaled and Weighted Operators
- Hierarchical Operators

9.2.1 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.
- **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.

9.2.2 Scaled and Weighted 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?".

These are the scaled and weighted aggregation operators:

Copyright © 2003, 2025, Oracle and/or its affiliates.

- **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.

9.2.3 Hierarchical Operators

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

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 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.

9.3 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 incompressible 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.

The following topics describe the ordering of aggregation operators.

Copyright © 2003, 2025, Oracle and/or its affiliates.

- Using the Same Operator for All Dimensions of a Cube
- Example: Mixing Aggregation Operators
- Using the Same Operator for All Dimensions of a Cube
- Example: Mixing Aggregation Operators

9.3.1 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
- Order Changes the Aggregation Results
- Order May Be Important

9.3.1.1 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

9.3.1.2 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

9.3.1.3 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

Copyright © 2003, 2025, Oracle and/or its affiliates.

- Hierarchical Weighted Average

9.3.2 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 9-5 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 9-5 Sum Method Followed by Maximum Method

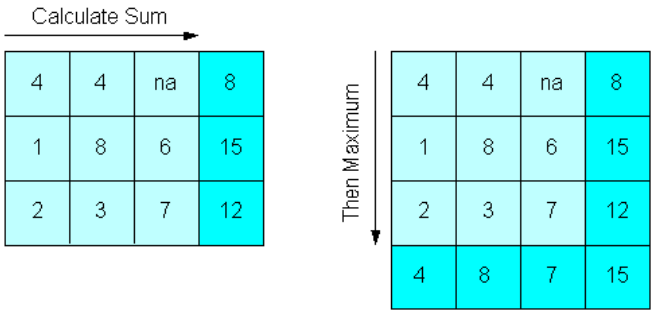
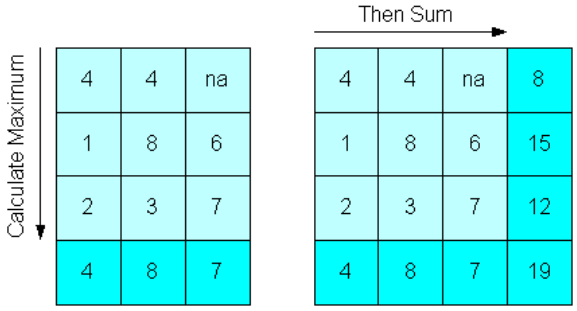


Figure 9-6 shows the same cube, except Maximum is calculated first down one dimension of the cube, then Sum is calculated across the other dimension. The maximum value of the sums in Figure 9-5 is 15, while the sum of the maximum values in Figure 9-6 is 19.

Figure 9-6 Max Method Followed by Sum Method



9.4 Example: Aggregating the Units Cube

This example describes changes to the default aggregation of the Units cube in the `GLOBAL` analytic workspace. These changes take effect in the next data refresh.

- Selecting the Aggregation Operators and Hierarchies
- Choosing the Percentage of Precomputed Values

Copyright © 2003, 2025, Oracle and/or its affiliates.

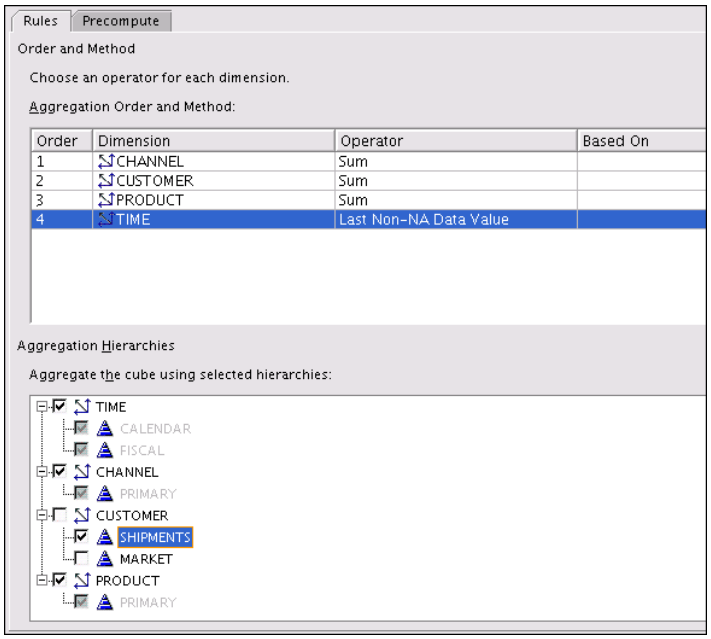
9.4.1 Selecting the Aggregation Operators and Hierarchies

Analytic Workspace Manager initially sets all dimensions to use the Sum operator and aggregates all levels of all dimensions. To change these default settings, use the Rules subtab of the Aggregation tab.

Figure 9-7 shows the operators for the Units Cube. Time is now set to Last Non-NA Data Value, and it is aggregated after the other dimensions. For operators like First and Last, the order in which the dimensions are aggregated can change the results.

Another change is that only the Shipments hierarchy of the Customer dimension is aggregated during data maintenance. Because the Market hierarchy is seldom queried, to save maintenance time and storage space the Global DBA chose not to calculate those aggregate values. However, response time is slower for queries that request Market aggregations.

Figure 9-7 Selecting the Aggregation Operators



9.4.2 Choosing the Percentage of Precomputed Values

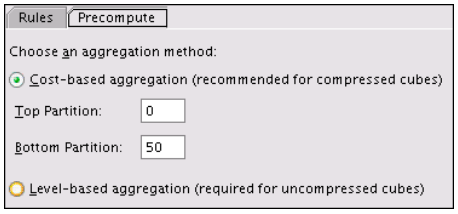
Analytic Workspace Manager initially chooses cost-based aggregation with 35% precomputed values for the bottom partitions and 0% for the top partition. An unpartitioned cube is also set to 35%. This setting means that 35% of the aggregate values is calculated and stored during data maintenance, and 65% is calculated in response to a query. These settings optimize data maintenance.

Increasing the materialization of the bottom partitions improves querying of both the bottom and the top partitions. Increasing the materialization of the top partition improves querying of the most aggregate data and any other hierarchies of the partitioned dimension.

Figure 9-8 shows the settings for the Units Cube. In this case, the Global DBA chose to keep the top partition at 0%, and to increase the bottom partitions from 35 to 50%. This change increases maintenance costs in time and storage space, but improves run-time performance of all partitions.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Figure 9-8 Setting Cost-Based Presummarization



Copyright © 2003, 2025, Oracle and/or its affiliates.

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
- Identifying Required Business Facts
- Designing a Dimensional Model for Global Computing

- Case Study Scenario
- Identifying Required Business Facts
- Designing a Dimensional Model for Global Computing

A.1 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 must 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 regarding the delayed response to report requests. Complaints have also been numerous in the IT department regarding 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 is no longer useful. I am

Copyright © 2003, 2025, Oracle and/or its affiliates.

only able to get information at the end of each month, and it does not have the details I need to do my job."

- Reporting Requirements
- Business Goals
- Information Requirements

A.1.1 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 email sales on a monthly and quarterly basis, and a comparison to past time periods. Information must identify when, how, and what is being sold by each channel.
- Margin information on products 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 concluded 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.

A.1.2 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.

A.1.3 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,

Copyright © 2003, 2025, Oracle and/or its affiliates.

you can determine how they look at the business, and what types of business analysis questions they want to answer.

- Business Analysis Questions
- Summary of Information Requirements

A.1.3.1 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?

Copyright © 2003, 2025, Oracle and/or its affiliates.

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?

A.1.3.2 Summary of Information Requirements

By examining the types of analyses that users want 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 must understand profitability trends.
- Global Computing must 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.

Copyright © 2003, 2025, Oracle and/or its affiliates.

- The ability to perform trend analysis is important to Global Computing.

A.2 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

A.3 Designing a Dimensional Model for Global Computing

"Business Goals" identifies the business facts that support analysis requirements at Global Computing. Next, we identify the dimensions, levels, and attributes in a data model. We 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
- Identifying Levels
- Identifying Hierarchies
- Identifying Stored Measures

A.3.1 Identifying Dimensions

Four dimensions are used to organize the facts in the database:

- Product shows how data varies by product.

Copyright © 2003, 2025, Oracle and/or its affiliates.

- 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.

A.3.2 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, the levels are Total and Channel.
- Global performs customer and geographic analysis along the line of shipments to customers and by market segmentation. Shipments and Market 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. The dimension has two hierarchies: Calendar and Fiscal.

All dimensions have a Total level as the highest level of summarization. Adding this highest level provides additional flexibility as application users analyze data.

A.3.3 Identifying Hierarchies

We can 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.

A.3.4 Identifying Stored Measures

"Identifying Required Business Facts" 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

Copyright © 2003, 2025, Oracle and/or its affiliates.

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.

Copyright © 2003, 2025, Oracle and/or its affiliates.

B

Keyboard Shortcuts

Keyboard shortcuts support accessibility in Analytic Workspace Manager. Most shortcuts work on all platforms, but Windows provides the most reliable results for all of them. If you use keyboard shortcuts for accessibility, then install Analytic Workspace Manager on a Windows platform.

The keyboard shortcuts are active within particular areas of the user interface:

- Menu Bar
- Navigation Tree
- Property Sheets
- Shuttle Keys
- Mapping Canvas

- Menu Bar
- Navigation Tree
- Property Sheets
- Shuttle Keys
- Mapping Canvas

B.1 Menu Bar

File menu: Alt+F

Tools menu: Alt+T

Help menu: Alt+H

B.2 Navigation Tree

To display a menu for the selected object, press Shift+F10. This is equivalent to clicking the right mouse button.

To close the menu for a selected object, press Esc.

To expand a folder, press the Right Arrow key.

To collapse a folder, press the Left Arrow key.

To move the cursor down the tree, press the Down Arrow key.

To move the cursor up the tree, press the Up Arrow key.

To move the cursor from the navigation tree to a property sheet, press Tab.

To move the cursor from a property sheet to the navigation tree, press Shift+Tab.

Copyright © 2003, 2025, Oracle and/or its affiliates.

B.3 Property Sheets

To move the cursor from the navigation tree to a property sheet, press Tab.

To move the cursor to the next tab, press the Right Arrow key.

To move the cursor to the previous tab, press the Left Arrow key.

To move the cursor from a property sheet to the navigation tree, press Shift+Tab.

To move the splitter between the navigation tree and a property sheet, press F8 Right Arrow or Left Arrow.

To change a menu choice in a table, press F2 Down Arrow.

B.4 Shuttle Keys

Move all: Alt+L

Move selected: Alt+D

Remove selected: Alt+R

Remove all: Alt+O

To select multiple items, press Ctrl+Arrow, then press the spacebar.

B.5 Mapping Canvas

Table mapping view: Ctrl+T

Graphical mapping view: Ctrl+G

Automatically arrange mappings: Ctrl+Alt+K

Automatically map star schema: Ctrl+M

Remove all mappings: Ctrl+D

Schema Viewer Navigator:

All keyboard shortcuts for the navigation tree are available, plus the following additions for the table view:

To copy the name of the selected column from the tree: Ctrl+C. To paste a column name into the selected field: Ctrl+V.

To expand the width of a column: Select the header and press Alt+Right Arrow.

To reduce the width of a column: Select the header and press Alt+Left Arrow.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Glossary

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`.

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, cube dimension.

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

Copyright © 2003, 2025, Oracle and/or its affiliates.

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.

See also cube dimension.

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.

See also cube dimension.

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.

Copyright © 2003, 2025, Oracle and/or its affiliates.

compressed cube

A cube with very sparse data that is stored in a compressed composite.

See also composite.

compression

See compressed cube.

consistent solve specification

See solve specification.

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.

cube dimension

A cube dimension is a dimensional object that stores a list of values. 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 the context of multidimensional analysis, a cube dimension is called a dimension.

See also dimension.

cube materialized view

A cube that has been enhanced with materialized view capabilities. A cube materialized view can be incrementally refreshed through the Oracle Database materialized view subsystem, and it can serve as a target for transparent rewrite of queries against the source tables.

Also called a cube-organized materialized view.

cube script

A sequence of steps that prepare the data for querying, such as loading and aggregating data.

cube view

A relational view of the data stored in a cube, which can be queried by SQL. It contains columns for the dimensions, measures, and calculated measures of the cube.

Copyright © 2003, 2025, Oracle and/or its affiliates.

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 security role

A group of users and database roles that is defined just for use in managing OLAP security policies.

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.

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.

Copyright © 2003, 2025, Oracle and/or its affiliates.

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.

In a relational table, a dimension is a type of object that defines hierarchical (parent-child) relationships between pairs of column sets.

See also cube dimension, hierarchy, measure dimension.

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 a cube dimension that provides information about all members of all hierarchies. It includes columns for the dimension keys, level, and attributes.

See also cube dimension, hierarchy view.

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.

Copyright © 2003, 2025, Oracle and/or its affiliates.

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.

hierarchy view

A relational view of a cube dimension that provides information about the members that belong to a particular hierarchy. It includes columns for the dimension keys, parents, levels of the hierarchy, and attributes.

Copyright © 2003, 2025, Oracle and/or its affiliates.

See also cube dimension, dimension view.

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.

See also cube materialized view.

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.

Copyright © 2003, 2025, Oracle and/or its affiliates.

See also calculated measure, fact, cube.

measure dimension

A dimension that has measures as dimension members.

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.

model

A set of interrelated 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.

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

Copyright © 2003, 2025, Oracle and/or its affiliates.

Contrast with OLTP.

OLAP DML

A set of commands, functions, and options used to manage dimensional data stored in analytic workspaces within Oracle Database.

Analytic Workspace Manager, the OLAP expression syntax, the OLAP Java API, and various applications and PL/SQL packages enable users to access dimensional data without using the OLAP DML directly, but those tools use the OLAP DML to accomplish the desired tasks.

The OLAP Data Manipulation Language (DML) operates exclusively within analytic workspaces, whose primary data structures are dimensions, variables, formulas, relations, and valuesets. These dimensional objects in analytic workspaces support the high-level dimensional objects in the database, such as cubes, cube dimensions, measures, attributes, and hierarchies.

Contrast with OLAP expression syntax.

OLAP expression syntax

An extension of the SQL syntax that is used to manipulate the data stored in dimensional database objects such as cubes, cube dimensions, attributes, and measures.

Contrast with OLAP DML.

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 as needed in response to a specific query. In a cube, calculated measures and custom members are typically calculated as needed. Aggregate data can be precomputed, calculated as needed, or a combination of the two methods.

Contrast with precompute.

override solve specification

See solve specification.

page

A unit for swapping data in and out of memory.

Copyright © 2003, 2025, Oracle and/or its affiliates.

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 as needed, 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.

security role

See data security role.

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,

Copyright © 2003, 2025, Oracle and/or its affiliates.

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.

solve specification

The aggregation method for each dimension of the cube.

solved data

A result set in which all derived data has been calculated. Data fetched from a 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 as needed.

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.

Copyright © 2003, 2025, Oracle and/or its affiliates.

star query

A join between a fact table and several 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 cube dimension, dimension member.

summary

See aggregation.

update window

The length of time available for loading 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 first-, second-, and third-level supervisors and so forth may not be meaningful for analysis.

See also hierarchy, level-based hierarchy.

Copyright © 2003, 2025, Oracle and/or its affiliates.

Index

A

ADVISOR privilege, *2*
aggregate functions, *21*
aggregation
  average operator, *2*
  calculated measures, *16*
  definition, *1*
  hierarchical average operator, *2*
  over attributes, *15*
  sum operator, *1*
  weighted operators, *2*
aggregation operators, *18*, *15*, *3*
aggregation order, *4*
aggregation percentages, *7*
aggregation step (cube scripts), *33*
ALL_AW_OBJ view, *4*
ALL_AW_PROP view, *4*
ALL_AW_PS view, *4*
analysis tools, *3*
analytic functions, *2*, *16*
Analytic Workspace Manager
  configuring, *3*, *5*, *26*, *27*
  installing, *3*
  opening, *3*
  using, *2*
analytic workspace security, *3*, *5*
analytic workspaces
  backing up and recovering, *16*
  creating, *3*
  database storage, *6*
  disk space consumption, *15*
  enhancing functionality, *4*
  identifying owners, *13*
  listing, *13*
  saving and re-creating, *40*
  size, *13*
analyze step (cube scripts), *33*
Application Express, *3*, *12*
arithmetic operations, *2*
attachment modes
  configuring, *5*
  selecting, *3*
  showing, *5*
attribute aggregation, *15*
attributes
  creating, *10*
  defined, *8*, *9*
authentication, *2*
Automatic Database Diagnostic Monitor, *11*
Automatic Storage Management, *3*
Automatic Workload Repository, *11*
average
  cumulative, *13*
  moving, *12*
average operator (aggregation), *2*
average rank, *10*
AVERAGE_RANK function, *16*
AVG function, *21*

B

backup and recovery, *16*
backup options, *16*
batch processing, *8*
BI Publisher, *3*
BI Suite, *6*
bind variables, *1*, *11*, *18*, *20*
branches (Application Express), *17*
build logs, *15*
BusinessObjects Enterprise, *6*

C

calculated measures
  and measure dimensions, *7*
  copying and pasting, *42*
  creating, *3*
  defined, *1*
  generator, *3*
calculation templates, *5*, *6*
calculations
  free-form, *14*
  in queries, *13*
  nested, *13*
  time ranges, *6*
changes, saving, *4*
character functions, *11*
clear data step (cube scripts), *33*
CLEAR LEAVES command, *22*
Cloud Control, *11*

Copyright © 2003, 2025, Oracle and/or its affiliates.

Cognos ReportNet, *6*
column links, *20*
configuring
  partitioning options, *27*
configuring Analytic Workspace Manager
  for a proxy server, *3*
  for partitioning options, *26*
  for plug-ins, *5*
  for showing attachment modes, *5*
connect string, for Analytic Workspace Manager, *5*
connections, defining, *4*
COUNT function, *21*
CREATE ANY DIMENSION privilege, *2*
CREATE ANY MATERIALIZED VIEW privilege, *2*
CREATE DIMENSION privilege, *2*
CREATE MATERIALIZED VIEW privilege, *2*
CREATE SESSION privilege, *2*
creation dates of analytic workspaces, *14*
CUBE JOIN, *17*
cube materialized views, *36*, *18*
CUBE SCAN operation, *20*
cube scripts, *33*
cube security, *4*
cube views, *31*, *2*
CUBE_TABLE function, *2*
CUBE_TEMPLATES table, *40*
cubes
  copying and pasting, *42*
  creating, *18*
  defined, *6*, *17*
  joining, *17*
  mapping, *19*
  partitioning, *24*
  requirements for materialized views, *36*
  saving and re-creating, *40*
cumulative calculations, *13*
cursors, *3*

D

dashboard, *3*
data dictionary views, *20*, *4*
data display, *17*, *31*
data loads, *15*, *28*
data maintenance, *32*
data model
  description of dimensional, *6*
  designing, *1*
  saving, *40*
Data Pump, *16*
data security, *2*
  disabling and enabling, *7*
  policies, *7*
  roles, *10*
data sources
  database objects, *2*
  mapping, *12*, *19*
database connections, defining, *4*
database integration, *1*
database security, *2*
DBA scripts download, *14*
DBA_AW_OBJ view, *4*
DBA_AW_PROP view, *4*
DBA_AW_PS view, *4*
DBA_AWS view, *13*
DBA_OBJECTS view, *14*
DBA_REGISTRY view, *13*
DBMS_AW_STATS PL/SQL package, *10*
DBMS_CUBE PL/SQL package, *32*
DBMS_LOB PL/SQL package, *13*
DBMS_METADATA PL/SQL package, *23*
DBMS_MVIEW PL/SQL package, *23*
DBMS_SCHEDULER PL/SQL package., *32*
DBMS_XPLAN PL/SQL package, *23*
dense rank, *11*
dimension hierarchies
  See hierarchies, *8*
dimension object security, *6*
dimension order, affecting aggregation, *5*
dimension security, *3*
dimension views, *4*
dimensions
  copying and pasting, *42*
  creating, *6*
  defined, *7*, *4*
  saving and re-creating, *40*
  viewing members, *17*
Discoverer Plus OLAP, *6*
disk space consumption, *15*
disks, spreading data across, *3*
displaying data, *31*
drillable reports, *3*
drilling, *11*, *20*
drilling (Application Express), *19*
dump files, *16*
dynamic performance tables, *12*

E

edits, saving, *4*
EIF files
  about, *18*
  creating analytic workspaces from, *42*
  saving analytic workspaces to, *42*
end date attributes, *10*
Enterprise Manager Cloud Control, *11*
execution plans, *18*
EXP_FULL_DATABASE privilege, *17*
EXPLAIN PLAN command, *18*
extensibility using plug-ins, *5*

Copyright © 2003, 2025, Oracle and/or its affiliates.

EXTENT MANAGEMENT LOCAL, *2*

F

FAST SOLVE method, *22*
filtering queries, *7*
free-form calculations, *14*
future periods, *8*

G

generator, calculated measures, *3*
Global Computing Company
  data requirements, *A-2*
GLOBAL QUERY REWRITE privilege, *23*
Global schema download, *1*
Gregorian calendar, *6*

H

hidden items (Application Express), *19*
HIER_PARENT function, *17*
hierarchical average operator (aggregation), *2*
hierarchical operators, *4*
hierarchical queries, *11*
hierarchies
  creating, *8*
  defined, *8*, *8*
  level-based, *7*
  supported types, *8*
hierarchy views, *4*

I

index, *7*
init.ora file, *1*
initialization parameters, *1*
installing Analytic Workspace Manager, *3*
installing OLAP option, validation, *13*
integration in database, *1*

J

JOB_QUEUE_PROCESSES parameter, *8*
joining cubes, *17*

L

LAG function, *8*
language support, *38*
layout template (BI Publisher), *4*
LEAD function, *8*
level-based dimensions, *4*
level-based hierarchy, *7*
levels
  creating, *7*
  defined, *8*
load step (cube scripts), *33*
loading data, *15*, *28*
localization, *38*
login names, *2*
LOVs (list of values), *9*, *16*

M

maintenance alternatives, *32*
maintenance scripts, *35*
Maintenance Wizard, *15*, *28*
mappings
  cube, *19*
  dimension, *12*
materialized views
  access privileges, *23*
  creating cube, *36*
  refresh logs, *19*
MAX function, *21*
maximum
  cumulative, *13*
  moving, *12*
measure dimension table
  mapping dimension to, *13*
measure dimensions
  aggregation method of cube, *18*
  and calculated measures, *7*
  mapping, *13*
measure folders
  creating, *39*
  saving and re-creating, *40*
measures
  copying and pasting, *42*
  creating, *19*
  defined, *7*
MIN function, *21*
minimum
  cumulative, *13*
  moving, *12*
moving calculations, *12*

N

natural keys, *6*
nested calculations, *13*
NO_USE_CUBE hint, *17*
normal hierarchies, *8*

O

object security, *2*, *3*, *5*
objects
  copying and pasting, *42*

Copyright © 2003, 2025, Oracle and/or its affiliates.

objects *(continued)*
mapping, *12*, *19*
saving and re-creating, *40*
OLAP data security roles, *10*
OLAP DML
calculated measures, *18*
expressions for data security policies, *8*
OLAP DML step (cube scripts), *33*
OLAP option, verifying installation, *13*
OLAP_DBA role, *2*
OLAP_USER role, *2*
OLAP_XS_ADMIN role, *2*, *8*
optimizer statistics, *10*
Oracle Application Express, *3*
Oracle Business Intelligence, *6*
Oracle Real Application Clusters, *3*, *10*
Oracle Real Application Security, *2*
Oracle Recovery Manager, *16*
OracleBI Discoverer Plus OLAP, *6*
OracleBI Spreadsheet Add-In, *6*
OracleBI Suite Enterprise Edition, *6*
owners of analytic workspaces, identifying, *13*

P

page definition (Application Express), *15*
parallel periods, *11*
parallel processing, *8*
parameter file, *2*
parent-child relations, *8*
partitioning
analyzing partition members, *27*
benefits, *24*
cubes, *24*
discussed, *7*
selecting partitions, *24*
performance counters, *12*
period to date, *9*
pfile settings, *2*
PL/SQL step (cube scripts), *33*
plug-ins
configuring, *5*
installing, *5*
prior periods, *8*
privileges, *2*
proxy server
configuring, *3*

Q

queries, filtering, *7*
query rewrite, *22*
query tools, *3*
QUERY_REWRITE_ENABLED parameter, *23*
querying dimensions and cubes, *1*

R

RAC
See Oracle Real Application Clusters
ragged hierarchies, *8*
rank, *10*
Real Application Clusters
See Oracle Real Application Clusters
refresh logs, *19*
refresh methods, *20*, *21*
Relational Schema Advisor, *37*, *19*
report entry (BI Publisher), *4*
report layout (BI Publisher), *8*
reports, *3*
RMAN, *16*

S

sample schema download, *1*
saving
analytic workspaces to EIF files, *42*
objects
to XML Templates, *41*
scaled operators, *3*
scheduling maintenance, *8*
security, *7*
about, *1*
data, *2*
materialized views, *23*
object, *5*
See also data security
server parameter file, *2*
SESSIONS parameter, *2*
share, *10*
single-row functions, *2*
size of analytic workspace, *13*
skip-level hierarchies, *8*
source data, *2*
Spreadsheet Add-In, *6*
static data dictionary views, *20*, *4*
step types, *33*
SUM function, *21*
sum operator (aggregation), *1*
surrogate keys, *6*
system tables, *5*

T

tablespaces, *2*
templates
BI Publisher, *5*
calculation, *5*
creating XML, *40*
saving object definitions to, *41*
time dimensions, *6*
time ranges in calculations, *6*

Copyright © 2003, 2025, Oracle and/or its affiliates.

time span attributes, *10*
total
    cumulative, *13*
    moving, *12*
transportable tablespaces, *16*

U

unique key attributes, *11*
upgrading metadata, *6*
USE_CUBE hint, *17*
user names, *2*
USER_AW_OBJ view, *4*
USER_AW_PROP view, *4*
USER_AW_PS view, *4*
USER_CUBE_DIM_VIEWS view, *4*
USER_CUBE_VIEW_COLUMNS view, *3*
USER_MVIEWS view, *19*

V

value-based dimensions, *4*
value-based hierarchies, *8*

W

weighted operators, *3*
weighted sum (aggregation), *2*
WHERE clause operations, *11*

X

XML templates
    about, *17*
    creating objects from, *41*
    saving object definitions to, *41*

Copyright © 2003, 2025, Oracle and/or its affiliates.