Oracle® OLAP

Reference

10*g* Release 1 (10.1)

Part No. B10334-02

December 2003

ORACLE®

Oracle OLAP Reference, 10*g* Release 1 (10.1)

Part No. B10334-02

Contents

3 Active Catalog Views

10 CWM2_OLAP_DIMENSION

11 CWM2_OLAP_DIMENSION_ATTRIBUTE

12 CWM2_OLAP_HIERARCHY

13 CWM2_OLAP_LEVEL

14 CWM2_OLAP_LEVEL_ATTRIBUTE

15 CWM2_OLAP_MEASURE

16 CWM2_OLAP_METADATA_REFRESH

17 CWM2_OLAP_PC_TRANSFORM

18 CWM2_OLAP_TABLE_MAP

21 DBMS_AW

22 DBMS_AW_UTILITIES

23 DBMS_AWM

25 OLAP_API_SESSION_INIT

26 OLAP_TABLE

Index

Send Us Your Comments

Oracle OLAP Reference, 10*g* Release 1 (10.1)

Part No. B10334-02

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the title and part number of the documentation and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: info_dev@oracle.com
- FAX: 781-238-9850. Attn: Oracle OLAP
- Postal service:
  Oracle Corporation
  Oracle OLAP Documentation
  10 Van de Graaff Drive
  Burlington, MA 01803
  U.S.A.

If you would like a reply, please give your name, address, telephone number, and electronic mail address (optional).

If you have problems with the software, please contact your local Oracle Support Services.

Preface

This reference manual describes the Oracle PL/SQL packages shipped with the OLAP option of the Oracle Database.

Intended Audience

This reference manual is intended for database administrators and application developers who perform the following tasks:

- Administer a database
- Administer analytic workspaces
- Build and maintain data warehouses or data marts
- Define metadata
- Develop analytical applications

To use this document, you need no prior knowledge of Oracle OLAP.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For additional information, visit the Oracle Accessibility Program Web site at http://www.oracle.com/accessibility/

Accessibility of Code Examples in Documentation JAWS, a Windows screen reader, may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, JAWS may not always read a line of text that consists solely of a bracket or brace.

Structure

This document contains the following chapters.

Chapter 1, "Creating Analytic Workspaces with DBMS_AWM"
This chapter explains how to use the DBMS_AWM package.

Chapter 2, "Creating OLAP Catalog Metadata with CWM2"
This chapter explains how to use the CWM2 packages.

Chapter 3, "Active Catalog Views"
This chapter describes the views in the Active Catalog.

Chapter 4, "Analytic Workspace Maintenance Views"
This chapter describes the views of analytic workspace maintenance information.

Chapter 5, "OLAP Catalog Metadata Views"
This chapter describes the views of OLAP Catalog metadata.

Chapter 6, "OLAP Fixed Views"
This chapter describes the dynamic performance views for Oracle OLAP.

Chapter 7, "CWM2_OLAP_CATALOG"
This chapter describes the syntax of the procedures in the CWM2_OLAP_CATALOG package.

Chapter 8, "CWM2_OLAP_CLASSIFY"
This chapter describes the metadata descriptors required by Oracle OLE DB for OLAP.

Chapter 9, "CWM2_OLAP_CUBE"

This chapter describes the syntax of the procedures in the CWM2_OLAP_CUBE package.

Chapter 10, "CWM2_OLAP_DIMENSION"

This chapter describes the syntax of the procedures in the CWM2_OLAP_DIMENSION package.

Chapter 11, "CWM2_OLAP_DIMENSION_ATTRIBUTE"

This chapter describes the syntax of the procedures in the CWM2_OLAP_DIMENSION_ATTRIBUTE package.

Chapter 12, "CWM2_OLAP_HIERARCHY"

This chapter describes the syntax of the procedures in the CWM2_OLAP_HIERARCHY package.

Chapter 13, "CWM2_OLAP_LEVEL"

This chapter describes the syntax of the procedures in the CWM2_OLAP_LEVEL package.

Chapter 14, "CWM2_OLAP_LEVEL_ATTRIBUTE"

This chapter describes the syntax of the procedures in the CWM2_OLAP_LEVEL_ATTRIBUTE package.

Chapter 15, "CWM2_OLAP_MEASURE"

This chapter describes the syntax of the procedures in the CWM2_OLAP_MEASURE package.

Chapter 16, "CWM2_OLAP_METADATA_REFRESH"

This chapter describes the syntax of the procedures in the CWM2_OLAP_METADATA_REFRESH package.

Chapter 17, "CWM2_OLAP_PC_TRANSFORM"

This chapter describes the syntax of the procedures in the CWM2_OLAP_PC_TRANSFORM package.

Chapter 18, "CWM2_OLAP_TABLE_MAP"

This chapter describes the syntax of the procedures in the CWM2_OLAP_TABLE_MAP package.

Chapter 19, "CWM2_OLAP_VALIDATE"

This chapter describes the syntax of the procedures in the CWM2_OLAP_VALIDATE package.

Chapter 20, "CWM2_OLAP_VERIFY_ACCESS"

This chapter describes the syntax of the procedures in the CWM2_OLAP_VERIFY package.

Chapter 21, "DBMS_AW"

This chapter describes the syntax of the procedures in the DBMS_AW package.

Chapter 22, "DBMS_AW_UTILITIES"

This chapter describes the syntax of the procedures in the DBMS_AW_UTILITIES package.

Chapter 23, "DBMS_AWM"

This chapter describes the syntax of the procedures in the DBMS_AWM package.

Chapter 24, "DBMS_ODM"

This chapter describes the syntax of the procedures in the DBMS_ODM package.

Chapter 25, "OLAP_API_SESSION_INIT"

This chapter describes the syntax of the procedures in the OLAP_API_SESSION_INIT package.

Chapter 26, "OLAP_TABLE"

This chapter describes the syntax of the OLAP_TABLE function.

Related Documents

For more information see these Oracle resources:

- *Oracle OLAP Application Developer's Guide*

Explains how SQL and Java applications can extend their analytic processing capabilities by using Oracle OLAP.

- *Oracle OLAP DML Reference*

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

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

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

- *Oracle OLAP Java API Reference*

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

- *Oracle OLAP Analytic Workspace Java API Reference*

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

- *Oracle Data Warehousing Guide*

  Discusses the database structures, concepts, and issues involved in creating a data warehouse to support online analytical processing solutions.

- *PL/SQL User's Guide and Reference*

  Explains the concepts and syntax of PL/SQL, Oracle's procedural extension of SQL.

Conventions

The following conventions are also used in this manual:

Convention	Meaning
. . .	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.
. . .	Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted

Convention	Meaning
boldface text	Boldface type in text indicates a term defined in the text, the glossary, or in both locations.
< >	Angle brackets enclose user-supplied names.
[ ]	Brackets enclose optional clauses from which you can choose one or none.
$	The dollar sign represents the DIGITAL Command Language prompt in Windows and the Bourne shell prompt in Digital UNIX

1

Creating Analytic Workspaces with DBMS_AWM

The `DBMS_AWM` package provides stored procedures for creating an analytic workspace cube from a star schema and enabling it for access by the OLAP API. The `DBMS_AWM` package is used by Analytic Workspace Manager. This chapter explains how to work with the `DBMS_AWM` procedures directly.

See Also:

- Chapter 23, "DBMS_AWM"
- Chapter 3, "Active Catalog Views"
- Chapter 4, "Analytic Workspace Maintenance Views"

This chapter contains the following topics:

Overview

If your data is stored in a star or snowflake schema, then you can use the DBMS_AWM package to simplify the process of loading it into an analytic workspace.

The first step is to create OLAP Catalog metadata that describes the functionality of your schema in multidimensional terms, that is, as a cube with dimensions, attributes, and measures. You can then use the DBMS_AWM package to instantiate these objects in an analytic workspace, create relational views of the workspace objects, and optionally generate a secondary set of OLAP Catalog metadata that maps to the workspace views.

Note: Analytic workspaces created by the DBMS_AWM procedures are in **database standard form**, ensuring compatibility with related Oracle OLAP tools and utilities. See *Oracle OLAP Application Developer's Guide* for information about standard form.

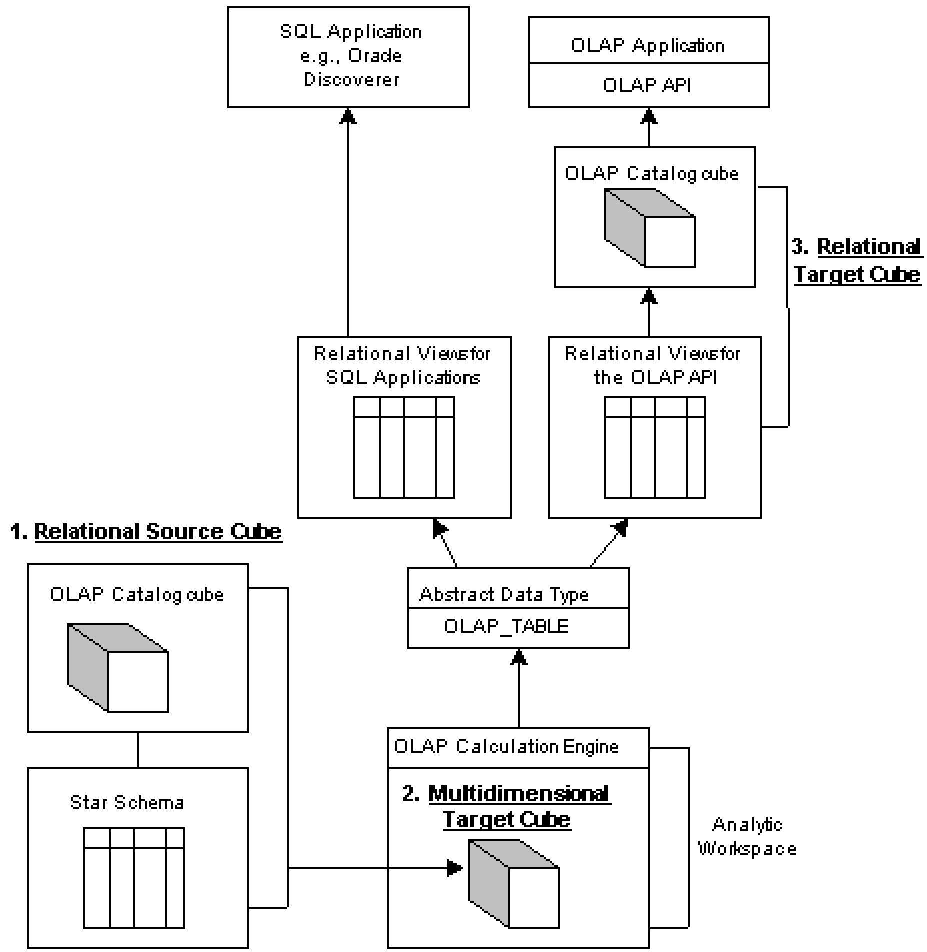
The DBMS_AWM package provides a feature–rich set of APIs that you can use to manage analytic workspaces. To effectively use these APIs, you will need to understand how the APIs work together to move data from a relational source to a multidimensional target and how they establish relational access to that target.

The basic flow of events involves the creation of three separate logical cubes:

1. **Relational Source Cube**. This cube must exist before you call any of the DBMS_AWM procedures. The cube's metadata is defined within the OLAP Catalog. Its data is unsolved (lowest level only) and stored in a star schema.
2. **Multidimensional Target Cube**. DBMS_AWM procedures define and populate this cube from the relational source cube. The cube's standard form metadata is defined in the analytic workspace. Its data is stored in the workspace, typically with full or partial summarization.
3. **Relational Target Cube**. DBMS_AWM procedures define this cube from the multidimensional target cube. The cube's metadata is defined within the OLAP Catalog. Its data is stored in the analytic workspace and accessed through relational views. The views present the data as fully solved (embedded totals for all level combinations).

The basic process of creating and enabling an analytic workspace with the DBMS_AWM package is illustrated in Figure 1–1.

Figure 1–1 Creating and Enabling an Analytic Workspace with DBMS_AWM



Creating OLAP Catalog Metadata for the Source Cube

Before you can use the DBMS_AWM procedures, you must create a cube in the OLAP Catalog and map it to the source fact table and dimension tables. The source tables must be organized in a basic star or snowflake schema.

You can use Enterprise Manager, or you can write scripts that use the CWM2 PL/SQL packages, as described in Chapter 2. You can also use Oracle Warehouse Builder to create OLAP Catalog metadata.

This cube is the **Relational Source Cube** identified in Figure 1–1.

Creating and Populating Workspace Dimensions

For each dimension of a cube defined in the OLAP Catalog, you must run a set of procedures in the DBMS_AWM package to accomplish the following general tasks:

1. Create a **dimension load specification**, which contains instructions for populating the dimension in the analytic workspace. The load specification may include a filter that identifies criteria for selecting data from the source dimension tables.
2. Create containers for the dimension in an analytic workspace.
3. Use the dimension load specification to populate the dimension in the analytic workspace from the source dimension tables.

See Also: "Creating and Refreshing a Workspace Dimension" on page 1-10.

Creating and Populating Workspace Cubes

After creating the cube's dimensions, run another set of procedures to create and populate the cube itself.

1. Create a **cube load specification**, which contains instructions for populating the cube's measures in the analytic workspace. The load specification may include a filter that identifies criteria for selecting data from the source fact table.
2. Create a **composite specification**, which contains instructions for ordering the cube's dimensions and storing sparse data in the analytic workspace.
3. Add the composite specification to the cube load specification.
4. Create containers for the cube in an analytic workspace.
5. Use the cube load specification to populate the cube's measures in the analytic workspace from the source fact table.

This cube is the **Multidimensional Target Cube** identified in Figure 1–1.

See Also: "Creating and Refreshing a Workspace Cube" on page 1-13 and "Managing Sparse Data and Optimizing the Workspace Cube" on page 1-16.

Aggregating the Cube's Data in the Analytic Workspace

For the workspace cube, run a set of procedures to accomplish the following:

1. Create an **aggregation specification**, which contains instructions for storing summary data in the analytic workspace.
2. Use the aggregation specification to aggregate the workspace cube.

See Also: "Aggregating the Data in an Analytic Workspace" on page 1-18.

Enabling Relational Access to the Workspace Cube

Once you have created, populated, and aggregated the cube in an analytic workspace, run another set of procedures to enable relational access. The **enablement process** consists of generating and running a set of enablement scripts. These scripts create the relational views that use the `OLAP_TABLE` function to access the workspace cube. The scripts may also create an OLAP Catalog cube that maps to the views.

The cube created by the enablement scripts is the **Relational Target Cube** identified in Figure 1–1.

To enable a workspace cube, you can either generate the scripts and run them yourself or you can use a one-step procedure to create and run the scripts automatically.

See Also: "Creating Relational Access to the Workspace Cube" on page 1-23.

Viewing Metadata Created by DBMS_AWM

Two sets of views reveal metadata related to analytic workspaces. The **Active Catalog views** reveal metadata stored within analytic workspaces. The **Analytic Workspace Maintenance views** reveal metadata stored within the OLAP Catalog.

Active Catalog Views

These views use OLAP_TABLE functions to return information about logical standard form objects within analytic workspaces. For example, you could query an Active Catalog view to obtain information about the dimensionality of a workspace cube. The Active Catalog view names have the prefix ALL_OLAP2_AW. For more information, see Chapter 3.

Analytic Workspace Maintenance Views

These views return information about building and maintaining analytic workspace cubes. For example, you could query an Analytic Workspace Maintenance view to obtain information about the load specifications associated with an analytic workspace dimension or cube. The Analytic Workspace Maintenance view names have the prefix ALL_AW. For more information, see Chapter 4.

Understanding the DBMS_AWM Procedures

The procedures in the DBMS_AWM package support methods on several types of logical entities. These entities are described in Table 1–1.

> **See Also:** Chapter 23, "DBMS_AWM"

Table 1–1 Logical Entities in the DBMS_AWM Package

Entity	Description
Dimension	A dimension in the OLAP Catalog and its corresponding dimension in an analytic workspace.
Cube	A cube in the OLAP Catalog and its corresponding cube in an analytic workspace.
Dimension Load Specification	Instructions for populating an analytic workspace dimension from the dimension tables of an OLAP Catalog dimension.
Cube Load Specification	Instructions for populating an analytic workspace cube from the fact table of an OLAP Catalog cube.
Cube Aggregation Specification	Instructions for creating summary data in an analytic workspace.
Cube Composite Specification	Instructions for ordering dimensions and storing sparse data in an analytic workspace.

Methods on Dimensions

The methods you can perform on a dimension are described in Table 1–2.

Table 1–2 Methods on Dimensions in DBMS_AWM

Method	Description	Procedure
Create	Create containers in an analytic workspace for a dimension defined in the OLAP Catalog.	CREATE_AWDIMENSION Procedure
Refresh	Use a dimension load specification to populate an analytic workspace dimension from the dimension tables of an OLAP Catalog dimension.	REFRESH_AWDIMENSION Procedure
Create access	Create a script to enable relational access to a dimension in an analytic workspace.	CREATE_AWCUBELOAD_SPEC Procedure
Delete access	Create a script to disable relational access to a dimension in an analytic workspace.	DELETE_AWDIMENSION_ACCESS Procedure
Set view name	Specify new names for the relational views of a dimension in an analytic workspace.	SET_AWDIMENSION_VIEW_NAME Procedure

Methods on Cubes

The methods you can perform on a cube are described in Table 1–3.

Table 1–3 Methods on Cubes in DBMS_AWM

Method	Description	Procedure
Create	Create containers in an analytic workspace for a cube defined in the OLAP Catalog.	CREATE_AWCUBE Procedure
Refresh	Use a cube load specification to populate the measures of an analytic workspace cube from the fact table of an OLAP Catalog cube.	REFRESH_AWCUBE Procedure
Aggregate	Use an aggregation specification to aggregate the cube in the analytic workspace.	AGGREGATE_AWCUBE Procedure

Table 1–3 (Cont.) Methods on Cubes in DBMS_AWM

Method	Description	Procedure
Create Access	Create a script to enable relational access to a cube in an analytic workspace.	CREATE_AWCUBE_ACCESS Procedure
Delete access	Create a script to disable relational access to a cube in an analytic workspace	DELETE_AWCUBE_ACCESS Procedure
Set view name	Specify new names for the relational views of a cube's data in an analytic workspace.	SET_AWCUBE_VIEW_NAME Procedure

Methods on Dimension Load Specifications

The methods you can perform on a dimension load specification are described in Table 1–4.

Table 1–4 Methods on Dimension Load Specifications in DBMS_AWM

Method	Description	Procedure
Create/Delete	Create or delete a dimension load specification.	CREATE_AWDIMLOAD_SPEC Procedure DELETE_AWDIMLOAD_SPEC Procedure
Reset information	Change various components of a dimension load specification.	SET_AWDIMLOAD_SPEC_DIMENSION Procedure SET_AWDIMLOAD_SPEC_LOADTYPE Procedure SET_AWDIMLOAD_SPEC_NAME Procedure SET_AWDIMLOAD_SPEC_PARAMETER Procedure
Add/Delete filter	Add or remove a filter from a dimension load specification.	ADD_AWDIMLOAD_SPEC_FILTER Procedure DELETE_AWDIMLOAD_SPEC_FILTER Procedure

Methods on Cube Load Specifications

The methods you can perform on a cube load specification are described in Table 1–5.

Table 1–5 Methods on Cube Load Specifications in DBMS_AWM

Method	Description	Procedure
Create/Delete	Create or delete a cube load specification.	CREATE_AWCUBELOAD_SPEC Procedure DELETE_AWCUBELOAD_SPEC Procedure
Reset information	Change various components of a cube load specification.	SET_AWCUBELOAD_SPEC_CUBE Procedure SET_AWCUBELOAD_SPEC_LOADTYPE Procedure SET_AWCUBELOAD_SPEC_NAME Procedure SET_AWCUBELOAD_SPEC_PARAMETER Procedure
Add/Delete filter	Add or remove a filter from a cube load specification.	ADD_AWCUBELOAD_SPEC_FILTER Procedure DELETE_AWCUBELOAD_SPEC_FILTER Procedure
Add/Delete composite specification	Add or remove a composite specification from a cube load specification.	ADD_AWCUBELOAD_SPEC_COMP Procedure DELETE_AWCUBELOAD_SPEC_COMP Procedure

Methods on Aggregation Specifications

The methods you can perform on an aggregation specification are described in Table 1–6.

Table 1–6 Methods on Aggregation Specifications in DBMS_AWM

Method	Description	Procedure
Create/Delete	Create or delete an aggregation specification.	CREATE_AWCUBEAGG_SPEC Procedure DELETE_AWCUBEAGG_SPEC_MEASURE Procedure
Set operator	Set the aggregation operator for a dimension.	SET_AWCUBEAGG_SPEC_AGGOP Procedure
Add/Delete levels	Add or remove levels from an aggregation specification.	ADD_AWCUBEAGG_SPEC_LEVEL Procedure DELETE_AWCUBEAGG_SPEC_LEVEL Procedure

Table 1–6 (Cont.) Methods on Aggregation Specifications in DBMS_AWM

Method	Description	Procedure
Add/Delete measures	Add or remove measures from an aggregation specification.	ADD_AWCUBEAGG_SPEC_MEASURE Procedure DELETE_AWCUBEAGG_SPEC_MEASURE Procedure

Methods on Composite Specifications

The methods you can perform on a composite specification are described in Table 1–7.

Table 1–7 Methods on Composite Specifications in DBMS_AWM

Method	Description	Procedure
Create/Delete	Create or delete a composite specification.	CREATE_AWCOMP_SPEC Procedure DELETE_AWCOMP_SPEC Procedure
Reset information	Change the name of the composite specification or associate it with a different cube.	SET_AWCOMP_SPEC_CUBE Procedure SET_AWCOMP_SPEC_NAME Procedure
Add/Delete members	Add or remove members from the specification. Members can be dimensions or composites.	ADD_AWCOMP_SPEC_MEMBER Procedure DELETE_AWCOMP_SPEC_MEMBER Procedure
Reset member information	Change information about members of the specification.	SET_AWCOMP_SPEC_MEMBER_NAME Procedure SET_AWCOMP_SPEC_MEMBER_POS Procedure SET_AWCOMP_SPEC_MEMBER_SEG Procedure
Add composite members	Add members to a composite in the specification.	ADD_AWCOMP_SPEC_COMP_MEMBER Procedure

Creating and Refreshing a Workspace Dimension

Once you have defined a dimension in the OLAP Catalog for your source dimension table, you can create the dimension in the analytic workspace.

Only one workspace dimension may be created from a given dimension in the OLAP Catalog. For example, if you have used the OLAP Catalog `PRODUCT` dimension as the source for the `PROD_AW` dimension in an analytic workspace, you cannot create another dimension `PROD_AW2` from the same source dimension in the same workspace.

Note: `CREATE_AWDIMENSION` opens the analytic workspace with read/write access. It updates the workspace, but it *does not* execute a SQL `COMMIT`.

The analytic workspace must already exist before you call `CREATE_AWDIMENSION` or any other procedures in the `DBMS_AWM` package.

Example 1–1 shows the procedure calls for defining and populating workspace objects for the `XADEMO.CHANNEL` dimension. The load specification includes a filter condition that causes only the row for `'DIRECT'` to be loaded.

Example 1–1 Creating the CHANNEL Dimension in an Analytic Workspace

```
--- SET UP
set serveroutput on
execute cwm2_olap_manager.set_echo_on;
execute cwm2_olap_manager.begin_log
          ('/users/myxademo/myscripts' , 'channel.log');

--- CREATE THE ANALYTIC WORKSPACE
execute dbms_aw.execute ('aw create ''myaw''');

--- CREATE AND POPULATE THE DIMENSION
execute dbms_awm.create_awdimension
          ('XADEMO', 'CHANNEL', 'MYSCHEMA','MYAW', 'AW_CHAN');
execute dbms_awm.create_awdimload_spec
          ('CHAN_LOAD', 'XADEMO', 'CHANNEL', 'FULL_LOAD');
execute dbms_awm.add_awdimload_spec_filter
          ('CHAN_LOAD', 'XADEMO', 'CHANNEL', 'XADEMO',
          'XADEMO_CHANNEL', '''CHAN_STD_CHANNEL'' = ''DIRECT''' );
execute dbms_awm.refresh_awdimension
          ('MYSCHEMA', 'MYAW', 'AW_CHAN', 'CHAN_LOAD');

--- COMMIT AND WRAP UP
commit;
execute cwm2_olap_manager.set_echo_off;
```

```
execute cwm2_olap_manager.end_log
```

When you query the Active Catalog view ALL_OLAP2_AW_DIMENSIONS, you will see the following row.

```
AW_OWNER    AW_NAME     AW_LOGICAL_NAME       SOURCE_OWNER  SOURCE_NAME
----------  ----------  -------------------  -------------  -----------
MYSCHEMA    MYAW        AW_CHAN               XADEMO      CHANNEL
```

Refreshing the Dimension's Metadata

CREATE_AWDIMENSION ensures that the generic standard form objects that support dimensions exist in the workspace, and it registers the specified dimension in the workspace. However, the metadata that defines the logical structure of this particular dimension is not instantiated in the workspace until you call REFRESH_AWDIMENSION.

For example, if you have just created a dimension AW_PROD in a workspace MYAW in XADEMO from a source dimension XADEMO.PRODUCT, you can query the Active Catalog to check the workspace.

```
SQL>select * from ALL_OLAP2_AW_DIMENSIONS WHERE AW_LOGICAL_NAME in 'AW_PROD';

AW_OWNER   AW_NAME         AW_LOGICAL_NAME           SOURCE_OWNER    SOURCE_NAME
---------- --------------- ------------------------- --------------- -----------
XADEMO     MYAW            AW_PROD                   XADEMO          PRODUCT
```

The following query shows that there are no levels associated with the dimension. The levels, hierarchies, attributes, and descriptions will be instantiated when the dimension is refreshed.

```
SQL>select * from ALL_OLAP2_AW_DIM_LEVELS where AW_LOGICAL_NAME in 'AW_PROD';

no rows selected
```

When To Refresh a Dimension

You must refresh a dimension whenever changes occur in the source dimension tables. These changes could be additions or deletions of dimension members, for example removing a product from a Product dimension, or they could be changes to the dimension's metadata, such as adding a Day level to a time dimension.

When you refresh a dimension, you must also refresh each cube in which it participates.

What To Do After a Dimension Refresh

When you refresh a dimension because of structural metadata changes to its hierarchies, you must re-enable the dimension and its related cubes. When you refresh a dimension because of data changes, you do not need to re-enable.

When you refresh a dimension whose cube has associated stored summaries in the analytic workspace (the result of an aggregation specification), you must also reaggregate the cube.

Creating and Refreshing a Workspace Cube

Once you have defined a cube in the OLAP Catalog for your star schema, you can create the cube in the analytic workspace.

You must call CREATE_AWDIMENSION to create each of the cube's dimensions before calling CREATE_AWCUBE to create the cube. To populate the cube, you must call REFRESH_AWDIMENSION to populate each of the cube's dimensions before calling REFRESH_AWCUBE to refresh the cube's measures. On subsequent refreshes, you only need to refresh the dimensions that have changed.

Within an analytic workspace, dimensions can be shared by more than one cube. When creating a new workspace cube, you will only call CREATE_AWDIMENSION for OLAP Catalog dimensions that have not been used as the source for dimensions of cubes that already exist in the workspace.

> **Note:** CREATE_AWCUBE opens the analytic workspace with read/write access. It updates the workspace, but it *does not* execute a SQL COMMIT.
>
> The analytic workspace must already exist before you call CREATE_AWCUBE or any other procedures in the DBMS_AWM package.

Example 1–2 shows the procedure calls for creating and populating the XADEMO.ANALYTIC_CUBE cube in an analytic workspace.

Example 1–2 Creating the ANALYTIC_CUBE Cube in an Analytic Workspace

```
--- SET UP
set serveroutput on
execute cwm2_olap_manager.set_echo_on;
execute cwm2_olap_manager.begin_log
```

```
          ('/users/myxademo/myscripts' , 'anacube.log');


--- CREATE THE ANALYTIC WORKSPACE
execute dbms_aw.execute ('aw create ''myaw''');

--- CREATE AND REFRESH THE DIMENSIONS
execute dbms_awm.create_awdimension
          ('XADEMO','CHANNEL','MYSCHEMA', 'MYAW', 'AW_CHAN');
execute dbms_awm.create_awdimension
          ('XADEMO','GEOGRAPHY','MYSCHEMA','MYAW', 'AW_GEOG');
execute dbms_awm.create_awdimension
          ('XADEMO','PRODUCT','MYSCHEMA', 'MYAW', 'AW_PROD');
execute dbms_awm.create_awdimension
          ('XADEMO','TIME','MYSCHEMA', 'MYAW', 'AW_TIME');
execute dbms_awm.refresh_awdimension
          ('MYSCHEMA', 'MYAW', 'AW_CHAN');
execute dbms_awm.refresh_awdimension
          ('MYSCHEMA', 'MYAW', 'AW_PROD');
execute dbms_awm.refresh_awdimension
          ('MYSCHEMA', 'MYAW', 'AW_GEOG');
execute dbms_awm.refresh_awdimension
          ('MYSCHEMA', 'MYAW', 'AW_TIME');

--- CREATE AND REFRESH THE CUBE
execute dbms_awm.create_awcube
          ('XADEMO', 'ANALYTIC_CUBE','MYSCHEMA', 'MYAW','AW_ANACUBE');
execute dbms_awm.create_awcubeload_spec
          ('AC_CUBELOADSPEC', 'XADEMO', 'ANALYTIC_CUBE', 'LOAD_DATA');
execute dbms_awm.refresh_awcube
          ('MYSCHEMA', 'MYAW', 'AW_ANACUBE', 'AC_CUBELOADSPEC');

--- COMMIT AND WRAP UP
commit;
execute cwm2_olap_manager.set_echo_off;
execute cwm2_olap_manager.end_log
```

When you query the Active Catalog view `ALL_OLAP2_AW_CUBES`, you will see the following row.

```
AW_OWNER    AW_NAME     AW_LOGICAL_NAME       SOURCE_OWNER  SOURCE_NAME
----------  ----------  -------------------  -------------  -------------
MYSCHEMA    MYAW        AW_ANACUBE            XADEMO        ANALYTIC_CUBE
```

Data Type Conversion

The measures in the source fact table may have numeric, text, or date data types. When REFRESH_AWCUBE loads the data into a workspace cube, it converts the RDBMS data types to types that are native to analytic workspaces. The data type conversion is described in Table 1–8.

If a source measure has a data type not described in Table 1–8, the measure is ignored by REFRESH_AWCUBE and none of its data or metadata is loaded into the analytic workspace.

Table 1–8 Conversion of RDBMS Data Types to Workspace Data Types

RDBMS Data Type	Analytic Workspace Data Type
NUMBER	DECIMAL
CHAR, LONG, VARCHAR, VARCHAR2	TEXT
NCHAR, NVARCHAR2	NTEXT
DATE	DATE

Refreshing the Cube's Metadata

CREATE_AWCUBE ensures that the generic standard form objects that support cubes exist in the workspace, and it registers the specified cube in the workspace. However, the metadata that defines the logical structure of this particular cube is not instantiated in the workspace until you call REFRESH_AWCUBE.

For example, if you have just created a cube AW_ANACUBE in a workspace MYAW in MYSCHEMA from the source cube XADEMO.ANALYTIC_CUBE, you can query the Active Catalog to check the workspace.

```
SQL>select * from ALL_OLAP2_AW_CUBES where AW_LOGICAL_NAME in 'AW_ANACUBE';

AW_OWNER   AW_NAME     AW_LOGICAL_NAME       SOURCE_OWNER    SOURCE_NAME
---------- ----------- --------------------- --------------- --------------
MYSCHEMA   MYAW        AW_ANACUBE            XADEMO          ANALYTIC_CUBE
```

The following query shows that there are no measures associated with the cube. The measures, dimensions, and descriptions will be instantiated when the cube is refreshed.

```
SQL>select * from ALL_OLAP2_AW_CUBE_MEASURES where AW_CUBE_NAME in 'AW_ANACUBE';

no rows selected
```

When To Refresh a Cube

You must refresh a cube whenever changes occur in the source fact table. These changes could be additions or deletions of data, for example updating sales figures, or they could be changes to the cube's metadata, such as adding a measure or renaming a description.

When you refresh a cube, you must first refresh any of its dimensions that have changed.

What To Do After a Cube Refresh

When you refresh a cube because of structural metadata changes to its dimension hierarchies, you must re-enable the cube and its related dimensions. When you refresh a cube because of data changes, you do not need to re-enable.

Everytime you refresh a cube that has an associated aggregation specification, you must reaggregate the cube.

If you make changes to the composite specification associated with a cube, you must drop the cube and re-create it in the analytic workspace. You cannot refresh a cube with a modified composite specification.

Managing Sparse Data and Optimizing the Workspace Cube

A **composite** is an object that is used to store sparse data compactly in a variable in an analytic workspace. A composite consists of a list of dimension-value combinations in which one value is taken from each of the dimensions on which the composite is based. Only the combinations for which data exists are included in the composite.

Composites are maintained automatically by the OLAP engine. With composites, you can keep your analytic workspace size to a minimum and promote good performance. For more information on composites, see the *Oracle OLAP DML Reference*. For information on managing sparsity and optimizing performance in your analytic workspaces, see the *Oracle OLAP Application Developer's Guide*

For example, you might have some products in your analytic cube that are not sold in all regions. The data cells for those combinations of `PRODUCT` and `GEOGRAPHY` would be empty. In this case, you might choose to define `PRODUCT` and `GEOGRAPHY` as a composite. The OLAP DML syntax for defining the dimensionality of the Costs measure in this cube could be as follows.

```
DEFINE prod_geog COMPOSITE <product geography>
DEFINE costs VARIABLE INTEGER <time channel prod_geog<product geography>>
```

To specify that a cube's data be loaded into an analytic workspace using this definition of the cube's dimensionality, you would define a **composite specification** for the cube. The composite specification would define the following expression.

```
<time channel prod_geog<product geography>>
```

Each member of a composite specification has a name, a type, and a position. Table 1–9 identifies this information for the preceding example.

Table 1–9 Composite Spec Members for XADEMO.ANALYTIC_CUBE

Member	Type	Position
TIME	dimension	1
CHANNEL	dimension	2
PROD_GEOG	composite	3
PRODUCT	dimension	4
GEOGRAPHY	dimension	5

Dimension Order

Dimension order determines how the cube's data is stored and accessed in the analytic workspace. The first dimension in the dimension's definition is the fastest-varying and the last is the slowest-varying.

By default, `REFRESH_AWCUBE` defines a workspace cube's dimensionality with Time as the fastest varying dimension followed by a composite of all the other dimensions. The dimensions in the composite are ordered according to their size. The dimension with the most members is first and the dimension with the least members is last. For example, the default dimensionality of the `ANALYTIC_CUBE` in an analytic workspace would be as follows.

```
<time comp_name<geography, product, channel>>
```

You can override the default dimensionality by specifying a composite specification and including it in the cube load specification.

For information on ordering dimensions and specifying segment size for dimension storage, see the *Oracle OLAP Application Developer's Guide*.

Creating and Modifying a Composite Specification

The statements in Example 1–3 create a composite specification called `comp1` for the `ANALYTIC_CUBE`.

Example 1–3 Defining a Cube's Dimensionality in an Analytic Workspace

```
exec dbms_awm.create_awcomp_spec
           ('comp1', 'xademo', 'analytic_cube');
exec dbms_awm.add_awcomp_spec_member
          ('comp1', 'xademo', 'analytic_cube', 'comp1_time', 'dimension',
           'xademo', 'time');
exec dbms_awm.add_awcomp_spec_member
          ('comp1', 'xademo', 'analytic_cube', 'comp1_channel', 'dimension',
           'xademo', 'channel');
exec dbms_awm.add_awcomp_spec_member
          ('comp1', 'xademo', 'analytic_cube', 'comp1_prod_geog', 'composite');
exec dbms_awm.add_awcomp_spec_comp_member
          ('comp1', 'xademo', 'analytic_cube', 'comp1_prod_geog',
           'comp1_product' ,'dimension', 'xademo', 'product');
exec dbms_awm.add_awcomp_spec_comp_member
          ('comp1', 'xademo', 'analytic_cube', 'comp1_prod_geog',
           'comp1_geography' ,'dimension', 'xademo', 'geography');
exec dbms_awm.add_awcubeload_spec_comp
           ('my_cube_load', 'xademo', 'analytic_cube', 'comp1');
```

You can modify a composite specification by applying it to a different cube or giving it a different name. You can rename, move, and change the segment size of a primary member of a composite specification. However, you cannot rename, move, or change the segment size of a member of a composite. To edit the composite itself, you must delete it and define a new composite.

Suppose that you wanted to make Channel, instead of Time, the fastest varying dimension of the cube in the analytic workspace. You could reposition Channel in the composite specification as follows.

```
exec dbms_awm.set_awcomp_spec_member_pos
          ('comp1', 'xademo', 'analytic_cube', 'comp1_channel', 1);
```

Aggregating the Data in an Analytic Workspace

The `DBMS_AWM` package allows you to store aggregate data for level combinations of measures in a workspace cube.

Stored aggregates in an analytic workspace are similar to materialized views for relational data. However, a workspace cube is always presented as fully solved with embedded totals when enabled for SQL access by an application. If you do not preaggregate any of the workspace data, all the aggregate data is still available but it must be calculated on the fly.

Preaggregating some or all of your workspace data will improve query performance in most circumstances. For information on choosing an aggregation strategy, refer to the *Oracle OLAP Application Developer's Guide*

Note: The aggregation process (AGGREGATE_AWCUBE) opens the analytic workspace with read/write access. It updates the workspace, but it *does not* execute a SQL COMMIT.

The cube refresh process stores detail data in the workspace and sets up the structures to support dynamic aggregation. If you want to preaggregate some or all of your data, you must create an aggregation specification and run a separate aggregation procedure for the workspace cube.

Creating an Aggregation Specification

Example 1–4 shows sample procedure calls for preaggregating the Costs and Quota measures of the analytic workspace cube AC2, which was created from XADEMO.ANALYTIC_CUBE.

The quarter totals (level 'L2' of TIME) for product groups (level 'L3' of PRODUCT), product divisions (level 'L2' of PRODUCT), and all channels (level 'STANDARD-2' of CHANNEL) are calculated and stored in the analytic workspace.

Example 1–4 Preaggregating Costs and Quota in an Analytic Workspace

```
execute dbms_awm.create_awcubeagg_spec
          ('AGG1', 'MYSCHEMA', 'MYAW', 'AC2');
execute dbms_awm.add_awcubeagg_spec_level
          ('AGG1', 'MYSCHEMA', 'MYAW', 'AC2', 'PRODUCT', 'L3');
execute dbms_awm.add_awcubeagg_spec_level
          ('AGG1', 'MYSCHEMA', 'MYAW', 'AC2', 'PRODUCT', 'L2');
execute dbms_awm.add_awcubeagg_spec_level
          ('AGG1', 'MYSCHEMA', 'MYAW', 'AC2', 'CHANNEL', 'STANDARD_2');
execute dbms_awm.add_awcubeagg_spec_level
          ('AGG1', 'MYSCHEMA', 'MYAW', 'AC2', 'TIME', 'L2');
execute dbms_awm.add_awcubeagg_spec_measure
```

```
          ('AGG1', 'XADEMOAW', 'UK', 'AC2', 'XXF_COSTS');
execute dbms_awm.add_awcubeagg_spec_measure
          ('AGG1', 'XADEMOAW', 'UK', 'AC2', 'XXF_QUOTA');
execute dbms_awm.aggregate_awcube('MYSCHEMA', 'MYAW', 'AC2', 'AGG1');
```

The following statements show the measures and the `PRODUCT` levels in the aggregation plan in the analytic workspace.

```
execute dbms_aw.execute  ('aw attach MYSCHEMA.MYAW ro');
execute dbms_aw.execute  ('fulldsc agg1');

DEFINE AGG1 DIMENSION TEXT
LD List of Measures which use this AggPlan
PROPERTY 'AW$CLASS' -
'IMPLEMENTATION'
PROPERTY 'AW$CREATEDBY' -
'AW$CREATE'
PROPERTY 'AW$LASTMODIFIED' -
'.*
PROPERTY 'AW$LOGICAL_NAME' -
'AGG1'
PROPERTY 'AW$PARENT_NAME' -
'AC2'
PROPERTY 'AW$ROLE' -
'AGGDEF'
PROPERTY 'AW$STATE' -
'ACTIVE'

execute dbms_aw.execute('rpr agg1')

AGG1
--------------
XXF.COSTS
XXF.QUOTA

execute dbms_aw.execute('fulldsc agg1_product');

DEFINE AGG1_PRODUCT VALUESET PRODUCT_LEVELLIST
LD List of Levels for this AggPlan
PROPERTY 'AW$AGGOPERATOR' -
'SUM'
PROPERTY 'AW$CLASS' -
'IMPLEMENTATION'
PROPERTY 'AW$CREATEDBY' -
'AW$CREATE'
```

```
PROPERTY 'AW$LASTMODIFIED' -
'.*
PROPERTY 'AW$PARENT_CUBE' -
'AC2'
PROPERTY 'AW$PARENT_DIM' -
'PRODUCT'
PROPERTY 'AW$PARENT_NAME' -
'AGG1'
PROPERTY 'AW$ROLE' -
'AGGDEF_LEVELS'
PROPERTY 'AW$STATE' -
'ACTIVE'

execute dbms_aw.execute('shw values(agg1_product)');

L3
L2
```

Choosing an Aggregation Method

An aggregation method specifies the operation used to summarize the data by level. The default aggregation method is addition. For example, sales data is typically aggregated over time by adding the values for each time period.

The OLAP Catalog supports a set of aggregation methods, which may be included to the definition of a cube. These aggregation methods are listed in Table 1–10.

When a workspace cube is refreshed, the aggregation operators specified in the OLAP Catalog are converted to the corresponding operators supported by the OLAP DML `RELATION` command. These operators are incorporated in the aggregation map that controls dynamic aggregation for the cube.

To specify a different operator for your stored aggregates, you can use the `SET_AWCUBEAGG_SPEC_AGGOP` procedure. This procedure enables you to specify any of the operators supported by the OLAP DML `RELATION` command to preaggregate your data.

Note: The DBMS_AWM package currently does not support weighted aggregation operators. For example, if the OLAP Catalog specifies a weighted sum or weighted average for aggregation along one of the cube's dimensions, it is converted to the scalar equivalent (sum or average) when the cube is refreshed in the analytic workspace. Weighted operators specified by SET_AWCUBEAGG_SPEC_AGGOP are similarly converted.

The OLAP Catalog and corresponding OLAP DML aggregation operators are described in Table 1–10.

Table 1–10 Aggregation Operators

OLAP Catalog	OLAP DML	DML Abbvr	Description
SUM	SUM	SU	Sum. Adds data values (default)
SCALED SUM	SSUM	SS	Converted to Sum.
WEIGHTED SUM	WSUM	WS	Converted to Sum.
AVERAGE	AVERAGE	AV	Average. Adds data values, then divides the sum by the number of data values that were added together.
HIERARCHICAL AVERAGE	HAVERAGE	HA	Hierarchical Average. Adds data values, then divides the sum by the number of the children in the dimension hierarchy.
WEIGHTED AVERAGE	WAVERAGE	WA	Converted to Average.
	HWAVERAGE	HW	Converted to Hierarchical Average.
MAX	MAX	MA	Maximum. The largest data value among the children of any parent data value.
MIN	MIN	MI	Minimum. The smallest data value among the children of any parent data value.
FIRST	FIRST	FI	First. The first non-NA data value.
	HFIRST	HF	Hierarchical First. The first data value that is specified by the hierarchy, even if that value is NA.
LAST	LAST	LA	Last. The last non-NA data value.
	HLAST	HL	Hierarchical Last. The last data value that is specified by the hierarchy, even if that value is NA.

Table 1–10 (Cont.) Aggregation Operators

OLAP Catalog	OLAP DML	DML Abbvr	Description
AND	AND	AN	(Boolean variables only) If any child data value is FALSE, then the data value of its parent is FALSE. A parent is TRUE only when all of its children are TRUE.
OR	OR	OR	(Default for Boolean variables) If any child data value is TRUE, then the data value of its parent is TRUE. A parent is FALSE only when all of its children are FALSE.
COUNT		NO	Converted to NOAGG.
	NOAGG	NO	Do not aggregate any data for this dimension.

Creating Relational Access to the Workspace Cube

Once you have created an analytic workspace cube and refreshed and aggregated its data, you can generate views that will allow applications to access that data using standard SQL. The DBMS_AWM procedures that generate the views are known as the **OLAP API Enabler** procedures. They generate views and OLAP Catalog metadata in the format required by the OLAP API and BI Beans, as follows.

- An embedded total dimension view for each dimension hierarchy.
- An embedded total fact view for each combination of dimension hierarchies.

If your analytic workspace will support different applications, then you need to generate views that conform to their requirements. You can use the OLAP_TABLE function, described in Chapter 26, to generate views in a variety of different formats.

To enable a workspace cube, you can either generate the scripts and run them yourself or you can use a one-step procedure to create and run the scripts automatically.

Procedure: Generate and Run the Enablement Scripts

Use the following steps to enable a workspace cube for access by the OLAP API and BI Beans:

1. Determine how your system is configured to write to files. The enabler procedures accept either a directory object or a directory path. If you specify a directory object, make sure that your user ID has been granted the appropriate access rights to it. If you specify a path, make sure that it is the value of the UTL_FILE_DIR initialization parameter for the instance.

2. Run the REFRESH_AWCUBE and REFRESH_AWDIMENSION procedures to refresh the cube. These procedures create metadata in the analytic workspace to track the generations of enablement view names.

   NOTE: If you use some other process to refresh the cube (for example, the OLAP Analytic Workspace Java API), this metadata is not created. If you want to specify your own names for the enablement views (as described in the following step), you must create this metadata by calling the REFRESH_AWDIMENSION_VIEW_NAME and REFRESH_AWCUBE_VIEW_NAME procedures.

3. The enablement process automatically provides system-generated names for the enablement views. To provide your own view names, call the SET_AWDIMENSION_VIEW_NAME and SET_AWCUBE_VIEW_NAME procedures.

4. Call the CREATE_AWDIMENSION_ACCESS procedure for each of the cube's dimensions. Set the access_type parameter to OLAP. Each procedure call will create an enablement script in a directory that you specify. The script will contain statements that create the dimension views and an OLAP Catalog dimension that maps to the views.

5. Call the CREATE_AWCUBE_ACCESS procedure. Set the access_type parameter to OLAP. This procedure call will create an enablement script in a directory that you specify. The script will contain statements that create the fact views and an OLAP Catalog cube that maps to the views.

6. Run the enablement scripts. The scripts will delete any previous generation of views and metadata before creating new views and metadata.

Procedure: Run the Enablement Scripts Automatically

To create and run the enablement scripts automatically, use the following steps:

1. Refresh the cube and its dimensions in the analytic workspace, as described in "Procedure: Generate and Run the Enablement Scripts" on page 1-23.

2. Call CREATE_AWDIMENSION_ACCESS_FULL for each of the cube's dimensions. This procedure creates the enablement scripts in temporary memory and runs the scripts to create the dimension views and OLAP Catalog metadata. The scripts delete any previous views and OLAP Catalog metadata before creating new views and metadata.

3. Call the procedure CREATE_AWCUBE_ACCESS_FULL to create the fact views for the cube. This procedure accomplishes the same basic steps as the corresponding procedure for dimensions.

The OLAP API Enabler Procedures

The OLAP API enabler procedures are listed in Table 1–11.

Table 1–11 The OLAP API Enabler Procedures

Procedure	Description
CREATE_AWCUBE_ACCESS Procedure	Creates a script that enables access to a cube in an analytic workspace.
CREATE_AWCUBE_ACCESS_FULL Procedure	Enables access to a cube in an analytic workspace.
CREATE_AWDIMENSION_ACCESS Procedure	Creates a script that enables access to a dimension in an analytic workspace.
CREATE_AWDIMENSION_ACCESS_FULL Procedure	Enables access to a dimension in an analytic workspace.
DELETE_AWCUBE_ACCESS Procedure	Creates a script that deletes the enablement views and metadata for a cube in an analytic workspace.
DELETE_AWCUBE_ACCESS_ALL Procedure	Deletes the enablement views and metadata for a cube in an analytic workspace.
DELETE_AWDIMENSION_ACCESS Procedure	Creates a script that deletes the enablement views and metadata for a dimension in an analytic workspace.
DELETE_AWDIMENSION_ACCESS_ALL Procedure	Deletes the enablement views and metadata for a dimension in an analytic workspace.
REFRESH_AWCUBE_VIEW_NAME Procedure	Creates metadata in the analytic workspace to support user-defined view names. (Not for use with `REFRESH_AWCUBE`)
REFRESH_AWDIMENSION_VIEW_NAME Procedure	Creates metadata in the analytic workspace to support user-defined view names. (Not for use with `REFRESH_AWDIMENSION`)
SET_AWCUBE_VIEW_NAME Procedure	Replaces the system-generated names for the views of an analytic workspace cube.
SET_AWDIMENSION_VIEW_NAME Procedure	Replaces the system-generated names for the views of an analytic workspace dimension.

Note: If you capture the SQL generated by Analytic Workspace Manager and use it to create your own scripts, you will need to edit the enablement procedure calls. Analytic Workspace Manager uses different versions of the enablement procedures. In your scripts, you must use the syntax described in this manual.

Enablement Metadata in the Analytic Workspace

The `REFRESH_AWDIMENSION` and `REFRESH_AWCUBE` procedures create metadata in the analytic workspace related to enablement. This metadata includes a set of default names for the views that will be created by the enablement scripts.

Whenever you refresh, new view names are generated. If you have previously created your own names (`SET_AWDIMENSION_VIEW_NAME` and `SET_AWCUBE_VIEW_NAME`), the refresh process uses them as the basis for the new names.

Note: If you use some other process to refresh the cube (for example, the OLAP Analytic Workspace Java API), you must run `REFRESH_AWDIMENSION_VIEW_NAME` and `REFRESH_AWCUBE_VIEW_NAME` before setting the view names .

If you refresh and there has been no change to the source cube's metadata, you do not need to re-create the enablement scripts.

Disabling Relational Access

The enablement procedures automatically delete any previous generation of views and OLAP Catalog metadata. However, in some circumstances, you might want to drop the views and metadata without re-creating them. In particular, if you drop the workspace cube or the workspace itself, you will need to clean up the orphaned views and metadata.

In this case, you can run the `DELETE_AWDIMENSION_ACCESS` and `DELETE_AWCUBE_ACCESS` procedures to generate scripts that will drop the views and metadata that enable relational access to the cube. These scripts do not delete any enablement metadata that is stored within the analytic workspace.

To delete all the enablement views and metadata for a dimension or a cube, use `DELETE_AWCUBE_ACCESS_ALL` and `DELETE_AWDIMENSION_ACCESS_ALL`.

Default Dimension View Names

`REFRESH_AWDIMENSION` constructs default names for the views. You can override the default names by calling `SET_AWDIMENSION_VIEW_NAME`.

The default view name is: *aaaa_bbbbb_ccccc_ddddd#*`view`, where:

aaaa is the first four characters of the analytic workspace owner

bbbbb is the first five characters of the analytic workspace name

ccccc is the first five characters of the analytic workspace dimension name

ddddd is the first five characters of the analytic workspace hierarchy name

is an automatically-generated sequence number between 1 and 9,999 to ensure uniqueness.

Default names are also generated for the abstract objects (ADTs) populated by `OLAP_TABLE`. For example, the workspace dimension `AWGEOG`, in a workspace called `AWTEST` in the `XADEMO` schema could have the following system-generated names for the `STANDARD` hierarchy.

Default Name	Description
`XADE_AWTES_AWGE0_STAND34VIEW`	Name of the relational view
`XADE_AWTES_AWGEOG34OBJ`	Name of the abstract object that defines a row in the abstract table of objects populated by `OLAP_TABLE`
`XADE_AWTES_AWGEOG34TBL`	Name of the abstract table type populated by `OLAP_TABLE`

Default Fact View Names

The `REFRESH_AWCUBE` procedure constructs default names for the views. You can override the default names by calling `SET_AWCUBE_VIEW_NAME`.

The default view name is: *aaaa_bbbbb_cccccccc#*`view`, where:

aaaa is the first four characters of the analytic workspace owner

bbbbb is the first five characters of the analytic workspace name

cccccccc is the first eight characters of the analytic workspace cube name

is an automatically-generated sequence number between 1 and 9,999 to ensure uniqueness.

Default names are also generated for the abstract objects (ADTs) populated by OLAP_TABLE. For example, the workspace cube AWCUBE, in a workspace called AWTEST in the XADEMO schema could have the following system-generated names.

Default Name	Description
XADE_AWTES_AWCUBE8VIEW	Name of the relational fact view for the first hierarchy combination.
XADE_AWTES_AWCUBE9VIEW	Name of the relational fact view for the second hierarchy combination.
XADE_AWTES_AWCUBE10VIEW	Name of the relational fact view for the third hierarchy combination.
XADE_AWTES_AWCUBE11VIEW	Name of the relational fact view for the fourth hierarchy combination.
XADE_AWTES_AWCUBE7OBJ	Name of the abstract object that defines a row in the abstract table of objects populated by OLAP_TABLE
XADE_AWTES_AWCUBE7TBL	Name of the abstract table type populated by OLAP_TABLE

Column Structure of Dimension Enablement Views

The enablement process generates a separate view for each dimension hierarchy. For example, a workspace cube with the four dimensions shown in Table 1–12 would have six separate dimension views since two of the dimensions have two hierarchies.

Table 1–12 Sample Dimension Hierarchies

Dimensions	Hierarchies	Number of Views
geography	standard	2
	consolidated	
product	standard	1
channel	standard	1
time	standard	2
	ytd	

The dimension views are level-based, and they include the full lineage of every level value in every row. This type of dimension table is considered **solved**, because the fact table related to this dimension includes embedded totals for all level combinations.

Each dimension view contains the columns described in Table 1–13.

Table 1–13 Dimension View Columns

Column	Description
ET key	The embedded-total key column stores the value of the lowest populated level in the row.
Parent ET key	The parent embedded-total key column stores the parent of each ET key value.
GID	The grouping ID column identifies the hierarchy level associated with each row, as described in "Grouping ID Column" on page 1-30.
Parent GID	The parent grouping ID column stores the parent of each GID value.
level columns	A column for each level of the dimension hierarchy. These columns provide the full ancestry of each dimension member within a single row.
level attribute columns	A column for each level attribute.

Sample Dimension View

For a standard geography hierarchy with levels for `TOTAL_US`, `REGION`, and `STATE`, the dimension view would contain columns like the ones that follow. Level attribute columns would also be included.

```
GID PARENT_GID  ET KEY     PARENT_ET_KEY TOTAL_US  REGION     STATE
--- ----------  ------     ------------  --------  ------     -----
0   1           MA         Northeast     USA       Northeast  MA
0   1           NY         Northeast     USA       Northeast  NY
0   1           GA         Southeast     USA       Southeast  GA
0   1           CA         Southwest     USA       Southwest  CA
0   1           AZ         Southwest     USA       Southwest  AZ
1   3           Northeast  USA           USA       Northeast
1   3           Southeast  USA           USA       Southeast
1   3           Southwest  USA           USA       Southwest
3   NA          USA        NA            USA
```

Grouping ID Column

The GID identifies the hierarchy level associated with each row by assigning a zero to each non-null value and a one to each null value in the level columns. The resulting binary number is the value of the GID.

For example, a GID of 1 is assigned to a row with the following three levels.

```
TOTAL_US  REGION      STATE
--------  ------      -----
USA       Southwest
0         0           1
```

A GID of 3 is assigned to a row with the following five levels.

```
TOTAL_GEOG COUNTRY REGION     STATE  CITY
---------- ------- -------    ------ -------
World      USA     Northeast
0          0       0           1      1
```

Column Structure of Enablement Fact Views

The `CREATE_AWCUBE_ACCESS` procedure generates a separate view for each dimension/hierarchy combination. For example, an analytic workspace cube with the four dimensions shown in Table 1–12, would have four separate fact views, one for each hierarchy combination show in Table 1–14.

Table 1–14 Sample Dimension/Hierarchy Combinations

Geography Dim	Product Dim	Channel Dim	Time Dim
geography/ standard	product/standard	channel/standard	time/standard
geography/ standard	product/standard	channel/standard	time/ytd
geography/ consolidated	product/standard	channel/standard	time/standard
geography/ consolidated	product/standard	channel/standard	time/ytd

The fact views are fully **solved**. They contain embedded totals for all level combinations. Each view has columns for the cube's measures, and key columns that link the fact view with its associated dimension views.

Each fact view contains the columns described in Table 1–15.

Table 1–15 Fact View Columns

Column	Description
ET key for each dimension/hierarchy	The ET key columns are foreign keys that map to the primary keys of the associated dimension tables, and are used to join the measure table with the dimension tables.
GID for each dimension/hierarchy	The GID column provides grouping IDs needed by the OLAP API for optimal response time. It is identical to the GID column of the associated dimension table.
measure columns	Columns for each of the cube's measures.
R2C	Information needed to dynamically calculate custom measures. See the ROWTOCELL keyword described in Table 26–3, " Components of the OLAP_TABLE Limit Map".
CUST_MEAS_TEXT*n*	100 sequentially numbered empty columns with a data type of VARCHAR2(1000). These columns return predefined custom measures with a text data type. These custom measures result from the execution of a formula within the analytic workspace and are managed by procedures in the DBMS_AW_UTILITIES package. For more information, see Chapter 22.
CUST_MEAS_NUM*n*	100 sequentially numbered empty columns with a data type of NUMBER(38,6). These columns return predefined custom measures with a numeric data type. These custom measures result from the execution of a formula within the analytic workspace and are managed by procedures in the DBMS_AW_UTILITIES package. For more information, see Chapter 22.

Example: Enable a Workspace Cube for Access by the OLAP API

The following example creates, refreshes, and enables a cube AWUSR.AWTEST based on the source cube XADEMO.ANALYTIC_CUBE.

Example 1–5 Create, Refresh, and Enable a Cube

```
-- SET UP
set serveroutput on size 1000000
execute cwm2_olap_manager.set_echo_on;
execute cwm2_olap_manager.begin_log ('/users/awuser/scripts' , 'awtest.log');

--- CREATE AW
execute dbms_aw.execute ('aw create ''AWTEST''');
```

```
-- CREATE DIMENSIONS
execute dbms_awm.create_awdimension
          ('XADEMO','CHANNEL',   'AWUSR', 'AWTEST', 'AWCHAN');
execute dbms_awm.create_awdimension
          ('XADEMO','GEOGRAPHY', 'AWUSR', 'AWTEST', 'AWGEOG');
execute dbms_awm.create_awdimension
          ('XADEMO','PRODUCT',   'AWUSR', 'AWTEST', 'AWPROD');
execute dbms_awm.create_awdimension
          ('XADEMO','TIME',      'AWUSR', 'AWTEST', 'AWTIME');

-- CREATE CUBE
execute dbms_awm.create_awcube
          ('XADEMO', 'ANALYTIC_CUBE','AWUSR', 'AWTEST','AWCUBE');

-- REFRESH DIMENSIONS
execute dbms_awm.refresh_awdimension ('AWUSR', 'AWTEST', 'AWCHAN');
execute dbms_awm.refresh_awdimension ('AWUSR', 'AWTEST', 'AWGEOG');
execute dbms_awm.refresh_awdimension ('AWUSR', 'AWTEST', 'AWPROD');
execute dbms_awm.refresh_awdimension ('AWUSR', 'AWTEST', 'AWTIME');

-- REFRESH CUBE
execute dbms_awm.refresh_awcube ('AWUSR', 'AWTEST', 'AWCUBE');

-- SET DIMENSION VIEW NAMES
exec dbms_awm.set_awdimension_view_name
          ('AWUSR', 'AWTEST', 'awprod', 'standard',     'prod_std_view');
exec dbms_awm.set_awdimension_view_name
          ('AWUSR', 'AWTEST', 'awchan', 'standard',     'chan_std_view');
exec dbms_awm.set_awdimension_view_name
          ('AWUSR', 'AWTEST', 'awgeog', 'consolidated', 'geog_csd_view');
exec dbms_awm.set_awdimension_view_name
          ('AWUSR', 'AWTEST', 'awgeog', 'standard',     'geog_std_view');
exec dbms_awm.set_awdimension_view_name
          ('AWUSR', 'AWTEST', 'awtime', 'standard',     'time_std_view');
exec dbms_awm.set_awdimension_view_name
          ('AWUSR', 'AWTEST', 'awtime', 'ytd',          'time_ytd_view');

-- SET CUBE VIEW NAMES
exec dbms_awm.set_awcube_view_name
          ('AWUSR', 'AWTEST', 'awcube', 1, 'AWCUBE_view1');
exec dbms_awm.set_awcube_view_name
          ('AWUSR', 'AWTEST', 'awcube', 2, 'AWCUBE_view2');
exec dbms_awm.set_awcube_view_name
          ('AWUSR', 'AWTEST', 'awcube', 3, 'AWCUBE_view3');
```

```
exec dbms_awm.set_awcube_view_name
          ('AWUSR', 'AWTEST', 'awcube', 4, 'AWCUBE_view4');

-- ENABLE DIMENSIONS
exec dbms_awm.create_AWdimension_access
          ('AWUSR', 'AWTEST', 'awprod', 'olap',
           '/users/awuser/scripts', 'awprod_views.sql', 'w');
exec dbms_awm.create_AWdimension_access
          ('AWUSR', 'AWTEST', 'awchan', 'olap',
           '/users/awuser/scripts', 'awchan_views.sql', 'w');
exec dbms_awm.create_AWdimension_access
          ('AWUSR', 'AWTEST', 'awgeog', 'olap',
           '/users/awuser/scripts', 'awgeog_views.sql', 'w');
exec dbms_awm.create_AWdimension_access
          ('AWUSR', 'AWTEST', 'awtime', 'olap',
           '/users/awuser/scripts', 'awtime_views.sql', 'w');

-- ENABLE CUBE
exec dbms_awm.create_AWcube_access
          ('AWUSR', 'AWTEST', 'awcube', 'olap',
           '/users/awuser/scripts', 'awcube_views.sql', 'w');

-- COMMIT and WRAPUP
commit;
execute cwm2_olap_manager.end_log;
```

The following queries show the resulting workspace cube and dimensions with their source cubes and dimensions in the OLAP Catalog.

```
select * from all_olap2_aw_dimensions where AW_OWNER = 'AWUSER';

AW_OWNER  AW_NAME  AW_LOGICAL_NAME AW_PHYSICAL_OBJECT SOURCE_OWNER  SOURCE_NAME
--------- --------- -------------- ----------------- ------------ ---------------
AWUSER     AWTEST    AWCHAN          AWCHAN             XADEMO        CHANNEL
AWUSER     AWTEST    AWGEOG          AWGEOG             XADEMO        GEOGRAPHY
AWUSER     AWTEST    AWPROD          AWPROD             XADEMO        PRODUCT
AWUSER     AWTEST    AWTIME          AWTIME             XADEMO        TIME

select * from all_olap2_aw_CUBEs where AW_OWNER = 'AWUSER';

AW_OWNER AW_NAME  AW_LOGICAL_NAME AW_PHYSICAL_OBJECT  SOURCE_OWNER  SOURCE_NAME
-------- -------- -------------- ------------------ ------------ ---------------
AWUSER    AWTEST   AWCUBE          AWCUBE              XADEMO        ANALYTIC_CUBE
```

The following queries show the system names and user names for the dimension enablement views.

```
select * from all_aw_dim_ENABLED_VIEWS where AW_OWNER = 'AWUSER';

AW_OWNER AW_NAME  DIMENSION_ HIERARCHY_   SYSTEM_VIEWNAME               USER_VIEWNAME
-------- -------- ---------- ----------   ----------------------------  ----------------
AWUSER    AWTEST   AWCHAN     STANDARD     AWUS_AWTES_AWCHA_STAND144VIEW  CHAN_STD_VIEW
AWUSER    AWTEST   AWGEOG     CONSOLIDATED AWUS_AWTES_AWGEO_CONSO145VIEW  GEOG_CSD_VIEW
AWUSER    AWTEST   AWGEOG     STANDARD     AWUS_AWTES_AWGEO_STAND146VIEW  GEOG_STD_VIEW
AWUSER    AWTEST   AWPROD     STANDARD     AWUS_AWTES_AWPRO_STAND147VIEW  PROD_STD_VIEW
AWUSER    AWTEST   AWTIME     STANDARD     AWUS_AWTES_AWTIM_STAND148VIEW  TIME_STD_VIEW
AWUSER    AWTEST   AWTIME     YTD          AWUS_AWTES_AWTIM_YTD149VIEW    TIME_YTD_VIEW
```

The following queries show the system names and user names for the cube enablement views. Included are the hierarchy combination numbers, in this case 1 - 4, and the hierarchy strings, consisting of each unique combination of dimension hierarchies for this cube.

```
select * from all_aw_CUBE_ENABLED_VIEWS where AW_OWNER = 'AWUSER';
```

```
AW_OWN AW_NA  CUBE_NAM  HIER  HIERCOMBO_STR                        SYSTEM_VIEWNAME          USER_VIEWNAME
------ ------ --------  ----  -----------------------------------  -----------------------  -----------
AWUSER  AWTEST  AWCUBE   1     DIM:AWCHAN/HIER:STANDARD;DIM:AWGEOG   AWUS_AWTES_AWCUBE151VIEW  AWCUBE_VIEW1
                              /HIER:CONSOLIDATED;DIM:AWPROD/HIER:
                              STANDARD;DIM:AWTIME/HIER:STANDARD
AWUSER  AWTEST  AWCUBE   2     DIM:AWCHAN/HIER:STANDARD;DIM:AWGEOG   AWUS_AWTES_AWCUBE152VIEW  AWCUBE_VIEW2
                              /HIER:CONSOLIDATED;DIM:AWPROD/HIER:
                              STANDARD;DIM:AWTIME/HIER:YTD
AWUSER   AWTEST  AWCUBE  3     DIM:AWCHAN/HIER:STANDARD;DIM:AWGEOG   AWUS_AWTES_AWCUBE153VIEW  AWCUBE_VIEW3
                              /HIER:STANDARD;DIM:AWPROD/HIER:STAN
                              DARD;DIM:AWTIME/HIER:STANDARD
AWUSER   AWTEST  AWCUBE  4     DIM:AWCHAN/HIER:STANDARD;DIM:AWGEOG   AWUS_AWTES_AWCUBE154VIEW   AWCUBE_VIEW4
                              /HIER:STANDARD;DIM:AWPROD/HIER:STAN
                              DARD;DIM:AWTIME/HIER:YTD
```

The final step is to run the enablement scripts to generate the views and OLAP Catalog metadata for the analytic workspace cube. The scripts produced by this example are described as follows.

Directory	Script	Description
`/users/awuser/scripts`	`awprod_views.sql`	Creates an abstract object, a table of objects, and a view for the `PRODUCT` dimension. Also creates and validates an OLAP Catalog dimension `AWUSER.AWPROD` that maps to the view.
`/users/awuser/scripts`	`awchan_views.sql`	Creates an abstract object, a table of objects, and a view for the `CHANNEL` dimension. Also creates and validates an OLAP Catalog dimension `AWUSER.AWCHAN` that maps to the view.

Directory	Script	Description
/users/awuser/scripts	awgeog_views.sql	Creates an abstract object, a table of objects, and a view for each hierarchy of the GEOGRAPHY dimension. Also creates and validates an OLAP Catalog dimension AWUSER.AWGEOG that maps to the view.
/users/awuser/scripts	awtime_views.sql	Creates an abstract object, a table of objects, and a view for each hierarchy of the TIME dimension. Also creates and validates an OLAP Catalog dimension AWUSER.AWTIME that maps to the view.
/users/awuser/scripts	awcube_views.sql	Creates an abstract object, a table of objects, and a separate view for each hierarchy combination of the AWCUBE cube. Also creates and validates an OLAP Catalog cube AWUSER.AWCUBE that maps to the view.

2

Creating OLAP Catalog Metadata with CWM2

The OLAP Catalog CWM2 PL/SQL packages provide stored procedures for creating, dropping, and updating OLAP metadata. This chapter explains how to work with the CWM2 procedures. For complete syntax descriptions, refer to the reference chapter for each package.

This chapter discusses the following topics:

- OLAP Metadata Entities
- Creating a Dimension
- Creating a Cube
- Mapping OLAP Metadata
- Validating and Committing OLAP Metadata
- Invoking the Procedures
- Directing Output
- Viewing OLAP Metadata

OLAP Metadata Entities

OLAP metadata entities are: **dimensions**, **hierarchies**, **levels**, **level attributes**, **dimension attributes**, **measures**, **cubes**, and **measure folders**. A separate PL/SQL package exists for each type of entity. The package provides procedures for creating, dropping, locking, and specifying descriptions for entities of that type. For example, to create a dimension, you would call CWM2_OLAP_DIMENSION.CREATE_

DIMENSION; to create a level, you would call CWM2_OLAP_LEVEL.CREATE_LEVEL, and so on.

Each entity of metadata is uniquely identified by its owner and its name.

When you create an OLAP metadata entity, you are simply adding a row to an OLAP Catalog table that identifies all the entities of that type. Creating an entity does not fully define a dimension or a cube, nor does it involve any mapping to warehouse dimension tables or fact tables.

Note: All OLAP Catalog metadata entities are defined as VARCHAR(30).

To fully construct a dimension or a cube, you must understand the hierarchical relationships between the component metadata entities.

Creating a Dimension

Creating a dimension entity is only the first step in constructing the OLAP metadata for a dimension. Each dimension must have at least one level. More typically, it will have multiple levels, hierarchies, and attributes. Table 2–1 shows the parent-child relationships between the metadata components of a dimension.

Table 2–1 Hierarchical Relationships Between Components of a Dimension

Parent Entity	Child Entity
dimension	dimension attribute, hierarchy, level
dimension attribute	level attribute
hierarchy	level
level	level attribute

Note: OLAP Catalog dimensions created with theCWM2 procedures are purely logical entities. They have no relationship to database dimension objects. However, OLAP Catalog dimensions created in Enterprise Manager *are* associated with database dimension objects.

Procedure: Create an OLAP Dimension

Generally, you will create hierarchies and dimension attributes after creating the dimension and before creating the dimension levels and level attributes. Once the levels and level attributes are defined, you can map them to columns in one or more warehouse dimension tables. The general steps are as follows:

1. Call procedures in `CWM2_OLAP_DIMENSION` to create the dimension.
2. Call procedures in `CWM2_OLAP_DIMENSION_ATTRIBUTE` to create dimension attributes. In general, you will need to define dimension attributes for `'long description'` and `'short description'`.

   The OLAP API requires the following dimension attributes for embedded total dimension tables (for example, views of analytic workspaces): `'ET Key'`, `'Parent ET Key'`, `'Grouping ID'`, and `'Parent Grouping ID'`. For more information, see Table 11–1, " Reserved Dimension Attributes".
3. Call procedures in `CWM2_OLAP_HIERARCHY` to define hierarchical relationships for the dimension's levels.
4. Call procedures in `CWM2_OLAP_LEVEL` to create levels and assign them to hierarchies.
5. Call procedures in `CWM2_OLAP_LEVEL_ATTRIBUTE` to create level attributes and assign them to dimension attributes. For `'long description'`, `'short description'` and other reserved dimension attributes, create level attributes with the same name for every level.

   The OLAP API requires the following level attributes for embedded total dimension tables (for example, views of analytic workspaces): `'ET Key'`, `'Parent ET Key'`, `'Grouping ID'`, and `'Parent Grouping ID'`. For more information, see Table 14–1, " Reserved Level Attributes".
6. Call procedures in `CWM2_OLAP_TABLE_MAP` to map the dimension's levels and level attributes to columns in dimension tables.

Example: Create a Product Dimension

The PL/SQL statements in Example 2–1 create a logical `CWM2` dimension, `PRODUCT_DIM`, for the `PRODUCTS` dimension table in the `SH` schema.

The following table shows the columns in the `PRODUCTS` table.

Column Name	Data Type
PROD_ID	NUMBER
PROD_NAME	VARCHAR2
PROD_DESC	VARCHAR2
PROD_SUBCATEGORY	VARCHAR2
PROD_SUBCAT_DESC	VARCHAR2
PROD_CATEGORY	VARCHAR2
PROD_CAT_DESC	VARCHAR2
PROD_WEIGHT_CLASS	NUMBER
PROD_UNIT_OF_MEASURE	VARCHAR2
PROD_PACK_SIZE	VARCHAR2
SUPPLIER_ID	NUMBER
PROD_STATUS	VARCHAR2
PROD_LIST_PRICE	NUMBER
PROD_MIN_PRICE	NUMBER
PROD_TOTAL	VARCHAR2

Example 2–1 Create an OLAP Dimension for the Products Table

```
---   CREATE THE PRODUCT DIMENSION    ---
exec cwm2_olap_dimension.create_dimension
          ('SH', 'PRODUCT_DIM', 'Product','Products', 'Product Dimension',
           'Product Dimension Values');

---   CREATE DIMENSION ATTRIBUTES  ---
exec cwm2_olap_dimension_attribute.create_dimension_attribute
          ('SH', 'PRODUCT_DIM', 'Long Description', 'Long Descriptions',
          'Long Desc', 'Long Product Descriptions', true);
exec cwm2_olap_dimension_attribute.create_dimension_attribute
          ('SH', 'PRODUCT_DIM', 'PROD_NAME_DIM', 'Product Name',
          'Prod Name', 'Product Name');

---   CREATE STANDARD HIERARCHY  ---
exec cwm2_olap_hierarchy.create_hierarchy
          ('SH', 'PRODUCT_DIM', 'STANDARD', 'Standard', 'Std Product',
           'Standard Product Hierarchy', 'Unsolved Level-Based');
```

```
exec cwm2_olap_dimension.set_default_display_hierarchy
          ('SH', 'PRODUCT_DIM', 'standard');

---   CREATE LEVELS  ---
exec cwm2_olap_level.create_level
          ('SH', 'PRODUCT_DIM', 'L4', 'Product ID', 'Product Identifiers',
          'Prod Key','Product Key');
exec cwm2_olap_level.create_level
          ('SH', 'PRODUCT_DIM', 'L3','Product Sub-Category',
          'Product Sub-Categories','Prod Sub-Category',
          'Sub-Categories of Products');
exec cwm2_olap_level.create_level
          ('SH', 'PRODUCT_DIM', 'L2','Product Category',
           'Product Categories', 'Prod Category', 'Categories of Products');
exec cwm2_olap_level.create_level
          ('SH', 'PRODUCT_DIM', 'L1', 'Total Product', 'Total Products',
           'Total Prod', 'Total Product');

---   CREATE LEVEL ATTRIBUTES  ---
exec cwm2_olap_level_attribute.create_level_attribute
          ('SH', 'PRODUCT_DIM', 'Long Description', 'L4', 'Long Description',
          'PRODUCT_LABEL', 'L4 Long Desc',
          'Long Labels for PRODUCT Identifiers', TRUE);
exec cwm2_olap_level_attribute.create_level_attribute
          ('SH', 'PRODUCT_DIM', 'Long Description', 'L3', 'Long Description',
          'SUBCATEGORY_LABEL', 'L3 Long Desc',
          'Long Labels for PRODUCT Sub-Categories', TRUE);
exec cwm2_olap_level_attribute.create_level_attribute
          ('SH', 'PRODUCT_DIM', 'Long Description', 'L2', 'Long Description',
          'CATEGORY_LABEL', 'L2 Long Desc',
          'Long Labels for PRODUCT Categories', TRUE);
exec cwm2_olap_level_attribute.create_level_attribute
          ('SH', 'PRODUCT_DIM', 'PROD_NAME_DIM', 'L4', 'PROD_NAME_LEV',
          'Product Name', 'Product Name', 'Product Name');

---   ADD LEVELS TO HIERARCHIES  ---
exec cwm2_olap_level.add_level_to_hierarchy
          ('SH', 'PRODUCT_DIM', 'STANDARD', 'L4', 'L3');
exec cwm2_olap_level.add_level_to_hierarchy
          ('SH', 'PRODUCT_DIM', 'STANDARD', 'L3', 'L2');
exec cwm2_olap_level.add_level_to_hierarchy
          ('SH', 'PRODUCT_DIM', 'STANDARD', 'L2', 'L1');
exec cwm2_olap_level.add_level_to_hierarchy
          ('SH', 'PRODUCT_DIM', 'STANDARD', 'L1');
```

```
---   CREATE MAPPINGS  ---
exec cwm2_olap_table_map.Map_DimTbl_HierLevel
          ('SH', 'PRODUCT_DIM', 'STANDARD', 'L4',
          'SH', 'PRODUCTS', 'PROD_ID');
exec cwm2_olap_table_map.Map_DimTbl_HierLevelAttr
          ('SH', 'PRODUCT_DIM', 'Long Description', 'STANDARD',
           'L4', 'Long Description', 'SH', 'PRODUCTS', 'PROD_DESC');
exec cwm2_olap_table_map.Map_DimTbl_HierLevelAttr
          ('SH', 'PRODUCT_DIM', 'PROD_NAME_DIM', 'STANDARD', 'L4',
          'PROD_NAME_LEV', 'SH', 'PRODUCTS', 'PROD_NAME');
exec cwm2_olap_table_map.Map_DimTbl_HierLevel
          ('SH', 'PRODUCT_DIM', 'STANDARD', 'L3','SH', 'PRODUCTS',
          'PROD_SUBCATEGORY');
exec cwm2_olap_table_map.Map_DimTbl_HierLevelAttr
          ('SH', 'PRODUCT_DIM', 'Long Description', 'STANDARD', 'L3',
          'Long Description', 'SH', 'PRODUCTS', 'PROD_SUBCATEGORY_DESC');
exec cwm2_olap_table_map.Map_DimTbl_HierLevel
          ('SH', 'PRODUCT_DIM', 'STANDARD', 'L2','SH', 'PRODUCTS',
          'PROD_CATEGORY');
exec cwm2_olap_table_map.Map_DimTbl_HierLevelAttr
          ('SH', 'PRODUCT_DIM', 'Long Description', 'STANDARD', 'L2',
          'Long Description', 'SH', 'PRODUCTS', 'PROD_CATEGORY_DESC');
exec cwm2_olap_table_map.Map_DimTbl_HierLevel
          ('SH', 'PRODUCT_DIM', 'STANDARD', 'L1','SH', 'PRODUCTS',
          'PROD_TOTAL');
```

Procedure: Create a Time Dimension

When constructing metadata for your time dimension tables, you will follow the same general procedure as for any other OLAP dimension. However, several additional requirements apply. The general steps for creating a time dimension are as follows:

1. Call procedures in `CWM2_OLAP_DIMENSION` to create the dimension. Specify `'TIME'` for the dimension type parameter.
2. Call procedures in `CWM2_OLAP_DIMENSION_ATTRIBUTE` to create dimension attributes. In addition to the dimension attributes needed for regular dimensions, define an `'End Date'` attribute and a '`Time Span`' attribute.
3. Call procedures in `CWM2_OLAP_HIERARCHY` to define hierarchical relationships for the dimension's levels. Typical hierarchies are Calendar and Fiscal.
4. Call procedures in `CWM2_OLAP_LEVEL` to create levels and assign them to hierarchies. Typical levels are Month, Quarter, and Year.

5. Call procedures in CWM2_OLAP_LEVEL_ATTRIBUTE to create level attributes and assign them to dimension attributes. In addition to the level attributes needed for regular dimension attributes, create 'End Date' and 'Time Span' attributes for each level and associate them with the 'End Date' and 'Time Span' dimension attributes.

6. Call procedures in CWM2_OLAP_TABLE_MAP to map the dimension's levels and level attributes to columns in dimension tables. Map the 'End Date' level attributes to columns with a Date data type. Map the 'Time Span' level attributes to columns with a numeric data type.

Example: Create a Time Dimension

The PL/SQL statements in Example 2–1 create a logical CWM2 time dimension, TIME_DIM, for the TIMES dimension table in the SH schema.

The TIMES table includes the following columns.

Column Name	Data Type
TIME_ID	DATE
TIME_ID_KEY	NUMBER
DAY_NAME	VARCHAR2(9)
CALENDAR_MONTH_NUMBER	NUMBER(2)
CALENDAR_MONTH_DESC	VARCHAR2(8)
CALENDAR_MONTH_DESC_KEY	NUMBER
END_OF_CAL_MONTH	DATE
CALENDAR_MONTH_NAME	VARCHAR2(9)
CALENDAR_QUARTER_DESC	CHAR(7)
CALENDAR_QUARTER_DESC_KEY	NUMBER
END_OF_CAL_QUARTER	DATE
CALENDAR_QUARTER_NUMBER	NUMBER(1)
CALENDAR_YEAR	NUMBER(4)
CALENDAR_YEAR_KEY	NUMBER
END_OF_CAL_YEAR	DATE

Example 2–2 Create an OLAP TIme Dimension

```
---   CREATE THE TIME DIMENSION
exec cwm2_olap_dimension.create_dimension
          ('SH', 'TIME_DIM', 'Time','Time', 'Time Dimension',
           'Time Dimension Values', 'TIME');

---   CREATE DIMENSION ATTRIBUTE END DATE
exec cwm2_olap_dimension_attribute.create_dimension_attribute
          ('SH', 'TIME_DIM', 'END DATE', 'End Date',
          'End Date', 'Last date of time period', true);

---   CREATE CALENDAR HIERARCHY
exec cwm2_olap_hierarchy.create_hierarchy
          ('SH', 'TIME_DIM', 'CALENDAR', 'Calendar', 'Calendar Hierarchy',
           'Calendar Hierarchy', 'Unsolved Level-Based');
exec cwm2_olap_dimension.set_default_display_hierarchy
          ('SH', 'TIME_DIM', 'CALENDAR');

---   CREATE LEVELS
exec cwm2_olap_level.create_level
          ('SH', 'TIME_DIM', 'MONTH', 'Month', 'Months', 'Month','Month');
exec cwm2_olap_level.create_level
          ('SH','TIME_DIM','QUARTER','Quarter','Quarters','Quarter','Quarter');
exec cwm2_olap_level.create_level
          ('SH', 'TIME_DIM', 'YEAR','Year','Years', 'Year', 'Year');

---   CREATE LEVEL ATTRIBUTES  ---
exec cwm2_olap_level_attribute.create_level_attribute
          ('SH', 'TIME_DIM', 'END DATE', 'Month', 'END DATE',
          'End Date', 'End Date',
          'Last date of time period', TRUE);
exec cwm2_olap_level_attribute.create_level_attribute
          ('SH', 'TIME_DIM', 'END DATE', 'Quarter', 'END DATE',
          'End Date', 'End Date',
          'Last date of time period', TRUE);
exec cwm2_olap_level_attribute.create_level_attribute
          ('SH', 'TIME_DIM', 'END DATE', 'Year', 'END DATE',
          'End Date', 'End Date',
          'Last date of time period', TRUE);

---   ADD LEVELS TO HIERARCHIES
exec cwm2_olap_level.add_level_to_hierarchy
          ('SH', 'TIME_DIM', 'CALENDAR', 'Month', 'Quarter');
exec cwm2_olap_level.add_level_to_hierarchy
```

```
        ('SH', 'TIME_DIM', 'CALENDAR', 'Quarter', 'Year');
exec cwm2_olap_level.add_level_to_hierarchy
        ('SH', 'TIME_DIM', 'CALENDAR', 'Year');

---   CREATE MAPPINGS
exec cwm2_olap_table_map.Map_DimTbl_HierLevel
        ('SH', 'TIME_DIM', 'CALENDAR', 'Year',
        'SH', 'TIMES', 'CALENDAR_YEAR_ID');
exec cwm2_olap_table_map.Map_DimTbl_HierLevelAttr
        ('SH', 'TIME_DIM', 'END DATE', 'CALENDAR',
         'Year', 'END DATE', 'SH', 'TIMES', 'END_OF_CAL_YEAR');
exec cwm2_olap_table_map.Map_DimTbl_HierLevel
        ('SH', 'TIME_DIM', 'CALENDAR', 'Quarter','SH', 'TIMES',
        'CALENDAR_QUARTER_NUMBER');
exec cwm2_olap_table_map.Map_DimTbl_HierLevelAttr
        ('SH', 'TIME_DIM', 'END DATE', 'CALENDAR',
         'Quarter', 'END DATE', 'SH', 'TIMES', 'END_OF_CAL_QUARTER');
exec cwm2_olap_table_map.Map_DimTbl_HierLevel
        ('SH', 'TIME_DIM', 'CALENDAR', 'Month','SH', 'TIMES',
        'CALENDAR_MONTH_NUMBER');
exec cwm2_olap_table_map.Map_DimTbl_HierLevelAttr
        ('SH', 'TIME_DIM', 'END DATE', 'CALENDAR',
         'Month', 'END DATE', 'SH', 'TIMES', 'END_OF_CAL_MONTH');
```

Creating a Cube

Creating a cube entity is only the first step in constructing the OLAP metadata for a cube. Each cube must have at least one dimension and at least one measure. More typically, it will have multiple dimensions and multiple measures.

Procedure: Create a Cube

The general steps for constructing a cube are as follows:

1. Follow the steps in "Procedure: Create an OLAP Dimension" for each of the cube's dimensions.
2. Call procedures in `CWM2_OLAP_CUBE` to create the cube and identify its dimensions.
3. Call procedures in `CWM2_OLAP_MEASURE` to create the cube's measures.
4. Call procedures in `CWM2_OLAP_TABLE_MAP` to map the cube's measures to columns in fact tables and to map foreign key columns in the fact tables to key columns in the dimension tables.

Example: Create a Costs Cube

The PL/SQL statements in Example 2–3 create a logical CWM2 cube object, ANALYTIC_CUBE, for the COSTS fact table in the SH schema. The dimensions of the cube are PRODUCT_DIM, shown in Example 2–1, and TIME_DIM, shown in Example 2–2.

The COSTS fact table has the following columns.

Column Name	Data Type
PROD_ID	NUMBER
TIME_ID	DATE
UNIT_COST	NUMBER
UNIT_PRICE	NUMBER

Example 2–3 Create an OLAP Cube for the COSTS Fact Table

```
---   CREATE THE ANALYTIC_CUBE CUBE  ---
cwm2_olap_cube.create_cube('SH', 'ANALYTIC_CUBE', 'Analytics',
     'Analytic Cube','Unit Cost and Price Analysis');

---  ADD THE DIMENSIONS TO THE CUBE  ---
cwm2_olap_cube.add_dimension_to_cube('SH', 'ANALYTIC_CUBE',
         'SH', 'TIME_DIM');
cwm2_olap_cube.add_dimension_to_cube('SH', 'ANALYTIC_CUBE',
         'SH', 'PRODUCT_DIM');

---   CREATE THE MEASURES  ---
cwm2_olap_measure.create_measure('SH', 'ANALYTIC_CUBE', 'UNIT_COST',
          'Unit Cost','Unit Cost', 'Unit Cost');
cwm2_olap_measure.create_measure('SH', 'ANALYTIC_CUBE', 'UNIT_PRICE',
          'Unit Price','Unit Price', 'Unit Price');

---   CREATE THE MAPPINGS  ---
cwm2_olap_table_map.Map_FactTbl_LevelKey
     ('SH', 'ANALYTIC_CUBE','SH', 'COSTS', 'LOWESTLEVEL',
      'DIM:SH.PRODUCTS/HIER:STANDARD/LVL:L4/COL:PROD_ID;
       DIM:SH.TIME/HIER:CALENDAR/LVL:L3/COL:MONTH;');
cwm2_olap_table_map.Map_FactTbl_Measure
     ('SH', 'ANALYTIC_CUBE','UNIT_COST', 'SH', 'COSTS', 'UNIT_COST',
      'DIM:SH.PRODUCTS/HIER:STANDARD/LVL:L4/COL:PROD_ID;
       DIM:SH.TIME/HIER:CALENDAR/LVL:L3/COL:MONTH;');
cwm2_olap_table_map.Map_FactTbl_Measure
```

```
('SH', 'ANALYTIC_CUBE','UNIT_PRICE', 'SH', 'COSTS', 'UNIT_PRICE',
 'DIM:SH.PRODUCTS/HIER:STANDARD/LVL:L4/COL:PROD_ID;
  DIM:SH.TIME/HIER:CALENDAR/LVL:L3/COL:MONTH;');
```

Mapping OLAP Metadata

OLAP metadata mapping is the process of establishing the links between logical metadata entities and the physical locations where the data is stored. Dimension levels and level attributes map to columns in dimension tables. Measures map to columns in fact tables. The mapping process also specifies the join relationships between a fact table and its associated dimension tables.

Note: The dimension tables and fact tables may be implemented as views. For example, the views you can generate using the `DBMS_AWM` package may be the data source for OLAP metadata. These views project an image of relational fact tables and dimension tables over an analytic workspace, where the data actually resides. For more information, see "CREATE_AWCUBE_ACCESS Procedure" on page 23-22.

Mapping to Columns

The `CWM2_OLAP_TABLE_MAP` package contains the mapping procedures for `CWM2` metadata. Dimension levels, level attributes, and measures can be mapped within the context of a hierarchy or with no hierarchical context.

Mapping Dimensions

Each level maps to one or more columns in a dimension table. All the columns of a multicolumn level must be mapped within the same table. All the levels of a dimension may be mapped to columns in the same table (a traditional star schema), or the levels may be mapped to columns in separate tables (snowflake schema).

Each level attribute maps to a single column in the same table as its associated level.

Mapping Measures

Each measure maps to a single column in a fact table. All the measures mapped within the same fact table must share the same dimensionality.

When more than one hierarchical context is possible within a cube (at least one of the cube's dimensions has multiple hierarchies), each combination of hierarchies may be mapped to a separate fact table. In this case, each table must have columns for each of the cube's measures, and the measure columns must appear in the same order in each table.

Joining Fact Tables with Dimension Tables

Once you have mapped the levels, level attributes, and measures, you can specify the mapping of logical foreign key columns in the fact table to level key columns in dimension tables.

The `MAP_FACTTBL_LEVELKEY` procedure defines the join relationships between a cube and its dimensions. This procedure takes as input: the cube name, the fact table name, a mapping string, and a storage type indicator specifying how data is stored in the fact table.

The storage type indicator can have either of the following values:

- **`'LOWESTLEVEL'`**
  A single fact table stores unsolved data for all the measures of a cube (star schema). If any of the cube's dimensions have more than one hierarchy, they must all have the same lowest level. Each foreign key column in the fact table maps to a level key column in a dimension table.

- **`'ET'`**
  Fact tables store completely solved data (with embedded totals) for specific hierarchies of the cube's dimensions. Typically, the data for each combination of hierarchies is stored in a separate fact table. Each fact table must have the same columns. Multiple hierarchies in dimensions do not have to share the same lowest level.

  An embedded total key and a grouping ID key (GID) in the fact table map to corresponding columns that identify a dimension hierarchy in a solved dimension table. The ET key identifies the lowest level value present in a row. The GID identifies the hierarchy level associated with each row. For more information, see "Grouping ID Column" on page 1-30. For more information on mapping the key relationships between fact tables and dimension tables, see "MAP_FACTTBL_LEVELKEY Procedure" on page 18-9.

  The OLAP API requires certain attributes for ET dimensions. See Table 11–1, " Reserved Dimension Attributes".

When the fact table and dimension tables are joined with a storage type of `LOWESTLEVEL`, the cube's hierarchies have a `solved_code` of '`UNSOLVED LEVEL-BASED`'.

When the fact tables and dimension tables are joined with a storage type of `ET`, the cube's hierarchies have a `solved_code` of `'SOLVED LEVEL-BASED'`.

See "SET_SOLVED_CODE Procedure" on page 12-8.

Validating and Committing OLAP Metadata

None of the `CWM2` procedures that create, map, or validate OLAP metadata includes a `COMMIT`.

To prepare metadata for the OLAP API, your script should first execute all the statements that create and map new metadata, then validate the metadata, then refresh OLAP API Metadata Reader tables. The refresh process includes a `COMMIT`. See "Refreshing Metadata Tables for the OLAP API" on page 2-16.

If you are preparing OLAP metadata for other types of applications, your script should include a `COMMIT` after creating, mapping, and validating the metadata.

Validating OLAP Metadata

To test the validity of OLAP metadata, use the `CWM2_OLAP_VALIDATE` and `CWM2_OLAP_VERIFY_ACCESS` packages. The validation procedures check the structural integrity of the metadata and ensure that it is correctly mapped to columns in dimension tables and fact tables. Additional validation specific to the OLAP API is done if requested.

The `CWM2_OLAP_VERIFY_ACCESS` package performs two additional checks after validating a cube. It checks that the `CWM2` metadata for the cube is consistent with the cached metadata tables queried by the OLAP API Metadata Reader. Additionally, it checks that the calling user has access to the source tables and columns.

See Also:

- "Refreshing Metadata Tables for the OLAP API" on page 2-16
- Chapter 19, "CWM2_OLAP_VALIDATE"
- Chapter 20, "CWM2_OLAP_VERIFY_ACCESS"

Note: Remember to validate metadata created or updated in Enterprise Manager as well as `CWM2` metadata.

When running the validation procedures, you can choose to generate a summary or detailed report of the validation process. See "Directing Output" on page 2-18 for information about viewing output on the screen or writing output to a file.

Example 2–4 shows the statements that validate the PRODUCT dimension in XADEMO and generate a detailed validation report. The report is displayed on the screen and written to a log file.

Example 2–4 Generate a Validation Report for the PRODUCT Dimension

```
set echo on
set linesize 135
set pagesize 50
set serveroutput on size 1000000

execute cwm2_olap_manager.set_echo_on;
execute cwm2_olap_manager.begin_log('/users/myxademo/myscripts' , 'x.log');

execute cwm2_olap_validate.validate_dimension
          ('xademo','product','default','yes');

execute cwm2_olap_manager.end_log;
execute cwm2_olap_manager.set_echo_off;
```

The validation report would look like this.

```
Validate Dimension: XADEMO.PRODUCT    Type of Validation: DEFAULT    Verbose Report: YES
Validating Dimension in OLAP Catalog 1
ENTITY TYPE                     ENTITY NAME                STATUS   COMMENT
  Dimension                     .                          VALID
  Dimension                     XADEMO.PRODUCT             VALID
        LevelAttribute          PROD_STD_TOP_LLABEL        VALID    DimensionAttribute "Long Description"
          LevelAttributeMap                                 VALID    Mapped to Column "XADEMO.XADEMO_PRODUCT
                                                                       .PROD_STD_TOP_LLABEL"
        LevelAttribute          PROD_STD_TOP_SLABEL         VALID    DimensionAttribute "Short Description"
          LevelAttributeMap                                 VALID    Mapped to Column "XADEMO.XADEMO_PRODUCT
                                                                       .PROD_STD_TOP_SLABEL"
    Hierarchy                   STANDARD                   VALID
      Level                     L4                         VALID    Hierarchy depth 1 (Lowest Level)
        LevelMap                                           VALID    Mapped to Column "XADEMO.XADEMO_PRODUCT
                                                                      .PROD_STD_PRODUCT"
        LevelAttribute          PROD_COLOR                 VALID    DimensionAttribute "Color"
          LevelAttributeMap                                VALID    Mapped to Column "XADEMO.XADEMO_PRODUCT
                                                                      .PROD_COLOR"
        LevelAttribute          PROD_SIZE                  VALID    DimensionAttribute "Size"
          LevelAttributeMap                                VALID    Mapped to Column "XADEMO.XADEMO_
                                                                       PRODUCT.PROD_SIZE"
```

```
  LevelAttribute          PROD_STD_PRODUCT_LLABEL    VALID    DimensionAttribute "Long Description"
    LevelAttributeMap                                 VALID    Mapped to Column "XADEMO.XADEMO_PRODUCT
                                                                 .PROD_STD_PRODUCT_LLABEL"
  LevelAttribute          PROD_STD_PRODUCT_SLABEL     VALID    DimensionAttribute "Short Description"
    LevelAttributeMap                                 VALID    Mapped to Column "XADEMO.XADEMO_PRODUCT
                                                                 .PROD_STD_PRODUCT_SLABEL"
Level                     L3                         VALID    Hierarchy depth 2
  LevelMap                                           VALID    Mapped to Column "XADEMO.XADEMO_PRODUCT
                                                                 .PROD_STD_GROUP"
  LevelAttribute          PROD_STD_GROUP_LLABEL      VALID    DimensionAttribute "Long Description"
    LevelAttributeMap                                 VALID    Mapped to Column "XADEMO.XADEMO_PRODUCT
                                                                 .PROD_STD_GROUP_LLABEL"
  LevelAttribute          PROD_STD_GROUP_SLABEL       VALID    DimensionAttribute "Short Description"
    LevelAttributeMap                                 VALID    Mapped to Column "XADEMO.XADEMO_PRODUCT
                                                               .PROD_STD_GROUP_SLABEL"
Level                     L2                         VALID    Hierarchy depth 3
  LevelMap                                           VALID    Mapped to Column "XADEMO.XADEMO_PRODUCT
                                                                 .PROD_STD_DIVISION"
  LevelAttribute          PROD_STD_DIVISION_LLABEL   VALID    DimensionAttribute "Long Description"
    LevelAttributeMap                                 VALID    Mapped to Column "XADEMO.XADEMO_PRODUCT
                                                                 .PROD_STD_DIVISION_LLABEL"
  LevelAttribute          PROD_STD_DIVISION_SLABEL    VALID    DimensionAttribute "Short Description"
    LevelAttributeMap                                 VALID    Mapped to Column "XADEMO.XADEMO_PRODUCT
                                                                 .PROD_STD_DIVISION_SLABEL"
Level                     L1                         VALID    Hierarchy depth 4 (Top Level)
  LevelMap                                           VALID    Mapped to Column "XADEMO.XADEMO_PRODUCT
                                                                 .PROD_STD_TOP"
```

Note: When a metadata entity is invalid, the Comment column of the validation report indicates whether the problem originates with this entity or with a different entity on which it depends. For example, if a level is invalid, its dependent level attributes will also be invalid.

Viewing Validity Status

You can check the validity status of cubes and dimensions by selecting the `INVALID` column of the `ALL_OLAP2_CUBES` and `ALL_OLAP2_DIMENSIONS` views. One of the following values is displayed:

`Y` -- The cube or dimension is invalid.

`N` -- The cube or dimension has met basic validation criteria.

`O` -- The cube has met basic validation criteria and additional criteria specific to the OLAP API.

For more information, see "ALL_OLAP2_CUBES" on page 5-5 and "ALL_OLAP2_DIMENSIONS" on page 5-7.

Refreshing Metadata Tables for the OLAP API

To make your metadata accessible to the OLAP API, use the CWM2_OLAP_METADATA_REFRESH package to refresh the OLAP API Metadata Reader tables.

Views built on these tables present a read API to the OLAP Catalog that is optimized for queries by the OLAP API Metadata Reader. The Metadata Reader views have public synonyms with the prefix MRV_OLAP2. For more information, see Chapter 16.

Note: You must refresh the Metadata Reader tables to ensure access by the OLAP API.

If you have scripts that call the CWM2 APIs to create OLAP metadata, include calls to validate the metadata and refresh the Metadata Reader tables.

If you use Enterprise Manager to create OLAP metadata, you must run the validate and refresh procedures separately, after the metadata has been created.

Invoking the Procedures

When using the OLAP Catalog write APIs, you should be aware of logic and conventions that are common to all the CWM2 procedures.

Security Checks and Error Conditions

Each CWM2 procedure first checks the calling user's security privileges. The calling user must have the OLAP_DBA role. Generally, the calling user must be the entity owner. If the calling user does not meet the security requirements, the procedure fails with an exception. For example, if your identity is jsmith, you cannot successfully execute CWM2_OLAP_HIERARCHY.DROP_HIERARCHY for a hierarchy owned by jjones.

After verifying the security requirements, each procedure checks for the existence of the entity and of its parent entities. All procedures, except CREATE procedures, return an error if the entity does not already exist. For example, if you call CWM2_

OLAP_LEVEL.SET_DESCRIPTION, and the level does not already exist, the procedure will fail.

Size Requirements for Parameters

CWM2 metadata entities are created with descriptions and display names. For example, the CREATE_CUBE procedure in the CWM2_OLAP_CUBE package requires the following parameters:

```
CREATE_CUBE (
          cube_owner            IN   VARCHAR2,
          cube_name             IN   VARCHAR2,
          display_name          IN   VARCHAR2,
          short_description     IN   VARCHAR2,
          description           IN   VARCHAR2);
```

Entity names and descriptions have size limitations based on the width of the columns where they are stored in the OLAP Catalog model tables. The size limitations are listed in Table 2–2.

Table 2–2 Size Limitations of CWM2 Metadata Entities

Metadata Entity	Maximum Size
entity owner	30 characters
entity name	30 characters
display name	30 characters
short description	240 characters
description	2000 characters

Case Requirements for Parameters

You can specify arguments to CWM2 procedures in lower case, upper case, or mixed case.

If the argument is a metadata entity name (for example, dimension_name) or a value that will be used in further processing by other procedures (for example, the solved_code of a hierarchy), the procedure converts the argument to upper case. For all other arguments, the case that you specify is retained.

Directing Output

There are several tools and settings you can use to help you develop and debug your `CWM2` scripts.

You can echo the output and messages from `CWM2` procedures to the SQL buffer. Use the following statement.

```
SQL>exec cwm2_olap_manager.set_echo_on;
```

By default, echoing is turned off. Once you have set echoing on, you can turn it off with the following statement.

```
SQL>exec cwm2_olap_manager.set_echo_off;
```

You can set SQL*Plus to display the contents of the SQL buffer on the screen with the following statement.

```
SQL>set serveroutput on
```

The default and minimum size of the SQL buffer is 2K. You can extend the size up to a maximum of 1MG with the following statement.

```
SQL>set serveroutput on size 1000000
```

You should set `serveroutput` to its maximum size to prevent buffer overflow conditions.

> **See Also:** *SQL*Plus User's Guide and Reference* for more information on setting `serveroutput`.

To accommodate larger amounts of output, you should direct output to a file. Use the following statement.

```
SQL>exec cwm2_olap_manager.begin_log('directory_path','filename');
```

For *`directory_path`* you can specify either a directory object to which your user ID has been granted the appropriate access, or a directory path set by the `UTL_FILE_DIR` initialization parameter for the instance.

To flush the contents of the buffer and turn off logging, use the following statement.

```
SQL>exec cwm2_olap_manager.end_log;
```

Viewing OLAP Metadata

A set of views, identified by the ALL_OLAP2 prefix, presents the metadata in the OLAP Catalog. The metadata may have been created with the CWM2 PL/SQL packages or with Enterprise Manager. The ALL_OLAP2 views are automatically populated whenever changes are made to the metadata.

A second set of views, identified by the MRV_OLAP prefix, also presents OLAP Catalog metadata. However, these views are structured specifically to support fast querying by the OLAP API's Metadata Reader. These views must be explicitly refreshed whenever changes are made to the metadata.

See Also:

- Chapter 5, "OLAP Catalog Metadata Views" for more information on the ALL_OLAP2 views.
- Chapter 16, "CWM2_OLAP_METADATA_REFRESH" for more information on refreshing metadata tables for the OLAP API.

3

Active Catalog Views

This chapter describes the relational views of standard form objects in analytic workspaces. Within the workspace, standard form objects are automatically created and populated by procedures in the DBMS_AWM package.

See Also:

- Chapter 1, "Creating Analytic Workspaces with DBMS_AWM"
- Chapter 23, "DBMS_AWM"
- "Views of Cached Active Catalog Metadata" on page 16-2

This chapter discusses the following topics:

- Standard Form Active Catalog
- Example: Query an Analytic Workspace Cube
- Summary of Active Catalog Views

Standard Form Active Catalog

OLAP processing depends on a data model composed of cubes, measures, dimensions, hierarchies, levels, and attributes. OLAP Catalog metadata defines this logical model for relational sources. Standard form metadata defines the logical model within analytic workspaces.

Procedures in the DBMS_AWM package create and maintain standard form metadata when creating and refreshing dimensions and cubes in analytic workspaces. Whereas OLAP Catalog metadata must be explicitly created by a DBA, standard form metadata is actively generated as part of workspace management. Views of this metadata are commonly referred to as the **Active Catalog**, because they are populated with information that is automatically generated within analytic workspaces.

Active Catalog views use the OLAP_TABLE function to return workspace data in relational format. See Chapter 26 for more information on OLAP_TABLE.

Note: To improve the performance of queries against the Active Catalog, you can refresh the cached metadata tables that underlie the MRV_OLAP2_AW views. For more information, see "Views of Cached Active Catalog Metadata" on page 16-2.

Standard Form Classes

Each standard form workspace object belongs to one of four classes:

- **Implementation class.** Objects in this class implement the logical model.
- **Catalogs class.** Objects in this class hold information about the logical model.
- **Features class.** Objects in this class hold information about specific objects in the logical model.
- **Extensions class.** Objects in this class are proprietary.

Active Catalog and Standard Form Classes

The primary source of information for the Active Catalog views is objects in the Catalogs class. This includes a list of all the cubes, measures, dimensions, levels, and attributes in analytic workspaces.

Active Catalog views also provide information that associates logical objects from the Catalogs class with their source objects in the OLAP Catalog and with their containers in the Implementation class.

Finally, two Active Catalog views provide all the standard form objects and all the properties of those objects.

Note: Active Catalog views provide information about standard form objects in all analytic workspaces accessible to the current user.

See Also:

- *Oracle OLAP Application Developer's Guide* for information about standard form analytic workspaces
- *Oracle OLAP DML Reference* for information about the OLAP DML and the native objects within analytic workspaces

Example: Query an Analytic Workspace Cube

Example 3–1 uses the XADEMO cube ANALYTIC_CUBE to illustrate two Active Catalog views.

Example 3–1 Query the Active Catalog for Information about a Workspace Cube

The following statements create the dimensions in the analytic workspace XADEMO.MY_AW.

```
execute dbms_awm.create_awdimension
           ('XADEMO','CHANNEL','XADEMO', 'MY_AW', 'AW_CHAN');
execute dbms_awm.create_awdimension
          ('XADEMO','PRODUCT','XADEMO', 'MY_AW', 'AW_PROD');
execute dbms_awm.create_awdimension
          ('XADEMO','GEOGRAPHY','XADEMO', 'MY_AW', 'AW_GEOG');
execute dbms_awm.create_awdimension
          ('XADEMO','TIME','XADEMO', 'MY_AW', 'AW_TIME');
```

You can view the logical dimensions in the analytic workspace with the following query.

```
SQL>select * from ALL_OLAP2_AW_DIMENSIONS;

AW_OWNER  AW_NAME  AW_LOGICAL_NAME  AW_PHYSICAL_OBJECT SOURCE_OWNER  SOURCE_NAME
-------- --------  ---------------  ------------------ -----------   ----------
XADEMO     MY_AW      AW_CHAN       AW_CHAN            XADEMO        CHANNEL
XADEMO     MY_AW      AW_PROD       AW_PROD            XADEMO        PRODUCT
XADEMO     MY_AW      AW_GEOG       AW_GEOG            XADEMO        GEOGRAPHY
XADEMO     MY_AW      AW_TIME       AW_TIME            XADEMO        TIME
```

The following statement creates the cube.

```
execute dbms_awm.create_awcube
          ('XADEMO','ANALYTIC_CUBE','XADEMO', 'MY_AW', 'MY_ANALYTIC_CUBE');
```

You can view the logical cube in the analytic workspace with the following query.

```
SQL>select * from ALL_OLAP2_AW_CUBES;

AW_OWNER AW_NAME AW_LOGICAL_NAME  AW_PHYSICAL_OBJECT  SOURCE_OWNER  SOURCE_NAME
-------- ------- ---------------  ------------------  ------------  -----------
XADEMO   MY_AW   MY_ANALYTIC_CUBE MY_ANALYTIC_CUBE    XADEMO       ANALYTIC_CUBE
```

The following query returns the analytic workspace cube with its associated dimensions.

```
SQL>select * from ALL_OLAP2_AW_CUBE_DIM_USES;

AW_OWNER AW_NAME AW_LOGICAL_NAME  DIMENSION_ DIMENSION_ DIMENSION_   DIMENSION_
                                  AW_OWNER   AW_NAME    SOURCE_OWNER SOURCE_NAME
-------- ------- ---------------- ---------  ---------- ------------ -----------
XADEMO   MY_AW   MY_ANALYTIC_CUBE  XADEMO     AW_CHAN    XADEMO       CHANNEL
XADEMO   MY_AW   MY_ANALYTIC_CUBE  XADEMO     AW_GEOG    XADEMO       GEOGRAPHY
XADEMO   MY_AW   MY_ANALYTIC_CUBE  XADEMO     AW_PROD    XADEMO       PRODUCT
XADEMO   MY_AW   MY_ANALYTIC_CUBE  XADEMO     AW_TIME    XADEMO       TIME
```

Summary of Active Catalog Views

The analytic workspace Active Catalog views are summarized in the following table.

Table 3–1 Active Catalog Views

PUBLIC Synonym	Description
ALL_OLAP2_AWS	List of analytic workspaces.
ALL_OLAP2_AW_ATTRIBUTES	List of dimension attributes in analytic workspaces.
ALL_OLAP2_AW_CUBES	List of cubes in analytic workspaces.
ALL_OLAP2_AW_CUBE_AGG_LVL	List of levels in aggregation plans in analytic workspaces.
ALL_OLAP2_AW_CUBE_AGG_MEAS	List of measures in aggregation plans in analytic workspaces.
ALL_OLAP2_AW_CUBE_AGG_OP	List of aggregation operators in aggregation plans in analytic workspaces.
ALL_OLAP2_AW_CUBE_AGG_SPECS	List of aggregation plans in analytic workspaces.
ALL_OLAP2_AW_CUBE_DIM_USES	List of cubes with their associated dimensions in analytic workspaces.
ALL_OLAP2_AW_CUBE_MEASURES	List of cubes with their associated measures in analytic workspaces.
ALL_OLAP2_AW_DIMENSIONS	List of dimensions in analytic workspaces.

Table 3–1 (Cont.) Active Catalog Views

PUBLIC Synonym	Description
ALL_OLAP2_AW_DIM_HIER_LVL_ORD	List of hierarchical levels in analytic workspaces.
ALL_OLAP2_AW_DIM_LEVELS	List of levels in analytic workspaces.
ALL_OLAP2_AW_PHYS_OBJ	List of standard form objects in analytic workspaces.
ALL_OLAP2_AW_PHYS_OBJ_PROP	List of properties associated with standard form objects in analytic workspaces.

ALL_OLAP2_AWS

ALL_OLAP2_AWS provides a list of all the analytic workspaces accessible to the current user. This includes both standard form and non-standard analytic workspaces.

Column	Datatype	NULL	Description
OWNER	VARCHAR2(30)	NOT NULL	Owner of the analytic workspace.
AW	VARCHAR2(30)		Name of the analytic workspace.
AW_ NUMBER	NUMBER	NOT_NULL	Unique identifier for the analytic workspace.

ALL_OLAP2_AW_ATTRIBUTES

ALL_OLAP2_AW_ATTRIBUTES lists attributes in standard form analytic workspaces.

The attributes associated with a dimension are created in an analytic workspace by the DBMS_AWM.REFRESH_AWDIMENSION procedure. See also "Refreshing the Dimension's Metadata" on page 1-12.

Column	Datatype	NULL	Description
AW_OWNER	VARCHAR2(30)		Owner of the analytic workspace.
AW_NAME	VARCHAR2(30)		Name of the analytic workspace.
AW_DIMENSION_NAME	VARCHAR2(1000)		Name of the dimension in the analytic workspace.
AW_LOGICAL_NAME	VARCHAR2(90)		Logical name for the attribute in the analytic workspace.

Column	Datatype	NULL	Description
AW_PHYSICAL_OBJECT	VARCHAR2(1000)		Standard form name for the attribute in the analytic workspace.
DISPLAY_NAME	VARCHAR2(1000)		Display name for the attribute.
DESCRIPTION	VARCHAR2(1000)		Description of the attribute.
ATTRIBUTE_TYPE	VARCHAR2(1000)		Type of attribute. See Table 11–1, " Reserved Dimension Attributes".
SOURCE_OWNER	VARCHAR2(1000)		Owner of the source attribute in the OLAP Catalog.
SOURCE_DIMENSION_NAME	VARCHAR2(1000)		Name of the source dimension in the OLAP Catalog.
SOURCE_NAME	VARCHAR2(1000)		Name of the source attribute in the OLAP Catalog.

ALL_OLAP2_AW_CUBES

ALL_OLAP2_AW_CUBES lists the cubes in standard form analytic workspaces.

Standard form cubes are created in analytic workspaces by the DBMS_AWM.CREATE_AWCUBE procedure.

Column	Datatype	NULL	Description
AW_OWNER	VARCHAR2(30)		Owner of the analytic workspace.
AW_NAME	VARCHAR2(30)		Name of the analytic workspace.
AW_LOGICAL_NAME	VARCHAR2(90)		Logical name for the cube in the analytic workspace.
AW_PHYSICAL_OBJECT	VARCHAR2(1000)		Standard form name for the cube in the analytic workspace.
SOURCE_OWNER	VARCHAR2(1000)		Owner of the source cube in the OLAP Catalog.
SOURCE_NAME	VARCHAR2(1000)		Name of the source cube in the OLAP Catalog.

ALL_OLAP2_AW_CUBE_AGG_LVL

ALL_OLAP2_AW_CUBE_AGG_LVL lists the levels in aggregation specifications in standard form analytic workspaces.

Aggregation specifications determine how summary data will be calculated and stored in the analytic workspace. Levels are added to aggregation specifications by the DBMS_AWM.ADD_AWCUBEAGG_LEVEL procedure.

Column	Datatype	NULL	Description
AW_OWNER	VARCHAR2(30)		Owner of the analytic workspace.
AW_NAME	VARCHAR2(30)		Name of the analytic workspace.
AW_CUBE_NAME	VARCHAR2(90)		Name of a cube in the analytic workspace.
AW_AGGSPEC_NAME	VARCHAR2(1000)		Name of an aggregation specification for the cube.
AW_DIMENSION_NAME	VARCHAR2(1000)		Name of a workspace dimension of the cube.
AW_LEVEL_NAME	VARCHAR2(1000)		Name of a workspace level of the dimension. This level is in the aggregation specification.

ALL_OLAP2_AW_CUBE_AGG_MEAS

ALL_OLAP2_AW_CUBE_AGG_MEAS lists the measures in aggregation specifications in standard form analytic workspaces.

Aggregation specifications determine how summary data will be calculated and stored in the analytic workspace. Measures are added to aggregation specifications by the DBMS_AWM.ADD_AWCUBEAGG_SPEC_MEASURE procedure.

Column	Datatype	NULL	Description
AW_OWNER	VARCHAR2(30)		Owner of the analytic workspace.
AW_NAME	VARCHAR2(30)		Name of the analytic workspace.
AW_CUBE_NAME	VARCHAR2(90)		Name of a cube in the analytic workspace.
AW_AGGSPEC_NAME	VARCHAR2(1000)		Name of an aggregation specification for the cube.
AW_MEASURE_NAME	VARCHAR2(1000)		Name of a workspace measure of the cube. This measure is in the aggregation specification

ALL_OLAP2_AW_CUBE_AGG_OP

ALL_OLAP2_AW_CUBE_AGG_OP lists the aggregation operators in aggregation specifications in standard form analytic workspaces.

Aggregation specifications determine how summary data will be calculated and stored in the analytic workspace. Aggregation operators are added to aggregation specifications by the DBMS_AWM.SET_AWCUBEAGG_SPEC_AGGOP procedure.

Column	Datatype	NULL	Description
AW_OWNER	VARCHAR2(30)		Owner of the analytic workspace.
AW_NAME	VARCHAR2(30)		Name of the analytic workspace.
AW_CUBE_NAME	VARCHAR2(90)		Name of a cube in the analytic workspace.
AW_MEASURE_NAME	VARCHAR2		Name of a workspace measure to aggregate.
AW_AGGSPEC_NAME	VARCHAR2(1000)		Name of an aggregation specification for the cube.
AW_DIMENSION_NAME	VARCHAR2(1000)		Name of a workspace dimension of the cube.
OPERATOR	VARCHAR2(1000)		Operator for aggregation along this dimension. See Table 1–10, " Aggregation Operators" for a list of valid operators.

ALL_OLAP2_AW_CUBE_AGG_SPECS

ALL_OLAP2_AW_CUBE_AGG_SPECS lists the aggregation specifications in standard form analytic workspaces.

Aggregation specifications determine how summary data will be calculated and stored in the analytic workspace. Aggregation specifications are created by the DBMS_AWM.CREATE_AWCUBEAGG_SPEC procedure.

Column	Datatype	NULL	Description
AW_OWNER	VARCHAR2(30)		Owner of the analytic workspace.
AW_NAME	VARCHAR2(30)		Name of the analytic workspace.
AW_CUBE_NAME	VARCHAR2(90)		Name of the cube in the analytic workspace.
AW_AGGSPEC_NAME	VARCHAR2(1000)		Name of an aggregation plan for the cube.

ALL_OLAP2_AW_CUBE_DIM_USES

ALL_OLAP2_AW_CUBE_DIM_USES lists the dimensions of cubes in standard form analytic workspaces.

Dimensions are associated with workspace cubes by the DBMS_AWM.CREATE_AWCUBE procedure.

Column	Datatype	NULL	Description
AW_OWNER	VARCHAR2(30)		Owner of the analytic workspace.
AW_NAME	VARCHAR2(30)		Name of the analytic workspace.
AW_LOGICAL_NAME	VARCHAR2(90)		Name of a cube in the analytic workspace.
DIMENSION_AW_OWNER	VARCHAR2(1000)		Owner of a workspace dimension of the cube.
DIMENSION_AW_NAME	VARCHAR2(1000)		Name of a workspace dimension of the cube.
DIMENSION_SOURCE_OWNER	VARCHAR2(1000)		Owner of the source dimension in the OLAP Catalog
DIMENSION_SOURCE_NAME	VARCHAR2(1000)		Name of the source dimension in the OLAP Catalog.

ALL_OLAP2_AW_CUBE_MEASURES

ALL_OLAP2_AW_CUBE_MEASURES lists the measures of cubes in standard form analytic workspaces.

Measures are associated with cubes by the DBMS_AWM.REFRESH_AWCUBE procedure. If individual measures were not specified by a call to DBMS_AWM.ADD_AWCUBELOAD_SPEC_MEASURE, then all the cube's measures are loaded when the cube is refreshed.

Column	Datatype	NULL	Description
AW_OWNER	VARCHAR2(30)		Owner of the analytic workspace.
AW_NAME	VARCHAR2(30)		Name of the analytic workspace.
AW_CUBE_NAME	VARCHAR2(90)		Name of a cube in the analytic workspace.
AW_MEASURE_NAME	VARCHAR2(1000)		Logical name of a measure of the cube.
AW_PHYSICAL_OBJECT	VARCHAR2(1000)		Standard form name of the measure.
MEASURE_SOURCE_NAME	VARCHAR2(1000)		Name of the source measure in the OLAP Catalog.
DISPLAY_NAME	VARCHAR2(1000)		Display name for the measure in the analytic workspace.
DESCRIPTION	VARCHAR2(1000)		Description of the measure in the analytic workspace.

Column	Datatype	NULL	Description
IS_AGGREGATEABLE	VARCHAR2(1000)		Whether or not this measure can be aggregated with the OLAP DML AGGREGATE command. The value is YES if the measure is implemented as an OLAP variable or if its underlying storage is a variable. For example, the measure could be implemented as a formula whose value is stored in a variable.

ALL_OLAP2_AW_DIMENSIONS

ALL_OLAP2_AW_DIMENSIONS lists the dimensions in standard form analytic workspaces.

Workspace dimensions are created by the DBMS_AWM.CREATE_AWDIMENSION procedure.

Column	Datatype	NULL	Description
AW_OWNER	VARCHAR2(30)		Owner of the analytic workspace.
AW_NAME	VARCHAR2(30)		Name of the analytic workspace.
AW_LOGICAL_NAME	VARCHAR2(90)		Logical name of the dimension in the analytic workspace.
AW_PHYSICAL_NAME	VARCHAR2(1000)		Standard form name of the dimension in the analytic workspace.
SOURCE_OWNER	VARCHAR2(1000)		Owner of the source dimension in the OLAP Catalog.
SOURCE_NAME	VARCHAR2(1000)		Name of the source dimension in the OLAP Catalog.

ALL_OLAP2_AW_DIM_HIER_LVL_ORD

ALL_OLAP2_AW_DIM_HIER_LVL_ORD lists the levels in hierarchies in standard form analytic workspaces. It includes the position of each level within the hierarchy.

Workspace dimensions are created by the DBMS_AWM.CREATE_AWDIMENSION procedure.

Column	Datatype	NULL	Description
AW_OWNER	VARCHAR2(30)		Owner of the analytic workspace.
AW_NAME	VARCHAR2(30)		Name of the analytic workspace.

Column	Datatype	NULL	Description
AW_DIMENSION_NAME	VARCHAR2(90)		Name of a dimension in the analytic workspace.
AW_HIERARCHY_NAME	VARCHAR2(1000)		Name of a hierarchy of the workspace dimension.
IS_DEFAULT_HIER	VARCHAR2(1000)		Whether or not this hierarchy is the default hierarchy
AW_LEVEL_NAME	VARCHAR2(1000)		Name of a level of the workspace hierarchy.
POSITION	NUMBER		The position of the level in the hierarchy

ALL_OLAP2_AW_DIM_LEVELS

ALL_OLAP2_AW_DIM_LEVELS lists the levels of dimensions in standard form analytic workspaces.

Workspace levels are created by the DBMS_AWM.CREATE_AWDIMENSION procedure.

Column	Datatype	NULL	Description
AW_OWNER	VARCHAR2(30)		Owner of the analytic workspace.
AW_NAME	VARCHAR2(30)		Name of the analytic workspace.
AW_LOGICAL_NAME	VARCHAR2(90)		Name of a dimension in the analytic workspace.
LEVEL_NAME	VARCHAR2(1000)		Name of a workspace level of the dimension.
DISPLAY_NAME	VARCHAR2(1000)		Display name of the level.
DESCRIPTION	VARCHAR2(1000)		Description of the level.

ALL_OLAP2_AW_PHYS_OBJ

ALL_OLAP2_AW_PHYS_OBJ lists the standard form objects in analytic workspaces.

Column	Datatype	NULL	Description
AW_OWNER	VARCHAR2(30)		Owner of the analytic workspace.
AW_NAME	VARCHAR2(30)		Name of the analytic workspace.
AW_OBJECT_NAME	VARCHAR2(90)		Name of the standard form object in the analytic workspace.

Column	Datatype	NULL	Description
AW_OBJECT_TYPE	VARCHAR2(1000)		Type of the standard form object. The type may be any of the native object types that can be defined with the OLAP DML, including: dimensions, relations, variables, formulas, composites, and valuesets.
AW_OBJECT_DATATYPE	VARCHAR2(1000)		Data type of the standard form object. The data type may be any of the native types supported by the OLAP DML, including text, boolean, or integer, or it may be a defined type specific to standard form.

ALL_OLAP2_AW_PHYS_OBJ_PROP

ALL_OLAP2_AW_PHYS_OBJ_PROP lists the standard form objects with their properties.

Column	Datatype	NULL	Description
AW_OWNER	VARCHAR2(30)		Owner of the analytic workspace.
AW_NAME	VARCHAR2(30)		Name of the analytic workspace.
AW_OBJECT_NAME	VARCHAR2(90)		Name of the standard form object in the analytic workspace.
AW_PROP_NAME	VARCHAR2(1000)		Name of a property of the standard form object.
AW_PROP_VALUE	VARCHAR2(1000)		Value of the property.

4

Analytic Workspace Maintenance Views

This chapter describes the views you can query to obtain information about maintaining standard form analytic workspaces.

See Also:

- Chapter 1, "Creating Analytic Workspaces with DBMS_AWM"
- Chapter 23, "DBMS_AWM"

This chapter discusses the following topics:

- Building and Maintaining Analytic Workspaces
- Example: Query Load and Enablement Parameters for Workspace Dimensions
- Summary of Analytic Workspace Maintenance Views

Building and Maintaining Analytic Workspaces

The `DBMS_AWM` package manages the life cycle of standard form analytic workspaces. This includes the creation of workspace cubes from relational sources, data loads, and the enablement of workspace cubes for relational access.

The `DBMS_AWM` package stores information about workspace builds in the OLAP Catalog. You can query the Analytic Workspace Maintenance views to obtain this information. For example, you could obtain a list of workspace cubes with their relational sources, a list of load specifications, or a list of composite specifications.

The `DBMS_AWM` package stores information about workspace enablement within the analytic workspace itself. The Analytic Workspace Maintenance views use `OLAP_TABLE` functions to return information about the enablement of workspace cubes.

You can query these views to obtain the names of enablement views and hierarchy combinations.

Example: Query Load and Enablement Parameters for Workspace Dimensions

The following example uses the XADEMO dimensions CHANNEL and TIME to illustrate several Analytic Workspace Maintenance views.

Example 4–1 Query Load Parameters and Enablement View Names for CHANNEL and TIME

The following statements create the dimensions AW_CHAN and AW_TIME in the analytic workspace MY_SCHEMA.MY_AW.

```
execute dbms_awm.create_awdimension
            ('XADEMO','CHANNEL','MY_SCHEMA', 'MY_AW', 'AW_CHAN');
execute dbms_awm.create_awdimension
           ('XADEMO','TIME','MY_SCHEMA', 'MY_AW', 'AW_TIME');
```

The following statements create the load specifications for the dimensions.

```
execute dbms_awm.create_awdimload_spec
           ('CHAN_DIMLOADSPEC', 'XADEMO', 'CHANNEL', 'FULL_LOAD');
execute dbms_awm.add_awdimload_spec_filter
           ('CHAN_DIMLOADSPEC', 'XADEMO', 'CHANNEL', 'XADEMO', 'XADEMO_CHANNEL',
            '''CHAN_STD_CHANNEL'' = ''DIRECT''' );
execute dbms_awm.create_awdimload_spec
           ('TIME_DIMLOADSPEC', 'XADEMO', 'TIME', 'FULL_LOAD');
execute dbms_awm.add_awdimload_spec_filter
           ('TIME_DIMLOADSPEC', 'XADEMO', 'TIME', 'XADEMO', 'XADEMO_TIME',
            '''TIME_STD_YEAR'' = ''1997''' );
```

The following query returns the filter conditions associated with the dimension load specifications.

```
SQL>select * from all_aw_load_dim_filters;

OWNER     DIMENSION_NAME   LOAD_NAME         TABLE_OWNER     TABLE_NAME      FILTER_CONDITION
-------- ---------------   ----------------  --------------  --------------- -----------------------------
XADEMO     TIME            TIME_DIMLOADSPEC    XADEMO        XADEMO_TIME     'TIME_STD_YEAR' = '1997'
XADEMO     CHANNEL         CHAN_DIMLOADSPEC    XADEMO        XADEMO_CHANNEL  'CHAN_STD_CHANNEL' = 'DIRECT'
```

The following statements load the dimensions in the analytic workspace. The system-generated names that will be used for the enablement views are created in the workspace as part of the load process.

```
execute dbms_awm.refresh_awdimension
          ('MY_SCHEMA', 'MY_AW', 'AWCHAN', 'CHAN_DIMLOADSPEC');
execute dbms_awm.refresh_awdimension
          ('MY_SCHEMA', 'MY_AW', 'AWTIME', 'TIME_DIMLOADSPEC');
```

The following query returns the system-generated enablement view names for the dimensions.

```
SQL>select * from all_aw_dim_enabled_views;

AW_OWNER   AW_NAME     DIMENSION_NAME  HIERARCHY_NAME   SYSTEM_VIEWNAME                  USER_VIEWNAME
---------- ---------- --------------- ---------------  -------------------------------- --------------
MY_SCHEMA    MY_AW     AWCHAN          STANDARD         MY_S_MY_AW_AWCHA_STAND35VIEW
MY_SCHEMA    MY_AW     AWTIME          STANDARD         MY_S_MY_AW_AWTIM_STAND36VIEW
MY_SCHEMA    MY_AW     AWTIME          YTD              MY_S_MY_AW_AWTIM_YTD37VIEW
```

Summary of Analytic Workspace Maintenance Views

The analytic workspace maintenance views are summarized in the following table.

Table 4–1 Analytic Workspace Maintenance Views

Public Synonym	Description
ALL_AW_CUBE_AGG_LEVELS	Describes the levels in aggregation specifications for cubes.
ALL_AW_CUBE_AGG_MEASURES	Describes the measures in aggregation specifications for cubes.
ALL_AW_CUBE_AGG_PLANS	Describes the aggregation specifications for cubes.
ALL_AW_CUBE_ENABLED_HIERCOMBO	Describes the hierarchy combinations associated with cubes.
ALL_AW_CUBE_ENABLED_VIEWS	Describes the fact views that can be generated for workspace cubes.
ALL_AW_DIM_ENABLED_VIEWS	Describes the dimension views that can be generated for workspace dimensions.
ALL_AW_LOAD_CUBES	Describes the load specifications for cubes.
ALL_AW_LOAD_CUBE_DIMS	Describes the composite specifications for cubes.
ALL_AW_LOAD_CUBE_FILTERS	Describes the filter conditions associated with load specifications for cubes.

Table 4–1 (Cont.) Analytic Workspace Maintenance Views

Public Synonym	Description
ALL_AW_LOAD_CUBE_MEASURES	Describes the measures in cube load specifications.
ALL_AW_LOAD_CUBE_PARMS	Describes parameters of cube load specifications.
ALL_AW_LOAD_DIMENSIONS	Describes the load specifications for dimensions.
ALL_AW_LOAD_DIM_FILTERS	Describes the filter conditions associated with load specifications for dimensions.
ALL_AW_LOAD_DIM_PARMS	Describes parameters of dimension load specifications.
ALL_AW_OBJ	Lists the objects in all analytic workspaces available to the current user. The workspaces may have been created by `DBMS_AWM` or by another tool, such as the OLAP Analytic Workspace API.
ALL_AW_PROP	Lists the OLAP DML properties and their values in all analytic workspaces available to the current user. The workspaces may have been created by `DBMS_AWM` or by another tool, such as the OLAP Analytic Workspace API.

ALL_AW_CUBE_AGG_LEVELS

`ALL_AW_CUBE_AGG_LEVELS` lists the levels in aggregation specifications for cubes.

Aggregation specifications determine how data will be aggregated along the dimensions of a cube in an analytic workspace. Aggregation specifications are created by the `DBMS_AWM.CREATE_AWCUBEAGG_SPEC` procedure.

Column	Datatype	NULL	Description
`owner`	`varchar2(240)`		Owner of the cube.
`cube_name`	`varchar2(240)`		Name of the cube.
`aggregation_name`	`varchar2(60)`		Name of the aggregation spec.
`dimension_owner`	`varchar2(30)`		Owner of the dimension to aggregate.
`dimension_name`	`varchar2(240)`		Name of the dimension to aggregate.
`level_name`	`archar2(240)`		Name of the level of aggregation for this dimension.

ALL_AW_CUBE_AGG_MEASURES

`ALL_AW_CUBE_AGG_MEASURES` lists the measures in aggregation specifications for cubes.

Aggregation specifications determine how the measures will be aggregated along the dimensions of a cube in an analytic workspace. Aggregation specifications are created by the DBMS_AWM.CREATE_AWCUBEAGG_SPEC procedure.

Column	Datatype	NULL	Description
cube_owner	varchar2(240)		Owner of the cube.
cube_name	varchar2(240)		Name of the cube.
aggregation_name	varchar2(60)		Name of the aggregation spec.
measure_name	varchar2(240)		Name of the measure to aggregate.

ALL_AW_CUBE_AGG_PLANS

ALL_AW_CUBE_AGG_PLANS lists the aggregation specifications for cubes.

Aggregation specifications determine how data will be aggregated along the dimensions of a cube in an analytic workspace. Aggregation specifications are created by the DBMS_AWM.CREATE_AWCUBEAGG_SPEC procedure.

Column	Datatype	NULL	Description
owner	varchar2(240)		Owner of the cube.
cube_name	varchar2(240)		Name of the cube.
aggregation_name	varchar2(60)		Name of the aggregation spec.

ALL_AW_CUBE_ENABLED_HIERCOMBO

ALL_AW_CUBE_ENABLED_HIERCOMBO lists the hierarchy combinations associated with cubes in analytic workspaces.

Each hierarchy combination is identified by a unique number. The OLAP API Enabler creates a separate fact view for each hierarchy combination.

The information in this view is available for all standard form cubes that have been refreshed. See the DBMS_AWM.REFRESH_AWCUBE procedure and the DBMS_AWM.CREATE_AWCUBE_ACCESS procedure.

Column	Datatype	NULL	Description
aw_owner	varchar2(30)		Owner of the analytic workspace.

Column	Datatype	NULL	Description
aw_name	varchar2(30)		Name of the analytic workspace.
cube_name	varchar2(1000)		Name of the cube in the analytic workspace.
hiercombo_num	number		Unique number that identifies the hierarchy combination.
hiercombo_str	varchar2(1000)		List of hierarchies that define the dimensionality of a fact view of the enabled cube.

ALL_AW_CUBE_ENABLED_VIEWS

ALL_AW_CUBE_ENABLED_VIEWS describes the fact views that can be generated for cubes in analytic workspaces.

Descriptions of the views are created when the cube is refreshed. The view is not instantiated until the DBMS_AWM.CREATE_AWCUBE_ACCESS has executed and the resulting script has been run.

ALL_AW_CUBE_ENABLED_VIEWS shows the descriptions of the views. The views themselves do not necessarily exist.

Metadata about fact views is generated by the DBMS_AWM.REFRESH_AWCUBE procedure. Scripts to create views of workspace cubes are created by the DBMS_AWM.CREATE_AWCUBE_ACCESS procedure.

Column	Datatype	NULL	Description
aw_owner	varchar2(30)		Owner of the analytic workspace.
aw_name	varchar2(30)		Name of the analytic workspace.
cube_name	varchar2(1000)		Name of the cube in the analytic workspace.
hiercombo_num	number		Unique number that identifies the hierarchy combination.
hiercombo_str	varchar2(1000)		List of hierarchies that define the dimensionality of a fact view of the enabled cube.
system_viewname	varchar2(1000)		Default view name created by the DBMS_AWM.REFRESH_AWCUBE procedure.
user_viewname	varchar2(1000)		User-defined view name specified by the DBMS_AWM.SET_AWCUBE_VIEWNAME procedure.

ALL_AW_DIM_ENABLED_VIEWS

ALL_AW_DIM_ENABLED_VIEWS describes the dimension views that can be generated for dimensions in analytic workspaces.

Descriptions of the views are created when the dimension is refreshed. The view is not instantiated until the DBMS_AWM.CREATE_AWDIMENSION_ACCESS has executed and the resulting script has been run.

ALL_AW_DIM_ENABLED_VIEWS shows the descriptions of the views. The views themselves do not necessarily exist.

Metadata about dimension views is generated by the DBMS_AWM.REFRESH_AWDIMENSION procedure. Scripts to create views of workspace dimensions are created by the DBMS_AWM.CREATE_AWDIMENSION_ACCESS procedure.

Column	Datatype	NULL	Description
aw_owner	varchar2(30)		Owner of the analytic workspace.
aw_name	varchar2(30)		Name of the analytic workspace.
dimension_name	varchar2(1000)		Name of the dimension in the analytic workspace.
hierarchy_name	varchar2(1000)		Name of the hierarchy in the analytic workspace.
system_viewname	varchar2(1000)		Default view name created by the DBMS_AWM.REFRESH_AWCUBE procedure.
user_viewname	varchar2(1000)		User-defined view name specified by the DBMS_AWM.SET_AWDIMENSION_VIEWNAME procedure.

ALL_AW_LOAD_CUBES

ALL_AW_LOAD_CUBES lists the load specifications for cubes.

Load specifications determine how data will be loaded from the source fact table into the analytic workspace. Cube load specifications are created by the DBMS_AWM.CREATE_AWCUBELOAD_SPEC procedure.

Column	Datatype	NULL	Description
cube_owner	varchar2(240)		Owner of the OLAP Catalog source cube.
cube_name	varchar2(240)		Name of the OLAP Catalog source cube.
load_name	varchar2(60)		Name of a load specification for the cube.

Column	Datatype	NULL	Description
load_type	varchar2(60)		'LOAD_DATA' -- Load the data from the fact table into the analytic workspace target cube. 'LOAD_PROGRAM' -- Create the load programs in the analytic workspace but do not execute them. You can run the program manually to load the data. Cube load program names are stored in the AW$LOADPGRGS property of the standard form cube in the analytic workspace.

ALL_AW_LOAD_CUBE_DIMS

ALL_AW_LOAD_CUBE_DIMS describes the composite specifications for cubes.

Composite specifications determines how the cube's dimensions will be optimized in the analytic workspace. Composite specifications are created by the DBMS_AWM.CREATE_AWCOMP_SPEC procedure.

Column	Datatype	NULL	Description
cube_owner	varchar2(240)		Owner of the OLAP Catalog source cube.
cube_name	varchar2(240)		Name of the OLAP Catalog source cube.
cubeload_name	varchar2(60)		Name of a load specification for the cube.
compspec_name	varchar2(30)		Name of a composite specification associated with this load specification.
composite_name	varchar2(30)		Name of a composite that is a member of the specification. A composite contains sparse dimensions of the cube.
segwidth	number		Segment width for storage of the data dimensioned by this member of the specification.
compspec_position	number		Position of the member within the specification.
dimension_owner	varchar2(30)		Owner of an OLAP Catalog source dimension that is a member of the specification.
dimension_name	varchar2(240)		Name of the OLAP Catalog source dimension that is a member of the specification.
composite_position	number		Position of the member within a composite member.

Column	Datatype	NULL	Description
nested_level	number		The level of nesting of the member of the specification. For example, a dense dimension would have a nesting level of 1. A sparse dimension within a composite would have a nesting level of 2, and a nested composite would have a nesting level of 3.
nested_type	varchar2(10)		Type of member of the specification. Either DIMENSION or COMPOSITE.
nested_name	varchar2(30)		Name of the member of the specification. This may be the name of a dimension or the name of a composite.

ALL_AW_LOAD_CUBE_FILTERS

ALL_AW_LOAD_CUBE_FILTS lists the filter conditions associated with load specifications for cubes.

Filter conditions are SQL WHERE clauses that identify a subset of the data to be loaded from the fact table to the analytic workspace.

Filter conditions are created by the DBMS_AWM.ADD_AWCUBELOAD_SPEC_FILTER procedure.

Column	Datatype	NULL	Description
owner	varchar2(240)		Owner of the OLAP Catalog source cube.
cube_name	varchar2(240)		Name of the OLAP Catalog source cube.
load_name	varchar2(60)		Name of a load specification for the cube.
table_owner	varchar2(30)		Owner of the fact table.
table_name	varchar2(30)		Name of the fact table.
filter_condition	varchar2(4000)		SQL WHERE clause.

ALL_AW_LOAD_CUBE_MEASURES

ALL_AW_LOAD_CUBE_MEASURES lists the measures in cube load specifications with their corresponding target measures in standard form analytic workspaces.

Measures are added to cube load specifications by the DBMS_AWM.ADD_AWCUBELOAD_SPEC_MEASURE procedure. This procedure enables you to specify a target name and display name for the measure in the analytic workspace. If you do not call this procedure, or if you do not specify the target names, the OLAP Catalog names are used.

Column	Datatype	NULL	Description
owner	varchar2(240)		Owner of the source cube in the OLAP Catalog.
cube_name	varchar2(240)		Name of the source cube in the OLAP Catalog.
load_name	varchar2(60)		Name of the load specification for the source cube.
measure_name	varchar2(240)		Name of a measure of the source cube.
measure_target_name	varchar2(60)		Name of the measure in the analytic workspace.
measure_target_display_name	varchar2(60)		Display name of the measure in the analytic workspace. This may be the display name from the OLAP Catalog, or it may be user-defined.
measure_target_description	varchar2(4000)		Description of the measure in the analytic workspace. This may be the description from the OLAP Catalog, or it may be user-defined.

ALL_AW_LOAD_CUBE_PARMS

ALL_AW_LOAD_CUBE_PARMS lists the parameters in cube load specifications.

Cube load specifications determine how a cube's data will be loaded from the fact table into the analytic workspace.

Parameters are set for cube load specifications by the DBMS_AWM.SET_AWCUBELOAD_SPEC_PARAMETER procedure.

Column	Datatype	NULL	Description
owner	varchar2(240)		Owner of the source cube in the OLAP Catalog.
cube_name	varchar2(240)		Name of the source cube in the OLAP Catalog.
load_name	varchar2(60)		Name of the load specification for the source cube.
parm_name	varchar2(16)		The name of the parameter. Currently only 'DISPLAY NAME' is available. If you do not set this parameter, the cube display name from the OLAP Catalog is used in the analytic workspace.
parm_value	varchar2(30)		The display name to use for the target cube in the analytic workspace.

ALL_AW_LOAD_DIMENSIONS

ALL_AW_LOAD_DIMENSIONS lists the load specifications for dimensions.

Dimension load specifications are created by the DBMS_AWM.CREATE_AWDIMLOAD_SPEC procedure.

Column	Datatype	NULL	Description
owner	varchar2(30)		Owner of the source dimension in the OLAP Catalog.
dimension_name	varchar2(30)		Name of the source dimension in the OLAP Catalog.
load_name	varchar2(60)		Name of the load specification.
load_type	varchar2(60)		'FULL_LOAD_ADDITIONS_ONLY' -- Only new dimension members will be loaded when the dimension is refreshed. (Default) 'FULL_LOAD' -- When the dimension is refreshed, all dimension members in the workspace will be deleted, then all the members of the source dimension will be loaded.

ALL_AW_LOAD_DIM_FILTERS

ALL_AW_LOAD_DIM_FILTERS lists the filter conditions associated with load specifications for dimensions.

Filter conditions are SQL WHERE clauses that identify a subset of the data to be loaded from the dimension table to the analytic workspace.

Filter conditions are created by the DBMS_AWM.ADD_AWDIMLOAD_SPEC_FILTER procedure.

Column	Datatype	NULL	Description
owner	varchar2(30)		Owner of the source dimension in the OLAP Catalog.
dimension_name	varchar2(30)		Name of the source dimension in the OLAP Catalog.
load_name	varchar2(60)		Name of the dimension load specification.
table_owner	varchar2(30)		Owner of the dimension table.
table_name	varchar2(30)		Name of the dimension table.
filter_condition	varchar2(4000)		SQL WHERE clause.

ALL_AW_LOAD_DIM_PARMS

ALL_AW_LOAD_DIM_PARMS lists the parameters in dimension load specifications.

Dimension load specifications determine how dimension members will be loaded from the dimension table into the analytic workspace.

Parameters are set for dimension load specifications by the DBMS_AWM.SET_AWDIMLOAD_SPEC_PARAMETER procedure.

Column	Datatype	NULL	Description
owner	varchar2(30)		Owner of the source dimension in the OLAP Catalog.
dimension_name	varchar2(30)		Name of the source dimension in the OLAP Catalog.
load_name	varchar2(60)		Name of the dimension load specification.
parm_name	varchar2(16)		'UNIQUE_RDBMS_KEY' -- Whether or not the members of this dimension are unique across all levels in the source tables. 'DISPLAY_NAME' -- Display name for the target dimension in the analytic workspace. 'PLURAL_DISPLAY_NAME' -- Plural display name for the target dimension in the analytic workspace.
parm_value	varchar2(4000)		Values of UNIQUE_RDBMS_KEY: NO-- Dimension member names are not unique across levels in the RDBMS tables. The corresponding dimension member names in the analytic workspace include the level name as a prefix. (Default) YES -- Dimension member names are unique across levels in the RDBMS tables. The corresponding dimension member names in the analytic workspace have the same names as in the source relational dimension. Value of DISPLAY_NAME is the display name for the target dimension in the analytic workspace. Value of PLURAL_DISPLAY_NAME is the plural display name for the target dimension in the analytic workspace.

ALL_AW_OBJ

ALL_AW_OBJ lists the current objects in all analytic workspaces that are accessible to the user. The workspaces may have been created by DBMS_AWM or by another tool, such as the OLAP Analytic Workspace API.

Column	Datatype	NULL	Description
OWNER	VARCHAR2	NOT NULL	User name of the analytic workspace owner
AW_NUMBER	NUMBER	NOT NULL	Unique identifier within the database for the analytic workspace
AW_NAME	VARCHAR2		Name of the analytic workspace
OBJ_ID	NUMBER		Unique identifier for the object within the analytic workspace
OBJ_NAME	VARCHAR2		Name of the object
OBJ_TYPE	NUMBER		Data type of the object
PART_NAME	VARCHAR2		Name of the partition for the object

ALL_AW_PROP

ALL_AW_PROP lists the current OLAP DML properties and their values in all analytic workspaces that are accessible to the user. The workspaces may have been created by DBMS_AWM or by another tool, such as the OLAP Analytic Workspace API.

Column	Datatype	NULL	Description
OWNER	VARCHAR2	NOT NULL	User name of the analytic workspace owner
AW_NUMBER	NUMBER	NOT NULL	Unique identifier within the database for the analytic workspace
AW_NAME	VARCHAR2		Name of the analytic workspace
OBJ_ID	NUMBER		Unique identifier for the object within the analytic workspace
OBJ_NAME	VARCHAR2		Name of the object
PROPERTY_NAME	VARCHAR2		Name of the property
PROPERTY_TYPE	VARCHAR2		Data type of the property value

Column	Datatype	NULL	Description
PROPERTY_VALUE	VARCHAR2		Value of the property

5

OLAP Catalog Metadata Views

This chapter describes the OLAP Catalog metadata views. All OLAP metadata, whether created with the CWM2 PL/SQL packages or with Enterprise Manager, is presented in these views.

See Also: Chapter 2, "Creating OLAP Catalog Metadata with CWM2".

Note: A second set of views, called the OLAP API Metadata Reader views, presents much of the same information as the OLAP Catalog views. The Metadata Reader views are structured to facilitate fast queries by the OLAP API. See Chapter 16 for more information.

This chapter discusses the following topics:

- Access to OLAP Catalog Views
- Views of the Dimensional Model
- Views of Mapping Information

Access to OLAP Catalog Views

The OLAP Catalog read API consists of two sets of corresponding views:

- ALL_ views displaying all valid OLAP metadata accessible to the current user.
- DBA_ views displaying all OLAP metadata (both valid and invalid) in the entire database. DBA_ views are intended only for administrators.

Note: The OLAP Catalog tables are owned by OLAPSYS. To create OLAP metadata in these tables, the user must have the OLAP_DBA role.

The columns of the ALL_ and DBA_ views are identical. Only the ALL_ views are listed in this chapter.

Views of the Dimensional Model

The following views show the basic dimensional model of OLAP metadata.

For more information on the logical model, see the *Oracle OLAP Application Developer's Guide*.

Table 5–1 OLAP Catalog Dimensional Model Views

View Name Synonym	Description
ALL_OLAP2_CATALOGS	List all measure folders (catalogs) within the Oracle instance.
ALL_OLAP2_CATALOG_ENTITY_USES	Lists the measures within each measure folder.
ALL_OLAP2_CUBES	Lists all cubes in an Oracle instance.
ALL_OLAP2_CUBE_DIM_USES	Lists the dimensions within each cube.
ALL_OLAP2_CUBE_MEASURES	Lists the measures within each cube.
ALL_OLAP2_CUBE_MEAS_DIM_USES	Shows how each measure is aggregated along each of its dimensions.
ALL_OLAP2_DIMENSIONS	Lists all OLAP dimensions in an Oracle instance.
ALL_OLAP2_DIM_ATTRIBUTES	Lists the dimension attributes within each dimension.
ALL_OLAP2_DIM_ATTR_USES	Shows how level attributes are associated with each dimension attribute.
ALL_OLAP2_DIM_HIERARCHIES	Lists the hierarchies within each dimension.
ALL_OLAP2_DIM_HIER_LEVEL_USES	Show how levels are ordered within each hierarchy.
ALL_OLAP2_DIM_LEVELS	Lists the levels within each dimension.
ALL_OLAP2_DIM_LEVEL_ATTRIBUTES	Lists the level attributes within each level.

Table 5–1 (Cont.) OLAP Catalog Dimensional Model Views

View Name Synonym	Description
ALL_OLAP2_ENTITY_DESC_USES	Lists the reserved attributes that have application-specific meanings. Examples are dimension attributes that are used for long and short descriptions and time-series calculations (end date, time span, period ago, and so on).
ALL_OLAP2_ENTITY_EXT_PARMS	Lists the OLE DB for OLAP extended metadata descriptors.
ALL_OLAP2_ENTITY_PARAMETERS	Lists the OLE DB for OLAP metadata descriptors.

Views of Mapping Information

The following views show how the basic dimensional model is mapped to relational tables or views.

Table 5–2 OLAP Catalog Mapping Views

View Synonym Name	Description
ALL_OLAP2_CUBE_MEASURE_MAPS	Shows the mapping of each measure to a column.
ALL_OLAP2_DIM_LEVEL_ATTR_MAPS	Shows the mapping of each level attribute to a column.
ALL_OLAP2_FACT_LEVEL_USES	Shows the joins between dimension tables and fact tables in a star or snowflake schema.
ALL_OLAP2_FACT_TABLE_GID	Shows the Grouping ID column for each hierarchy in each fact table.
ALL_OLAP2_HIER_CUSTOM_SORT	Shows the default sort order for level columns within hierarchies.
ALL_OLAP2_JOIN_KEY_COLUMN_USES	Shows the joins between two levels in a hierarchy.
ALL_OLAP2_LEVEL_KEY_COL_USES	Shows the mapping of each level to a unique key column.

ALL_OLAP2_AGGREGATION_USES

`ALL_OLAP2_AGGREGATION_USES` lists the aggregation operators associated with cubes that map to relational tables organized as star or snowflake schemas.

Column	Data Type	NULL	Description
OWNER	VARCHAR2(30)	NOT NULL	Owner of the cube.

Column	Data Type	NULL	Description
CUBE_NAME	VARCHAR2(30)	NOT NULL	Name of the cube.
DIMENSION_NAME	VARCHAR2(30)	NOT NULL	Name of the dimensions of the cube.
HIERARCHY_NAME	VARCHAR2(30)		Name of the hierarchies of the cube's dimensions.
DIM_HIER_COMBO_ID	NUMBER	NOT NULL	Identifier of a hierarchy combination within the cube.
AGGREGATION_NAME	VARCHAR2(240)		Name of the aggregation operator for this dimension. (See Table 1–10, " Aggregation Operators" on page 1-22.
AGGREGATION_ORDER	NUMBER		The order of precedence of the aggregation operator.
TABLE_OWNER	VARCHAR2(30)		Owner of the table that contains the weightby factors for weighted operators. If the operator is not weighted, this column is null.
TABLE_NAME	VARCHAR2(30)		Name of the table that contains the weightby factors for weighted operators. If the operator is not weighted, this column is null.
COLUMN_NAME	VARCHAR2(30)		Name of the column that contains the weightby factors for weighted operators. If the operator is not weighted, this column is null.

ALL_OLAP2_CATALOGS

ALL_OLAP2_CATALOGS lists all the measure folders (catalogs) within the Oracle instance.

Column	Data Type	NULL	Description
CATALOG_ID	NUMBER	NOT NULL	ID of the measure folder.
CATALOG_NAME	VARCHAR2(30)	NOT NULL	Name of the measure folder.
PARENT_CATALOG_ID	NUMBER		ID of the parent measure folder. This column is null for measure folders at the root of the measure folder tree.
DESCRIPTION	VARCHAR2(2000)		Description of the measure folder.

ALL_OLAP2_CATALOG_ENTITY_USES

ALL_OLAP2_CATALOG_ENTITY_USES lists the measures within each measure folder.

Column	Data Type	NULL	Description
CATALOG_ID	NUMBER	NOT NULL	ID of the measure folder.
ENTITY_OWNER	VARCHAR2(30)	NOT NULL	Owner of the measure's cube.
ENTITY_NAME	VARCHAR2(30)	NOT NULL	Name of the measure's cube.
CHILD_ENTITY_NAME	VARCHAR2(30)	NOT NULL	Name of the measure in the measure folder.

ALL_OLAP2_CUBES

ALL_OLAP2_CUBES lists all cubes in an Oracle instance .

Column	Data Type	NULL	Description
OWNER	VARCHAR2(30)	NOT NULL	Owner of the cube that contains the measure.
CUBE_NAME	VARCHAR2(30)	NOT NULL	Name of the cube that contains the measure.
INVALID	VARCHAR2(2)	NOT NULL	Whether or not this cube is in an invalid state. See "Validating and Committing OLAP Metadata" on page 2-13.
DISPLAY_NAME	VARCHAR2(30)		Display name for the cube.
DESCRIPTION	VARCHAR2(2000)		Description of the cube.
MV_SUMMARYCODE	VARCHAR2(2)		If this cube has an associated materialized view, the MV summary code specifies whether it is in Grouping Set (**groupingset**) or Rolled Up (**rollup**) form. See Chapter 24, "DBMS_ODM".

ALL_OLAP2_CUBE_DIM_USES

ALL_OLAP2_CUBE_DIM_USES lists the dimensions within each cube.

A dimension may be associated more than once with the same cube, but each association is specified in a separate row, under its own unique dimension alias.

Column	Data Type	NULL	Description
CUBE_DIMENSION_USE_ID	NUMBER	NOT NULL	ID of the association between a cube and a dimension.
OWNER	VARCHAR2(30)	NOT NULL	Owner of the cube.
CUBE_NAME	VARCHAR2(30)	NOT NULL	Name of the cube.
DIMENSION_OWNER	VARCHAR2(30)	NOT NULL	Owner of the dimension.
DIMENSION_NAME	VARCHAR2(30)	NOT NULL	Name of the dimension.
DIMENSION_ALIAS	VARCHAR2(30)		Alias of the dimension, to provide unique identity of dimension use within the cube.
DEFAULT_CALC_HIERARCHY_NAME	VARCHAR2(30)		The default hierarchy to be used for drilling up or down within the dimension.
DEPENDENT_ON_DIM_USE_ID	NUMBER		ID of the cube/dimension association on which this cube/dimension association depends.

ALL_OLAP2_CUBE_MEASURES

ALL_OLAP2_CUBE_MEASURES lists the measures within each cube .

Column	Data Type	NULL	Description
OWNER	VARCHAR2(30)	NOT NULL	Owner of the cube that contains the measure.
CUBE_NAME	VARCHAR2(30)	NOT NULL	Name of the cube that contains the measure.
MEASURE_NAME	VARCHAR2(30)	NOT NULL	Name of the measure.
DISPLAY_NAME	VARCHAR2(30)		Display name for the measure.
DESCRIPTION	VARCHAR2(2000)		Description of the measure.

ALL_OLAP2_CUBE_MEASURE_MAPS

ALL_OLAP2_CUBE_MEASURE_MAPS shows the mapping of each measure to a column.

Column	Data Type	NULL	Description
OWNER	VARCHAR2(30)	NOT NULL	Owner of the cube.
CUBE_NAME	VARCHAR2(30)	NOT NULL	Name of the cube.

Column	Data Type	NULL	Description
MEASURE_NAME	VARCHAR2(30)	NOT NULL	Name of the measure contained in this cube.
DIM_HIER_COMBO_ID	NUMBER	NOT NULL	ID of the association between this measure and one combination of its dimension hierarchies.
FACT_TABLE_OWNER	VARCHAR2(30)	NOT NULL	Owner of the fact table.
FACT_TABLE_NAME	VARCHAR2(30)	NOT NULL	Name of the fact table.
COLUMN_NAME	VARCHAR2(30)	NOT NULL	Name of the column in the fact table where this measure's data is stored.

ALL_OLAP2_CUBE_MEAS_DIM_USES

ALL_OLAP2_CUBE_MEAS_DIM_USES shows how each measure is aggregated along each of its dimensions. The default aggregation method is addition.

Column	Data Type	NULL	Description
OWNER	VARCHAR2(30)	NOT NULL	Owner of the cube that contains this measure.
CUBE_NAME	VARCHAR2(30)	NOT NULL	Name of the cube that contain this measure.
MEASURE_NAME	VARCHAR2(30)	NOT NULL	Name of the measure.
DIMENSION_OWNER	VARCHAR2(30)	NOT NULL	Owner of a dimension associated with this measure.
DIMENSION_NAME	VARCHAR2(30)	NOT NULL	Name of the dimension.
DIMENSION_ALIAS	VARCHAR2(30)		Alias of the dimension.
DEFAULT_AGGR_FUNCTION_USE_ID	NUMBER		The default aggregation method used to aggregate this measure's data over this dimension. If this column is null, the aggregation method is addition.

ALL_OLAP2_DIMENSIONS

ALL_OLAP2_DIMENSIONS lists all the OLAP dimensions in the Oracle instance.

OLAP dimensions created with the CWM2 APIs have no association with database dimension objects. OLAP dimensions created in Enterprise Manager are based on database dimension objects.

Column	Data Type	NULL	Description
OWNER	VARCHAR2(30)	NOT NULL	Owner of the dimension.
DIMENSION_NAME	VARCHAR2(30)	NOT NULL	Name of the dimension.
PLURAL_NAME	VARCHAR2(30)		Plural name for the dimension. Used for display.
DISPLAY_NAME	VARCHAR2(30)		Display name for the dimension.
DESCRIPTION	VARCHAR2(2000)		Description of the dimension.
DEFAULT_ DISPLAY_ HIERARCHY	VARCHAR2(30)	NOT NULL	Default display hierarchy for the dimension.
INVALID	VARCHAR2(1)	NOT NULL	Whether or not the dimension is valid. See "Validating and Committing OLAP Metadata" on page 2-13
DIMENSION_TYPE	VARCHAR2(10)		Not used.

ALL_OLAP2_DIM_ATTRIBUTES

ALL_OLAP2_DIM_ATTRIBUTES lists the dimension attributes within each dimension.

Column	Data Type	NULL	Description
OWNER	VARCHAR2(30)	NOT NULL	Owner of the dimension.
DIMENSION_NAME	VARCHAR2(30)	NOT NULL	Name of the dimension.
ATTRIBUTE_NAME	VARCHAR2(30)	NOT NULL	Name of the dimension attribute.
DISPLAY_NAME	VARCHAR2(30)		Display name for the dimension attribute.
DESCRIPTION	VARCHAR2(2000)		Description of the dimension attribute.
DESC_ID	NUMBER		If the attribute is reserved, its type is listed in this column. Examples of reserved dimension attributes are long and short descriptions and time-related attributes, such as end date, time span, and period ago.

ALL_OLAP2_DIM_ATTR_USES

ALL_OLAP2_DIM_ATTR_USES shows how level attributes are associated with each dimension attribute.

The same level attribute can be included in more than one dimension attribute.

Column	Data Type	NULL	Description
OWNER	VARCHAR2(30)	NOT NULL	Owner of the dimension.
DIMENSION_NAME	VARCHAR2(30)	NOT NULL	Name of the dimension.
DIM_ATTRIBUTE_NAME	VARCHAR2(30)	NOT NULL	Name of the dimension attribute.
LEVEL_NAME	VARCHAR2(30)	NOT NULL	Name of a level within the dimension.
LVL_ATTRIBUTE_NAME	VARCHAR2(30)	NOT NULL	Name of an attribute for this level. This level attribute is included in the dimension attribute.

ALL_OLAP2_DIM_HIERARCHIES

ALL_OLAP2_DIM_HIERARCHIES lists the hierarchies within each dimension.

Column	Data Type	NULL	Description
OWNER	VARCHAR2(30)	NOT NULL	Owner of the dimension.
DIMENSION_NAME	VARCHAR2(30)	NOT NULL	Name of the dimension.
HIERARCHY_NAME	VARCHAR2(30)	NOT NULL	Name of the hierarchy.
DISPLAY_NAME	VARCHAR2(30)		Display name for the hierarchy.
DESCRIPTION	VARCHAR2(2000)		Description of the hierarchy.
SOLVED_CODE	VARCHAR2(2)	NOT NULL	The solved code may be one of the following: UNSOLVED LEVEL-BASED, for a hierarchy that contains no embedded totals and is stored in a level-based dimension table. SOLVED LEVEL-BASED, for a hierarchy that contains embedded totals, has a grouping ID, and is stored in a level-based dimension table. SOLVED VALUE-BASED, for a hierarchy that contains embedded totals and is stored in a parent-child dimension table. For information about mapping hierarchies with different solved codes, see "Joining Fact Tables with Dimension Tables" on page 2-12.

ALL_OLAP2_DIM_HIER_LEVEL_USES

ALL_OLAP2_DIM_HIER_LEVEL_USES shows how levels are ordered within each hierarchy.

Within separate hierarchies, the same parent level may be hierarchically related to a different child level.

Column	Data Type	NULL	Description
OWNER	VARCHAR2(30)	NOT NULL	Owner of the dimension.
DIMENSION_NAME	VARCHAR2(30)	NOT NULL	Name of the dimension.
HIERARCHY_NAME	VARCHAR2(30)	NOT NULL	Name of the hierarchy.
PARENT_LEVEL_NAME	VARCHAR2(30)	NOT NULL	Name of the parent level.
CHILD_LEVEL_NAME	VARCHAR2(30)	NOT NULL	Name of the child level.
POSITION	NUMBER	NOT NULL	Position of this parent-child relationship within the hierarchy, with position 1 being the most detailed.

ALL_OLAP2_DIM_LEVELS

ALL_OLAP2_DIM_LEVELS lists the levels within each dimension.

Column	Data Type	NULL	Description
OWNER	VARCHAR2(30)	NOT NULL	Owner of the dimension containing this level.
DIMENSION_NAME	VARCHAR2(30)	NOT NULL	Name of the dimension containing this level.
LEVEL_NAME	VARCHAR2(30)	NOT NULL	Name of the level.
DISPLAY_NAME	VARCHAR2(30)		Display name for the level.
DESCRIPTION	VARCHAR2(2000)		Description of the level.
LEVEL_TABLE_OWNER	VARCHAR2(30)	NOT NULL	Owner of the dimension table that contains the columns for this level.
LEVEL_TABLE_NAME	VARCHAR2(30)	NOT NULL	Name of the dimension table that contains the columns for this level.

ALL_OLAP2_DIM_LEVEL_ATTRIBUTES

ALL_OLAP2_DIM_LEVEL_ATTRIBUTES lists the level attributes within each level.

Column	Data Type	NULL	Description
OWNER	VARCHAR2(30)	NOT NULL	Owner of the dimension containing the level.
DIMENSION_NAME	VARCHAR2(30)	NOT NULL	Name of the dimension containing the level.
ATTRIBUTE_NAME	VARCHAR2(30)		Name of the level attribute. If no attribute name is specified, the column name is used.
DISPLAY_NAME	VARCHAR2(30)		Display name for the level attribute.
DESCRIPTION	VARCHAR2(2000)		Description of the level attribute.
DETERMINED_BY_LEVEL_NAME	VARCHAR2(30)	NOT NULL	Name of the level.

ALL_OLAP2_DIM_LEVEL_ATTR_MAPS

ALL_OLAP2_DIM_LEVEL_ATTR_MAPS shows the mapping of each level attribute to a column.

The mapping of level attributes to levels is dependent on hierarchy. The same level may have different attributes when it is used in different hierarchies.

Column	Data Type	NULL	Description
OWNER	VARCHAR2(30)	NOT NULL	Owner of the dimension.
DIMENSION_NAME	VARCHAR2(30)	NOT NULL	Name of the dimension.
HIERARCHY_NAME	VARCHAR2(30)		Name of the hierarchy containing this level.
ATTRIBUTE_NAME	VARCHAR2(30)		Name of a dimension attribute grouping containing this level attribute.
LVL_ATTRIBUTE_NAME	VARCHAR2(30)	NOT NULL	Name of the level attribute, or name of the column if the level attribute name is not specified.
LEVEL_NAME	VARCHAR2(30)	NOT NULL	Name of the level.
TABLE_OWNER	VARCHAR2(30)	NOT NULL	Owner of the dimension table containing the level and level attribute.
TABLE_NAME	VARCHAR2(30)	NOT NULL	Name of the dimension table containing the level and level attribute columns.
COLUMN_NAME	VARCHAR2(30)	NOT NULL	Name of the column containing the level attribute.
DTYPE	VARCHAR2(10)	NOT NULL	Data type of the column containing the level attribute.

ALL_OLAP2_ENTITY_DESC_USES

ALL_OLAP2_ENTITY_DESC_USES lists the reserved attributes and shows whether or not dimensions are time dimensions.

Column	Data Type	NULL	Description
DESCRIPTOR_ID	NUMBER	NOT NULL	Name of the reserved attribute or dimension type. The reserved dimension attributes are listed in Table 11–1, " Reserved Dimension Attributes" on page 11-2. The reserved level attributes are listed in Table 14–1, " Reserved Level Attributes" on page 14-2.
ENTITY_OWNER	VARCHAR2(30)	NOT NULL	Owner of the metadata entity.
ENTITY_NAME	VARCHAR2(30)	NOT NULL	Name of the metadata entity.
CHILD_ENTITY_NAME	VARCHAR2(30)		Name of the child entity (if applicable). A dimension attribute is a child entity of a dimension. A level attribute is a child entity of a dimension attribute.
SECONDARY_CHILD_ENTITY_NAME	VARCHAR2(30)		Name of the secondary child entity name (if applicable). A dimension attribute is a child entity of a dimension. A level attribute is a child entity of a dimension attribute. A level attribute could be the secondary child entity of a dimension.

ALL_OLAP2_ENTITY_EXT_PARMS

ALL_OLAP2_ENTITY_EXT_PARMS lists the following OLE DB metadata descriptors: Default Member, Dense Indicator, Fact Table Join, and Estimated Cardinality.

The OLE DB metadata descriptors are described in Table 8–1, "OLAP Catalog Metadata Descriptors" on page 8-1.

Column	Data Type	NULL	Description
DESCRIPTOR_ID	NUMBER(38)		ID of the metadata descriptor.

Column	Data Type	NULL	Description
DESCRIPTOR_NAME	VARCHAR2(240)		One of the following metadata descriptor names: **Default Member** — The default dimension member within a hierarchy. The Default Member descriptor is set by the CWM2_OLAP_CLASSIFY.ADD_ENTITY_DEFAULTMEMBER_USE procedure (described on page 8-5). **Dense Indicator** — Specifies whether the data is sparse or dense over a dimension of a cube. The Dense Indicator descriptor is set by the CWM2_OLAP_CLASSIFY.ADD_ENTITY_DENSEINDICATOR_USE procedure (described on page 8-6). **Fact Table Join** — Specifies the key columns in a dimension table that satisfy the foreign key columns in the fact table. The Fact Table Join descriptor applies only to CWM2 metadata. The Fact Table Join descriptor is set by the CWM2_OLAP_CLASSIFY.ADD_ENTITY_FACTJOIN_USE procedure (described on page 8-8). **Estimated Cardinality** — The Estimated Cardinality descriptor is set by the CWM2_OLAP_CLASSIFY.ADD_ENTITY_CARDINALITY_USE procedure (described on page 8-4).
ENTITY_OWNER	VARCHAR2(240)		Schema of the cube or dimension.
ENTITY_NAME	VARCHAR2(240)		Name of the cube or dimension.
CHILD_ENTITY_NAME	VARCHAR2(30)		Name of a child of the cube or dimension. For example, a dimension attribute is a child of a dimension, and a measure is a child of a cube. If the descriptor applies to a cube or dimension, this parameter is NULL.
SECONDARY_CHILD_ENTITY_NAME	VARCHAR2(30)		Name of a child of the child entity. For example, a level attribute is a child of a level, which is a child of a dimension. If the descriptor applies to a cube or dimension, or a child of a cube or dimension, this parameter is NULL.
PARAMETER_NAME	VARCHAR2(80)		User-defined label for the descriptor.
PARAMETER_VALUE	VARCHAR2(4000)		Value of the descriptor. For the Fact Table Join descriptor, this parameter contains the table owner.
PARAMETER_VALUE2	VARCHAR2(4000)		Table name for Fact Table Join descriptor.
PARAMETER_VALUE3	VARCHAR2(4000)		Column name for Fact Table Join descriptor.

Column	Data Type	NULL	Description
PARAMETER_ VALUE4	VARCHAR2(4000)		Hierarchy name for Fact Table Join descriptor.
POSITION	NUMBER		Position in mult-column key for Fact Table Join descriptor.

ALL_OLAP2_ENTITY_PARAMETERS

ALL_OLAP2_ENTITY_PARAMETERS lists the OLE DB metadata descriptors not listed in ALL_OLAP2_ENTITY_EXT_PARMS. Additionally, it includes all the descriptors from ALL_OLAP2_ENTITY_DESC_USES.

The OLE DB metadata descriptors are described in Table 8–1, "OLAP Catalog Metadata Descriptors" on page 8-1.

Column	Data Type	NULL	Description
DESCRIPTOR_ID	NUMBER(38)		ID of metadata descriptor.
DESCRIPTOR_NAME	VARCHAR2(240)		Name of the metadata descriptor.
ENTITY_OWNER	VARCHAR2(240)		Schema of the cube or dimension.
ENTITY_NAME	VARCHAR2(240)		Name of the cube or dimension.
CHILD_ENTITY_ NAME	VARCHAR2(240)		Name of a child of the cube or dimension. For example, a dimension attribute is a child of a dimension, and a measure is a child of a cube. If the descriptor applies to a cube or dimension, this parameter is NULL.
SECONDARY_ CHILD_ENTITY_ NAME	VARCHAR2(240)		Name of a child of the child entity. For example, a level attribute is a child of a level, which is a child of a dimension. If the descriptor applies to a cube or dimension, or a child of a cube or dimension, this parameter is NULL.
PARAMETER_NAME	VARCHAR2(30)		User-defined label for the descriptor.
PARAMETER_VALUE	VARCHAR2(80)		Value of the descriptor.

ALL_OLAP2_FACT_LEVEL_USES

ALL_OLAP2_FACT_LEVEL_USES shows the joins between dimension tables and fact tables in a star or snowflake schema. For more information, see "Joining Fact Tables with Dimension Tables" on page 2-12.

Column	Data Type	NULL	Description
OWNER	VARCHAR2(30)	NOT NULL	Owner of the cube.
CUBE_NAME	VARCHAR2(30)	NOT NULL	Name of the cube.
DIMENSION_OWNER	VARCHAR2(30)	NOT NULL	Owner of the dimension.
DIMENSION_NAME	NUMBER	NOT NULL	Name of the dimension.
DIMENSION_ALIAS	VARCHAR2(30)		Dimension alias (if applicable).
HIERARCHY_NAME		NOT NULL	Name of the hierarchy.
DIM_HIER_COMBO_ID	NUMBER	NOT NULL	ID of the dimension hierarchy combination associated with this fact table.
LEVEL_NAME	VARCHAR2(30)		Name of the level within the hierarchy where the mapping occurs.
FACT_TABLE_OWNER	VARCHAR2(30)	NOT NULL	Owner of the fact table.
FACT_TABLE_NAME	VARCHAR2(30)	NOT NULL	Name of the fact table.
COLUMN_NAME	VARCHAR2(30)	NOT NULL	Name of the foreign key column in the fact table.
POSITION	NUMBER		Position of this column within a multi-column key.
DIMENSION_KEYMAP_TYPE	VARCHAR2(30)	NOT NULL	Type of key mapping for the fact table. Values may be: LL (Lowest Level), when only lowest-level dimension members are stored in the key column. The fact table is unsolved. ET (Embedded Totals), when dimension members for all level combinations are stored in the key column. The fact table is solved (contains embedded totals for all level combinations).
FOREIGN_KEY_NAME	VARCHAR2(30)		Name of the foreign key constraint applied to the foreign key column. Constraints are not used by the CWM2 APIs.

.

ALL_OLAP2_FACT_TABLE_GID

ALL_OLAP2_FACT_TABLE_GID shows the Grouping ID column for each hierarchy in each fact table. For more information, see "Grouping ID Column" on page 1-30.

Column	Data Type	NULL	Description
OWNER	VARCHAR2(30)	NOT NULL	Owner of the cube.
CUBE_NAME	VARCHAR2(30)	NOT NULL	Name of the cube.
DIMENSION_OWNER	VARCHAR2(30)	NOT NULL	Owner of the dimension.
DIMENSION_NAME	VARCHAR2(30)	NOT NULL	Name of the dimension
HIERARCHY_NAME	VARCHAR2(30)	NOT NULL	Name of the hierarchy.
DIM_HIER_COMBO_ID	NUMBER	NOT NULL	ID of the dimension-hierarchy association.
FACT_TABLE_OWNER	VARCHAR2(30)	NOT NULL	Owner of the fact table.
FACT_TABLE_NAME	VARCHAR2(30)	NOT NULL	Name of the fact table.
COLUMN_NAME	VARCHAR2(30)	NOT NULL	Name of the GID column.

ALL_OLAP2_HIER_CUSTOM_SORT

ALL_OLAP2_HIER_CUSTOM_SORT shows the sort order for level columns within hierarchies. Custom sorting information is optional.

Custom sorting information specifies how to sort the members of a hierarchy based on columns in the dimension table. The specific columns in the dimension tables may be the same as the key columns or may be related attribute columns.

Custom sorting can specify that the column be sorted in ascending or descending order, with nulls first or nulls last. Custom sorting can be applied at multiple levels of a dimension.

Column	Data Type	NULL	Description
OWNER	VARCHAR2(30)	NOT NULL	Owner of the dimension.
DIMENSION_NAME	VARCHAR2(30)	NOT NULL	Name of the dimension.
HIERARCHY_NAME	VARCHAR2(30)	NOT NULL	Name of the hierarchy.
TABLE_OWNER	VARCHAR2(30)	NOT NULL	Owner of the dimension table.
TABLE_NAME	VARCHAR2(30)	NOT NULL	Name of the dimension table.
COLUMN_NAME	VARCHAR2(30)	NOT NULL	Name of the column to be sorted.

Column	Data Type	NULL	Description
POSITION	NUMBER	NOT NULL	Represents the position within a multi-column SORT_POSITION. In most cases, a single column represents SORT_POSITION, and the value of POSITION is 1.
SORT_POSITION	NUMBER	NOT NULL	Position within the sort order of the level to be sorted.
SORT_ORDER	VARCHAR2(4)	NOT NULL	Sort order. Can be either Ascending or Descending.
NULL_ORDER	VARCHAR2(5)	NOT NULL	Where to insert null values in the sort order. Can be either Nulls First or Nulls Last.

ALL_OLAP2_JOIN_KEY_COLUMN_USES

ALL_OLAP2_JOIN_KEY_COLUMN_USES shows the joins between two levels in a hierarchy. The joins are between dimension tables in a snowflake schema, and between level columns in a star schema.

If the level is mapped to more than one column, each column mapping is represented in a separate row in the view.

Column	Data Type	NULL	Description
OWNER	VARCHAR2(30)	NOT NULL	Owner of the dimension.
DIMENSION_NAME	VARCHAR2(30)	NOT NULL	Name of the dimension.
HIERARCHY_NAME	VARCHAR2(30)	NOT NULL	Name of the hierarchy.
CHILD_LEVEL_NAME	VARCHAR2(30)	NOT NULL	Child level in the hierarchy.
TABLE_OWNER	VARCHAR2(30)	NOT NULL	Owner of the dimension table.
TABLE_NAME	VARCHAR2(30)	NOT NULL	Name of the dimension table.
COLUMN_NAME	VARCHAR2(30)	NOT NULL	Name of the child level column in the dimension table. In a star schema, this is the column associated with CHILD_LEVEL_NAME. In a snowflake schema, this is the parent column of CHILD_LEVEL_NAME in the same dimension table.
POSITION	NUMBER		Position of column within the key. Applies to multi-column keys only (where the level is mapped to more than one column).

Column	Data Type	NULL	Description
JOIN_KEY_TYPE	VARCHAR2(30)	NOT NULL	The key is of type SNOWFLAKE if the join key is a logical foreign key. The key is of type STAR if the join key refers to a column within the same table.

ALL_OLAP2_LEVEL_KEY_COL_USES

ALL_OLAP2_LEVEL_KEY_COL_USES shows the mapping of each level to a unique key column.

If the level is mapped to more than one column, each column mapping is represented in a separate row in the view.

Column	Data Type	NULL	Description
OWNER	VARCHAR2(30)	NOT NULL	Owner of the dimension.
DIMENSION_NAME	VARCHAR2(30)	NOT NULL	Name of the dimension.
HIERARCHY_NAME	VARCHAR2(30)		Name of the hierarchy that includes this level.
CHILD_LEVEL_NAME	VARCHAR2(30)	NOT NULL	Name of the level.
TABLE_OWNER	VARCHAR2(30)	NOT NULL	Owner of the dimension table.
TABLE_NAME	VARCHAR2(30)	NOT NULL	Name of the dimension table.
COLUMN_NAME	VARCHAR2(30)	NOT NULL	Name of the column that stores CHILD_LEVEL_NAME.
POSITION	NUMBER		Position of the column within the key. Applies to multi-column keys only (where the level is mapped to more than one column).

6

OLAP Fixed Views

Oracle collects statistics in fixed tables, and creates user-accessible views from these tables. This chapter describes the fixed views that contain data on Oracle OLAP.

> **See Also:** For additional information about fixed tables and views, refer to the following:
>
> - *Oracle Database Reference*
> - *Oracle Database Performance Tuning Guide*

This chapter contains the following topics:

- System Tables Referenced by OLAP Fixed Views
- Summary of OLAP Fixed Views
- V$AW_AGGREGATE_OP
- V$AW_ALLOCATE_OP
- V$AW_CALC
- V$AW_LONGOPS
- V$AW_OLAP
- V$AW_SESSION_INFO

System Tables Referenced by OLAP Fixed Views

Each Oracle database instance maintains a set of virtual tables that record current database activity and store data about the instance. These tables are called the **V$** tables. They are also referred to as the **dynamic performance tables**, because they store information that pertains primarily to performance. Views of the V$ tables are sometimes called **fixed views** because they cannot be altered or removed by the database administrator.

The V$ tables collect data on internal disk structures and memory structures. They are continuously updated while the database is in use. Among them are tables that collect data on Oracle OLAP.

The `SYS` user owns the V$ tables. In addition, any user with the `SELECT CATALOG` role can access the tables. The system creates views from these tables and creates public synonyms for the views. The views are also owned by `SYS`, but the DBA can grant access to them to a wider range of users.

The names of the OLAP V$ tables begin with `V$AW`. The view names also begin with `V$AW`. The following sample SQL*Plus session shows the list of OLAP system tables.

```
% sqlplus '/ as sysdba'
.
.
.
SQL> SELECT name FROM v$fixed_table WHERE name LIKE 'V$AW%';

NAME
- - - - - - - - - - - - - - -
V$AW_AGGREGATE_OP
V$AW_ALLOCATE_OP
V$AW_CALC
V$AW_LONGOPS
V$AW_OLAP
V$AW_SESSION_INFO
```

> **See Also:** For more information on the V$ views in the Database, see the *Oracle Database Reference*.

Summary of OLAP Fixed Views

Table 6–1 briefly describes each OLAP fixed view.

Table 6–1 OLAP Fixed Views

Fixed View	Description
V$AW_AGGREGATE_OP	Lists the aggregation operators available in the OLAP DML.
V$AW_ALLOCATE_OP	Lists the allocation operators available in the OLAP DML.
V$AW_CALC	Collects information about the use of cache space.

Table 6–1 (Cont.) OLAP Fixed Views

Fixed View	Description
V$AW_LONGOPS	Collects status information about SQL fetches.
V$AW_OLAP	Collects information about the status of active analytic workspaces.
V$AW_SESSION_INFO	Collects information about each active session.

V$AW_AGGREGATE_OP

`V$AW_AGGREGATE_OP` lists the aggregation operators available in the OLAP DML. You can use this view in an application to provide a list of choices.

Column	Datatype	NULL	Description
NAME	VARCHAR2		Operator keyword used in the OLAP DML RELATION command
LONGNAME	VARCHAR2		Descriptive name for the operator
DEFAULT_WEIGHT	NUMBER		Default weight factor for weighted operators

V$AW_ALLOCATE_OP

`V$AW_ALLOCATE_OP` lists the allocation operators available in the OLAP DML. You can use this view in an application to provide a list of choices.

Column	Datatype	NULL	Description
NAME	VARCHAR2		Operator keyword used in the OLAP DML RELATION command
LONGNAME	VARCHAR2		Descriptive name for the operator

V$AW_CALC

`V$AW_CALC` reports on the effectiveness of various caches used by Oracle OLAP. Because OLAP queries tend to be iterative, the same data is typically queried repeatedly during a session. The caches provide much faster access to data that has already been calculated during a session than would be possible if the data had to be recalculated for each query.

The more effective the caches are, the better the response time experienced by users. An ineffective cache (that is, one with few hits and many misses) probably indicates that the data is not being stored optimally for the way it is being viewed. To improve runtime performance, you may need to reorder the dimensions of the variables (that is, change the order of fastest to slowest varying dimensions).

Oracle OLAP uses the following caches:

- **Aggregate cache**. An optional cache used by the AGGREGATE function in the OLAP DML. The AGGREGATE function calculates aggregate data at runtime in response to a query. When a cache is maintained, AGGREGATE can retrieve data that was previously calculated during the session instead of recalculating it each time the data is queried.
- **Session cache**. Oracle OLAP maintains a cache for each session for storing the results of calculations. When the session ends, the contents of the cache are discarded.
- **Page pool**. A cache allocated from the program global area (PGA) in the database, which Oracle OLAP maintains for the session. The page pool is associated with a particular session and is shared by all attached analytic workspaces. If the page pool becomes too full, then Oracle OLAP writes some of the pages to the database cache. When an UPDATE command is issued in the OLAP DML, the changed pages associated with that analytic workspace are written to the permanent LOB, using temporary segments as the staging area for streaming the data to disk. The size of the page pool is controlled by the OLAP_PAGE_POOL initialization parameter.
- **Database cache**. The larger cache maintained by the Oracle RDBMS for the database instance.

> **See Also:** *Oracle OLAP DML Reference* for full discussions of data storage issues and aggregation. See the CACHE command for information about defining an aggregate cache.

Column	Datatype	Description
AGGREGATE_CACHE_HITS	NUMBER	The number of times a dimension member is found in the aggregate cache (a hit). The number of hits for run-time aggregation can be increased by fetching data across the dense dimension.
AGGREGATE_CACHE_MISSES	NUMBER	The number of times a dimension member is not found in the aggregate cache and must be read from disk (a miss).

Column	Datatype	Description
SESSION_CACHE_HITS	NUMBER	The number of times the data is found in the session cache (a hit).
SESSION_CACHE_MISSES	NUMBER	The number of times the data is not found in the session cache (a miss).
POOL_HITS	NUMBER	The number of times the data is found in a page in the OLAP page pool (a hit).
POOL_MISSES	NUMBER	The number of times the data is not found in the OLAP page pool (a miss).
POOL_NEW_PAGES	NUMBER	The number of newly created pages in the OLAP page pool that have not yet been written to the workspace LOB.
POOL_RECLAIMED_PAGES	NUMBER	The number of previously unused pages that have been recycled with new data.
CACHE_WRITES	NUMBER	The number of times the data from the OLAP page pool has been written to the database cache.
POOL_SIZE	NUMBER	The number of pages in the OLAP page pool.

V$AW_LONGOPS

`V$AW_LONGOPS` provides status information about active SQL cursors initiated in the OLAP DML.

A cursor can be initiated within the OLAP DML using `SQL FETCH`, `SQL IMPORT`, or `SQL EXECUTE`, that is, SQL statements that can be declared and executed.

Column	Datatype	Description
SESSION_ID	NUMBER	The identifier for the session in which the fetch is executing. This table can be joined with V$SESSION to get the user name.
CURSOR_NAME	VARCHAR2	The name assigned to the cursor in an OLAP DML `SQL DECLARE CURSOR` or `SQL PREPARE CURSOR` command.
COMMAND	VARCHAR2	An OLAP DML command (`SQL IMPORT`, `SQL FETCH`, or `SQL EXECUTE`) that is actively fetching data from relational tables.

Column	Datatype	Description
STATUS	VARCHAR2	One of the following values: ▪ EXECUTING. The command has begun executing. ▪ FETCHING. Data is being fetched into the analytic workspace. ▪ FINISHED. The command has finished executing. This status appears very briefly before the record disappears from the table.
ROWS_PROCESSED	NUMBER	The number of rows already inserted, updated, or deleted.
START_TIME	TIMESTAMP	The time the command started executing.

V$AW_OLAP

V$AW_OLAP provides a record of active sessions and their use with analytic workspaces. A row is generated whenever an analytic workspace is created or attached. The first row for a session is created when the first DML command is issued. It identifies the SYS.EXPRESS workspace, which is attached automatically to each session. Rows related to a particular analytic workspace are deleted when the workspace is detached from the session or the session ends.

Column	Datatype	Description
SESSION_ID	NUMBER	A unique numerical identifier for a session.
AW_NUMBER	NUMBER	A unique numerical identifier for an analytic workspace.
ATTACH_MODE	VARCHAR2(10)	READ ONLY or READ WRITE.
GENERATION	NUMBER	The generation of an analytic workspace. Each UPDATE creates a new generation. Sessions attaching the same workspace between UPDATE commands share the same generation.
TEMP_SPACE_PAGES	NUMBER	The number of pages stored in temporary segments for the analytic workspace.
TEMP_SPACE_READS	NUMBER	The number of times data has been read from a temporary segment and not from the page pool.
LOB_READS	NUMBER	The number of times data has been read from the table where the analytic workspace is stored (the permanent LOB).

Column	Datatype	Description
POOL_CHANGED_PAGES	NUMBER	The number of pages in the page pool that have been modified in this analytic workspace.
POOL_UNCHANGED_PAGES	NUMBER	The number of pages in the page pool that have not been modified in this analytic workspace.

V$AW_SESSION_INFO

V$AW_SESSION_INFO provides information about each active session.

A transaction is a single exchange between a client session and Oracle OLAP. Multiple OLAP DML commands can execute within a single transaction, such as in a call to the DBMS_AW.EXECUTE procedure.

Column	Datatype	Description
CLIENT_TYPE	VARCHAR2(64)	OLAP
SESSION_STATE	VARCHAR2(64)	TRANSACTING, NOT_TRANSACTING, EXCEPTION_HANDLING, CONSTRUCTING, CONSTRUCTED, DECONSTRUCTING, or DECONSTRUCTED
SESSION_HANDLE	NUMBER	The session identifier
USERID	VARCHAR2(64)	The database user name under which the session opened
CURR_DML_COMMAND	VARCHAR2(64)	The DML command currently being executed
PREV_DML_COMMAND	VARCHAR2(64)	The DML command most recently completed.
TOTAL_TRANSACTION	NUMBER	The total number of transactions executed within the session; this number provides a general indication of the level of activity in the session
TOTAL_TRANSACTION_TIME	NUMBER	The total elapsed time in milliseconds in which transactions were being executed
AVERAGE_TRANSACTION_TIME	NUMBER	The average elapsed time in milliseconds to complete a transaction
TRANSACTION_CPU_TIME	NUMBER	The total CPU time in milliseconds used to complete the most recent transaction
TOTAL_TRANSACTION_CPU_TIME	NUMBER	The total CPU time used to execute all transactions in this session; this total does not include transactions that are currently in progress

Column	Datatype	Description
AVERAGE_TRANSACTION_CPU_TIME	NUMBER	The average CPU time to complete a transaction; this average does not include transactions that are currently in progress

7

CWM2_OLAP_CATALOG

The CWM2_OLAP_CATALOG package provides procedures for managing measure folders.

Note: The term **catalog**, when used in the context of the CWM2_OLAP_CATALOG package, refers to a measure folder.

See Also:

- Chapter 15, "CWM2_OLAP_MEASURE"
- Chapter 2, "Creating OLAP Catalog Metadata with CWM2"

This chapter discusses the following topics:

- Understanding Measure Folders
- Example: Creating a Measure Folder
- Summary of CWM2_OLAP_CATALOG Subprograms

Understanding Measure Folders

A measure folder is an OLAP metadata entity. This means that it is a logical object, identified by name and owner, within the OLAP Catalog.

Use the procedures in the CWM2_OLAP_CATALOG package to create, populate, drop, and lock measure folders, and to specify descriptive information for display purposes.

Measure folders provide a mechanism for grouping related measures. They can contain measures and nested measure folders. Access to measure folders is schema-independent. All measure folders are visible to any client. However, access to the measures themselves depends on the client's access rights to the underlying tables.

> **See Also:** *Oracle OLAP Application Developer's Guide* for more information on measure folders and the OLAP metadata model.

Example: Creating a Measure Folder

The following statements create a measure folder called `PHARMACEUTICALS` and add the measure `UNIT_COST` from the cube `SH.COST_CUBE`. The measure folder is at the root level.

```
execute cwm2_olap_catalog.create_catalog
     ('PHARMACEUTICALS', 'Pharmaceutical Sales and Planning');
execute cwm2_olap_catalog.add_catalog_entity
     ('PHARMACEUTICALS',  'SH', 'COST_CUBE', UNIT_COST');
```

Summary of CWM2_OLAP_CATALOG Subprograms

Table 7–1 CWM2_OLAP_CATALOG Subprograms

Subprogram	Description
ADD_CATALOG_ENTITY Procedure on page 7-3	Adds a measure to a measure folder.
CREATE_CATALOG Procedure on page 7-4	Creates a measure folder.
DROP_CATALOG Procedure on page 7-4	Drops a measure folder.
LOCK_CATALOG Procedure on page 7-5	Locks a measure folder.
REMOVE_CATALOG_ENTITY Procedure on page 7-5	Removes a measure from a measure folder.
SET_CATALOG_NAME Procedure on page 7-6	Sets the name of a measure folder.
SET_DESCRIPTION Procedure on page 7-6	Sets the description of a measure folder.
SET_PARENT_CATALOG Procedure on page 7-7	Sets the parent folder of a measure folder.

ADD_CATALOG_ENTITY Procedure

This procedure adds a measure to a measure folder.

Syntax

```
ADD_CATALOG_ENTITY  (
               catalog_name      IN   VARCHAR2,
               cube_owner        IN   VARCHAR2,
               cube_name         IN   VARCHAR2,
               measure_name      IN   VARCHAR2);
```

Parameters

Table 7–2 ADD_CATALOG_ENTITY Procedure Parameters

Parameter	Description
catalog_name	Name of the measure folder.
cube_owner	Owner of the cube.
cube_name	Name of the cube.
measure_name	Name of the measure to be added to the measure folder.

CREATE_CATALOG Procedure

This procedure creates a new measure folder.

Descriptions and display properties must also be established as part of measure folder creation. Once the measure folder has been created, you can override these properties by calling other procedures in this package.

Syntax

```
CREATE_CATALOG (
               catalog_name       IN   VARCHAR2,
               description        IN   VARCHAR2,
               parent_catalog     IN   VARCHAR2 DEFAULT NULL);
```

Parameters

Table 7–3 CREATE_CATALOG Procedure Parameters

Parameter	Description
catalog_name	Name of the measure folder.
description	Description of the measure folder.
parent_catalog	Optional parent measure folder.

DROP_CATALOG Procedure

This procedure drops a measure folder. If the measure folder contains other measure folders, they are also dropped.

Syntax

```
DROP_CATALOG (
              catalog_name     IN   VARCHAR2);
```

Parameters

Table 7–4 DROP_CATALOG Procedure Parameters

Parameter	Description
catalog_name	Name of the measure_folder.

LOCK_CATALOG Procedure

This procedure locks the measure folder's metadata for update by acquiring a database lock on the row that identifies the measure folder in the CWM2 model table.

Syntax

```
LOCK_CATALOG (
              catalog_name     IN   VARCHAR2,
              wait_for_lock    IN   BOOLEAN DEFAULT FALSE);
```

Parameters

Table 7–5 LOCK_CATALOG Procedure Parameters

Parameter	Description
catalog_name	Name of the measure folder
wait_for_lock	(Optional) Whether or not to wait for the measure folder to be available when it is already locked by another user. If you do not specify a value for this parameter, the procedure does not wait to acquire the lock.

REMOVE_CATALOG_ENTITY Procedure

This procedure removes a measure from a measure folder.

Syntax

```
REMOVE_CATALOG_ENTITY  (
               catalog_name      IN   VARCHAR2,
               cube_owner        IN   VARCHAR2,
               cube_name         IN   VARCHAR2,
               measure_name      IN   VARCHAR2);
```

Parameters

Table 7–6 REMOVE_CATALOG_ENTITY Procedure Parameters

Parameter	Description
catalog_name	Name of the measure folder.
cube_owner	Owner of the cube.
cube_name	Name of the cube.
measure_name	Name of the measure to be removed from the measure folder.

SET_CATALOG_NAME Procedure

This procedure sets the name for a measure folder.

Syntax

```
SET_CATALOG_NAME (
               old_catalog_name     IN   VARCHAR2,
               new_catalog_name     IN   VARCHAR2);
```

Parameters

Table 7–7 SET_CATALOG_NAME Procedure Parameters

Parameter	Description
old_catalog_name	Old measure folder name.
new_catalog_name	New measure folder name.

SET_DESCRIPTION Procedure

This procedure sets the description for a measure folder.

Syntax

```
SET_DESCRIPTION (
              catalog_name     IN   VARCHAR2,
              description      IN   VARCHAR2);
```

Parameters

Table 7–8 SET_DESCRIPTION Procedure Parameters

Parameter	Description
catalog_name	Name of the measure folder
description	Description of the measure folder.

SET_PARENT_CATALOG Procedure

This procedure sets a parent measure folder for a measure folder.

Syntax

```
SET_PARENT_CATALOG (
              catalog_name            IN   VARCHAR2,
              parent_catalog_name     IN   VARCHAR2   DEFAULT NULL);
```

Parameters

Table 7–9 SET_PARENT_CATALOG Procedure Parameters

Parameter	Description
catalog_name	Name of the measure folder.
parent_catalog_name	Name of the parent measure folder. If the measure folder is at the root level, this parameter is null.

8

CWM2_OLAP_CLASSIFY

The CWM2_OLAP_CLASSIFY package provides procedures for managing metadata extensions for the OLAP API.

This chapter discusses the following topics:

- OLAP Catalog Metadata Descriptors
- Example: Creating Descriptors
- Summary of CWM2_OLAP_CLASSIFY Subprograms

OLAP Catalog Metadata Descriptors

The OLAP Catalog metadata descriptors provide additional information about your data. These descriptors can be used by the OLAP API.

The OLAP Catalog metadata descriptors are described in Table 8–1, "OLAP Catalog Metadata Descriptors".

You can view the descriptors that have been set for your OLAP Catalog metadata in the views ALL_OLAP2_ENTITY_EXT_PARMS (described on page 5-12) and ALL_OLAP2_ENTITY_PARAMETERS (described on page 5-14).

Table 8–1 OLAP Catalog Metadata Descriptors

Descriptor	Applies To	Description
`Level Standard`	level	The level is not in a time dimension.
`Level Year`	level	The year level in a time dimension.
`Level HalfYear`	level	The half year level in a time dimension.
`Level Quarter`	level	The quarter level in a time dimension.

Table 8–1 OLAP Catalog Metadata Descriptors

Descriptor	Applies To	Description
`Level Month`	level	The month level in a time dimension.
`Level Week`	level	The week level in a time dimension.
`Level Day`	level	The day level in a time dimension.
`Level Hour`	level	The hour level in a time dimension.
`Level Minute`	level	The minutes level in a time dimension.
`Level Second`	level	The seconds level in a time dimension.
`Value Separator`	dimension	The separator character used by the OLAP API to construct the names of dimension members. The default separator is "::".
`Skip Level`	hierarchy	Whether or not the hierarchy supports skip levels. An example of a skip level hierarchy is City-State-Country, where Washington D.C. is a City whose parent is a Country.
`Measure Format`	measure	The display format for a measure.
`Measure Unit`	measure	The unit of measurement of a measure.
`Fact Table Join`	hierarchy	The key columns in a dimension table that satisfy the join to a fact table. This descriptor applies to CWM2 metadata only.
`Default Member`	hierarchy	The default dimension member in a hierarchy.
`Dense Indicator`	dimension	Whether or not the data over a given dimension of a cube is dense or sparse.
`Estimated Cardinality`	level	Estimated number of dimension members in a given level.

Example: Creating Descriptors

The following examples show how to set some of the metadata descriptors.

Note: If you have used Enterprise Manager to create your OLAP metadata, be sure to respect the case of metadata names.

The following statements specify the quarter, month, and year levels in the time dimension `XADEMO.TIME`.

```
execute cwm2_olap_classify.add_entity_descriptor_use
     ('Level Year', 'LEVEL', 'XADEMO', 'TIME', 'L1');
execute cwm2_olap_classify.add_entity_descriptor_use
     ('Level Quarter', 'LEVEL', 'XADEMO', 'TIME', 'L2');
execute cwm2_olap_classify.add_entity_descriptor_use
     ('Level Month', 'LEVEL', 'XADEMO', 'TIME', 'L3');
```

The following statement indicates that the value separator used by the OLAP API to contruct dimesion member names for `XADEMO.TIME` is the default ("::").

```
execute cwm2_olap_classify.add_entity_descriptor_use
     ('Value Separator', 'DIMENSION', 'XADEMO', 'TIME', NULL, NULL,
      'Value Separator','::');
```

The following statement indicates that the data in the cube `XADEMO.ANALYTIC_CUBE` is dense over Time and Geography, but sparse over Channel and Product.

```
execute cwm2_olap_classify.add_entity_denseindicator_use
     ('XADEMO', 'ANALYTIC_CUBE', 'XADEMO', 'TIME', 'YES');
execute cwm2_olap_classify.add_entity_denseindicator_use
     ('XADEMO', 'ANALYTIC_CUBE', 'XADEMO', 'GEOGRAPHY', 'YES');
execute cwm2_olap_classify.add_entity_denseindicator_use
     ('XADEMO', 'ANALYTIC_CUBE', 'XADEMO', 'CHANNEL', 'NO');
execute cwm2_olap_classify.add_entity_denseindicator_use
     ('XADEMO', 'ANALYTIC_CUBE', 'XADEMO', 'PRODUCT', 'NO');
```

The following statement removes the `Dense Indicator` descriptors from `XADEMO.ANALYTIC_CUBE`.

```
execute cwm2_olap_classify.remove_entity_descriptor_use
  ('Dense Indicator', 'DENSE INDICATOR','XADEMO', 'ANALYTIC_CUBE',
   'XADEMO', 'CHANNEL');
execute cwm2_olap_classify.remove_entity_descriptor_use
  ('Dense Indicator', 'DENSE INDICATOR','XADEMO', 'ANALYTIC_CUBE',
   'XADEMO', 'PRODUCT');
execute cwm2_olap_classify.remove_entity_descriptor_use
  ('Dense Indicator', 'DENSE INDICATOR','XADEMO', 'ANALYTIC_CUBE',
   'XADEMO', 'GEOGRAPHY');
execute cwm2_olap_classify.remove_entity_descriptor_use
  ('Dense Indicator', 'DENSE INDICATOR','XADEMO', 'ANALYTIC_CUBE',
   'XADEMO', 'TIME');
```

Summary of CWM2_OLAP_CLASSIFY Subprograms

Table 8–2 CWM2_OLAP_CLASSIFY Subprograms

Subprogram	Description
ADD_ENTITY_CARDINALITY_USE on page 8-4	Adds the `Estimated Cardinality` descriptor to a level of a hierarchy.
ADD_ENTITY_DEFAULTMEMBER_USE on page 8-5	Adds the `Default Member` descriptor to a hierarchy.
ADD_ENTITY_DENSEINDICATOR_USE on page 8-6	Adds the `Dense Indicator` descriptor to a dimension of a cube.
ADD_ENTITY_DESCRIPTOR_USE on page 8-7	Applies a descriptor to a metadata entity.
ADD_ENTITY_FACTJOIN_USE on page 8-8	Adds the `Fact Table Join` descriptor to a CWM2 hierarchy.
REMOVE_ENTITY_DESCRIPTOR_USE on page 8-10	Removes a descriptor from a metadata entity.

ADD_ENTITY_CARDINALITY_USE

This procedure adds the `Estimated Cardinality` descriptor to a level of a hierarchy.

The OLAP Catalog metadata descriptors are described in Table 8–1, "OLAP Catalog Metadata Descriptors".

Syntax

```
ADD_ENTITY_CARDINALITY_USE  (
               dimension_owner        IN   VARCHAR2,
               dimension_name         IN   VARCHAR2,
               hierarchy_name         IN   VARCHAR2,
               level_name             IN   VARCHAR2,
               estimated_cardinality  IN   NUMBER);
```

Parameters

Table 8–3 ADD_ENTITY_CARDINALITY_USE Procedure Parameters

Parameter	Description
dimension_owner	Owner of the dimension.
dimension_name	Name of the dimension.
hierarchy_name	Hierarchy within the dimension. If the dimension has no hierarchy, specify NULL.
level_name	Level within the hierarchy.
estimated_cardinality	Estimated number of dimension members in the level.

Example

The following statement sets the estimated cardinality of a level in the Standard hierarchy of the Geography dimension.

```
execute cwm2_olap_classify.add_entity_cardinality_use
          ('XADEMO', 'GEOGRAPHY', 'STANDARD', 'L4', 60);
```

ADD_ENTITY_DEFAULTMEMBER_USE

This procedure adds the `Default Member` descriptor to a hierarchy.

The OLAP Catalog metadata descriptors are described in Table 8–1, "OLAP Catalog Metadata Descriptors".

Syntax

```
ADD_ENTITY_DEFAULTMEMBER_USE  (
               dimension_owner        IN   VARCHAR2,
               dimension_name         IN   VARCHAR2,
               hierarchy_name         IN   VARCHAR2,
               default_member         IN   VARCHAR2,
               default_member_level   IN   VARCHAR2,
               position               IN   NUMBER DEFAULT NULL);
```

Parameters

Table 8–4 ADD_ENTITY_DEFAULTMEMBER_USE Procedure Parameters

Parameter	Description
`dimension_owner`	Owner of the dimension.
`dimension_name`	Name of the dimension.
`hierarchy_name`	Name of the hierarchy.
`default_member`	Name of a dimension member in the hierarchy.
`default_member_level`	Level of the default dimension member.
`position`	Position of the default member within a multi-column key. If position is not meaningful, this parameter is NULL (default).

Example

The following statement sets the default member of the Standard hierarchy in the Geography dimension to Paris.

```
execute cwm2_olap_classify.add_entity_defaultmember_use
          ('XADEMO', 'GEOGRAPHY', 'STANDARD', 'Paris', 'L4');
```

ADD_ENTITY_DENSEINDICATOR_USE

This procedure adds the `Dense Indicator` descriptor to a dimension of a cube.

The OLAP Catalog metadata descriptors are described in Table 8–1, "OLAP Catalog Metadata Descriptors".

Syntax

```
ADD_ENTITY_DENSEINDICATOR_USE  (
               cube_owner        IN   VARCHAR2,
               cube_name         IN   VARCHAR2,
               dimension_owner   IN   VARCHAR2,
               dimension_name    IN   VARCHAR2,
               dense_indicator   IN   VARCHAR2 );
```

Parameters

Table 8–5 ADD_ENTITY_DENSEINDICATOR_USE Procedure Parameters

Parameter	Description
cube_owner	Owner of the cube.
cube_name	Name of the cube.
dimension_owner	Owner of the dimension.
dimension_name	Name of the dimension.
dense_indicator	YES indicates that the data over this dimension is dense. This means that data exists for most dimension members. NO indicates that the data over this dimension is sparse. This means that there is no data for many of the dimension members.

Example

See "Example: Creating Descriptors" on page 8-2.

ADD_ENTITY_DESCRIPTOR_USE

This procedure adds a descriptor to a metadata entity.

The OLAP Catalog metadata descriptors are described in Table 8–1, "OLAP Catalog Metadata Descriptors".

Syntax

```
ADD_ENTITY_DESCRIPTOR_USE  (
               descriptor_name              IN   VARCHAR2,
               entity_type                  IN   VARCHAR2,
               entity_owner                 IN   VARCHAR2,
               entity_name                  IN   VARCHAR2,
               entity_child_name            IN   VARCHAR2 DEFAULT NULL,
               entity_secondary_child_name  IN   VARCHAR2 DEFAULT NULL,
               parameter_name               IN   VARCHAR2 DEFAULT NULL,
               parameter_value              IN   VARCHAR2 DEFAULT NULL);
```

Parameters

Table 8–6 ADD_ENTITY_DESCRIPTOR_USE Procedure Parameters

Parameter	Description
`descriptor_name`	Name of the descriptor.

`entity_type`	Type of metadata entity to which the descriptor applies. Types are:
	`DIMENSION`
	`HIERARCHY`
	`LEVEL`
	`LEVEL ATTRIBUTE`
	`DIMENSION ATTRIBUTE`
	`CUBE`
	`MEASURE`

`entity_owner`	Schema of the cube or dimension.
`entity_name`	Name of the cube or dimension.
`entity_child_name`	Name of a child of the cube or dimension. For example, a dimension attribute is a child of a dimension, and a measure is a child of a cube. If the descriptor applies to a cube or dimension, this parameter is NULL.
`entity_secondary_child_name`	Name of a child of the child entity. For example, a level attribute is a child of a level, which is a child of a dimension. If the descriptor applies to a cube or dimension, or a child of a cube or dimension, this parameter is NULL.
`parameter_name`	Label for the descriptor. You can specify any label that you choose.
`parameter_value`	Value of the descriptor.

Example

See "Example: Creating Descriptors" on page 8-2.

ADD_ENTITY_FACTJOIN_USE

This procedure adds the `Fact Table Join` descriptor to a cube. The `Fact Table Join` descriptor applies to CWM2 metadata only.

The OLAP Catalog metadata descriptors are described in Table 8–1, "OLAP Catalog Metadata Descriptors".

Syntax

```
ADD_ENTITY_FACTJOIN_USE  (
               cube_owner              IN   VARCHAR2,
               cube_name               IN   VARCHAR2,
               dimension_owner         IN   VARCHAR2,
               dimension_name          IN   VARCHAR2,
               hierarchy_name          IN   VARCHAR2,
               dim_table_owner         IN   VARCHAR2,
               dim_table_name          IN   VARCHAR2,
               dim_table_column_name   IN   VARCHAR2,
               position                IN   NUMBER DEFAULT NULL);
```

Parameters

Table 8–7 ADD_ENTITY_FACTJOIN_USE Procedure Parameters

Parameter	Description
`cube_owner`	Owner of the cube.
`cube_name`	Name of the cube.
`dimension_owner`	Owner of a dimension of the cube.
`dimension_name`	Name of the dimension.
`hierarchy_name`	Name of a hierarchy of the dimension.
`dim_table_owner`	Owner of the dimension table.
`dim_table_name`	Name of the dimension table.
`dim_table_column_name`	Key column in the dimension table that maps to a foreign key column in the fact table.
`position`	Position of the key column in a multi-column key. If the key is in a single column, this parameter is NULL (Default).

Example

The following statement adds Fact Table Join descriptor to the Standard hierarchy of the Geography dimension of the `ANALYTIC_CUBE`.

```
execute cwm2_olap_classify.add_entity_factjoin_use
          ('XADEMO', 'ANALYTIC_CUBE', 'XADEMO','GEOGRAPHY, 'STANDARD',
           'XADEMO', 'XADEMO_GEOGRAPHY', 'GEOG_STD_CITY');
```

REMOVE_ENTITY_DESCRIPTOR_USE

This procedure removes a descriptor from an entity.

The OLAP Catalog metadata descriptors are described in Table 8–1, "OLAP Catalog Metadata Descriptors".

Syntax

```
REMOVE_ENTITY_DESCRIPTOR_USE  (
              descriptor_name              IN   VARCHAR2,
              entity_type                  IN   VARCHAR2,
              entity_owner                 IN   VARCHAR2,
              entity_name                  IN   VARCHAR2,
              entity_child_name            IN   VARCHAR2 DEFAULT NULL,
              entity_secondary_child_name  IN   VARCHAR2 DEFAULT NULL);
```

Parameters

Table 8–8 REMOVE_ENTITY_DESCRIPTOR_USE Procedure Parameters

Parameter	Description
descriptor_name	Name of the descriptor to remove.
entity_type	Type of metadata entity to which the descriptor applies. Types are: DIMENSION HIERARCHY LEVEL LEVEL ATTRIBUTE DIMENSION ATTRIBUTE CUBE MEASURE ESTIMATED CARDINALITY DEFAULT MEMBER DENSE INDICATOR FACT TABLE JOIN
entity_owner	Schema of the cube or dimension.
entity_name	Name of the cube or dimension.
entity_child_name	Name of a child of the cube or dimension. For example, a dimension attribute is a child of a dimension, and a measure is a child of a cube. If the descriptor applies to a cube or dimension, this parameter is NULL.

Table 8–8 (Cont.) REMOVE_ENTITY_DESCRIPTOR_USE Procedure Parameters

Parameter	Description
`entity_secondary_child_name`	Name of a child of the child entity. For example, a level attribute is a child of a level, which is a child of a dimension. If the descriptor applies to a cube or dimension, or a child of a cube or dimension, this parameter is NULL.

Example

See "Example: Creating Descriptors" on page 8-2.

9

CWM2_OLAP_CUBE

The CWM2_OLAP_CUBE package provides procedures managing cubes.

> **See Also:** Chapter 2, "Creating OLAP Catalog Metadata with CWM2"

This chapter discusses the following topics:

- Understanding Cubes
- Example: Creating a Cube
- Summary of CWM2_OLAP_CUBE Subprograms

Understanding Cubes

A cube is an OLAP metadata entity. This means that it is a logical object, identified by name and owner, within the OLAP Catalog.

A cube is a multidimensional framework to which you can assign measures. A measure represents data stored in fact tables. The fact tables may be relational tables or views. The views may reference data stored in analytic workspaces.

Use the procedures in the CWM2_OLAP_CUBE package to create, drop, and lock cubes, to associate dimensions with cubes, and to specify descriptive information for display purposes.

You must create the cube before using the CWM2_OLAP_MEASURE package to create the cube's measures.

See Also:

- Chapter 15, "CWM2_OLAP_MEASURE"
- *Oracle OLAP Application Developer's Guide* for more information about cubes and the OLAP metadata model.

Example: Creating a Cube

The following statements drop the cube SALES_CUBE, re-create it, and add the dimensions TIME_DIM, GEOG_DIM, and PRODUCT_DIM.

Dropping the cube removes the cube entity, along with its measures, from the OLAP Catalog. However, dropping the cube does not cause the cube's dimensions to be dropped.

```
execute cwm2_olap_cube.drop_cube('JSMITH', 'SALES_CUBE');
execute cwm2_olap_cube.create_cube
     ('JSMITH', 'SALES_CUBE', 'Sales', 'Sales Cube',
      'Sales dimensioned over geography, product, and time' );
execute cwm2_olap_cube.add_dimension_to_cube
     ('JSMITH', 'SALES_CUBE', 'JSMITH', 'TIME_DIM');
execute cwm2_olap_cube.add_dimension_to_cube
     ('JSMITH', 'SALES_CUBE', 'JSMITH', 'GEOG_DIM');
execute cwm2_olap_cube.add_dimension_to_cube
     ('JSMITH', 'SALES_CUBE', 'JSMITH', 'PRODUCT_DIM');
```

Summary of CWM2_OLAP_CUBE Subprograms

Table 9–1 CWM2_OLAP_CUBE Subprograms

Subprogram	Description
ADD_DIMENSION_TO_CUBE Procedure on page 9-3	Adds a dimension to a cube.
CREATE_CUBE Procedure on page 9-4	Creates a cube.
DROP_CUBE Procedure on page 9-5	Drops a cube.
LOCK_CUBE Procedure on page 9-5	Locks a cube's metadata for update.
REMOVE_DIMENSION_FROM_CUBE Procedure on page 9-6	Removes a dimension from a cube.
SET_AGGREGATION_OPERATOR Procedure on page 9-6	Sets the aggregation operators for rolling up the cube's data.
SET_CUBE_NAME Procedure on page 9-8	Sets the name of a cube.
SET_DEFAULT_CUBE_DIM_CALC_HIER Procedure on page 9-9	Sets the default calculation hierarchy for a dimension of the cube.
SET_DESCRIPTION Procedure on page 9-9	Sets the description for a cube.
SET_DISPLAY_NAME Procedure on page 9-10	Sets the display name for a cube.
SET_MV_SUMMARY_CODE Procedure on page 9-10	Sets the format for materialized views associated with a cube.
SET_SHORT_DESCRIPTION Procedure on page 9-11	Sets the short description for a cube.

ADD_DIMENSION_TO_CUBE Procedure

This procedure adds a dimension to a cube.

Syntax

```
ADD_DIMENSION_TO_CUBE  (
          cube_owner          IN   VARCHAR2,
          cube_name           IN   VARCHAR2,
          dimension_owner     IN   VARCHAR2,
          dimension_name      IN   VARCHAR2);
```

Parameters

Table 9–2 ADD_DIMENSION_TO_CUBE Procedure Parameters

Parameter	Description
`cube_owner`	Owner of the cube.
`cube_name`	Name of the cube.
`dimension_owner`	Owner of the dimension to be added to the cube.
`dimension_name`	Name of the dimension to be added to the cube.

CREATE_CUBE Procedure

This procedure creates a new cube in the OLAP Catalog.

Descriptions and display properties must also be established as part of cube creation. Once the cube has been created, you can override these properties by calling other procedures in this package.

Syntax

```
CREATE_CUBE (
          cube_owner            IN   VARCHAR2,
          cube_name             IN   VARCHAR2,
          display_name          IN   VARCHAR2,
          short_description     IN   VARCHAR2,
          description           IN   VARCHAR2);
```

Parameters

Table 9–3 CREATE_CUBE Procedure Parameters

Parameter	Description
`cube_owner`	Owner of the cube.
`cube_name`	Name of the cube.
`display_name`	Display name for the cube.
`short_description`	Short description of the cube.
`description`	Description of the cube.

DROP_CUBE Procedure

This procedure drops a cube from the OLAP Catalog.

Note: When a cube is dropped, its associated measures are also dropped. However, the cube's dimensions are not dropped. They might be mapped within the context of a different cube.

Syntax

```
DROP_CUBE (
         cube_owner     IN   VARCHAR2,
         cube_name      IN   VARCHAR2);
```

Parameters

Table 9–4 DROP_CUBE Procedure Parameters

Parameter	Description
cube_owner	Owner of the cube.
cube_name	Name of the cube.

LOCK_CUBE Procedure

This procedure locks the cube's metadata for update by acquiring a database lock on the row that identifies the cube in the CWM2 model table.

Syntax

```
LOCK_CUBE (
           cube_owner         IN   VARCHAR2,
           cube_name          IN   VARCHAR2.
           wait_for_lock      IN   BOOLEAN DEFAULT FALSE);
```

Parameters

Table 9–5 LOCK_CUBE Procedure Parameters

Parameter	Description
cube_owner	Owner of the cube.
cube_name	Name of the cube.

Table 9–5 (Cont.) LOCK_CUBE Procedure Parameters

Parameter	Description
wait_for_lock	(Optional) Whether or not to wait for the cube to be available when it is already locked by another user. If you do not specify a value for this parameter, the procedure does not wait to acquire the lock.

REMOVE_DIMENSION_FROM_CUBE Procedure

This procedure removes a dimension from a cube.

Syntax

```
REMOVE_DIMENSION_FROM_CUBE  (
          cube_owner          IN   VARCHAR2,
          cube_name           IN   VARCHAR2,
          dimension_owner     IN   VARCHAR2,
          dimension_name      IN   VARCHAR2);
```

Parameters

Table 9–6 REMOVE_DIMENSION_FROM_CUBE Procedure Parameters

Parameter	Description
cube_owner	Owner of the cube.
cube_name	Name of the cube.
dimension_owner	Owner of the dimension to be removed from the cube.
dimension_name	Name of the dimension to be removed from the cube.

SET_AGGREGATION_OPERATOR Procedure

This procedure sets the aggregation operator for rolling up a cube's data over its dimensions. The cube must be mapped to a star schema, with a storage type indicator of 'LOWESTLEVEL'. (See "Joining Fact Tables with Dimension Tables" on page 2-12.)

The aggregation operators supported by the OLAP Catalog are listed in Table 1–10, " Aggregation Operators" on page 1-22.

When no aggregation operator is specified, the operator is addition. The view ALL_OLAP2_AGGREGATION_USES lists the non-default aggregation operators that have been specified for cubes. See "ALL_OLAP2_AGGREGATION_USES" on page 5-3.

Syntax

```
SET_AGGREGATION_OPERATOR (
          cube_owner        IN   VARCHAR2,
          cube_name         IN   VARCHAR2,
          aggop_spec        IN   VARCHAR2);
```

Parameters

Table 9–7 SET_AGGREGATION_OPERATOR Procedure Parameters

Parameter	Description
`cube_owner`	Owner of the cube.
`cube_name`	Name of the cube.
`aggop_spec`	A string that specifies the aggregation operators for the cube. Each aggregation operator that you specify applies to all of the cube's measures over a given hierarchy of a given dimension of the cube. If you do not specify a hierarchy, the operator applies to all hierarchies of the dimension. By default, the aggregation operator is addition. Enclose the string in single quotes, and separate each dimension/operator clause with a semicolon as follows: '`DIM:`*`dim1_owner.dim1_name`*`/AGGOP:`*`operator`*; `DIM:`*`dim2_owner.dim2_name`*`/AGGOP:`*`operator`*;...........' If the operator should apply to a specific hierarchy of a dimension, use the optional '`HIER`' clause after the `DIM` clause: `/HIER:`*`hiername1`* For weighted operators, the '`AGGOP`' clause may optionally be followed with a `WEIGHTBY` clause: `/WEIGHTBY:`*`TblOwner.TblName.ColName;`* *NOTE:* The cube's data will be aggregated in the order of the dimension clauses in the aggregation specification.

Example

The following example specifies that data in the `ANALYTIC_CUBE` should be aggregated using addition over the Standard hierarchies of the Product and Channel dimensions, using the `MAX` operator over the Standard hierarchy of Geography, and using `AVERAGE` over the Year to Date hierarchy of the Time dimension. Any unspecified hierarchies will use addition.

```
execute cwm2_olap_cube.set_aggregation_operator
```

```
            ('XADEMO', 'ANALYTIC_CUBE',
             'DIM:XADEMO.PRODUCT/HIER:STANDARD/AGGOP:SUM;
              DIM:XADEMO.GEOGRAPHY/HIER:STANDARD/AGGOP:MAX;
              DIM:XADEMO.TIME/HIER:YTD/AGGOP:AVERAGE;
              DIM:XADEMO.CHANNEL/HIER:STANDARD/AGGOP:SUM;');
```

The following example shows the same specification including a weighted operator for Product.

```
execute cwm2_olap_cube.set_aggregation_operator
            ('XADEMO', 'ANALYTIC_CUBE',
             'DIM:XADEMO.PRODUCT/HIER:STANDARD/AGGOP:SUM/
                                WEIGHTBY:XADEMO.XADEMO_SALES_VIEW.COSTS;
              DIM:XADEMO.GEOGRAPHY/HIER:STANDARD/AGGOP:MAX;
              DIM:XADEMO.TIME/HIER:YTD/AGGOP:AVERAGE;
              DIM:XADEMO.CHANNEL/HIER:STANDARD/AGGOP:SUM;');
```

In the following example, aggregation operators are specified for all hierarchicies of each dimension.

```
execute cwm2_olap_cube.set_aggregation_operator
            ('XADEMO', 'ANALYTIC_CUBE',
              DIM:XADEMO.PRODUCT/AGGOP:SUM;
              DIM:XADEMO.GEOGRAPHY/AGGOP:MAX;
              DIM:XADEMO.TIME/AGGOP:AVERAGE;
              DIM:XADEMO.CHANNEL/AGGOP:SUM;');
```

See Also

"Aggregating the Cube's Data in the Analytic Workspace" on page 1-5

SET_CUBE_NAME Procedure

This procedure sets the name for a cube.

Syntax

```
SET_CUBE_NAME (
          cube_owner        IN   VARCHAR2,
          cube_name         IN   VARCHAR2,
          set_cube_name     IN   VARCHAR2);
```

Parameters

Table 9–8 SET_CUBE_NAME Procedure Parameters

Parameter	Description
cube_owner	Owner of the cube.
cube_name	Original name of the cube.
set_cube_name	New name for the cube.

SET_DEFAULT_CUBE_DIM_CALC_HIER Procedure

This procedure sets the default calculation hierarchy for a dimension of this cube.

Syntax

```
SET_DEFAULT_CUBE_DIM_CALC_HIER (
          cube_owner       IN   VARCHAR2,
          cube_name        IN   VARCHAR2,
          dimension_owner  IN   VARCHAR2,
          dimension_name   IN   VARCHAR2,
          hierarchy_name   IN   VARCHAR2);
```

Parameters

Table 9–9 SET_DEFAULT_CUBE_DIM_CALC_HIER Procedure Parameters

Parameter	Description
cube_owner	Owner of the cube.
cube_owner	Name of the cube.
dimension_owner	Owner of the dimension.
dimension_name	Name of the dimension.
hierarchy_name	Name of the hierarchy to be used by default for this dimension.

SET_DESCRIPTION Procedure

This procedure sets the description for a cube.

Syntax

```
SET_DESCRIPTION (
          cube_owner     IN   VARCHAR2,
          cube_name      IN   VARCHAR2,
          description    IN   VARCHAR2);
```

Parameters

Table 9–10 SET_DESCRIPTION Procedure Parameters

Parameter	Description
cube_owner	Owner of the cube.
cube_name	Name of the cube.
description	Description of the cube.

SET_DISPLAY_NAME Procedure

This procedure sets the display name for a cube.

Syntax

```
SET_DISPLAY_NAME (
          cube_owner     IN   VARCHAR2,
          cube_name      IN   VARCHAR2,
          display_name   IN   VARCHAR2);
```

Parameters

Table 9–11 SET_DISPLAY_NAME Procedure Parameters

Parameter	Description
cube_owner	Owner of the cube.
cube_name	Name of the cube.
display_name	Display name for the cube.

SET_MV_SUMMARY_CODE Procedure

This procedure specifies the form of materialized views for this cube. Materialized views may be in Grouping Set (`groupingset`) or Rolled Up (`rollup`) form.

In a materialized view in Rolled Up form, all the dimension key columns are populated, and data may only be accessed when its full lineage is specified.

In a materialized view in Grouping Set form, dimension key columns may contain null values, and data may be accessed simply by specifying one or more levels.

Syntax

```
SET_MV_SUMMARY_CODE (
          cube_owner            IN   VARCHAR2,
          cube_name             IN   VARCHAR2,
          summary_code          IN   VARCHAR2);
```

Parameters

Table 9–12 SET_MV_SUMMARY_CODE Procedure Parameters

Parameter	Description
`cube_owner`	Owner of the cube.
`cube_name`	Name of the cube.
`summary_code`	One of the following case-insensitive values: ■ `rollup`, for Rolled Up form. ■ `groupingset`, for Grouping Set form.

SET_SHORT_DESCRIPTION Procedure

This procedure sets the short description for a cube.

Syntax

```
SET_SHORT_DESCRIPTION (
          cube_owner            IN   VARCHAR2,
          cube_name             IN   VARCHAR2,
          short_description     IN   VARCHAR2);
```

Parameters

Table 9–13 SET_SHORT_DESCRIPTION Procedure Parameters

Parameter	Description
`cube_owner`	Owner of the cube.

Table 9–13 (Cont.) SET_SHORT_DESCRIPTION Procedure Parameters

Parameter	Description
`cube_name`	Name of the cube.
`short_description`	Short description of the cube.

10

CWM2_OLAP_DIMENSION

The CWM2_OLAP_DIMENSION package provides procedures for managing dimensions.

> **See Also:** Chapter 2, "Creating OLAP Catalog Metadata with CWM2"

This chapter discusses the following topics:

- Understanding Dimensions
- Example: Creating a CWM2 Dimension
- Summary of CWM2_OLAP_DIMENSION Subprograms

Understanding Dimensions

A dimension is an OLAP metadata entity. This means that it is a logical object, identified by name and owner, within the OLAP Catalog. Logical OLAP dimensions are fully described in .

> **Note:** Dimensions in CWM2 map directly to columns in dimension tables and have no relationship to Oracle database dimension objects.

Use the procedures in the CWM2_OLAP_DIMENSION package to create, drop, and lock CWM2 dimension entities and to specify descriptive information for display purposes. To fully define a CWM2 dimension, follow the steps listed in "Creating a Dimension" on page 2-2.

> **See Also:** *Oracle OLAP Application Developer's Guide* for more information on dimensions and the OLAP metadata model.

Example: Creating a CWM2 Dimension

The following statement creates a `CWM2` dimension entity, `PRODUCT_DIM`, in the `JSMITH` schema. The display name is `Product`, and the plural name is `Products`. The short description is `Prod`, and the description is `Product`.

```
execute cwm2_olap_dimension.create_dimension
     ('JSMITH', 'PRODUCT_DIM', 'Product', 'Products', 'Prod', 'Product');
```

The following statements change the short description to `Product` and the long description to `Product Dimension`.

```
execute cwm2_olap_dimension.set_short_description
     ('JSMITH', 'PRODUCT_DIM', 'Product');
execute cwm2_olap_dimension.set_description
     ('JSMITH', 'PRODUCT_DIM', 'Product Dimension');
```

Summary of CWM2_OLAP_DIMENSION Subprograms

Table 10–1 CWM2_OLAP_DIMENSION Subprograms

Subprogram	Description
CREATE_DIMENSION Procedure on page 10-3	Creates a dimension.
DROP_DIMENSION Procedure on page 10-4	Drops a dimension.
LOCK_DIMENSION Procedure on page 10-5	Locks the dimension metadata for update.
SET_DEFAULT_DISPLAY_HIERARCHY Procedure on page 10-5	Sets the default hierarchy for a dimension.
SET_DESCRIPTION Procedure on page 10-6	Sets the description for a dimension.
SET_DIMENSION_NAME Procedure on page 10-6	Sets the name of a dimension.
SET_DISPLAY_NAME Procedure on page 10-7	Sets the display name for a dimension.
SET_PLURAL_NAME Procedure on page 10-7	Sets the plural name for a dimension.
SET_SHORT_DESCRIPTION Procedure on page 10-8	Sets the short description for a dimension.

CREATE_DIMENSION Procedure

This procedure creates a new dimension entity in the OLAP Catalog.

By default the new dimension is a normal dimension, but you can specify the value `TIME` for the `dimension_type` parameter to create a time dimension.

Descriptions and display properties must also be established as part of dimension creation. Once the dimension has been created, you can override these properties by calling other procedures in this package.

Syntax

```
CREATE_DIMENSION (
          dimension_owner       IN   VARCHAR2,
          dimension_name        IN   VARCHAR2,
          display_name          IN   VARCHAR2,
          plural_name           IN   VARCHAR2,
```

```
          short_description     IN   VARCHAR2,
          description           IN   VARCHAR2,
          dimension_type        IN   VARCHAR2 DEFAULT NULL);
```

Parameters

Table 10–2 CREATE_DIMENSION Procedure Parameters

Parameter	Description
`dimension_owner`	Owner of the dimension.
`dimension_name`	Name of the dimension.
`display_name`	Display name for the dimension.
`plural_name`	Plural name for the dimension.
`short_description`	Short description of the dimension.
`description`	Description of the dimension.
`dimension_type`	(Optional) Type of the dimension. Specify the value `TIME` to create a time dimension. If you do not specify this parameter, the dimension is created as a normal dimension.

DROP_DIMENSION Procedure

This procedure drops a dimension entity from the OLAP Catalog. All related levels, hierarchies, and dimension attributes are also dropped.

Syntax

```
DROP_DIMENSION (
        dimension_owner     IN   VARCHAR2,
        dimension_name      IN   VARCHAR2);
```

Parameters

Table 10–3 DROP_DIMENSION Procedure Parameters

Parameter	Description
`dimension_owner`	Owner of the dimension.
`dimension_name`	Name of the dimension.

LOCK_DIMENSION Procedure

This procedure locks the dimension metadata for update by acquiring a database lock on the row that identifies the dimension in the CWM2 model table.

Syntax

```
LOCK_DIMENSION (
          dimension_owner     IN   VARCHAR2,
          dimension_name      IN   VARCHAR2.
          wait_for_lock       IN   BOOLEAN DEFAULT FALSE);
```

Parameters

Table 10–4 LOCK_DIMENSION Procedure Parameters

Parameter	Description
dimension_owner	Owner of the dimension.
dimension_name	Name of the dimension.
wait_for_lock	(Optional) Whether or not to wait for the dimension to be available when it is already locked by another user. If you do not specify a value for this parameter, the procedure does not wait to acquire the lock.

SET_DEFAULT_DISPLAY_HIERARCHY Procedure

This procedure sets the default hierarchy to be used for display purposes.

Syntax

```
SET_DEFAULT_DISPLAY_HIERARCHY  (
          dimension_owner     IN   VARCHAR2,
          dimension_name      IN   VARCHAR2,
          hierarchy_name      IN   VARCHAR2);
```

Parameters

Table 10–5 SET_DEFAULT_DISPLAY_HIERARCHY Procedure Parameters

Parameter	Description
dimension_owner	Owner of the dimension.
dimension_name	Name of the dimension.

Table 10–5 (Cont.) SET_DEFAULT_DISPLAY_HIERARCHY Procedure Parameters

Parameter	Description
hierarchy_name	Name of one of the dimension's hierarchies.

SET_DESCRIPTION Procedure

This procedure sets the description for a dimension.

Syntax

```
SET_DESCRIPTION (
          dimension_owner     IN   VARCHAR2,
          dimension_name      IN   VARCHAR2,
          description         IN   VARCHAR2);
```

Parameters

Table 10–6 SET_DESCRIPTION Procedure Parameters

Parameter	Description
dimension_owner	Owner of the dimension.
dimension_name	Name of the dimension.
description	Description of the dimension.

SET_DIMENSION_NAME Procedure

This procedure sets the name for a dimension.

Syntax

```
SET_DIMENSION_NAME (
          dimension_owner        IN   VARCHAR2,
          dimension_name         IN   VARCHAR2,
          set_dimension_name     IN   VARCHAR2);
```

Parameters

Table 10–7 SET_DIMENSION_NAME Procedure Parameters

Parameter	Description
dimension_owner	Owner of the dimension.

Table 10–7 (Cont.) SET_DIMENSION_NAME Procedure Parameters

Parameter	Description
`dimension_name`	Original name of the dimension.
`set_dimension_name`	New name for the dimension.

SET_DISPLAY_NAME Procedure

This procedure sets the display name for a dimension.

Syntax

```
SET_DISPLAY_NAME (
          dimension_owner     IN   VARCHAR2,
          dimension_name      IN   VARCHAR2,
          display_name        IN   VARCHAR2);
```

Parameters

Table 10–8 SET_DISPLAY_NAME Procedure Parameters

Parameter	Description
`dimension_owner`	Owner of the dimension.
`dimension_name`	Name of the dimension.
`display_name`	Display name for the dimension.

SET_PLURAL_NAME Procedure

This procedure sets the plural name of a dimension.

Syntax

```
SET_PLURAL_NAME  (
          dimension_owner     IN   VARCHAR2,
          dimension_name      IN   VARCHAR2,
          plural_name         IN   VARCHAR2);
```

Parameters

Table 10–9 SET_PLURAL_NAME Procedure Parameters

Parameter	Description
`dimension_owner`	Owner of the dimension.
`dimension_name`	Name of the dimension.
`plural_name`	Plural name for the dimension.

SET_SHORT_DESCRIPTION Procedure

This procedure sets the short description for a dimension.

Syntax

```
SET_SHORT_DESCRIPTION (
          dimension_owner       IN   VARCHAR2,
          dimension_name        IN   VARCHAR2,
          short_description     IN   VARCHAR2);
```

Parameters

Table 10–10 SET_SHORT_DESCRIPTION Procedure Parameters

Parameter	Description
`dimension_owner`	Owner of the dimension.
`dimension_name`	Name of the dimension.
`short_description`	Short description of the dimension.

11

CWM2_OLAP_DIMENSION_ATTRIBUTE

The `CWM2_OLAP_DIMENSION_ATTRIBUTE` package provides procedures managing dimension attributes.

> **See Also:** Chapter 2, "Creating OLAP Catalog Metadata with CWM2".

This chapter discusses the following topics:

- Understanding Dimension Attributes
- Example: Creating a Dimension Attribute
- Summary of CWM2_OLAP_DIMENSION_ATTRIBUTE Subprograms

Understanding Dimension Attributes

A dimension attribute is an OLAP metadata entity. This means that it is a logical object, identified by name and owner, within the OLAP Catalog.

Dimension attributes define sets of level attributes for a dimension. Dimension attributes may include level attributes for some or all of the dimension's levels. For time dimensions, the dimension attributes `end date` and `time span` must be defined for all levels.

Use the procedures in the `CWM2_OLAP_DIMENSION_ATTRIBUTE` package to create, drop, and lock dimension attributes and to specify descriptive information for display purposes.

Several dimension attribute names are reserved, because they have special significance within `CWM2`. The level attributes comprising a reserved dimension attribute will be mapped to columns containing specific information. The reserved dimension attributes are listed in Table 11–1.

Table 11–1 Reserved Dimension Attributes

Dimension Attribute	Description
`Long Description`	A long description of the dimension member.
`Short Description`	A short description of the dimension member.
`End Date`	For a time dimension, the last date in a time period. (Required)
`Time Span`	For a time dimension, the number of days in a time period. (Required)
`Prior Period`	For a time dimension, the time period before this time period.
`Year Ago Period`	For a time dimension, the period a year before this time period.
`ET Key`	For an embedded total dimension, the embedded total key, which identifies the dimension member. (Required)
`Parent ET Key`	For an embedded total dimension, the dimension member that is the parent of the ET key. (Required)
`Grouping ID`	For an embedded total dimension, the grouping ID (GID), which identifies the hierarchical level for a row of the dimension table. (Required)
`Parent Grouping ID`	For an embedded total dimension, the dimension member that is the parent of the grouping ID. (Required)

The parent dimension must already exist before you can create dimension attributes for it. To fully define a dimension, follow the steps listed in "Creating a Dimension" on page 2-2.

See Also:

- Chapter 14, "CWM2_OLAP_LEVEL_ATTRIBUTE"
- *Oracle OLAP Application Developer's Guide* for more information about dimension attributes and the OLAP metadata model

Example: Creating a Dimension Attribute

The following statement creates a dimension attribute, `PRODUCT_DIM_BRAND`, for the `PRODUCT_DIM` dimension in the `JSMITH` schema. The display name is `Brand`. The short description is `Brand Name`, and the description is `Product Brand Name`.

```
execute cwm2_olap_dimension_attribute.create_dimension_attribute
     ('JSMITH', 'PRODUCT_DIM', 'PRODUCT_DIM_BRAND',
```

```
     'Brand', 'Brand Name', 'Product Brand Name');
```

The following statement creates a dimension attribute, `'Short Description'`, for the `PRODUCT_DIM` dimension in the `JSMITH` schema. `Short Description` is a reserved dimension attribute.

```
execute cwm2_olap_dimension_attribute.create_dimension_attribute
     ('JSMITH', 'PRODUCT_DIM', 'Short Description',
      'Short Product Names', 'Short Desc Product',
      'Short Name of Products', TRUE);
```

Summary of CWM2_OLAP_DIMENSION_ATTRIBUTE Subprograms

Table 11–2 CWM2_OLAP_DIMENSION_ATTRIBUTE Subprograms

Subprogram	Description
CREATE_DIMENSION_ATTRIBUTE Procedure on page 11-4	Creates a dimension attribute.
DROP_DIMENSION_ATTRIBUTE Procedure on page 11-5	Drops a dimension attribute.
LOCK_DIMENSION_ATTRIBUTE Procedure on page 11-6	Locks the dimension attribute for update.
SET_DESCRIPTION Procedure on page 11-7	Sets the description for a dimension attribute.
SET_DIMENSION_ATTRIBUTE_NAME Procedure on page 11-7	Sets the name of a dimension attribute.
SET_DISPLAY_NAME Procedure on page 11-8	Sets the display name for a dimension attribute.
SET_SHORT_DESCRIPTION Procedure on page 11-9	Sets the short description for a dimension attribute.

CREATE_DIMENSION_ATTRIBUTE Procedure

This procedure creates a new dimension attribute.

If the dimension attribute is reserved, you can specify the reserved name as the dimension attribute name or as a type associated with a name that you specify. The reserved dimension attributes are listed in Table 11–1, " Reserved Dimension Attributes".

If the dimension attribute name should be reserved for mapping specific groups of level attributes, you can set the `RESERVED_DIMENSION_ATTRIBUTE` argument to `TRUE`. For more information, see Table 11–1, " Reserved Dimension Attributes".

Descriptions and display properties must also be established as part of dimension attribute creation. Once the dimension attribute has been created, you can override these properties by calling other procedures in this package.

Syntax

```
CREATE_DIMENSION_ATTRIBUTE (
```

```
dimension_owner              IN   VARCHAR2,
dimension_name               IN   VARCHAR2,
dimension_attribute_name     IN   VARCHAR2,
display_name                 IN   VARCHAR2,
short_description            IN   VARCHAR2,
description                  IN   VARCHAR2,
  type                       IN   VARCHAR2            );
  use_name_as_type           IN   BOOLEAN DEFAULT FALSE);
```

Parameters

Table 11–3 CREATE_DIMENSION_ATTRIBUTE Procedure Parameters

Parameter	Description
dimension_owner	Owner of the dimension.
dimension_name	Name of the dimension.
dimension_attribute_name	Name of the dimension attribute.
display_name	Display name for the dimension attribute.
short_description	Short description of the dimension attribute.
description	Description of the dimension attribute.
type or use_name_as_type	This argument can be one of the following: ■ type a VARCHAR2 argument whose value is one of the reserved names from Table 11–1, " Reserved Dimension Attributes". Specify this argument if you want to create your own name for a reserved dimension attribute. ■ use_name_as_type a BOOLEAN argument that defaults to FALSE. This argument specifies whether or not the dimension attribute name is a reserved name. If this argument is TRUE, the value of the dimension_attribute_name argument must be a reserved name from Table 11–1, " Reserved Dimension Attributes". If you do not specify a value for this argument, the dimension attribute is not reserved.

DROP_DIMENSION_ATTRIBUTE Procedure

This procedure drops a dimension attribute.

Syntax

```
DROP_DIMENSION_ATTRIBUTE (
         dimension_owner               IN   VARCHAR2,
         dimension_name                IN   VARCHAR2,
         dimension_attribute_name      IN   VARCHAR2);
```

Parameters

Table 11–4 DROP_DIMENSION_ATTRIBUTE Procedure Parameters

Parameter	Description
dimension_owner	Owner of the dimension.
dimension_name	Name of the dimension.
dimension_attribute_name	Name of the dimension attribute.

LOCK_DIMENSION_ATTRIBUTE Procedure

This procedure locks the dimension attribute for update by acquiring a database lock on the row that identifies the dimension attribute in the CWM2 model table.

Syntax

```
LOCK_DIMENSION_ATTRIBUTE (
           dimension_owner              IN   VARCHAR2,
           dimension_name               IN   VARCHAR2,
           dimension_attribute_name     IN   VARCHAR2,
           wait_for_lock                IN   BOOLEAN DEFAULT FALSE);
```

Parameters

Table 11–5 LOCK_DIMENSION_ATTRIBUTE Procedure Parameters

Parameter	Description
dimension_owner	Owner of the dimension.
dimension_name	Name of the dimension.
dimension_attribute_name	Name of the dimension attribute.

Table 11–5 (Cont.) LOCK_DIMENSION_ATTRIBUTE Procedure Parameters

Parameter	Description
wait_for_lock	(Optional) Whether or not to wait for the dimension attribute to be available when it is already locked by another user. If you do not specify a value for this parameter, the procedure does not wait to acquire the lock.

SET_DESCRIPTION Procedure

This procedure sets the description for a dimension attribute.

Syntax

```
SET_DESCRIPTION (
          dimension_owner             IN   VARCHAR2,
          dimension_name              IN   VARCHAR2,
          dimension_attribute_name    IN   VARCHAR2,
          description                 IN   VARCHAR2);
```

Parameters

Table 11–6 SET_DESCRIPTION Procedure Parameters

Parameter	Description
dimension_owner	Owner of the dimension.
dimension_name	Name of the dimension.
dimension_attribute_name	Name of the dimension attribute.
description	Description of the dimension attribute.

SET_DIMENSION_ATTRIBUTE_NAME Procedure

This procedure sets the name for a dimension attribute.

If the dimension attribute is reserved, you can specify the reserved name as the dimension attribute name or as a type associated with a name that you specify. The reserved dimension attributes are listed in Table 11–1, " Reserved Dimension Attributes".

Syntax

```
SET_DIMENSION_ATTRIBUTE_NAME (
          dimension_owner                  IN   VARCHAR2,
          dimension_name                   IN   VARCHAR2,
          dimension_attribute_name         IN   VARCHAR2,
          set_dimension_attribute_name     IN   VARCHAR2,
            type                           IN   VARCHAR2             );
            use_name_as_type               IN   BOOLEAN DEFAULT FALSE);
```

Parameters

Table 11–7 SET_DIMENSION__ATTRIBUTE_NAME Procedure Parameters

Parameter	Description
`dimension_owner`	Owner of the dimension.
`dimension_name`	Name of the dimension.
`dimension_attribute_name`	Original name for the dimension attribute.
`set_dimension_attribute_name`	New name for the dimension attribute.
`type` or `use_name_as_type`	This argument can be one of the following: ■ `type` a VARCHAR2 argument whose value is one of the reserved names from Table 11–1, " Reserved Dimension Attributes". Specify this argument if you want to create your own name for a reserved dimension attribute. ■ `use_name_as_type` a BOOLEAN argument that defaults to FALSE. This argument specifies whether or not the dimension attribute name is a reserved name. If this argument is TRUE, the value of the `dimension_attribute_name` argument must be a reserved name from Table 11–1, " Reserved Dimension Attributes". If you do not specify a value for this argument, the dimension attribute is not reserved.

SET_DISPLAY_NAME Procedure

This procedure sets the display name for a dimension attribute.

Syntax

```
SET_DISPLAY_NAME (
          dimension_owner              IN   VARCHAR2,
          dimension_name               IN   VARCHAR2,
          dimension_attribute_name     IN   VARCHAR2,
          display_name                 IN   VARCHAR2);
```

Parameters

Table 11–8 SET_DISPLAY_NAME Procedure Parameters

Parameter	Description
dimension_owner	Owner of the dimension.
dimension_name	Name of the dimension.
dimension_attribute_name	Name of the dimension attribute.
display_name	Display name for the dimension attribute.

SET_SHORT_DESCRIPTION Procedure

This procedure sets the short description for a dimension attribute.

Syntax

```
SET_SHORT_DESCRIPTION (
          dimension_owner             IN   VARCHAR2,
          dimension_name              IN   VARCHAR2,
          dimension_attribute_name    IN   VARCHAR2,
          short_description           IN   VARCHAR2);
```

Parameters

Table 11–9 SET_SHORT_DESCRIPTION Procedure Parameters

Parameter	Description
dimension_owner	Owner of the dimension.
dimension_name	Name of the dimension.
dimension_attribute_name	Name of the dimension attribute.
short_description	Short description of the dimension attribute.

12

CWM2_OLAP_HIERARCHY

The CWM2_OLAP_HIERARCHY package provides procedures managing hierarchies.

See Also: Chapter 2, "Creating OLAP Catalog Metadata with CWM2".

This chapter discusses the following topics:

- Understanding Hierarchies
- Example: Creating a Hierarchy
- Summary of CWM2_OLAP_HIERARCHY Subprograms

Understanding Hierarchies

A hierarchy is an OLAP metadata entity. This means that it is a logical object, identified by name and owner, within the OLAP Catalog.

Hierarchies define parent-child relationships between sets of levels in a dimension. There can be multiple hierarchies associated with a single dimension, and the same level can be used in multiple hierarchies. Hierarchies are fully described in .

Use the procedures in the CWM2_OLAP_HIERARCHY package to create, drop, and lock hierarchies and to specify descriptive information for display purposes.

The parent dimension must already exist in the OLAP Catalog before you can create hierarchies for it.

See Also:

- Chapter 13, "CWM2_OLAP_LEVEL"
- *Oracle OLAP Application Developer's Guide* for more information about hierarchies and the OLAP metadata model

Example: Creating a Hierarchy

The following statement creates a dimension hierarchy `PRODUCT_DIM_ROLLUP`, for the `PRODUCT_DIM` dimension in the `JSMITH` schema. The display name is `Standard.` The short description is `Std Product`, and the description is `Standard Product Hierarchy`. The solved code is `SOLVED LEVEL-BASED`, meaning that this hierarchy will be mapped to an embedded total dimension table, and that the fact table associated with this dimension hierarchy will store fully solved data.

```
execute cwm2_olap_hierarchy.create_hierarchy
     ('JSMITH', 'PRODUCT_DIM', 'PRODUCT_DIM_ROLLUP',
      'Standard', 'Std Product', 'Standard Product Hierarchy',
      'SOLVED LEVEL-BASED');
```

Summary of CWM2_OLAP_HIERARCHY Subprograms

Table 12–1 CWM2_OLAP_HIERARCHY Subprograms

Subprogram	Description
CREATE_HIERARCHY Procedure on page 12-3	Creates a hierarchy.
DROP_HIERARCHY Procedure on page 12-4	Drops a hierarchy.
LOCK_HIERARCHY Procedure on page 12-5	Locks the hierarchy for update.
SET_DESCRIPTION Procedure on page 12-6	Sets the description for a hierarchy.
SET_DISPLAY_NAME Procedure on page 12-6	Sets the display name for a hierarchy.
SET_HIERARCHY_NAME Procedure on page 12-7	Sets the name of a hierarchy.
SET_SHORT_DESCRIPTION Procedure on page 12-7	Sets the short description for a hierarchy.
SET_SOLVED_CODE Procedure on page 12-8	Sets the solved code for a hierarchy.

CREATE_HIERARCHY Procedure

This procedure creates a new hierarchy in the OLAP Catalog.

You must specify descriptions and display properties as part of hierarchy creation. Once the hierarchy has been created, you can override these properties by calling other procedures in the `CWM2_OLAP_HIERARCHY` package.

Syntax

```
CREATE_HIERARCHY (
          dimension_owner       IN   VARCHAR2,
          dimension_name        IN   VARCHAR2,
          hierarchy_name        IN   VARCHAR2,
          display_name          IN   VARCHAR2,
          short_description     IN   VARCHAR2,
          description           IN   VARCHAR2,
          solved_code           IN   VARCHAR2);
```

Parameters

Table 12–2 CREATE_HIERARCHY Procedure Parameters

Parameter	Description
dimension_owner	Owner of the dimension.
dimension_name	Name of the dimension.
hierarchy_name	Name of the hierarchy.
display_name	Display name for the hierarchy.
short_description	Short description of the hierarchy.
description	Description of the hierarchy.
solved_code	Specifies whether or not the hierarchy includes embedded totals and whether it is mapped to a level-based dimension table or a parent-child dimension table. For information about mapping hierarchies with different solved codes, see "Joining Fact Tables with Dimension Tables" on page 2-12. Values for this parameter are: ■ UNSOLVED LEVEL-BASED, for a hierarchy that contains no embedded totals and is stored in a level-based dimension table ■ SOLVED LEVEL-BASED, for a hierarchy that contains embedded totals, has a grouping ID, and is stored in a level-based dimension table ■ SOLVED VALUE-BASED, for a hierarchy that contains embedded totals and is stored in a parent-child dimension table

DROP_HIERARCHY Procedure

This procedure drops a hierarchy from the OLAP Catalog.

Syntax

```
DROP_HIERARCHY (
        dimension_owner     IN   VARCHAR2,
        dimension_name      IN   VARCHAR2,
        hierarchy_name      IN   VARCHAR2);
```

Parameters

Table 12–3 DROP_HIERARCHY Procedure Parameters

Parameter	Description
dimension_owner	Owner of the dimension.
dimension_name	Name of the dimension.
hierarchy_name	Name of the hierarchy.

LOCK_HIERARCHY Procedure

This procedure locks the hierarchy metadata for update by acquiring a database lock on the row that identifies the hierarchy in the CWM2 model table.

Syntax

```
LOCK_HIERARCHY (
          dimension_owner     IN   VARCHAR2,
          dimension_name      IN   VARCHAR2,
          hierarchy_name      IN   VARCHAR2,
          wait_for_lock       IN   BOOLEAN DEFAULT FALSE);
```

Parameters

Table 12–4 LOCK_HIERARCHY Procedure Parameters

Parameter	Description
dimension_owner	Owner of the dimension.
dimension_name	Name of the dimension.
hierarchy_name	Name of the hierarchy.
wait_for_lock	(Optional) Whether or not to wait for the hierarchy to be available when it is already locked by another user. If you do not specify a value for this parameter, the procedure does not wait to acquire the lock.

SET_DESCRIPTION Procedure

This procedure sets the description for a hierarchy.

Syntax

```
SET_DESCRIPTION (
          dimension_owner    IN   VARCHAR2,
          dimension_name     IN   VARCHAR2,
          hierarchy_name     IN   VARCHAR2,
          description        IN   VARCHAR2);
```

Parameters

Table 12–5 SET_DESCRIPTION Procedure Parameters

Parameter	Description
dimension_owner	Owner of the dimension.
dimension_name	Name of the dimension.
hierarchy_name	Name of the hierarchy.
description	Description of the hierarchy.

SET_DISPLAY_NAME Procedure

This procedure sets the display name for a dimension.

Syntax

```
SET_DISPLAY_NAME (
          dimension_owner     IN   VARCHAR2,
          dimension_name      IN   VARCHAR2,
          hierarchy_name      IN   VARCHAR2,
          display_name        IN   VARCHAR2);
```

Parameters

Table 12–6 SET_DISPLAY_NAME Procedure Parameters

Parameter	Description
dimension_owner	Owner of the dimension.
dimension_name	Name of the dimension.

Table 12–6 (Cont.) SET_DISPLAY_NAME Procedure Parameters

Parameter	Description
`hierarchy_name`	Name of the hierarchy.
`display_name`	Display name for the hierarchy.

SET_HIERARCHY_NAME Procedure

This procedure sets the name for a hierarchy.

Syntax

```
SET_HIERARCHY_NAME (
          dimension_owner         IN   VARCHAR2,
          dimension_name          IN   VARCHAR2,
          hierarchy_name          IN   VARCHAR2,
          set_hierarchy_name      IN   VARCHAR2);
```

Parameters

Table 12–7 SET_HIERARCHY_NAME Procedure Parameters

Parameter	Description
`dimension_owner`	Owner of the dimension.
`dimension_name`	Name of the dimension.
`hierarchy_name`	Original name for the hierarchy.
`set_hierarchy_name`	New name for the hierarchy.

SET_SHORT_DESCRIPTION Procedure

This procedure sets the short description for a hierarchy.

Syntax

```
SET_SHORT_DESCRIPTION (
          dimension_owner         IN   VARCHAR2,
          dimension_name          IN   VARCHAR2,
          hierarchy_name          IN   VARCHAR2,
          short_description       IN   VARCHAR2);
```

Parameters

Table 12–8 SET_SHORT_DESCRIPTION Procedure Parameters

Parameter	Description
dimension_owner	Owner of the dimension.
dimension_name	Name of the dimension.
hierarchy_name	Name of the hierarchy.
short_description	Short description of the hierarchy.

SET_SOLVED_CODE Procedure

This procedure sets the solved code for a hierarchy. The solved code specifies whether or not the data dimensioned by this hierarchy includes embedded totals and whether it is mapped to a level-based dimension table or a parent-child dimension table. If mapped to a parent-child dimension table, it cannot be accessed by the OLAP API.

For more information on mapping solved and unsolved data, see "Joining Fact Tables with Dimension Tables" on page 2-12.

Syntax

```
SET_SOLVED_CODE (
          dimension_owner     IN   VARCHAR2,
          dimension_name      IN   VARCHAR2,
          hierarchy_name      IN   VARCHAR2,
          solved_code         IN   VARCHAR2);
```

Parameters

Table 12–9 SET_SOLVED_CODE Procedure Parameters

Parameter	Description
dimension_owner	Owner of the dimension.
dimension_name	Name of the dimension.
hierarchy_name	Name of the hierarchy.

Table 12–9 (Cont.) SET_SOLVED_CODE Procedure Parameters

Parameter	Description
`solved_code`	Specifies whether or not the hierarchy includes embedded totals and whether it is mapped to a level-based dimension table or a parent-child dimension table. For information about mapping hierarchies with different solved codes, see "Joining Fact Tables with Dimension Tables" on page 2-12. Values for this parameter are: ■ `UNSOLVED LEVEL-BASED`, for a hierarchy that contains no embedded totals and is stored in a level-based dimension table ■ `SOLVED LEVEL-BASED`, for a hierarchy that contains embedded totals, has a grouping ID, and is stored in a level-based dimension table ■ `SOLVED VALUE-BASED`, for a hierarchy that contains embedded totals and is stored in a parent-child dimension table

13

CWM2_OLAP_LEVEL

The CWM2_OLAP_LEVEL package provides procedures for managing levels.

> **See Also:** Chapter 2, "Creating OLAP Catalog Metadata with CWM2".

This chapter discusses the following topics:

- Understanding Levels
- Example: Creating a Level
- Summary of CWM2_OLAP_LEVEL Subprograms

Understanding Levels

A level is an OLAP metadata entity. This means that it is a logical object, identified by name and owner, within the OLAP Catalog.

Dimension members are organized in levels that map to columns in dimension tables or views. Levels are typically organized in hierarchies. Every dimension must have at least one level. Levels are fully described in

Use the procedures in the CWM2_OLAP_LEVEL package to create, drop, and lock levels, to assign levels to hierarchies, and to specify descriptive information for display purposes.

The parent dimension and the parent hierarchy must already exist in the OLAP Catalog before you can create a level.

See Also:

- Chapter 12, "CWM2_OLAP_HIERARCHY"
- *Oracle OLAP Application Developer's Guide* for more information about levels and the OLAP metadata model

Example: Creating a Level

The following statements create four levels for the PRODUCT_DIM dimension and assign them to the PRODUCT_DIM_ROLLUP hierarchy.

```
execute cwm2_olap_level.create_level
     ('JSMITH', 'PRODUCT_DIM', 'TOTALPROD_LVL',
      'Total Product', 'All Products', 'Total',
      'Equipment and Parts of standard product hierarchy');
execute cwm2_olap_level.create_level
     ('JSMITH', 'PRODUCT_DIM', 'PROD_CATEGORY_LVL',
      'Product Category', 'Product Categories', 'Category',
      'Categories of standard product hierarchy');
execute cwm2_olap_level.create_level
     ('JSMITH', 'PRODUCT_DIM', 'PROD_SUBCATEGORY_LVL',
      'Product Sub-Category', 'Product Sub-Categories', 'Sub-Category',
      'Sub-Categories of standard product hierarchy');
execute cwm2_olap_level.create_level
     ('JSMITH', 'PRODUCT_DIM', 'PRODUCT_LVL',
      'Product', 'Products', 'Product',
      'Individual products of standard product hierarchy');

execute cwm2_olap_level.add_level_to_hierarchy
     ('JSMITH', 'PRODUCT_DIM', 'PRODUCT_DIM_ROLLUP',
      'PRODUCT_LVL', 'PROD_SUBCATEGORY_LVL');
execute cwm2_olap_level.add_level_to_hierarchy
     ('JSMITH', 'PRODUCT_DIM', 'PRODUCT_DIM_ROLLUP',
      'PROD_SUBCATEGORY_LVL', 'PROD_CATEGORY_LVL');
execute cwm2_olap_level.add_level_to_hierarchy
     ('JSMITH', 'PRODUCT_DIM', 'PRODUCT_DIM_ROLLUP',
      'PROD_CATEGORY_LVL', 'TOTALPROD_LVL');
execute cwm2_olap_level.add_level_to_hierarchy
     ('JSMITH', 'PRODUCT_DIM', 'PRODUCT_DIM_ROLLUP', 'TOTALPROD_LVL');
```

Summary of CWM2_OLAP_LEVEL Subprograms

Table 13–1 CWM2_OLAP_LEVEL Subprograms

Subprogram	Description
ADD_LEVEL_TO_HIERARCHY Procedure on page 13-3	Adds a level to a hierarchy.
CREATE_LEVEL Procedure on page 13-4	Creates a level.
DROP_LEVEL Procedure on page 13-5	Drops a level.
LOCK_LEVEL Procedure on page 13-5	Locks the level metadata for update.
REMOVE_LEVEL_FROM_HIERARCHY Procedure on page 13-6	Removes a level from a hierarchy.
SET_DESCRIPTION Procedure on page 13-6	Sets the description for a level.
SET_DISPLAY_NAME Procedure on page 13-7	Sets the display name for a level.
SET_LEVEL_NAME Procedure on page 13-8	Sets the name of a level.
SET_PLURAL_NAME Procedure on page 13-8	Sets the plural name for a level.
SET_SHORT_DESCRIPTION Procedure on page 13-9	Sets the short description for a level.

ADD_LEVEL_TO_HIERARCHY Procedure

This procedure adds a level to a hierarchy.

Syntax

```
ADD_LEVEL_TO_HIERARCHY (
          dimension_owner     IN   VARCHAR2,
          dimension_name      IN   VARCHAR2,
          hierarchy_name      IN   VARCHAR2,
          level_name          IN   VARCHAR2,
          parent_level_name   IN   VARCHAR2  DEFAULT NULL);
```

Parameters

Table 13–2 ADD_LEVEL_TO_HIERARCHY Procedure Parameters

Parameter	Description
dimension_owner	Owner of the dimension.
dimension_name	Name of the dimension.
hierarchy_name	Name of the hierarchy.
level_name	Name of the level to add to the hierarchy.
parent_level_name	Name of the level's parent in the hierarchy. If you do not specify a parent, then the added level is the root of the hierarchy.

CREATE_LEVEL Procedure

This procedure creates a new level in the OLAP Catalog.

You must specify descriptions and display properties as part of level creation. Once the level has been created, you can override these properties by calling other procedures in the CWM2_OLAP_LEVEL package.

Syntax

```
CREATE_LEVEL (
          dimension_owner       IN   VARCHAR2,
          dimension_name        IN   VARCHAR2,
          level_name            IN   VARCHAR2,
          display_name          IN   VARCHAR2,
          plural_name           IN   VARCHAR2,
          short_description     IN   VARCHAR2,
          description           IN   VARCHAR2);
```

Parameters

Table 13–3 CREATE_LEVEL Procedure Parameters

Parameter	Description
dimension_owner	Owner of the dimension.
dimension_name	Name of the dimension.
level_name	Name of the level.

Table 13–3 (Cont.) CREATE_LEVEL Procedure Parameters

Parameter	Description
display_name	Display name for the level.
plural_name	Plural name for the level.
short_description	Short description of the level.
description	Description of the level.

DROP_LEVEL Procedure

This procedure drops a level from the OLAP Catalog. All related level attributes are also dropped.

Syntax

```
DROP_LEVEL (
         dimension_owner     IN   VARCHAR2,
         dimension_name      IN   VARCHAR2,
         level_name          IN   VARCHAR2);
```

Parameters

Table 13–4 DROP_LEVEL Procedure Parameters

Parameter	Description
dimension_owner	Owner of the dimension.
dimension_name	Name of the dimension.
level_name	Name of the level.

LOCK_LEVEL Procedure

This procedure locks the level metadata for update by acquiring a database lock on the row that identifies the level in the CWM2 model table.

Syntax

```
LOCK_LEVEL (
           dimension_owner     IN   VARCHAR2,
           dimension_name      IN   VARCHAR2,
           level_name          IN   VARCHAR2,
           wait_for_lock       IN   BOOLEAN DEFAULT FALSE);
```

Parameters

Table 13–5 LOCK_LEVEL Procedure Parameters

Parameter	Description
dimension_owner	Owner of the dimension.
dimension_name	Name of the dimension.
level_name	Name of the level.
wait_for_lock	(Optional) Whether or not to wait for the level to be available when it is already locked by another user. If you do not specify a value for this parameter, the procedure does not wait to acquire the lock.

REMOVE_LEVEL_FROM_HIERARCHY Procedure

This procedure removes a level from a hierarchy.

Syntax

```
REMOVE_LEVEL_FROM_HIERARCHY (
          dimension_owner     IN   VARCHAR2,
          dimension_name      IN   VARCHAR2,
          hierarchy_name      IN   VARCHAR2,
          level_name          IN   VARCHAR2);
```

Parameters

Table 13–6 REMOVE_LEVEL_FROM_HIERARCHY Procedure Parameters

Parameter	Description
dimension_owner	Owner of the dimension.
dimension_name	Name of the dimension.
hierarchy_name	Name of the hierarchy.
level_name	Name of the level to remove from the hierarchy.

SET_DESCRIPTION Procedure

This procedure sets the description for a level.

Syntax

```
SET_DESCRIPTION (
          dimension_owner     IN   VARCHAR2,
          dimension_name      IN   VARCHAR2,
          level_name          IN   VARCHAR2,
          description         IN   VARCHAR2);
```

Parameters

Table 13–7 SET_DESCRIPTION Procedure Parameters

Parameter	Description
`dimension_owner`	Owner of the dimension.
`dimension_name`	Name of the dimension.
`level_name`	Name of the level.
`description`	Description of the level.

SET_DISPLAY_NAME Procedure

This procedure sets the display name for a level.

Syntax

```
SET_DISPLAY_NAME (
          dimension_owner     IN   VARCHAR2,
          dimension_name      IN   VARCHAR2,
          level_name          IN   VARCHAR2,
          display_name        IN   VARCHAR2);
```

Parameters

Table 13–8 SET_DISPLAY_NAME Procedure Parameters

Parameter	Description
`dimension_owner`	Owner of the dimension.
`dimension_name`	Name of the dimension.
`level_name`	Name of the level.
`display_name`	Display name for the level.

SET_LEVEL_NAME Procedure

This procedure sets the name for a level.

Syntax

```
SET_LEVEL_NAME (
          dimension_owner   IN   VARCHAR2,
          dimension_name    IN   VARCHAR2,
          level_name        IN   VARCHAR2,
          set_level_name    IN   VARCHAR2);
```

Parameters

Table 13–9 SET_LEVEL_NAME Procedure Parameters

Parameter	Description
dimension_owner	Owner of the dimension.
dimension_name	Name of the dimension.
level_name	Original name for the level.
set_level_name	New name for the level.

SET_PLURAL_NAME Procedure

This procedure sets the plural name of a level.

Syntax

```
SET_PLURAL_NAME  (
          dimension_owner     IN   VARCHAR2,
          dimension_name      IN   VARCHAR2,
          level_name          IN   VARCHAR2,
          plural_name         IN   VARCHAR2);
```

Parameters

Table 13–10 SET_PLURAL_NAME Procedure Parameters

Parameter	Description
dimension_owner	Owner of the dimension.
dimension_name	Name of the dimension.

Table 13–10 (Cont.) SET_PLURAL_NAME Procedure Parameters

Parameter	Description
level_name	Name of the level.
plural_name	Plural name for the level.

SET_SHORT_DESCRIPTION Procedure

This procedure sets the short description for a level.

Syntax

```
SET_SHORT_DESCRIPTION (
          dimension_owner       IN   VARCHAR2,
          dimension_name        IN   VARCHAR2,
          level_name            IN   VARCHAR2,
          short_description     IN   VARCHAR2);
```

Parameters

Table 13–11 SET_SHORT_DESCRIPTION Procedure Parameters

Parameter	Description
dimension_owner	Owner of the dimension.
dimension_name	Name of the dimension.
level_name	Name of the level.
short_description	Short description of the level.

14

CWM2_OLAP_LEVEL_ATTRIBUTE

The CWM2_OLAP_LEVEL_ATTRIBUTE package provides procedures for managing level attributes.

> **See Also:** Chapter 2, "Creating OLAP Catalog Metadata with CWM2".

This chapter discusses the following topics:

- Understanding Level Attributes
- Example: Creating Level Attributes
- Summary of CWM2_OLAP_LEVEL_ATTRIBUTE Subprograms

Understanding Level Attributes

A level attribute is an OLAP metadata entity. This means that it is a logical object, identified by name and owner, within the OLAP Catalog.

A level attribute is a child entity of a level and a dimension attribute. A level attribute stores descriptive information about its related level. For example, a level containing product identifiers might have an associated level attribute that contains color information for each product.

Each level attribute maps to a column in a dimension table. The level attribute column must be in the same table as the column (or columns) for its associated level. Level attributes are fully described in .

Use the procedures in the CWM2_OLAP_LEVEL_ATTRIBUTE package to create, drop, and lock level attributes, to assign level attributes to levels and dimension attributes, and to specify descriptive information for display purposes.

Several level attribute names are reserved, because they have special significance within CWM2. Reserved level attributes are associated with reserved dimension attributes of the same name. Reserved level attributes will be mapped to columns containing specific information. The reserved level attributes are listed in Table 14–1.

Table 14–1 Reserved Level Attributes

Dimension Attribute	Description
`Long Description`	A long description of the dimension member.
`Short Description`	A short description of the dimension member.
`End Date`	For a time dimension, the last date in a time period. (Required)
`Time Span`	For a time dimension, the number of days in a time period. (Required)
`Prior Period`	For a time dimension, the time period before this time period.
`Year Ago Period`	For a time dimension, the period a year before this time period.
`ET Key`	For an embedded total dimension, the embedded total key, which identifies the dimension member at the lowest level in a row of the dimension table. (Required)
`Parent ET Key`	For an embedded total dimension, the dimension member that is the parent of the ET key. (Required)
`Grouping ID`	For an embedded total dimension, the grouping ID (GID), which identifies the hierarchical level for a row of the dimension table. (Required)
`Parent Grouping ID`	For an embedded total dimension, the dimension member that is the parent of the grouping ID. (Required)

The parent dimension, parent level, and parent dimension attribute must already exist in the OLAP Catalog before you can create a level attribute.

See Also:

- Chapter 11, "CWM2_OLAP_DIMENSION_ATTRIBUTE"
- *Oracle OLAP Application Developer's Guide* for more information about level attributes and the OLAP metadata model

Example: Creating Level Attributes

The following statements create a color attribute for the lowest level and long descriptions for all four levels of the PRODUCT_DIM dimension.

```
execute cwm2_olap_level_attribute.create_level_attribute
     ('JSMITH', 'PRODUCT_DIM', 'Product Color', 'PRODUCT_LVL', 'Product Color',
      'PROD_STD_COLOR', 'Prod Color', 'Product Color');

execute cwm2_olap_level_attribute.create_level_attribute
     ('JSMITH', 'PRODUCT_DIM', 'Long Description', 'PRODUCT_LVL',
      'Long Description','PRODUCT_STD_LLABEL', 'Product',
      'Long Labels for individual products of the PRODUCT hierarchy', TRUE);

execute cwm2_olap_level_attribute.create_level_attribute
     ('JSMITH', 'PRODUCT_DIM', 'Long Description', 'PROD_SUBCATEGORY_LVL',
      'Long Description', 'PROD_STD_LLABEL', 'Product Sub Category',
      'Long Labels for subcategories of the PRODUCT hierarchy', TRUE);

execute cwm2_olap_level_attribute.create_level_attribute
     ('JSMITH', 'PRODUCT_DIM', 'Long Description', 'PROD_CATEGORY_LVL',
      'Long Description', 'PROD_STD_LLABEL', 'Product Category',
      'Long Labels for categories of the PRODUCT hierarchy', TRUE);

execute cwm2_olap_level_attribute.create_level_attribute
     ('JSMITH', 'PRODUCT_DIM', 'Long Description', 'TOTALPROD_LVL',
      'Long Description', 'PROD_STD_LLABEL', 'Total Product',
      'Long Labels for total of the PRODUCT hierarchy', TRUE);
```

Summary of CWM2_OLAP_LEVEL_ATTRIBUTE Subprograms

Table 14–2 CWM2_OLAP_LEVEL_ATTRIBUTE Subprograms

Subprogram	Description
CREATE_LEVEL_ATTRIBUTE Procedure on page 14-4	Creates a level attribute.
DROP_LEVEL_ATTRIBUTE Procedure on page 14-6	Drops a level attribute.
LOCK_LEVEL_ATTRIBUTE Procedure on page 14-7	Locks the level attribute metadata for update.
SET_DESCRIPTION Procedure on page 14-7	Sets the description for a level attribute.
SET_DISPLAY_NAME Procedure on page 14-8	Sets the display name for a level attribute.
SET_LEVEL_ATTRIBUTE_NAME Procedure on page 14-9	Sets the name of a level attribute.
SET_SHORT_DESCRIPTION Procedure on page 14-10	Sets the short description for a level attribute.

CREATE_LEVEL_ATTRIBUTE Procedure

This procedure creates a new level attribute in the OLAP Catalog and associates the level attribute with a level and with a dimension attribute.

If the level attribute is reserved, you can specify the reserved name as the level attribute name or as a type associated with a name that you specify. The reserved level attributes are listed in Table 14–1, " Reserved Level Attributes".

You must specify descriptions and display properties as part of level attribute creation. Once the level attribute has been created, you can override these properties by calling other procedures in the `CWM2_OLAP_LEVEL_ATTRIBUTE` package.

Syntax

```
CREATE_LEVEL_ATTRIBUTE (
          dimension_owner            IN   VARCHAR2,
          dimension_name             IN   VARCHAR2,
          dimension_attribute_name   IN   VARCHAR2,
          level_name                 IN   VARCHAR2,
          level_attribute_name       IN   VARCHAR2,
          display_name               IN   VARCHAR2,
          short_description          IN   VARCHAR2,
          description                IN   VARCHAR2,
            type                     IN   VARCHAR2               );
            use_name_as_type         IN   BOOLEAN DEFAULT FALSE);
```

Parameters

Table 14–3 CREATE_LEVEL_ATTRIBUTE Procedure Parameters

Parameter	Description
`dimension_owner`	Owner of the dimension.
`dimension_name`	Name of the dimension.
`dimension_attribute_name`	Name of the dimension attribute that includes this level attribute.
`level_name`	Name of the level.
`level_attribute_name`	Name of the level attribute.
`display_name`	Display name for the level attribute.
`short_description`	Short description of the level attribute.
`description`	Description of the level attribute.

Table 14–3 (Cont.) CREATE_LEVEL_ATTRIBUTE Procedure Parameters

Parameter	Description

`type`
or
`use_name_as_type`
	This argument can be one of the following:
	■ `type`
	a VARCHAR2 argument whose value is one of the reserved names from Table 14–1, " Reserved Level Attributes". Specify this argument if you want to create your own name for a reserved level attribute.
	■ `use_name_as_type`
	a BOOLEAN argument that defaults to FALSE. This argument specifies whether or not the level attribute name is a reserved name. If this argument is TRUE, the value of the `level_attribute_name` argument must be a reserved name from Table 14–1, " Reserved Level Attributes".
	If you do not specify a value for this argument, the level attribute is not reserved.

DROP_LEVEL_ATTRIBUTE Procedure

This procedure drops a level attribute from the OLAP Catalog.

Syntax

```
DROP_LEVEL_ATTRIBUTE (
          dimension_owner            IN   VARCHAR2,
          dimension_name             IN   VARCHAR2,
          dimension_attribute_name   IN   VARCHAR2,
          level_name                 IN   VARCHAR2,
          level_attribute_name       IN   VARCHAR2);
```

Parameters

Table 14–4 DROP_LEVEL_ATTRIBUTE Procedure Parameters

Parameter	Description
`dimension_owner`	Owner of the dimension.
`dimension_name`	Name of the dimension.
`dimension_attribute_name`	Name of the dimension attribute.
`level_name`	Name of the level.

Table 14–4 (Cont.) DROP_LEVEL_ATTRIBUTE Procedure Parameters

Parameter	Description
`level_attribute_name`	Name of the level attribute.

LOCK_LEVEL_ATTRIBUTE Procedure

This procedure locks the level attribute metadata for update by acquiring a database lock on the row that identifies the level attribute in the `CWM2` model table.

Syntax

```
LOCK_LEVEL_ATTRIBUTE (
          dimension_owner            IN   VARCHAR2,
          dimension_name             IN   VARCHAR2,
          dimension_attribute_name   IN   VARCHAR2,
          level_name                 IN   VARCHAR2,
          level_attribute_name       IN   VARCHAR2,
          wait_for_lock              IN   BOOLEAN DEFAULT FALSE);
```

Parameters

Table 14–5 LOCK_LEVEL_ATTRIBUTE Procedure Parameters

Parameter	Description
`dimension_owner`	Owner of the dimension.
`dimension_name`	Name of the dimension.
`dimension_attribute_name`	Name of the dimension attribute.
`level_name`	Name of the level.
`level_attribute_name`	Name of the level attribute.
`wait_for_lock`	(Optional) Whether or not to wait for the level attribute to be available when it is already locked by another user. If you do not specify a value for this parameter, the procedure does not wait to acquire the lock.

SET_DESCRIPTION Procedure

This procedure sets the description for a level attribute.

Syntax

```
SET_DESCRIPTION (
          dimension_owner            IN   VARCHAR2,
          dimension_name             IN   VARCHAR2,
          dimension_attribute_name   IN   VARCHAR2,
          level_name                 IN   VARCHAR2,
          level_attribute_name       IN   VARCHAR2,
          description                IN   VARCHAR2);
```

Parameters

Table 14–6 SET_DESCRIPTION Procedure Parameters

Parameter	Description
dimension_owner	Owner of the dimension.
dimension_name	Name of the dimension.
dimension_attribute_name	Name of the dimension attribute.
level_name	Name of the level.
level_attribute_name	Name of the level attribute.
description	Description of the level attribute.

SET_DISPLAY_NAME Procedure

This procedure sets the display name for a level attribute.

Syntax

```
SET_DISPLAY_NAME (
          dimension_owner            IN   VARCHAR2,
          dimension_name             IN   VARCHAR2,
          dimension_attribute_name   IN   VARCHAR2,
          level_name                 IN   VARCHAR2,
          level_attribute_name       IN   VARCHAR2,
          display_name               IN   VARCHAR2);
```

Parameters

Table 14–7 SET_DISPLAY_NAME Procedure Parameters

Parameter	Description
dimension_owner	Owner of the dimension.
dimension_name	Name of the dimension.
dimension_attribute_name	Name of the dimension attribute.
level_name	Name of the level.
level_attribute_name	Name of the level attribute.
display_name	Display name for the level attribute.

SET_LEVEL_ATTRIBUTE_NAME Procedure

This procedure sets the name for a level attribute.

If the level attribute is reserved, you can specify the reserved name as the level attribute name or as a type associated with a name that you specify. The reserved level attributes are listed in Table 14–1, " Reserved Level Attributes".

Syntax

```
SET_LEVEL_ATTRIBUTE_NAME (
          dimension_owner             IN   VARCHAR2,
          dimension_name              IN   VARCHAR2,
          dimension_attribute_name    IN   VARCHAR2,
          level_name                  IN   VARCHAR2,
          level_attribute_name        IN   VARCHAR2,
          set_level_attribute_name    IN   VARCHAR2,
            type                      IN   VARCHAR2             );
            use_name_as_type          IN   BOOLEAN DEFAULT FALSE);
```

Parameters

Table 14–8 SET_LEVEL_ATTRIBUTE_NAME Procedure Parameters

Parameter	Description
dimension_owner	Owner of the dimension.

Table 14–8 (Cont.) SET_LEVEL_ATTRIBUTE_NAME Procedure Parameters

Parameter	Description
`dimension_name`	Name of the dimension.
`dimension_attribute_name`	Name of the dimension attribute.
`level_name`	Name for the level.
`level_attribute_name`	Original name for the level attribute.
`set_level_attribute_name`	New name for the level attribute.
`type` or `use_name_as_type`	This argument can be one of the following: ■ `type` a VARCHAR2 argument whose value is one of the reserved names from Table 14–1, " Reserved Level Attributes". Specify this argument if you want to create your own name for a reserved level attribute. ■ `use_name_as_type` a BOOLEAN argument that defaults to FALSE. This argument specifies whether or not the level attribute name is a reserved name. If this argument is TRUE, the value of the `level_attribute_name` argument must be a reserved name from Table 14–1, " Reserved Level Attributes". If you do not specify a value for this argument, the level attribute is not reserved.

SET_SHORT_DESCRIPTION Procedure

This procedure sets the short description for a level attribute.

Syntax

```
SET_SHORT_DESCRIPTION (
          dimension_owner             IN    VARCHAR2,
          dimension_name              IN    VARCHAR2,
          dimension_attribute_name    IN    VARCHAR2,
          level_name                  IN    VARCHAR2,
          level_attribute_name        IN    VARCHAR2,
          short_description           IN    VARCHAR2);
```

Parameters

Table 14–9 SET_SHORT_DESCRIPTION Procedure Parameters

Parameter	Description
`dimension_owner`	Owner of the dimension.
`dimension_name`	Name of the dimension.
`dimension_attribute_name`	Name of the dimension attribute.
`level_name`	Name of the level.
`level_attribute_name`	Name of the level attribute.
`short_description`	Short description of the level attribute.

15

CWM2_OLAP_MEASURE

The CWM2_OLAP_MEASURE package provides procedures for managing measures.

> **See Also:** Chapter 2, "Creating OLAP Catalog Metadata with CWM2".

This chapter discusses the following topics:

- Understanding Measures
- Example: Creating a Measure
- Summary of CWM2_OLAP_MEASURE Subprograms

Understanding Measures

A measure is an OLAP metadata entity. This means that it is a logical object, identified by name and owner, within the OLAP Catalog.

Measures represent data stored in fact tables. The fact tables may be relational tables or views. The views may reference data stored in analytic workspaces.

Measures exist within the context of cubes, which fully specify the dimensionality of the measures' data. Measures are fully described in .

Use the procedures in the CWM2_OLAP_MEASURE package to create, drop, and lock measures, to associate a measure with a cube, and to specify descriptive information for display purposes.

The parent cube must already exist in the OLAP Catalog before you can create a measure.

See Also:

- Chapter 9, "CWM2_OLAP_CUBE"
- *Oracle OLAP Application Developer's Guide* for more information about measures and the OLAP metadata model

Example: Creating a Measure

The following statements create the SALES_AMOUNT and SALES_QUANTITY measures for the SALES_CUBE cube.

```
execute cwm2_olap_measure.create_measure
     ('JSMITH', 'SALES_CUBE', 'SALES_AMOUNT', 'Sales Amount',
      '$ Sales', 'Dollar Sales');
execute cwm2_olap_measure.create_measure
     ('JSMITH', 'SALES_CUBE', 'SALES_QUANTITY', 'Sales Quantity',
      'Sales Quantity', 'Quantity of Items Sold');
```

Summary of CWM2_OLAP_MEASURE Subprograms

Table 15–1 CWM2_OLAP_MEASURE Subprograms

Subprogram	Description
CREATE_MEASURE Procedure on page 15-3	Creates a measure.
DROP_MEASURE Procedure on page 15-4	Drops a measure.
LOCK_MEASURE Procedure on page 15-4	Locks a measure's metadata for update.
SET_DESCRIPTION Procedure on page 15-5	Sets the description for a measure.
SET_DISPLAY_NAME Procedure on page 15-6	Sets the display name for a measure.
SET_MEASURE_NAME Procedure on page 15-6	Sets the name of a measure.
SET_SHORT_DESCRIPTION Procedure on page 15-7	Sets the short description for a measure.

CREATE_MEASURE Procedure

This procedure creates a new measure in the OLAP Catalog.

A measure can only be created in the context of a cube. The cube must already exist before you create the measure.

Descriptions and display properties must also be established as part of measure creation. Once the measure has been created, you can override these properties by calling other procedures in this package.

Syntax

```
CREATE_MEASURE (
          cube_owner            IN   VARCHAR2,
          cube_name             IN   VARCHAR2,
          measure_name          IN   VARCHAR2,
          display_name          IN   VARCHAR2,
          short_description     IN   VARCHAR2,
          description           IN   VARCHAR2);
```

Parameters

Table 15–2 CREATE_MEASURE Procedure Parameters

Parameter	Description
`cube_owner`	Owner of the cube.
`cube_name`	Name of the cube.
`measure_name`	Name of the measure.
`display_name`	Display name for the measure.
`short_description`	Short description of the measure.
`description`	Description of the measure.

DROP_MEASURE Procedure

This procedure drops a measure from a cube.

Syntax

```
DROP_MEASURE (
         cube_owner      IN   VARCHAR2,
         cube_name        IN   VARCHAR2,
         measure_name    IN   VARCHAR2);
```

Parameters

Table 15–3 DROP_MEASURE Procedure Parameters

Parameter	Description
`cube_owner`	Owner of the cube.
`cube_name`	Name of the cube.
`measure_name`	Name of the measure to be dropped from the cube.

LOCK_MEASURE Procedure

This procedure locks the measure's metadata for update by acquiring a database lock on the row that identifies the measure in the `CWM2` model table.

Syntax

```
LOCK_MEASURE (
          cube_owner        IN   VARCHAR2,
          cube_name         IN   VARCHAR2.
          measure_name       IN   VARCHAR2,
          wait_for_lock     IN   BOOLEAN DEFAULT FALSE);
```

Parameters

Table 15–4 LOCK_MEASURE Procedure Parameters

Parameter	Description
cube_owner	Owner of the cube.
cube_name	Name of the cube.
measure_name	Name of the measure to be locked.
wait_for_lock	(Optional) Whether or not to wait for the measure to be available when it is already locked by another user. If you do not specify a value for this parameter, the procedure does not wait to acquire the lock.

SET_DESCRIPTION Procedure

This procedure sets the description for a measure.

Syntax

```
SET_DESCRIPTION (
          cube_owner     IN   VARCHAR2,
          cube_name      IN   VARCHAR2,
          measure_name   IN   VARCHAR2,
          description    IN   VARCHAR2);
```

Parameters

Table 15–5 SET_DESCRIPTION Procedure Parameters

Parameter	Description
cube_owner	Owner of the cube.
cube_name	Name of the cube.
measure_name	Name of the measure.

Table 15–5 (Cont.) SET_DESCRIPTION Procedure Parameters

Parameter	Description
`description`	Description of the measure.

SET_DISPLAY_NAME Procedure

This procedure sets the display name for a measure.

Syntax

```
SET_DISPLAY_NAME (
          cube_owner     IN   VARCHAR2,
          cube_name      IN   VARCHAR2,
          measure_name   IN   VARCHAR2,
          display_name   IN   VARCHAR2);
```

Parameters

Table 15–6 SET_DISPLAY_NAME Procedure Parameters

Parameter	Description
`cube_owner`	Owner of the cube.
`cube_name`	Name of the cube.
`measure_name`	Name of the measure.
`display_name`	Display name for the measure.

SET_MEASURE_NAME Procedure

This procedure sets the name for a measure.

Syntax

```
SET_MEASURE_NAME (
          cube_owner        IN   VARCHAR2,
          cube_name         IN   VARCHAR2,
          measure_name      IN   VARCHAR2,
          set_cube_name     IN   VARCHAR2);
```

Parameters

Table 15–7 SET_MEASURE_NAME Procedure Parameters

Parameter	Description
cube_owner	Owner of the cube.
cube_name	Name of the cube.
measure_name	Original name of the measure.
set_cube_name	New name for the measure.

SET_SHORT_DESCRIPTION Procedure

This procedure sets the short description for a measure.

Syntax

```
SET_SHORT_DESCRIPTION (
          cube_owner            IN   VARCHAR2,
          cube_name             IN   VARCHAR2,
          measure_name          IN   VARCHAR2,
          short_description     IN   VARCHAR2);
```

Parameters

Table 15–8 SET_SHORT_DESCRIPTION Procedure Parameters

Parameter	Description
cube_owner	Owner of the cube.
cube_name	Name of the cube.
measure_name	Name of the measure.
short_description	Short description of the measure.

16

CWM2_OLAP_METADATA_REFRESH

The CWM2_OLAP_METADATA_REFRESH package provides procedures that refresh the cached metadata tables used by the OLAP API.

See Also:

- "Validating and Committing OLAP Metadata" on page 2-13.
- Chapter 5, "OLAP Catalog Metadata Views"
- Chapter 3, "Active Catalog Views"

This chapter discusses the following topics:

- Views of Cached OLAP Catalog Metadata
- Views of Cached Active Catalog Metadata
- Summary of CWM2_OLAP_METADATA_REFRESH Subprograms

Views of Cached OLAP Catalog Metadata

The Metadata Reader views, named with the prefix MRV_OLAP2, present a read API to a set of cache tables for OLAP Catalog metadata. These views and tables are structured to facilitate queries by the OLAP API Metadata Reader.

The cache tables, unlike the OLAP Catalog model tables, are not automatically refreshed when changes are made to the metadata. You must call the MR_REFRESH procedure to refresh the cache tables.

Views of the OLAP Catalog model tables, described in Chapter 5, "OLAP Catalog Metadata Views", have the prefix ALL_OLAP2. Most of the MRV_OLAP2 views have the same name and column structure as the corresponding ALL_OLAP2 views.

Note: If the tables that underlie the MRV_OLAP2 views are not consistent with the OLAP Catalog metadata tables, the OLAP API may not be able to access the metadata.

Views of Cached Active Catalog Metadata

The MRV_OLAP2_AW views , present a read API to a set of cache tables for the Active Catalog. These views and tables are structured to facilitate query performance.

The cache tables, unlike the Active Catalog, are not automatically refreshed when changes are made to analytic workspaces. You must call the MR_AC_REFRESH procedure to refresh the cache tables.

The Active Catalog views, described in Chapter 3, "Active Catalog Views", have the prefix ALL_OLAP2_AW. The MRV_OLAP2_AW views have the same name and column structure as the corresponding ALL_OLAP2_AW views.

Note: If the tables that underlie the MRV_OLAP2_AW views are not consistent with the standard form metadata in analytic workspaces, you may not be able to obtain accurate information from them.

Summary of CWM2_OLAP_METADATA_REFRESH Subprograms

Table 16–1 CWM2_OLAP_METADATA_REFRESH Subprograms

Subprogram	Description
MR_REFRESH Procedure	Refreshes the cached metadata tables used by the OLAP API Metadata Reader.
MR_AC_REFRESH Procedure	Refreshes the cached Active Catalog metadata tables.

MR_REFRESH Procedure

This procedure refreshes the metadata tables that underlie the MRV_OLAP2 views. These tables must be updated to support queries by the OLAP API Metadata Reader.

Execute MR_REFRESH as the final statement in any script that creates, drops, or updates OLAP Catalog metadata for the OLAP API.

Execute MR_REFRESH after creating or modifying OLAP Catalog metadata in Enterprise Manager.

The MR_REFRESH procedure includes a COMMIT. The updates to the metadata tables are saved permanently in the database.

Syntax

```
MR_REFRESH;
```

MR_AC_REFRESH Procedure

This procedure refreshes the metadata tables that underlie the MRV_OLAP2_AW views. These tables must be updated to support queries against the Active Catalog cache tables.

Execute MR_AC_REFRESH as the final statement in any script that uses the DBMS_AWM package to create, modify, or enable analytic workspaces.

The MR_AC_REFRESH procedure includes a COMMIT.

Syntax

```
MR_AC_REFRESH;
```

17

CWM2_OLAP_PC_TRANSFORM

The CWM2_OLAP_PC_TRANSFORM package contains a procedure for generating a SQL script that creates a solved, level-based dimension table from a parent-child dimension table. .

After running the script and creating the new table, you can define OLAP metadata so that OLAP API applications can access the dimension.

See Also:

- *Oracle OLAP Application Developer's Guide* for information about types of data warehouse tables supported by OLAP Catalog metadata.
- Chapter 12, "CWM2_OLAP_HIERARCHY" for information about creating OLAP Catalog metadata for dimension hierarchies.

This chapter discusses the following topics:

- Prerequisites
- Parent-Child Dimensions
- Solved, Level-Based Dimensions
- Example: Creating a Solved, Level-Based Dimension Table
- Summary of CWM2_OLAP_PC_TRANSFORM Subprograms

Prerequisites

Before running the CWM2_OLAP_PC_TRANSFORM.CREATE_SCRIPT procedure, ensure that the RDBMS is enabled to write to a file. To specify a directory, you can use either a directory object to which your user ID has been granted the appropriate access, or a path set by the UTL_FILE_DIR initialization parameter for the instance.

A parent-child dimension table must exist and be accessible to the CWM2_OLAP_PC_TRANSFORM.CREATE_SCRIPT procedure.

Parent-Child Dimensions

A **parent-child dimension table** is one in which the hierarchical relationships are defined by a parent column and a child column. Since the hierarchy is defined by the relationship between the *values* within two columns, a parent-child dimension is sometimes referred to as having a **value-based hierarchy**.

Sample Parent-Child Dimension Table Columns

The following example illustrates the relationships between the values in the child and parent columns. A description column, which is an attribute of the child, is also included.

```
CHILD          PARENT       DESCRIPTION
-------------  -----------  -------------
World                       World
USA            World        United States of America
Northeast      USA          North East Region
Southeast      USA          South East Region
MA             Northeast    Massachusetts
Boston         MA           Boston, MA
Burlington     MA           Burlington, MA
NY             Northeast    New York State
New York City  NY           New York, NY
GA             Southeast    Georgia
Atlanta        GA           Atlanta,GA
Canada         World        Canada
```

If you choose to create OLAP Catalog metadata to represent a parent-child dimension, set the solved_code for the hierarchy to 'SOLVED VALUE-BASED', as described in Chapter 12, "CWM2_OLAP_HIERARCHY".

Note: You can create OLAP Catalog metadata to represent value-based hierarchies, but this type of hierarchy is not accessible to applications that use the OLAP API.

Solved, Level-Based Dimensions

The script generated by OLAP_PC_TRANSFORM.CREATE_SCRIPT creates a table that stores the values from the parent-child table in levels.

The resulting level-based dimension table includes the full lineage of every level value in every row. This type of dimension table is **solved**, because the fact table related to this dimension includes embedded totals for all level combinations.

If you want to enable parent-child dimension tables for access by the OLAP API, you must convert them to solved, level-based dimension tables. The OLAP API requires that dimensions have levels and that they include a GID (Grouping ID) column and an Embedded Total (ET) key column. GIDs and ET key columns are described in Example: Creating a Solved, Level-Based Dimension Table.

The following example illustrates how the parent-child relationships in would be represented as solved levels.

```
TOT_GEOG COUNTRY REGION    STATE CITY           DESCRIPTION
-------- ------- --------- ----- -------        ----------------------
World    USA     Northeast  MA   Boston         Boston, MA
World    USA     Northeast  MA   Burlington     Burlington, MA
World    USA     Northeast  NY   New York City  New York, NY
World    USA     Southeast  GA   Atlanta        Atlanta, GA
World    USA     Northeast  MA                  Massachusetts
World    USA     Northeast  NY                  New York State
World    USA     Southeast  GA                  Georgia
World    USA     Northeast                      North East Region
World    USA     Southeast                      South East Region
World    USA                                    United States of America
World    Canada                                 Canada
World                                           World
```

When creating OLAP Catalog metadata to represent a solved, level-based dimension hierarchy, specify a solved_code of 'SOLVED LEVEL-BASED', as described in Chapter 12, "CWM2_OLAP_HIERARCHY".

Example: Creating a Solved, Level-Based Dimension Table

Assuming a parent-child dimension table with the PARENT and CHILD columns shown in , you could use a command like the following to represent these columns in a solved, level-based dimension table.

```
execute cwm2_olap_pc_transform.create_script
     ('/dat1/scripts/myscripts' ,
```

```
'jsmith' ,
'input_tbl' ,
'PARENT' ,
'CHILD' ,
'output_tbl' ,
'jsmith_data');
```

This statement creates a script in the directory `/dat1/scripts/myscripts`. The script will convert the parent-child table `input_tbl` to the solved, level-based table `output_tbl`. Both tables are in the `jsmith_data` tablespace of the `jsmith` schema.

You can run the resulting script with the following command.

```
@create_output_tbl
```

You can view the resulting table with the following command.

```
select * from output_tbl_view
```

The resulting table would look like this.

```
GID SHORT_DESC    LONG_DESC     CHILD1 CHILD2 CHILD3    CHILD4 CHILD5
--- -----------   ------------  ------ -----  --------  ------ -------
0   Boston        Boston        World  USA    Northeast  MA    Boston
0   Burlington    Burlington    World  USA    Northeast  MA    Burlington
0   New York City New York City World  USA    Northeast  NY    New York City
0   Atlanta       Atlanta       World  USA    Southeast  GA    Atlanta
1   MA            MA            World  USA    Northeast  MA
1   NY            MA            World  USA    Northeast  NY
1   GA            GA            World  USA    Southeast  GA
3   Northeast     Northeast     World  USA    Northeast
3   Southeast     Southeast     World  USA    Southeast
7   USA           USA           World  USA
7   Canada        Canada        World  Canada
15  World         World         World
```

Grouping ID Column

The script automatically creates a GID column, as required by the OLAP API. The GID identifies the hierarchy level associated with each row by assigning a zero to each non-null value and a one to each null value in the level columns. The resulting binary number is the value of the GID. For example, a GID of 3 is assigned to the row with the level values World, USA, Northeast, since the three highest levels are assigned zeros and the two lowest levels are assigned ones.

```
CHILD1 CHILD2 CHILD3    CHILD4 CHILD5
------ -----  --------  ------ -------
World  USA    Northeast
0      0      0          1      1
```

Embedded Total Key Column

The script automatically generates columns for long description and short description. If you have columns in the input table that contain this information, you can specify them as parameters to the CREATE_SCRIPT procedure.

If you do not specify a column for the short description, the script creates the column and populates it with the lowest-level child value represented in each row. If you do not specify a column for the long description, the script simply replicates the short description.

The ET key column required by the OLAP API is the short description column that is created by default.

Summary of CWM2_OLAP_PC_TRANSFORM Subprograms

Table 17–1 CWM2_OLAP_PC_TRANSFORM

Subprogram	Description
CREATE_SCRIPT Procedure on page 17-6	Generates a script that converts a parent-child table to an embedded-total table.

CREATE_SCRIPT Procedure

This procedure generates a script that converts a parent-child dimension table to an embedded-total dimension table.

Syntax

```
CREATE_SCRIPT (
          directory          IN   VARCHAR2,
          schema             IN   VARCHAR2,
          pc_table           IN   VARCHAR2,
          pc_parent          IN   VARCHAR2,
          pc_child           IN   VARCHAR2,
          slb_table          IN   VARCHAR2,
          slb_tablespace     IN   VARCHAR2,
          pc_root            IN   VARCHAR2   DEFAULT  NULL,
          number_of_levels   IN   NUMBER     DEFAULT  NULL,
          level_names        IN   VARCHAR2   DEFAULT  NULL,
          short_description  IN   VARCHAR2   DEFAULT  NULL,
          long_description   IN   VARCHAR2   DEFAULT  NULL,
          attribute_names     IN   VARCHAR2   DEFAULT  NULL);
```

Parameters

Table 17–2 CREATE_SCRIPT Procedure Parameters

Parameter	Description
`directory`	The directory that will contain the generated script. This may be either a directory object or a directory path specified in the `UTL_FILE_DIR` initialization parameter.
`schema`	Schema containing the parent-child table. This schema will also contain the solved, level-based table.
`pc_table`	Name of the parent-child table.

Table 17–2 (Cont.) CREATE_SCRIPT Procedure Parameters

Parameter	Description
pc_parent	Name of the column in pc_table that contains the parent values .
pc_child	Name of the column in pc_table that contains the child values.
slb_table	Name of the solved, level-based table that will be created.
slb_tablespace	Name of the tablespace where the solved, level-based table will be created.
pc_root	One of the following: null - Root of the parent-child hierarchy is identified by null in the parent column. (default) *condition* - Root of the parent-child hierarchy is a condition, for example: 'long_des = "All Countries"'
number_of_levels	One of the following: null - The number of levels in the solved, level-based table will be all the levels of the hierarchy in the parent-child table. (default) *number* - The number of levels to be created in the solved, level-based table.
level_names	One of the following: null - The column names in the solved, level-based table will be the source child column name concatenated with the level number. (default) *list* - A comma-delimited list of column names for the solved, level-based table.
short_description	One of the following: null - There is no short description in the parent-child table. The highest level non-null child value in each row of the solved, level-based table will be used as the short description. This constitutes the ET key column (default) *column name* - Name of the column in the parent-child table that contains the short description. This column will be copied from the parent-child table to the solved, level-based table.

Table 17–2 (Cont.) CREATE_SCRIPT Procedure Parameters

Parameter	Description
`long_description`	One of the following: `null` - There is no long description in the parent-child table. The short description will be used. (default) *`column name`* - Name of the column in the parent-child table that contains the long description. This column will be copied from the parent-child table to the solved, level-based table.
`attribute_names`	One of the following: `null` - There are no attributes in the parent-child table. (default) *`list`* - A comma-delimited list of attribute columns in the parent-child table. These columns will be copied from the parent-child table to the solved, level-based table

Usage Notes

1. If a table with the same name as the solved, level-based table already exists, the script will delete it.
2. You can reduce the time required to generate the script by specifying the number of levels in the `number_of_levels` parameter. If you do not specify a value for this parameter, the `CREATE_SCRIPT` procedure calculates all the levels from the parent-child table.
3. To define additional characteristics of the solved, level-based table, you can modify the generated script file before executing it.

18

CWM2_OLAP_TABLE_MAP

The CWM2_OLAP_TABLE_MAP package provides procedures for mapping OLAP metadata entities to columns in your data warehouse dimension tables and fact tables.

See Also: Chapter 2, "Creating OLAP Catalog Metadata with CWM2"

This chapter discusses the following topics:

- Understanding OLAP Metadata Mapping
- Example: Mapping a Dimension
- Example: Mapping a Cube
- Summary of CWM2_OLAP_TABLE_MAP Subprograms

Understanding OLAP Metadata Mapping

The CWM2_OLAP_TABLE_MAP package provides procedures for linking OLAP metadata entities to columns in fact tables and dimension tables and for establishing the join relationships between a fact table and its associated dimension tables.

Dimension levels and level attributes are mapped to columns in dimension tables. Typically, they are mapped by hierarchy. Measures are mapped to columns in fact tables.

The join relationship between the fact table and dimension tables may be specified for solved or unsolved data stored in a single fact table, or for solved data stored in a single fact table for each hierarchy combination.

See Also: "Mapping OLAP Metadata" on page 2-11.

Example: Mapping a Dimension

The following statements map the four levels of the STANDARD hierarchy in the XADEMO.PRODUCT_AW dimension to columns in the XADEMO_AW_VIEW_PRODUCT dimension table. A long description attribute is mapped for each level.

```
execute cwm2_olap_table_map.Map_DimTbl_HierLevel
    ('XADEMO', 'PRODUCT_AW', 'STANDARD', 'L4',
     'XADEMO', 'XADEMO_AW_VIEW_PRODUCT', 'L4', 'L3');
execute cwm2_olap_table_map.Map_DimTbl_HierLevelAttr
    ('XADEMO', 'PRODUCT_AW', 'Long Description', 'STANDARD', 'L4',
     'Long Description', 'XADEMO', 'XADEMO_AW_VIEW_PRODUCT', 'PROD_STD_LLABEL');

execute cwm2_olap_table_map.Map_DimTbl_HierLevel
    ('XADEMO', 'PRODUCT_AW', 'STANDARD', 'L3',
     'XADEMO', 'XADEMO_AW_VIEW_PRODUCT', 'L3', 'L2');
execute cwm2_olap_table_map.Map_DimTbl_HierLevelAttr
    ('XADEMO', 'PRODUCT_AW', 'Long Description', 'STANDARD', 'L3',
     'Long Description', 'XADEMO', 'XADEMO_AW_VIEW_PRODUCT', 'PROD_STD_LLABEL');

execute cwm2_olap_table_map.Map_DimTbl_HierLevel
    ('XADEMO', 'PRODUCT_AW', 'STANDARD', 'L2',
     'XADEMO', 'XADEMO_AW_VIEW_PRODUCT', 'L2', 'L1');
execute cwm2_olap_table_map.Map_DimTbl_HierLevelAttr
    ('XADEMO', 'PRODUCT_AW', 'Long Description', 'STANDARD', 'L2',
     'Long Description', 'XADEMO', 'XADEMO_AW_VIEW_PRODUCT', 'PROD_STD_LLABEL');

execute cwm2_olap_table_map.Map_DimTbl_HierLevel
    ('XADEMO', 'PRODUCT_AW', 'STANDARD', 'L1',
     'XADEMO', 'XADEMO_AW_VIEW_PRODUCT', 'L1', null);

execute cwm2_olap_table_map.Map_DimTbl_HierLevelAttr
    ('XADEMO', 'PRODUCT_AW', 'Long Description', 'STANDARD', 'L1',
     'Long Description', 'XADEMO', 'XADEMO_AW_VIEW_PRODUCT', 'PROD_STD_LLABEL');
```

Example: Mapping a Cube

The following statement maps the dimension join keys for a cube named ANALYTIC_CUBE_AW in the XADEMO schema. Join key relationships are specified for four dimension/hierarchy combinations:

```
PRODUCT_AW/STANDARD
```

```
CHANNEL_AW/STANDARD
TIME_AW/YTD
GEOGRAPHY_AW/CONSOLIDATED.
```

The fact table is called `XADEMO_AW_SALES_VIEW_4`. It stores lowest level data and embedded totals for all level combinations.

```
execute cwm2_olap_table_map.Map_FactTbl_LevelKey
        ('XADEMO', 'ANALYTIC_CUBE_AW','XADEMO', 'XADEMO_AW_SALES_VIEW_4', 'ET',
         'DIM:XADEMO.PRODUCT_AW/HIER:STANDARD/GID:PRODUCT_GID/LVL:L4/COL:PRODUCT_ET;
          DIM:XADEMO.CHANNEL_AW/HIER:STANDARD/GID:CHANNEL_GID/LVL:STANDARD_1/COL:CHANNEL_ET;
          DIM:XADEMO.TIME_AW/HIER:YTD/GID:TIME_YTD_GID/LVL:L3/COL:TIME_YTD_ET;
          DIM:XADEMO.GEOGRAPHY_AW/HIER:CONSOLIDATED/GID:GEOG_CONS_GID/LVL:L4/COL:GEOG_CONS_ET;');
```

The following statement maps the `F.SALES_AW` measure to the `SALES` column in the fact table.

```
execute cwm2_olap_table_map.Map_FactTbl_Measure
        ('XADEMO', 'ANALYTIC_CUBE_AW', 'F.SALES_AW',
         'XADEMO', 'XADEMO_AW_SALES_VIEW_4', 'SALES',
         'DIM:XADEMO.PRODUCT_AW/HIER:STANDARD/LVL:L4/COL:PRODUCT_ET;
          DIM:XADEMO.CHANNEL_AW/HIER:STANDARD/LVL:STANDARD_1/COL:CHANNEL_ET;
          DIM:XADEMO.TIME_AW/HIER:YTD/LVL:L3/COL:TIME_YTD_ET;
          DIM:XADEMO.GEOGRAPHY_AW/HIER:CONSOLIDATED/LVL:L4/COL:GEOG_CONS_ET;');
```

Summary of CWM2_OLAP_TABLE_MAP Subprograms

Table 18–1 CWM2_OLAP_TABLE_MAP

Subprogram	Description
MAP_DIMTBL_HIERLEVELATTR Procedure on page 18-5	Maps a hierarchical level attribute to a column in a dimension table.
MAP_DIMTBL_HIERLEVEL Procedure on page 18-5	Maps a hierarchical level to one or more columns in a dimension table.
MAP_DIMTBL_HIERSORTKEY Procedure on page 18-6	Sorts the members of a hierarchy within a column of a dimension table.
MAP_DIMTBL_LEVELATTR Procedure on page 18-7	Maps a non-hierarchical level attribute to a column in a dimension table
MAP_DIMTBL_LEVEL Procedure on page 18-8	Maps a non-hierarchical level to one or more columns in a dimension table.
MAP_FACTTBL_LEVELKEY Procedure on page 18-9	Maps the dimensions of a cube to a fact table.
MAP_FACTTBL_MEASURE Procedure on page 18-11	Maps a measure to a column in a fact table.
REMOVEMAP_DIMTBL_HIERLEVELATTR Procedure on page 18-12	Removes the mapping of a hierarchical level attribute from a column in a dimension table.
REMOVEMAP_DIMTBL_HIERLEVEL Procedure on page 18-13	Removes the mapping of a hierarchical level from one or more columns in a dimension table.
REMOVEMAP_DIMTBL_HIERSORTKEY Procedure on page 18-14	Removes custom sorting criteria associated with columns in a dimension table.
REMOVEMAP_DIMTBL_LEVELATTR Procedure on page 18-14	Removes the mapping of a non-hierarchical level attribute from a column in a dimension table.
REMOVEMAP_DIMTBL_LEVEL Procedure on page 18-15	Removes the mapping of a non-hierarchical level from one or more columns in a dimension table.
REMOVEMAP_FACTTBL_LEVELKEY Procedure on page 18-16	Removes the mapping of a cube's dimensions from a fact table.
REMOVEMAP_FACTTBL_MEASURE Procedure on page 18-16	Removes the mapping of a measure from a column in a fact table.

MAP_DIMTBL_HIERLEVELATTR Procedure

This procedure maps a level attribute to a column in a dimension table.

The attribute being mapped is associated with a level in the context of a hierarchy.

Syntax

```
MAP_DIMTBL_HIERLEVELATTR (
          dimension_owner            IN   VARCHAR2,
          dimension_name             IN   VARCHAR2,
          dimension_attribute_name    IN   VARCHAR2,
          hierarchy_name             IN   VARCHAR2,
          level_name                 IN   VARCHAR2,
          level_attribute_name        IN   VARCHAR2,
          table_owner                IN   VARCHAR2,
          table_name                 IN   VARCHAR2,
          attrcol                     IN   VARCHAR2);
```

Parameters

Table 18–2 MAP_DIMTBL_HIERLEVELATTR Procedure Parameters

Parameter	Description
dimension_owner	Owner of the dimension.
dimension_name	Name of the dimension.
dimension_attribute_name	Name of the dimension attribute.
hierarchy_name	Name of the hierarchy.
level_name	Name of the level.
level_attribute_name	Name of the level attribute associated with this level.
table_owner	Owner of the dimension table.
table_name	Name of the dimension table.
attrcol	Column in the dimension table to which this level attribute should be mapped.

MAP_DIMTBL_HIERLEVEL Procedure

This procedure maps a level to one or more columns in a dimension table.

The level being mapped is identified within the context of a hierarchy.

Syntax

```
MAP_DIMTBL_HIERLEVEL (
        dimension_owner     IN   VARCHAR2,
        dimension_name      IN   VARCHAR2,
        hierarchy_name      IN   VARCHAR2,
        level_name          IN   VARCHAR2,
        table_owner          IN   VARCHAR2,
        table_name          IN   VARCHAR2,
        keycol               IN   VARCHAR2,
        parentcol           IN   VARCHAR2 DEFAULT NULL);
```

Parameters

Table 18–3 MAP_DIMTBL_HIERLEVEL Procedure Parameters

Parameter	Description
dimension_owner	Owner of the dimension.
dimension_name	Name of the dimension.
hierarchy_name	Name of the hierarchy.
level_name	Name of the level.
table_owner	Owner of the dimension table.
table_name	Name of the dimension table.
keycol	Column in the dimension table to which this level should be mapped. This column will be the key for this level column in the fact table. If the level is stored in more than one column, separate the column names with commas. These columns will be the multicolumn key for these level columns in the fact table.
parentcol	Column that stores the parent level in the hierarchy. If you do not specify this parameter, the level is the root of the hierarchy.

MAP_DIMTBL_HIERSORTKEY Procedure

This procedure specifies how to sort the members of a hierarchy within a column of a dimension table. The column may be the key column or it may be a related attribute column. Custom sorting can specify that the column be sorted in ascending or descending order, with nulls first or nulls last.

Custom sorting information is optional and can be applied at multiple levels of a dimension.

Syntax

```
MAP_DIMTBL_HIERSORTKEY (
          dimension_owner     IN   VARCHAR2,
          dimension_name      IN   VARCHAR2,
          hierarchy_name      IN   VARCHAR2,
          sortcol             IN   VARCHAR2);
```

Parameters

Table 18–4 MAP_DIMTBL_HIERSORTKEY Procedure Parameters

Parameter	Description
`dimension_owner`	Owner of the dimension.
`dimension_name`	Name of the dimension.
`hierarchy_name`	Name of the hierarchy.
`sortcol`	A string specifying how to sort the values stored in a given column of a dimension table. The string specifies the table name, the column name, whether to sort in ascending or descending order, and whether to place nulls first or last. The string should be enclosed in single quotes, and it should be in the following form. `'TBL:`*`tableowner.tablename`*`/COL:`*`columnname`* `/ORD:ASC|DSC/NULL:FIRST|LAST;'`

MAP_DIMTBL_LEVELATTR Procedure

This procedure maps a level attribute to a column in a dimension table.

The attribute being mapped is associated with a level that has no hierarchical context. Typically, this level is the only level defined for this dimension.

Syntax

```
MAP_DIMTBL_LEVELATTR (
          dimension_owner              IN   VARCHAR2,
          dimension_name               IN   VARCHAR2,
          dimension_attribute_name     IN   VARCHAR2,
          level_name                   IN   VARCHAR2,
          level_attribute_name          IN   VARCHAR2,
          table_owner                   IN   VARCHAR2,
          table_name                    IN   VARCHAR2,
          attrcol                       IN   VARCHAR2);
```

Parameters

Table 18–5 MAP_DIMTBL_LEVELATTR Procedure Parameters

Parameter	Description
dimension_owner	Owner of the dimension.
dimension_name	Name of the dimension.
dimension_attribute_name	Name of the dimension attribute.
level_name	Name of the level.
level_attribute_name	Name of the level attribute associated with this level.
table_owner	Owner of the dimension table.
table_name	Name of the dimension table.
attrcol	Column in the dimension table to which this level attribute should be mapped.

MAP_DIMTBL_LEVEL Procedure

This procedure maps a level to one or more columns in a dimension table.

The level being mapped has no hierarchical context. Typically, this level is the only level defined for this dimension.

Syntax

```
MAP_DIMTBL_LEVEL (
        dimension_owner     IN   VARCHAR2,
        dimension_name      IN   VARCHAR2,
        level_name          IN   VARCHAR2,
        table_owner          IN   VARCHAR2,
        table_name           IN   VARCHAR2,
        keycol               IN   VARCHAR2);
```

Parameters

Table 18–6 MAP_DIMTBL_LEVEL Procedure Parameters

Parameter	Description
`dimension_owner`	Owner of the dimension.
`dimension_name`	Name of the dimension.
`level_name`	Name of the level.
`table_owner`	Owner of the dimension table.
`table_name`	Name of the dimension table.
`keycol`	Column in the dimension table to which this level should be mapped. This column will be the key for this level column in the fact table. If the level is stored in more than one column, separate the column names with commas. These columns will be the multicolumn key for these level columns in the fact table.

MAP_FACTTBL_LEVELKEY Procedure

This procedure creates the join relationships between a fact table and a set of dimension tables. A join must be specified for each of the dimensions of the cube. Each dimension is joined in the context of one of its hierarchies.

For example, if you had a cube with three dimensions, and each dimension had only one hierarchy, you could fully map the cube with one call to `MAP_FACTTBL_LEVELKEY`.

However, if you had a cube with three dimensions, but two of the dimensions each had two hierarchies, you would need to call `MAP_FACTTBL_LEVELKEY` four times to fully map the cube. For dimensions `Dim1`, `Dim2`, and `Dim3`, where `Dim1` and `Dim3` each have two hierarchies, you would specify the following mapping strings in each call to `MAP_FACTTBL_LEVELKEY`, as follows.

```
Dim1_Hier1, Dim2_Hier, Dim3_Hier1
Dim1_Hier1, Dim2_Hier, Dim3_Hier2
Dim1_Hier2, Dim2_Hier, Dim3_Hier1
Dim1_Hier2, Dim2_Hier, Dim3_Hier2
```

Typically the data for each hierarchy combination would be stored in a separate fact table.

For more information, see"Joining Fact Tables with Dimension Tables" on page 2-12.

Syntax

```
MAP_FACTTBL_LEVELKEY (
         cube_owner          IN   VARCHAR2,
         cube_name           IN   VARCHAR2,
         facttable_owner     IN   VARCHAR2,
         facttable_name      IN   VARCHAR2,
         storetype           IN   VARCHAR2,
         dimkeymap           IN   VARCHAR2,
         dimktype            IN   VARCHAR2 DEFAULT NULL);
```

Parameters

Table 18–7 MAP_FACTTBL_LEVELKEY Procedure Parameters

Parameter	Description
`cube_owner`	Owner of the cube.
`cube_name`	Name of the cube.
`facttable_owner`	Owner of the fact table.
`facttable_name`	Name of the fact table.
`storetype`	One of the following: 'LOWESTLEVEL', for a fact table that stores only lowest level data 'ET', for a fact table that stores embedded totals for all level combinations in addition to lowest level data

Table 18–7 (Cont.) MAP_FACTTBL_LEVELKEY Procedure Parameters

Parameter	Description
`dimkeymap`	A string specifying the mapping for each dimension of the data in the fact table. For each dimension you must specify a hierarchy and the lowest level to be mapped within that hierarchy. Enclose the string in single quotes, and separate each dimension specification with a semicolon as follows: '`DIM:`*`dimname1`*`/HIER:`*`hiername1`* `/GID:`*`gid_columnname1`*`/LVL:`*`levelname1`* `/COL:`*`map_columnname1`*; `DIM:`*`dimname2`*`/HIER:`*`hiername2`* `/GID:`*`gid_columnname2`*`/LVL:`*`levelname2`* `/COL:`*`map_columnname2`*;...........' Note that the `GID` clause of the mapping string is only applicable to embedded totals. If you specify '`LOWESTLEVEL`' for the *storetype* argument, do not include a `GID` clause in the mapping string. This string must also be specified as an argument to the `MAP_FACTTBL_MEASURE` procedure.
`dimktype`	This parameter is not currently used.

MAP_FACTTBL_MEASURE Procedure

This procedure maps a measure to a column in a fact table.

Syntax

```
MAP_FACTTBL_MEASURE (
          cube_owner        IN   VARCHAR2,
          cube_name         IN   VARCHAR2,
          measure_name       IN   VARCHAR2,
          facttable_owner   IN   VARCHAR2,
          facttable_name    IN   VARCHAR2,
          column_name        IN   VARCHAR2,
          dimkeymap         IN   VARCHAR2);
```

Parameters

Table 18–8 MAP_FACTTBL_MEASURE Procedure Parameters

Parameter	Description
`cube_owner`	Owner of the cube.
`cube_name`	Name of the cube.
`measure_name`	Name of the measure to be mapped.
`facttable_owner`	Owner of the fact table.
`facttable_name`	Name of the fact table.
`column_name`	Column in the fact table to which the measure will be mapped.
`dimkeymap`	A string specifying the mapping for each of the measure's dimensions. For each dimension you must specify a hierarchy and the lowest level to be mapped within that hierarchy. Enclose the string in single quotes, and separate each dimension specification with a semicolon as follows: '`DIM:`*`dimname1`*`/HIER:`*`hiername1`* `/GID:`*`gid_columnname1`*`/LVL:`*`levelname1`* `/COL:`*`map_columnname1`*; `DIM:`*`dimname2`*`/HIER:`*`hiername2`* `/GID:`*`gid_columnname2`*`/LVL:`*`levelname2`* `/COL:`*`map_columnname2`*;...........' Note that the `GID` clause of the mapping string is only applicable to embedded totals. If you specify '`LOWESTLEVEL`' for the *storetype* argument, do not include a `GID` clause in the mapping string. This string must also be specified as an argument to the `MAP_FACTTBL_LEVELKEY` procedure.

REMOVEMAP_DIMTBL_HIERLEVELATTR Procedure

This procedure removes the relationship between a level attribute and a column in a dimension table. The attribute is identified by the hierarchy that contains its associated level.

Upon successful completion of this procedure, the level attribute is a purely logical metadata entity. It has no data associated with it.

Syntax

```
REMOVEMAP_DIMTBL_HIERLEVELATTR (
          dimension_owner             IN   VARCHAR2,
          dimension_name              IN   VARCHAR2,
          dimension_attribute_name     IN   VARCHAR2,
          hierarchy_name              IN   VARCHAR2,
          level_name                  IN   VARCHAR2,
          level_attribute_name         IN   VARCHAR2);
```

Parameters

Table 18–9 REMOVEMAP_DIMTBL_HIERLEVELATTR Procedure Parameters

Parameter	Description
dimension_owner	Owner of the dimension.
dimension_name	Name of the dimension.
dimension_attribute_name	Name of the dimension attribute.
hierarchy_name	Name of the hierarchy.
level_name	Name of the level.
level_attribute_name	Name of the level attribute associated with this level.

REMOVEMAP_DIMTBL_HIERLEVEL Procedure

This procedure removes the relationship between a level of a hierarchy and one or more columns in a dimension table.

Upon successful completion of this procedure, the level is a purely logical metadata entity. It has no data associated with it.

Syntax

```
REMOVEMAP_DIMTBL_HIERLEVEL (
         dimension_owner     IN   VARCHAR2,
         dimension_name      IN   VARCHAR2,
         hierarchy_name      IN   VARCHAR2,
         level_name          IN   VARCHAR2);
```

Parameters

Table 18–10 REMOVEMAP_DIMTBL_HIERLEVEL Procedure Parameters

Parameter	Description
`dimension_owner`	Owner of the dimension.
`dimension_name`	Name of the dimension.
`hierarchy_name`	Name of the hierarchy.
`level_name`	Name of the level.

REMOVEMAP_DIMTBL_HIERSORTKEY Procedure

This procedure removes custom sorting criteria associated with columns in a dimension table.

Syntax

```
REMOVEMAP_DIMTBL_HIERSORTKEY (
          dimension_owner     IN   VARCHAR2,
          dimension_name      IN   VARCHAR2,
          hierarchy_name      IN   VARCHAR2);
```

Parameters

Table 18–11 REMOVEMAP_DIMTBL_HIERSORTKEY Procedure Parameters

Parameter	Description
`dimension_owner`	Owner of the dimension.
`dimension_name`	Name of the dimension.
`hierarchy_name`	Name of the hierarchy.

REMOVEMAP_DIMTBL_LEVELATTR Procedure

This procedure removes the relationship between a level attribute and a column in a dimension table.

Upon successful completion of this procedure, the level attribute is a purely logical metadata entity. It has no data associated with it.

Syntax

```
REMOVEMAP_DIMTBL_LEVELATTR (
          dimension_owner             IN   VARCHAR2,
          dimension_name              IN   VARCHAR2,
          dimension_attribute_name    IN   VARCHAR2,
          level_name                  IN   VARCHAR2,
          level_attribute_name         IN   VARCHAR2);
```

Parameters

Table 18–12 REMOVEMAP_DIMTBL_LEVELATTR Procedure Parameters

Parameter	Description
`dimension_owner`	Owner of the dimension.
`dimension_name`	Name of the dimension.
`dimension_attribute_name`	Name of the dimension attribute.
`level_name`	Name of the level.
`level_attribute_name`	Name of the level attribute associated with this level.

REMOVEMAP_DIMTBL_LEVEL Procedure

This procedure removes the relationship between a level and one or more columns in a dimension table.

Upon successful completion of this procedure, the level is a purely logical metadata entity. It has no data associated with it.

Syntax

```
REMOVEMAP_DIMTBL_LEVEL (
         dimension_owner     IN   VARCHAR2,
         dimension_name      IN   VARCHAR2,
         level_name          IN   VARCHAR2);
```

Parameters

Table 18–13 REMOVEMAP_DIMTBL_LEVEL Procedure Parameters

Parameter	Description
`dimension_owner`	Owner of the dimension.

Table 18–13 (Cont.) REMOVEMAP_DIMTBL_LEVEL Procedure Parameters

Parameter	Description
`dimension_name`	Name of the dimension.
`level_name`	Name of the level.

REMOVEMAP_FACTTBL_LEVELKEY Procedure

This procedure removes the relationship between the key columns in a fact table and the level columns of a dimension hierarchy in a dimension table.

Syntax

```
REMOVEMAP_FACTTBL_LEVELKEY (
          cube_owner        IN   VARCHAR2,
          cube_name         IN   VARCHAR2,
          facttable_owner   IN   VARCHAR2,
          facttable_name    IN   VARCHAR2 DEFAULT );
```

Parameters

Table 18–14 REMOVEMAP_FACTTBL_LEVELKEY Procedure Parameters

Parameter	Description
`cube_owner`	Owner of the cube.
`cube_name`	Name of the cube.
`facttable_owner`	Owner of the fact table.
`facttable_name`	Name of the fact table.

REMOVEMAP_FACTTBL_MEASURE Procedure

This procedure removes the relationship between a measure column in a fact table and a logical measure associated with a cube.

Upon successful completion of this procedure, the measure is a purely logical metadata entity. It has no data associated with it.

Syntax

```
REMOVEMAP_FACTTBL_MEASURE (
          cube_owner        IN   VARCHAR2,
          cube_name         IN   VARCHAR2,
          measure_name       IN   VARCHAR2,
          facttable_owner    IN   VARCHAR2,
          facttable_name     IN   VARCHAR2,
          column_name        IN   VARCHAR2,
          dimkeymap          IN   VARCHAR2);
```

Parameters

Table 18–15 REMOVEMAP_FACTTBL_MEASURE Procedure Parameters

Parameter	Description
cube_owner	Owner of the cube.
cube_name	Name of the cube.
measure_name	Name of the measure.
facttable_owner	Owner of the fact table.
facttable_name	Name of the fact table.
column_name	Column in the fact table to which the measure is mapped.
dimkeymap	A string specifying the mapping for each of the measure's dimensions. For each dimension you must specify a hierarchy and the lowest level to be mapped within that hierarchy. Enclose the string in single quotes, and separate each dimension specification with a semicolon as follows: 'DIM:*dimname1*/HIER:*hiername1* /GID:*gid_columnname1*/LVL:*levelname1* /COL:*map_columnname1*; DIM:*dimname2*/HIER:*hiername2* /GID:*gid_columnname2*/LVL:*levelname2* /COL:*map_columnname2*;...........' Note that the GID clause of the mapping string is only applicable to embedded totals. If the measure contained only detail data and was mapped with a storage type of 'LOWESTLEVEL', do not include a GID clause in the mapping string. This string must also be specified as an argument to the MAP_ FACTTBL_MEASURE and MAP_FACTTBL_LEVELKEY procedures.

19

CWM2_OLAP_VALIDATE

The CWM2_OLAP_VALIDATE package provides procedures for validating OLAP metadata.

See Also:

- "Validating OLAP Metadata" on page 2-13
- Chapter 20, "CWM2_OLAP_VERIFY_ACCESS"

This chapter discusses the following topics:

- About OLAP Catalog Metadata Validation
- Summary of CWM2_OLAP_VALIDATE Subprograms

About OLAP Catalog Metadata Validation

The validation process checks the structural integrity of the metadata and ensures that it is correctly mapped to columns in dimension tables and fact tables. Additional validation specific to the OLAP API is done if requested.

The procedures in CWM2_OLAP_VALIDATE validate the OLAP metadata created by Enterprise Manager as well as the metadata created by CWM2 procedures.

See Also: "Validating and Committing OLAP Metadata" on page 2-13 for additional information.

Structural Validation

Structural validation ensures that cubes and dimensions have all their required components parts. All the procedures in CWM2_OLAP_VALIDATE perform structural validation by default.

Cubes

To be structurally valid, a cube must meet the following criteria:

- It must have at least one valid dimension.
- It must have at least one measure.

Dimensions

To be structurally valid, a dimension must meet the following criteria:

- It must have at least one level.
- It may have one or more hierarchies. Each hierarchy must have at least one level.
- It may have one or more dimension attributes. Each dimension attribute must have at least one level attribute.

Mapping Validation

Mapping validation ensures that the metadata has been properly mapped to columns in tables or views. All the procedures in `CWM2_OLAP_VALIDATE` perform mapping validation by default.

Cubes

To be valid, a cube's mapping must meet the following criteria:

- It must be mapped to one or more fact tables.
- All of the cube's measures must be mapped to existing columns in a fact table. If there are multiple fact tables, all the measures must be in each one.
- Every dimension/hierarchy combination must be mapped to one of the fact tables.

Dimensions

To be valid, a dimension's mapping must meet the following criteria:

- All levels must be mapped to existing columns in a dimension table.
- Level attributes must be mapped to columns in the same table as the corresponding levels.

Validation Type

All the procedures in CWM2_OLAP_VALIDATE package take a validation type argument. The validation type can be one of the following:

DEFAULT -- Validates the basic structure of the metadata and its mapping to the source tables. To be valid, the metadata must meet the criteria specified in "Structural Validation" and "Mapping Validation" on page 19-2.

OLAP API -- Performs default validation plus the following:

- Validates that each dimension of an ET-style cube has dimension and level attributes 'ET KEY' and 'GROUPING ID' for all levels.
- Validates that time dimensions have dimension and level attributes 'END DATE' and 'TIME SPAN' for all levels.

Summary of CWM2_OLAP_VALIDATE Subprograms

Table 19–1 CWM2_OLAP_VALIDATE

Subprogram	Description
VALIDATE_ALL_CUBES Procedure on page 19-4	Validates all the cubes in the OLAP Catalog.
VALIDATE_ALL_DIMENSIONS Procedure on page 19-5	Validates all the dimensions in the OLAP Catalog.
VALIDATE_CUBE Procedure on page 19-5	Validates an OLAP Catalog cube.
VALIDATE_DIMENSION Procedure on page 19-6	Validates an OLAP Catalog dimension.
VALIDATE_OLAP_CATALOG Procedure on page 19-7	Validates all the cubes and all the dimensions in the OLAP Catalog.

VALIDATE_ALL_CUBES Procedure

This procedure validates all the cubes the OLAP Catalog. This includes validation of all the dimensions associated with the cubes.

Cube validity status is displayed in the view ALL_OLAP2_CUBES.

Syntax

```
VALIDATE_ALL_CUBES (
          type_of_validation     IN   VARCHAR2 DEFAULT 'DEFAULT',
          verbose_report         IN   VARCHAR2 DEFAULT 'YES');
```

Parameters

Table 19–2 VALIDATE_ALL_CUBES Procedure Parameters

Parameter	Description
type_of_validation	`'DEFAULT'` or `'OLAP API'`. See "Validation Type" on page 19-3.
verbose_report	`'YES'` or `'NO'`. Whether `to` report all validation checks or only major events and errors. By default, all validation checks are reported.

VALIDATE_ALL_DIMENSIONS Procedure

This procedure validates all the dimensions in the OLAP Catalog.

Dimension validity status is displayed in the view ALL_OLAP2_DIMENSIONS.

Syntax

```
VALIDATE_ALL_DIMENSIONS (
               type_of_validation     IN   VARCHAR2 DEFAULT 'DEFAULT',
               verbose_report         IN   VARCHAR2 DEFAULT 'YES');
```

Parameters

Table 19–3 VALIDATE_ALL_DIMENSIONS Procedure Parameters

Parameter	Description
type_of_validation	'DEFAULT' or 'OLAP API'. See "Validation Type" on page 19-3.
verbose_report	'YES' or 'NO'. Whether to report all validation checks or only major events and errors. By default, all validation checks are reported.

VALIDATE_CUBE Procedure

This procedure validates an OLAP Catalog cube. This includes validation of all the dimensions associated with the cube.

The validity status of a cube is displayed in the view ALL_OLAP2_CUBES.

Syntax

```
VALIDATE_CUBE (
          cube_owner          IN   VARCHAR2,
          cube_name            IN   VARCHAR2,
          type_of_validation  IN   VARCHAR2 DEFAULT 'DEFAULT',
          verbose_report      IN   VARCHAR2 DEFAULT 'YES');
```

Parameters

Table 19–4 VALIDATE_CUBE Procedure Parameters

Parameter	Description
cube_owner	Owner of the cube.

Table 19–4 (Cont.) VALIDATE_CUBE Procedure Parameters

Parameter	Description
cube_name	Name of the cube.
type_of_validation	'DEFAULT' or 'OLAP API'. See "Validation Type" on page 19-3.
verbose_report	'YES' or 'NO'. Whether to report all validation checks or only major events and errors. By default, all validation checks are reported.

VALIDATE_DIMENSION Procedure

This procedure validates an OLAP Catalog dimension.

The validity status of an OLAP dimension is displayed in the view ALL_OLAP2_DIMENSIONS.

Syntax

```
VALIDATE_DIMENSION (
          dimension_owner      IN   VARCHAR2,
          dimension_name        IN   VARCHAR2,
          type_of_validation   IN    VARCHAR2 DEFAULT 'DEFAULT',
          verbose_report       IN   VARCHAR2 DEFAULT 'YES');
```

Parameters

Table 19–5 VALIDATE_DIMENSION Procedure Parameters

Parameter	Description
dimension_owner	Owner of the dimension.
dimension_name	Name of the dimension.
type_of_validation	'DEFAULT' or 'OLAP API'. See "Validation Type" on page 19-3.
verbose_report	'YES' or 'NO'. Whether to report all validation checks or only major events and errors. By default, all validation checks are reported.

VALIDATE_OLAP_CATALOG Procedure

This procedure validates all the metadata in the OLAP Catalog. This includes all the cubes (with their dimensions) and all the dimensions that are not associated with cubes.

`VALIDATE_OLAP_CATALOG` validates each standalone dimension in alphabetical order, then it validates each cube in alphabetical order.

Syntax

```
VALIDATE_OLAP_CATALOG (
          type_of_validation     IN   VARCHAR2 DEFAULT 'DEFAULT',
          verbose_report         IN   VARCHAR2 DEFAULT 'YES');
```

Parameters

Table 19–6 VALIDATE_OLAP_CATALOG Procedure Parameters

Parameter	Description
type_of_validation	`'DEFAULT'` or `'OLAP API'`. See "Validation Type" on page 19-3.
verbose_report	`'YES'` or `'NO'`. Whether `to` report all validation checks or only major events and errors. By default, all validation checks are reported.

20

CWM2_OLAP_VERIFY_ACCESS

The CWM2_OLAP_VERIFY_ACCESS package provides a procedure for validating an OLAP cube and verifying its accessibility to the OLAP API.

See Also:

- "Validating and Committing OLAP Metadata" on page 2-13
- Chapter 16, "CWM2_OLAP_METADATA_REFRESH"
- Chapter 19, "CWM2_OLAP_VALIDATE"

This chapter discusses the following topics:

- Validating the Accessibility of an OLAP Cube
- Summary of CWM2_OLAP_VERIFY_ACCESS Subprograms

Validating the Accessibility of an OLAP Cube

Cube validation procedures in the CWM2_OLAP_VALIDATE package validate the logical structure of an OLAP cube and check that it is correctly mapped to columns in dimension tables and fact tables. However, a cube may be entirely valid according to this criteria and still be inaccessible to your application.

For this reason, you may need to use the CWM2_OLAP_VERIFY_ACCESS package to check that the following additional criteria have also been met:

- The metadata tables used by the OLAP API Metadata Reader must be refreshed with the latest changes in the cube's metadata. If these MRV$ tables have not been updated, you must run the procedures in the CWM2_OLAP_METADATA_REFRESH package to enable access by the OLAP API.

- The identity of the application must have access to the source data that underlies the cube. The validation procedures in `CWM2_OLAP_VALIDATE` run under the `SYS` identity. These procedures may indicate that the cube is entirely valid, and yet the application may not be able to access it. If this is the case, you must grant the appropriate rights to the calling user.

Summary of CWM2_OLAP_VERIFY_ACCESS Subprograms

Table 20–1 CWM2_OLAP_VERIFY_ACCESS

Subprogram	Description
VERIFY_CUBE_ACCESS Procedure on page 20-3	Validates the cube and verifies its accessibility to an OLAP application.

VERIFY_CUBE_ACCESS Procedure

This procedure first validates a cube by calling the `VALIDATE_CUBE` procedure in the `CWM2_OLAP_VALIDATE` package. Additionally it checks that an OLAP API application running under the identity of the calling user has access to the cube.

Cube accessibility requirements are described in "Validating the Accessibility of an OLAP Cube" on page 20-1.

Syntax

```
VERIFY_CUBE_ACCESS (
          cube_owner             IN   VARCHAR2,
          cube_name              IN   VARCHAR2,
          type_of_validation     IN   VARCHAR2 DEFAULT 'DEFAULT',
          verbose_report         IN   VARCHAR2 DEFAULT 'YES');
```

Parameters

Table 20–2 VERIFY_CUBE_ACCESS Procedure Parameters

Parameter	Description
`cube_owner`	Owner of the cube.
`cube_name`	Name of the cube.
type_of_validation	`'DEFAULT'` or `'OLAP API'`. See "Validation Type" on page 19-3.
verbose_report	`'YES'` or `'NO'`. Whether `to` report all validation checks or only major events and errors. By default, all validation checks are reported.

21

DBMS_AW

The `DBMS_AW` package provides procedures and functions for performing operations within analytic workspaces. With `DBMS_AW`, you can:

- Embed OLAP DML commands in SQL statements
- Write queries that return the data resulting from calculations within the workspace
- Obtain information to help you manage aggregate data within the workspace

See Also:

- *Oracle OLAP DML Reference* for information on analytic workspace objects and the syntax of individual OLAP DML commands
- *PL/SQL User's Guide and Reference* for information about the package

This chapter includes the following topics:

- Embedding OLAP DML in SQL Statements
- Embedding Custom Measures in SELECT Statements
- Using the Aggregate Advisor
- Summary of DBMS_AW Subprograms

Embedding OLAP DML in SQL Statements

With the `DBMS_AW` package you can perform the full range of OLAP processing within analytic workspaces. You can import data from legacy workspaces, relational tables, or flat files. You can define OLAP objects and perform complex calculations.

Note: If you use the `DBMS_AW` package to create analytic workspaces from scratch, you may not be able to use OLAP utilities that require standard form. You will have to develop your own relational views of the workspaces using the `OLAP_TABLE` function. To make the workspaces accessible to the OLAP API, you will have to create your own metadata for the views using the `CWM2` packages.

Methods for Executing OLAP DML Commands

The `DBMS_AW` package provides several procedures for executing ad hoc OLAP DML commands. Using the `EXECUTE` or `INTERP_SILENT` procedures or the `INTERP` or `INTERCLOB` functions, you can execute a single OLAP DML command or a series of commands separated by semicolons.

Which procedures you use will depend on how you want to direct output and on the size of the input and output buffers. For example, the `EXECUTE` procedure directs output to a printer buffer, the `INTERP_SILENT` procedure suppresses output, and the `INTERP` function returns the session log.

Guidelines for Using Quotation Marks in OLAP DML Commands

The SQL processor evaluates the embedded OLAP DML commands, either in whole or in part, before sending them to Oracle OLAP for processing. Follow these guidelines when formatting the OLAP DML commands in the `olap-commands` parameter of `DBMS_AW` procedures:

- Wherever you would normally use single quote (`'`) in an OLAP DML command, use two single quotes (`''`). The SQL processor strips one of the single quotes before it sends the OLAP DML command to Oracle OLAP.
- In the OLAP DML, a double quote (`"`) indicates the beginning of a comment.

Embedding Custom Measures in SELECT Statements

The OLAP_EXPRESSION function in the DBMS_AW package dynamically executes a single-row numeric function in an analytic workspace and returns the results. You can embed OLAP_EXPRESSION functions in the WHERE and ORDER BY clauses of SELECT statements.

You can use variants of OLAP_EXPRESSION to calculate text, date, or boolean expressions.

The following script was used to create a view named MEASURE_VIEW, which is used in Example 21–1 and Example 21–2 to illustrate the use of OLAP_EXPRESSION.

Sample View: MEASURE_VIEW

```
CREATE TYPE measure_row AS OBJECT (
   time                        VARCHAR2(12),
   geography                   VARCHAR2(30),
   product                     VARCHAR2(30),
   channel                     VARCHAR2(30),
   sales                       NUMBER(16),
   cost                        NUMBER(16),
   promotions                  NUMBER(16),
   quota                       NUMBER(16),
   units                       NUMBER(16),
   r2c                         RAW(32));
/

CREATE TYPE measure_table AS TABLE OF measure_row;
/

CREATE OR REPLACE VIEW measure_view AS
SELECT sales, cost, promotions, quota, units,
      time, geography, product, channel, r2c
   FROM TABLE(CAST(OLAP_TABLE(
     'xademo DURATION SESSION',
     'measure_table',
     '',
     'MEASURE sales FROM analytic_cube_f.sales
      MEASURE cost FROM analytic_cube_f.costs
      MEASURE promotions FROM analytic_cube_f.promo
      MEASURE quota FROM analytic_cube_f.quota
      MEASURE units FROM analytic_cube_f.units
      DIMENSION time FROM time WITH
```

```
        HIERARCHY time_member_parentrel
           INHIERARCHY time_member_inhier
      DIMENSION geography FROM geography WITH
         HIERARCHY geography_member_parentrel
            INHIERARCHY geography_member_inhier
      DIMENSION product FROM product WITH
         HIERARCHY product_member_parentrel
            INHIERARCHY product_member_inhier
      DIMENSION channel FROM channel WITH
         HIERARCHY channel_member_parentrel
            INHIERARCHY channel_member_inhier
   ROW2CELL r2c')
      AS measure_table))
   WHERE sales IS NOT NULL;
/
COMMIT
/
GRANT SELECT ON measure_view TO PUBLIC;
```

Example 21–1 OLAP_EXPRESSION: Time Series Function with a WHERE Clause

This example uses the view described in "Sample View: MEASURE_VIEW" on page 21-3.

The following `SELECT` statement calculates an expression with an alias of `PERIODAGO`, and limits the result set to calculated values greater than 200,000. The calculation uses the `LAG` function to return the value of the previous time period.

```
SELECT time, cost, OLAP_EXPRESSION(r2c,
   'LAG(analytic_cube_f.costs, 1, time,
      LEVELREL time_member_levelrel)') periodago
FROM measure_view
WHERE geography = 'L1.WORLD' AND
CHANNEL = 'STANDARD_2.TOTALCHANNEL' AND
PRODUCT = 'L1.TOTALPROD' and
OLAP_EXPRESSION(r2c, 'LAG(analytic_cube_f.costs, 1, time,
   LEVELREL time_member_levelrel)') > 200000;
```

This `SELECT` statement produces these results.

```
TIME               COST  PERIODAGO
------------ ---------- ----------
L1.1997         1078031 2490243.07
L2.Q1.97         615399 560379.445
L2.Q2.96         649004 615398.858
L2.Q2.97         462632 649004.473
```

```
L2.Q3.96          582693 462632.064
L2.Q4.96          698166 582693.091
L3.AUG96          194498 209476.344
L3.FEB96          186762 252738.981
L3.JAN96          185755 205214.946
                .
                .
                .
```

Example 21–2 OLAP_EXPRESSION: Numeric Calculation with an ORDER BY CLause

This example uses the view described in "Sample View: MEASURE_VIEW" on page 21-3.

This example subtracts costs from sales to calculate profit, and gives this expression an alias of PROFIT. The rows are ordered by geographic areas from most to least profitable.

```
SELECT geography, sales, cost, OLAP_EXPRESSION(r2c,
   'analytic_cube_f.sales - analytic_cube_f.costs') profit
FROM measure_view
WHERE
channel = 'STANDARD_2.TOTALCHANNEL' AND
product = 'L1.TOTALPROD' AND
time = 'L3.APR97'
ORDER BY OLAP_EXPRESSION(r2c,
   'analytic_cube_f.sales - analytic_cube_f.costs') DESC;
```

This SELECT statement produces these results.

```
GEOGRAPHY                           SALES       COST     PROFIT
------------------------------ ---------- ---------- ----------
L1.WORLD                          9010260     209476 8800783.17
L2.EUROPE                         3884776      95204 3789571.85
L2.AMERICAS                       2734436      55322 2679114.66
L2.ASIA                           1625379      37259 1588120.61
L3.USA                            1603043      27547 1575496.86
L2.AUSTRALIA                       765668      21692 743976.058
L3.UK                              733090      19144 713945.952
L3.CANADA                          731734      19666 712067.455
L4.NEWYORK                         684008       8020 675987.377
L3.GERMANY                         659428      12440 646988.197
L3.FRANCE                          596767      19307 577460.113
                .
                .
                .
```

Using the Aggregate Advisor

The management of aggregate data within analytic workspaces can have significant performance implications. To determine an optimal set of dimension member combinations to preaggregate, you can use the `ADVISE_REL` and `ADVISE_CUBE` procedures in the `DBMS_AW` package. These procedures are known together as the **Aggregate Advisor.**

Based on a percentage that you specify, `ADVISE_REL` suggests a set of dimension members to preaggregate. The `ADVISE_CUBE` procedure suggests a set of members for each dimension of a cube. The Aggregate Advisor procedures require database standard form.

> **See Also:** *Oracle OLAP Application Developer's Guide* for information on standard form analytic workspaces.

Aggregation Facilities within the Workspace

Instructions for storing aggregate data are specified in a workspace object called an aggmap. The OLAP DML `AGGREGATE` command uses the aggmap to preaggregate the data. Any data that is not preaggregated is aggregated dynamically by the `AGGREGATE` function when the data is queried.

Choosing a balance between static and dynamic aggregation depends on many factors including disk space, available memory, and the nature and frequency of the queries that will run against the data. After weighing these factors, you may arrive at a percentage of the data to preaggregate.

Once you have determined the percentage of the data to preaggregate, you can use the Aggregate Advisor. These procedures analyze the distribution of dimension members within hierarchies and identify an optimal set of dimension members to preaggregate.

Example: Using the ADVISE_REL Procedure

Based on a precompute percentage that you specify, the `ADVISE_REL` procedure analyzes a family relation, which represents a dimension with all its hierarchical relationships, and returns a list of dimension members.

`ADVISE_CUBE` applies similar heuristics to each dimension in an aggmap for a cube.

See Also:

- "ADVISE_REL Procedure" on page 21-13
- ADVISE_CUBE Procedure on page 21-12

Example 21–3 on page 21-9 uses a sample Customer dimension to illustrate the ADVISE_REL procedure.

Sample Dimension: Customer in the Global Analytic Workspace

The Customer dimension in GLOBAL_AW.GLOBAL has two hierarchies: SHIPMENTS_ROLLUP with four levels, and MARKET_ROLLUP with three levels. The dimension has 106 members. This number includes all members at each level and all level names.

The members of the Customer dimension are integer keys whose text values are defined in long and short descriptions.

The following OLAP DML commands illustrate some aspects of the standard form representation of the Customer dimension.

```
" ---- Number of members of Customer dimension
>show statlen(customer)
106

" ---- Hierarchies in Customer dimension;
>rpr w 40 customer_hierlist
CUSTOMER_HIERLIST
----------------------------------------
MARKET_ROLLUP
SHIPMENTS_ROLLUP

" ---- Levels in Customer dimension
>rpr w 40 customer_levellist
CUSTOMER_LEVELLIST
----------------------------------------
ALL_CUSTOMERS
REGION
WAREHOUSE
TOTAL_MARKET
MARKET_SEGMENT
ACCOUNT
SHIP_TO
" ---- In the MARKET_ROLLUP hierarchy, ACCOUNT is the leaf level.
" ---- In the SHIPMENTS_HIER hierarchy, SHIP_TO is the leaf level.
" ---- MARKET_HIER                      SHIPMENTS_HIER
" ------------------------------------------------------------
```

```
" ---- TOTAL_MARKET                     ALL_CUSTOMERS
" ---- MARKET_SEGMENT                   REGIONS
" ---- ACCOUNT                          WAREHOUSE
" ----                                 SHIP_TO
" ----
" ---- Parent relation showing parent-child relationships in the Customer dimension
>limit customer to last 20            "Only show the last 20 members
>rpr w 10 down customer w 20 customer_parentrel
           -----------CUSTOMER_PARENTREL------------
           ------------CUSTOMER_HIERLIST------------
CUSTOMER      MARKET_ROLLUP       SHIPMENTS_ROLLUP
---------- ------------------- --------------------
103        44                  21
104        45                  21
105        45                  21
106        45                  21
7          NA                  NA
1          NA                  NA
8          NA                  1
9          NA                  1
10         NA                  1
11         NA                  8
12         NA                  10
13         NA                  9
14         NA                  9
15         NA                  8
16         NA                  9
17         NA                  8
18         NA                  8
19         NA                  9
20         NA                  9
21         NA                  10
" ---- Show text descriptions for the same twenty dimension members
>report w 15 down customer w 35 across customer_hierlist: <customer_short_description>
ALL_LANGUAGES: AMERICAN_AMERICA
                ---------------------------CUSTOMER_HIERLIST---------------------------
                -----------MARKET_ROLLUP----------- ---------SHIPMENTS_ROLLUP----------
CUSTOMER            CUSTOMER_SHORT_DESCRIPTION          CUSTOMER_SHORT_DESCRIPTION
--------------- ----------------------------------- -----------------------------------
103             US Marine Svcs Washington           US Marine Svcs Washington
104             Warren Systems New York             Warren Systems New York
105             Warren Systems Philladelphia        Warren Systems Philladelphia
106             Warren Systems Boston               Warren Systems Boston
7               Total Market                        NA
1               NA                                  All Customers
8               NA                                  Asia Pacific
9               NA                                  Europe
10              NA                                  North America
11              NA                                  Australia
12              NA                                  Canada
```

```
13              NA                                  France
14              NA                                  Germany
15              NA                                  Hong Kong
16              NA                                  Italy
17              NA                                  Japan
18              NA                                  Singapore
19              NA                                  Spain
20              NA                                  United Kingdom
21              NA                                  United States
```

Example 21–3 ADVISE_REL: Suggested Preaggregation of the Customer Dimension

This example uses the GLOBAL Customer dimension described in Sample Dimension: Customer in the Global Analytic Workspace on page 21-7.

The following PL/SQL statements assume that you want to preaggregate 25% of the Customer dimension. ADVISE_REL returns the suggested set of members in a valueset.

```
SQL>SET SERVEROUTPUT ON
SQL>EXECECUTE dbms_aw.execute('aw attach global_aw.global');
SQL>EXECECUTE dbms_aw.execute('define customer_preagg valueset customer');
SQL>EXECECUTE dbms_aw.advise_rel('customer_parentrel', 'customer_preagg', 25);
SQL>EXECECUTE dbms_aw.execute('show values(customer_preagg)');
31
2
4
5
6
7
1
8
9
20
21
```

The Customer members returned are shown below with their text descriptions, related levels, and related hierarchies.

Customer Member	Description	Hierarchy	Level
31	Kosh Enterprises	MARKET_ROLLUP	ACCOUNT
2	Consulting	MARKET_ROLLUP	MARKET_SEGMENT
4	Government	MARKET_ROLLUP	MARKET_SEGMENT
5	Manufacturing	MARKET_ROLLUP	MARKET_SEGMENT

Customer Member	Description	Hierarchy	Level
6	Reseller	MARKET_ROLLUP	MARKET_SEGMENT
7	TOTAL_MARKET	MARKET_ROLLUP	TOTAL_MARKET
1	ALL_CUSTOMERS	SHIPMENTS_ROLLUP	ALL_CUSTOMERS
8	Asia Pacific	SHIPMENTS_ROLLUP	REGION
9	Europe	SHIPMENTS_ROLLUP	REGION
20	United Kingdom	SHIPMENTS_ROLLUP	WAREHOUSE
21	United States	SHIPMENTS_ROLLUP	WAREHOUSE

Summary of DBMS_AW Subprograms

The following table describes the subprograms provided in DBMS_AW.

Table 21–1 DBMS_AW Subprograms

Subprogram	Description
ADVISE_CUBE Procedure on page 21-12	Suggests how to preaggregate a standard form cube, based on a specified percentage of the cube's data.
ADVISE_REL Procedure on page 21-13	Suggests how to preaggregate a standard form dimension, based on a specified percentage of the dimension's members.
AW_ATTACH Procedure on page 21-14	Attaches an analytic workspace to a session.
AW_COPY Procedure on page 21-15	Creates a new analytic workspace and populates it with the object definitions and data from another analytic workspace.
AW_CREATE Procedure on page 21-16	Creates a new, empty analytic workspace.
AW_DELETE on page 21-17	Deletes an analytic workspace
AW_DETACH Procedure on page 21-18	Detaches an analytic workspace from a session.
AW_RENAME Procedure on page 21-19	Changes the name of an analytic workspace.
AW_UPDATE Procedure on page 21-19	Saves changes made to an analytic workspace.
"EXECUTE Procedure" on page 21-20	Executes one or more OLAP DML commands. Input and output is limited to 4K. Typically used in an interactive session using an analytic workspace.
"GETLOG Function" on page 21-21	Returns the session log from the last execution of the INTERP or INTERPCLOB functions.
"INTERP Function" on page 21-22	Executes one or more OLAP DML commands. Input is limited to 4K and output to 4G. Typically used in applications when the 4K limit on output for the EXECUTE procedure is too restrictive.

Table 21–1 (Cont.) DBMS_AW Subprograms

Subprogram	Description
"INTERPCLOB Function" on page 21-23	Executes one or more OLAP DML commands. Input and output are limited to 4G. Typically used in applications when the 4K input limit of the INTERP function is too restrictive.
"INTERP_SILENT Procedure" on page 21-25	Executes one or more OLAP DML commands and suppresses the output. Input is limited to 4K and output to 4G.
"OLAP_EXPRESSION Function" on page 21-26	Returns the result set of a single-row numeric function calculated in an analytic workspace.
"OLAP_EXPRESSION_BOOL Function" on page 21-27	Returns the result set of a single-row boolean function calculated in an analytic workspace.
"OLAP_EXPRESSION_DATE Function" on page 21-28	Returns the result set of a single-row date function calculated in an analytic workspace.
"OLAP_EXPRESSION_TEXT Function" on page 21-29	Returns the result set of a single-row text function calculated in an analytic workspace.
"PRINTLOG Procedure" on page 21-30	Prints a session log returned by the INTERP, INTERCLOB, or GETLOG functions.

ADVISE_CUBE Procedure

The ADVISE_CUBE procedure helps you determine how to preaggregate a standard form cube in an analytic workspace. When you specify a percentage of the cube's data to preaggregate, ADVISE_CUBE recommends a set of members to preaggregate from each of the cube's dimensions.

The ADVISE_CUBE procedure takes an aggmap and a precompute percentage as input. The aggmap must have a precompute clause in each of its RELATION statements. The precompute clause must consist of a valueset. Based on the precompute percentage that you specify, ADVISE_CUBE returns a set of dimension members in each valueset.

Syntax

```
ADVISE_CUBE (
          aggmap_name              IN   VARCHAR2
          precompute_percentage    IN   INTEGER DEFAULT 20);
```

Parameters

Table 21–2 ADVISE_CUBE Procedure Parameters

Parameter	Description
aggmap_name	The name of an aggmap associated with the cube. Each RELATION statement in the aggmap must have a precompute clause containing a valueset. ADVISE_CUBE returns a list of dimension members in each valueset. If the valueset is not empty, ADVISE_CUBE deletes its contents before adding new values.
precompute_percentage	A percentage of the cube's data to preaggregate. The default is 20%.

Example

This example illustrates the ADVISE_CUBE procedure with a cube called UNITS dimensioned by PRODUCT and TIME. ADVISE_CUBE returns the dimension combinations to include if you want to preaggregate 40% of the cube's data.

```
SET SERVEROUTPUT ON
--- View valuesets
SQL>EXECUTE dbms_aw.execute('describe prodvals');
     DEFINE PRODVALS VALUESET PRODUCT
SQL>EXECUTE dbms_aw.execute('describe timevals');
     DEFINE TIMEVALS VALUESET TIME
--- View aggmap
SQL>EXECUTE dbms_aw.execute ('describe units_agg');
     DEFINE UNITS_AGG AGGMAP
          RELATION product_parentrel PRECOMPUTE (prodvals)
          RELATION time_parentrel PRECOMPUTE (timevals)
SQL>EXECUTE dbms_aw.advise_cube ('units_agg', 40);
----
---- The results are returned in the prodvals and timevals valuesets
```

See Also

"Using the Aggregate Advisor" on page 21-6

ADVISE_REL Procedure

The ADVISE_REL procedure helps you determine how to preaggregate a standard form dimension in an analytic workspace. When you specify a percentage of the dimension to preaggregate, ADVISE_REL recommends a set of dimension members.

The ADVISE_REL procedure takes a family relation, a valueset, and a precompute percentage as input. The family relation is a standard form object that specifies the hierarchical relationships between the members of a dimension. The valueset must be defined from the dimension to be analyzed. Based on the precompute percentage that you specify, ADVISE_REL returns a set of dimension members in the valueset.

Syntax

```
ADVISE_REL (
          family_relation_name    IN   VARCHAR2,
          valueset_name           IN   VARCHAR2,
          precompute_percentage   IN   INTEGER DEFAULT 20);
```

Parameters

Table 21–3 ADVISE_REL Procedure Parameters

Parameter	Description
family_relation_name	The name of a family relation, which specifies a dimension and the hierarchical relationships between the dimension members.
valueset_name	The name of a valueset to contain the results of the procedure. The valueset must be defined from the dimension in the family relation. If the valueset is not empty, ADVISE_REL deletes its contents before adding new values.
precompute_percentage	A percentage of the dimension to preaggregate. The default is 20%.

See Also

"Using the Aggregate Advisor" on page 21-6

AW_ATTACH Procedure

The AW_ATTACH procedure attaches an existing analytic workspace to your SQL session so that you can access its contents. The analytic workspace remains attached until you explicitly detach it, or you end your session.

AW_ATTACH can also be used to create a new analytic workspace, but the AW_CREATE procedure is provided specifically for that purpose.

Syntax

```
DBMS_AW.AW_ATTACH (
    awname        IN VARCHAR2,
    forwrite      IN BOOLEAN DEFAULT FALSE,
    createaw      IN BOOLEAN DEFAULT FALSE,
    attargs       IN VARCHAR2 DEFAULT NULL,
    tablespace    IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 21–4 AW_ATTACH Procedure Parameters

Parameter	Description
awname	The name of an existing analytic workspace, unless *createaw* is specified as TRUE. See the description of *createaw*.
forwrite	TRUE attaches the analytic workspace in read/write mode, giving you exclusive access and full administrative rights to the analytic workspace. FALSE attaches the analytic workspace in read-only mode.
createaw	TRUE creates an analytic workspace named *awname*. If *awname* already exists, then an error is generated. FALSE attaches an existing analytic workspace named *awname*.
attargs	Keywords for attaching an analytic workspace, such as FIRST or LAST, as described in the *Oracle OLAP DML Reference* under the AW command.

Example

The following SQL call attaches an analytic workspace named GLOBAL in read/write mode.

```
EXECUTE DBMS_AW.AW_ATTACH('global', TRUE);
```

The next SQL call attaches GLOBAL_PROGRAMS in read-only mode as the last user-owned analytic workspace. If GLOBAL_PROGRAMS is already attached, this call just changes its position in the list of analytic workspaces.

```
EXECUTE DBMS_AW.AW_ATTACH('global_programs', false, false, 'last');
```

AW_COPY Procedure

The AW_COPY procedure creates a new analytic workspace and copies into it both the object definitions and the data from another analytic workspace.

Syntax

```
DBMS_AW.AW_COPY (
    oldname         IN VARCHAR2,
    newname         IN VARCHAR2,
    tablespace      IN VARCHAR2 DEFAULT NULL,
    partnum         IN NUMBER DEFAULT 8);
```

Parameters

Table 21–5 AW_COPY Procedure Parameters

Parameter	Description
`oldname`	The name of an existing analytic workspace.
`newname`	A name for the new analytic workspace.
`tablespace`	The name of a tablespace in which *newname* will be stored. If this parameter is omitted, then the analytic workspace is created in the user's default tablespace.
`partnum`	The number of partitions that will be created for the `AW$`*newname* table.

Example

The following command creates a new analytic workspace named `DEMO` and copies the contents of `GLOBAL` into it. The workspace is stored in a table named `AW$DEMO`, which has three partitions and is stored in the user's default tablespace.

```
EXECUTE DBMS_AW.AW_COPY('global', 'demo', null, 3);
```

AW_CREATE Procedure

The `AW_CREATE` procedure creates a new, empty analytic workspace.

Syntax

```
DBMS_AW.AW_CREATE (
    awname        IN VARCHAR2 ,
    tablespace    IN VARCHAR2 DEFAULT NULL ,
    partnum       IN NUMBER DEFAULT 8 );
```

Parameters

Table 21–6 AW_CREATE Procedure Parameters

Parameter	Description
`awname`	The name of a new analytic workspace. The name must comply with the naming requirements for a table in an Oracle database. This procedure creates a table named AW$*awname*, in which the analytic workspace is stored.
`tablespace`	The tablespace in which the analytic workspace will be created. If you omit this parameter, the analytic workspace is created in your default tablespace.
`partnum`	The number of partitions that will be created for the `AW$`*awname* table.

Example

The following command creates a new, empty analytic workspace named `GLOBAL`. The new analytic workspace is stored in a table named `AW$GLOBAL` with eight partitions in the user's default tablespace.

```
EXECUTE DBMS_AW.AW_CREATE('global');
```

The next command creates an analytic workspace named `DEMO` in the `GLOBAL_AW` schema. `AW$DEMO` will have two partitions and will be stored in the `GLOBAL` tablespace.

```
EXECUTE DBMS_AW.AW_CREATE('global_aw.demo', 'global', 2);
```

AW_DELETE

The `AW_DELETE` procedure deletes an existing analytic workspace.

Syntax

```
DBMS_AW.AW_DELETE (
    awname        IN VARCHAR2 );
```

Parameters

Table 21–7 AW_DELETE Procedure Parameters

Parameter	Description
awname	The name of an existing analytic workspace that you want to delete along with all of its contents. You must be the owner of *awname* or have DBA rights to delete it, and it cannot currently be attached to your session. The AW$*awname* file is deleted from the database.

Example

The following SQL call deletes the GLOBAL analytic workspace in the user's default schema.

```
EXECUTE DBMS_AW.AW_DELETE('global');
```

AW_DETACH Procedure

The AW_DETACH procedure detaches an analytic workspace from your session so that its contents are no longer accessible. All changes that you have made since the last update are discarded. Refer to "AW_UPDATE Procedure" on page 21-19 for information about saving changes to an analytic workspace.

Syntax

```
DBMS_AW.AW_DETACH (
    awname        IN VARCHAR2);
```

Parameters

Table 21–8 AW_DETACH Procedure Parameters

Parameter	Description
awname	The name of an attached analytic workspace that you want to detach from your session.

Example

The following command detaches the GLOBAL analytic workspace.

```
EXECUTE DBMS_AW.AW_DETACH('global');
```

AW_RENAME Procedure

The AW_RENAME procedure changes the name of an analytic workspace.

Syntax

```
DBMS_AW.AW_RENAME (
    oldname       IN VARCHAR2 DEFAULT NULL,
    newname       IN VARCHAR2 );
```

Parameters

Table 21–9 AW_RENAME Procedure Parameters

Parameter	Description
oldname	The current name of the analytic workspace. The analytic workspace cannot be attached to any session.
newname	The new name of the analytic workspace.

Example

The following command changes the name of the GLOBAL analytic workspace to DEMO.

```
EXECUTE DBMS_AW.AW_RENAME('global', 'demo');
```

AW_UPDATE Procedure

The AW_UPDATE procedure saves the changes made to an analytic workspace in its permanent database table. For the updated version of this table to be saved in the database, you must issue a SQL COMMIT statement before ending your session.

Syntax

```
DBMS_AW.AW_UPDATE (
    awname     IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 21–10 AW_UPDATE Procedure Parameters

Parameter	Description
`awname`	Saves changes to *awname* by copying them to a table named `AW$`*awname*. If this parameter is omitted, then changes are saved for all analytic workspaces attached in read/write mode.

Example

The following command saves changes to the `GLOBAL` analytic workspace to a table named `AW$GLOBAL`.

```
EXECUTE DBMS_AW.AW_UPDATE('global');
```

EXECUTE Procedure

The `EXECUTE` procedure executes one or more OLAP DML commands and directs the output to a printer buffer. It is typically used to manipulate analytic workspace data within an interactive SQL session.

When you are using SQL*Plus, you can direct the printer buffer to the screen by issuing the following command:

```
SET SERVEROUT ON
```

If you are using a different program, refer to its documentation for the equivalent setting.

Input and output is limited to 4K. For larger values, refer to the `INTERP` and `INTERPCLOB` functions in this package.

This procedure does not print the output of the DML commands when you have redirected the output by using the OLAP DML `OUTFILE` command.

Syntax

```
EXECUTE (
          olap_commands     IN    VARCHAR2
          text              OUT   VARCHAR2);
```

Parameters

Table 21–11 EXECUTE Procedure Parameters

Parameter	Description
`olap-commands`	One or more OLAP DML commands separated by semicolons. See "Guidelines for Using Quotation Marks in OLAP DML Commands" on page 21-2.
text	Output from the OLAP engine in response to the OLAP commands.

Example

The following sample SQL*Plus session attaches an analytic workspace named `XADEMO`, creates a formula named `COST_PP` in `XADEMO`, and displays the new formula definition.

```
SQL> SET SERVEROUT ON

SQL> EXECUTE DBMS_AW.EXECUTE('AW ATTACH xademo RW; DEFINE cost_pp FORMULA
LAG(analytic_cube_f.costs, 1, time, LEVELREL time_levelrel)');

PL/SQL procedure successfully completed.

SQL> EXECUTE DBMS_AW.EXECUTE('DESCRIBE cost_pp');

DEFINE COST_PP FORMULA DECIMAL <CHANNEL GEOGRAPHY PRODUCT TIME>
EQ lag(analytic_cube_f.costs, 1, time, levelrel time.levelrel)

PL/SQL procedure successfully completed.
```

GETLOG Function

This function returns the session log from the last execution of the `INTERP` or `INTERPCLOB` functions in this package.

To print the session log returned by this function, use the `DBMS_AW.PRINTLOG` procedure.

Syntax

```
GETLOG()
          RETURN CLOB;
```

Returns

The session log from the latest call to INTERP or INTERPCLOB.

Example

The following example shows the session log returned by a call to INTERP, then shows the identical session log returned by GETLOG.

```
SQL> SET SERVEROUT ON SIZE 1000000
SQL> EXECUTE DBMS_AW.PRINTLOG(DBMS_AW.INTERP('AW ATTACH xademo; LISTNAMES AGGMAP'));
2 AGGMAPs
------------------------------------------
ANALYTIC_CUBE.AGGMAP.1
SALES_MULTIKEY_CUBE.AGGMAP.1

PL/SQL procedure successfully completed.


SQL> EXECUTE DBMS_AW.PRINTLOG(DBMS_AW.GETLOG());
2 AGGMAPs
------------------------------------------
ANALYTIC_CUBE.AGGMAP.1
SALES_MULTIKEY_CUBE.AGGMAP.1

PL/SQL procedure successfully completed.
```

INTERP Function

The INTERP function executes one or more OLAP DML commands and returns the session log in which the commands are executed. It is typically used in applications when the 4K limit on output for the EXECUTE procedure may be too restrictive.

Input to the INTERP function is limited to 4K. For larger input values, refer to the INTERPCLOB function of this package.

This function does not return the output of the DML commands when you have redirected the output by using the OLAP DML OUTFILE command.

You can use the INTERP function as an argument to the PRINTLOG procedure in this package to view the session log. See the example.

Syntax

```
INTERP (
          olap-commands     IN   VARCHAR2)
     RETURN CLOB;
```

Parameters

Table 21–12 INTERP Function Parameters

Parameter	Description
`olap-commands`	One or more OLAP DML commands separated by semi-colons. See "Guidelines for Using Quotation Marks in OLAP DML Commands" on page 21-2.

Returns

The log file for the Oracle OLAP session in which the OLAP DML commands were executed.

Example

The following sample SQL*Plus session attaches an analytic workspace named XADEMO and lists the members of the PRODUCT dimension.

```
SQL> SET SERVEROUT ON SIZE 1000000
SQL> EXECUTE DBMS_AW.PRINTLOG(DBMS_AW.INTERP('AW ATTACH cloned; REPORT product'));
PRODUCT
--------------
L1.TOTALPROD
L2.ACCDIV
L2.AUDIODIV
L2.VIDEODIV
L3.AUDIOCOMP
L3.AUDIOTAPE
     .
     .
     .
PL/SQL procedure successfully completed.
```

INTERPCLOB Function

The INTERPCLOB function executes one or more OLAP DML commands and returns the session log in which the commands are executed. It is typically used in applications when the 4K limit on input for the INTERP function may be too restrictive.

This function does not return the output of the OLAP DML commands when you have redirected the output by using the OLAP DML OUTFILE command.

You can use the INTERPCLOB function as an argument to the PRINTLOG procedure in this package to view the session log. See the example.

Syntax

```
INTERPCLOB (
          olap-commands     IN   CLOB)
     RETURN CLOB;
```

Parameters

Table 21–13 INTERPCLOB Function Parameters

Parameter	Description
`olap-commands`	One or more OLAP DML commands separated by semi-colons. See "Guidelines for Using Quotation Marks in OLAP DML Commands" on page 21-2.

Returns

The log for Oracle OLAP session in which the OLAP DML commands were executed.

Example

The following sample SQL*Plus session creates an analytic workspace named `ELECTRONICS`, imports its contents from an EIF file stored in the `dbs` directory alias, and displays the contents of the analytic workspace.

```
SQL> SET SERVEROUT ON SIZE 1000000
SQL> EXECUTE DBMS_AW.PRINTLOG(DBMS_AW.INTERPCLOB('AW CREATE electronics; IMPORT
ALL FROM EIF FILE ''dbs/electronics.eif'' DATA DFNS; DESCRIBE'));

DEFINE GEOGRAPHY DIMENSION TEXT WIDTH 12
LD Geography Dimension Values
DEFINE PRODUCT DIMENSION TEXT WIDTH 12
LD Product Dimension Values
DEFINE TIME DIMENSION TEXT WIDTH 12
LD Time Dimension Values
DEFINE CHANNEL DIMENSION TEXT WIDTH 12
LD Channel Dimension Values
          .
          .
          .
PL/SQL procedure successfully completed.
```

INTERP_SILENT Procedure

The INTERP_SILENT procedure executes one or more OLAP DML commands and suppresses all output from them. It does not suppress error messages from the OLAP command interpreter.

Input to the INTERP_SILENT function is limited to 4K. If you want to display the output of the OLAP DML commands, use the EXECUTE procedure, or the INTERP or INTERPCLOB functions.

Syntax

```
INTERP_SILENT (
          olap-commands     IN   VARCHAR2);
```

Parameters

Table 21–14 INTERP_SILENT Function Parameters

Parameter	Description
olap-commands	One or more OLAP DML commands separated by semi-colons. See "Guidelines for Using Quotation Marks in OLAP DML Commands" on page 21-2.

Example

The following commands show the difference in message handling between EXECUTE and INTERP_SILENT. Both commands attach the XADEMO analytic workspace in read-only mode. However, EXECUTE displays a warning message, while INTERP_SILENT does not.

```
SQL> EXECUTE DBMS_AW.EXECUTE('AW ATTACH xademo');
IMPORTANT: Analytic workspace XADEMO is read-only. Therefore, you will
not be able to use the UPDATE command to save changes to it.

PL/SQL procedure successfully completed.


SQL> EXECUTE DBMS_AW.INTERP_SILENT('AW ATTACH xademo');

PL/SQL procedure successfully completed.
```

OLAP_EXPRESSION Function

The OLAP_EXPRESSION function enables you to execute single-row numeric functions in an analytic workspace and thus generate custom measures in SELECT statements. In addition to calculating an expression, OLAP_EXPRESSION can be used in the WHERE and ORDER BY clauses to modify the result set of a SELECT.

Syntax

```
OLAP_EXPRESSION(
          r2c          IN   RAW(32),
          expression   IN   VARCHAR2 )
      RETURN NUMBER;
```

Parameters

Table 21–15 OLAP_EXPRESSION Function Parameters

Parameter	Description
r2c	The name of a column populated by a ROW2CELL clause in a call to OLAP_TABLE. ROW2CELL is a component of a limit map parameter of the OLAP_TABLE function. See "Limit Map Parameter" on page 26-6.
expression	A numeric calculation that will be performed in the analytic workspace.

Returns

An evaluation of *expression* for each row of the table object returned by the OLAP_TABLE function.

To return text, boolean, or date data, use the OLAP_EXPRESSION_TEXT, OLAP_EXPRESSION_BOOL, or OLAP_EXPRESSION_DATE functions in this package.

Note

You can use OLAP_EXPRESSION only with a table object returned by the OLAP_TABLE function. The returned table object must have a column populated by a ROW2CELL. Refer to Chapter 26, "OLAP_TABLE" for more information about using this function.

Example

See "Embedding Custom Measures in SELECT Statements" on page 21-3.

OLAP_EXPRESSION_BOOL Function

The OLAP_EXPRESSION_BOOL function enables you to execute single-row boolean functions in an analytic workspace and thus generate custom measures in SELECT statements. In addition to calculating an expression, OLAP_EXPRESSION_BOOL can be used in the WHERE and ORDER BY clauses to modify the result set of a SELECT.

Syntax

```
OLAP_EXPRESSION_BOOL(
          r2c           IN    RAW(32),
          expression    IN    VARCHAR2 )
     RETURN NUMBER;
```

Parameters

Table 21–16 OLAP_EXPRESSION_BOOL Function Parameters

Parameter	Description
r2c	The name of a column populated by a ROW2CELL clause in a call to OLAP_TABLE. ROW2CELL is a component of a limit map parameter of the OLAP_TABLE function. See "Limit Map Parameter" on page 26-6.
expression	A boolean calculation that will be performed in the analytic workspace.

Returns

An evaluation of *expression* for each row of the table object returned by the OLAP_TABLE function.

Return values are numbers 1 (true) or 0 (false).

To return text, numeric, or date data, use the OLAP_EXPRESSION_TEXT, OLAP_EXPRESSION, or OLAP_EXPRESSION_DATE functions in this package.

Note

You can use OLAP_EXPRESSION_BOOL only with a table object returned by the OLAP_TABLE function. The returned table object must have a column populated by a ROW2CELL. Refer to Chapter 26, "OLAP_TABLE" for more information about using this function.

Example

See "Embedding Custom Measures in SELECT Statements" on page 21-3.

OLAP_EXPRESSION_DATE Function

The OLAP_EXPRESSION_DATE function enables you to execute single-row date functions in an analytic workspace and thus generate custom measures in SELECT statements. In addition to calculating an expression, OLAP_EXPRESSION_DATE can be used in the WHERE and ORDER BY clauses to modify the result set of a SELECT.

Syntax

```
OLAP_EXPRESSION_DATE(
          r2c          IN   RAW(32),
          expression   IN   VARCHAR2 )
RETURN DATE;
```

Parameters

Table 21–17 OLAP_EXPRESSION_DATE Function Parameters

Parameter	Description
r2c	The name of a column populated by a ROW2CELL clause in a call to OLAP_TABLE. ROW2CELL is a component of a limit map parameter of the OLAP_TABLE function. See "Limit Map Parameter" on page 26-6.
expression	A date calculation that will be performed in the analytic workspace.

Returns

An evaluation of *expression* for each row of the table object returned by the OLAP_TABLE function.

To return text, boolean, or numeric data, use the OLAP_EXPRESSION_TEXT, OLAP_EXPRESSION_BOOL, or OLAP_EXPRESSION functions in this package.

Note

You can use OLAP_EXPRESSION_DATE only with a table object returned by the OLAP_TABLE function. The returned table object must have a column populated by a ROW2CELL. Refer to Chapter 26, "OLAP_TABLE" for more information about using this function.

Example

See "Embedding Custom Measures in SELECT Statements" on page 21-3.

OLAP_EXPRESSION_TEXT Function

The OLAP_EXPRESSION_TEXT function enables you to execute single-row text functions in an analytic workspace and thus generate custom measures in SELECT statements. In addition to calculating an expression, OLAP_EXPRESSION_TEXT can be used in the WHERE and ORDER BY clauses to modify the result set of a SELECT.

Syntax

```
OLAP_EXPRESSION_TEXT(
          r2c          IN   RAW(32),
          expression   IN   VARCHAR2 )
RETURN VARCHAR2;
```

Parameters

Table 21–18 OLAP_EXPRESSION_TEXT Function Parameters

Parameter	Description
r2c	The name of a column populated by a ROW2CELL clause in a call to OLAP_TABLE. ROW2CELL is a component of a limit map parameter of the OLAP_TABLE function. See "Limit Map Parameter" on page 26-6.
expression	A text calculation that will be performed in the analytic workspace.

Returns

An evaluation of *expression* for each row of the table object returned by the OLAP_TABLE function.

To return numeric, boolean, or date data, use the OLAP_EXPRESSION, OLAP_EXPRESSION_BOOL, or OLAP_EXPRESSION_DATE functions in this package.

Note

You can use OLAP_EXPRESSION_TEXT only with a table object returned by the OLAP_TABLE function. The returned table object must have a column populated by a ROW2CELL. Refer to Chapter 26, "OLAP_TABLE" for more information about using this function.

Example

See "Embedding Custom Measures in SELECT Statements" on page 21-3.

PRINTLOG Procedure

This procedure sends a session log returned by the INTERP, INTERPCLOB, or GETLOG functions of this package to the print buffer, using the DBMS_OUTPUT package in PL/SQL.

When you are using SQL*Plus, you can direct the printer buffer to the screen by issuing the following command:

```
SET SERVEROUT ON SIZE 1000000
```

The SIZE clause increases the buffer from its default size of 4K.

If you are using a different program, refer to its documentation for the equivalent setting.

Syntax

```
DBMS_AW.PRINTLOG (
          session-log     IN   CLOB);
```

Parameters

Table 21–19 PRINTLOG Procedure Parameters

Parameter	Description
session-log	The log of a session.

Example

The following example shows the session log returned by the `INTERP` function.

```
SQL> SET SERVEROUT ON SIZE 1000000
SQL> EXECUTE DBMS_AW.PRINTLOG(DBMS_AW.INTERP('DESCRIBE analytic_cube_f.profit'));

DEFINE ANALYTIC_CUBE.F.PROFIT FORMULA DECIMAL <CHANNEL
GEOGRAPHY PRODUCT TIME>
EQ analytic_cube.f.sales - analytic_cube.f.costs

PL/SQL procedure successfully completed.
```

22

DBMS_AW_UTILITIES

The DBMS_AW_UTILITIES package contains procedures for managing custom measures in analytic workspaces.

See Also:

- *Oracle OLAP Application Developer's Guide* for more information on analytic workspaces.
- Chapter 1, "Creating Analytic Workspaces with DBMS_AWM" for information on creating relational views of analytic workspaces.

This chapter contains the following topics:

- About Custom Measures
- Querying Custom Measures
- Example: Creating a Custom Measure
- Summary of DBMS_AW_UTILITIES Subprograms

About Custom Measures

You can use the DBMS_AW_UTILITIES package to define custom measures within database standard form analytic workspaces and associate the custom measures with columns in relational views. You can define temporary custom measures for use during the current session, or you can save them permanently.

Note: Standard form analytic workspaces, and the relational views that expose their contents, are created by procedures in the DBMS_AWM package.

A custom measure is derived from one or more stored measures. It is calculated at run-time and returned in columns of a view that is structured like a fact table. An example of a custom measure is PROFITS, which is calculated by subtracting the COSTS measure from the SALES measure.

Custom measures created by DBMS_AW_UTILITIES are defined as formulas in an analytic workspace. A formula is a workspace schema object representing a calculation. The result set of a formula includes a value for each workspace dimension member currently in status.

See Also: *Oracle OLAP DML Reference* for information on defining formulas and setting dimension status with the OLAP DML.

Querying Custom Measures

When the CREATE_CUSTOM_MEASURE procedure successfully creates a new custom measure, it provides the following information.

```
Custom Measure cust_meas_name created in Workspace workspace_name.
Custom Measure cust_meas_name mapped to column col_name in View view_name.
```

You can query the specified column to obtain the results of custom measure calculations.

Alternatively, you can query the following tables to obtain information about custom measures created with CREATE_CUSTOM_MEASURE. These tables also provide the name of the columns that contain the results of custom measure calculations.

- olapsys.CWM2$_AW_PERM_CUST_MEAS_MAP — This table provides information about permanent custom measures. This table is only available to users with DBA privileges.
- olapsys.CWM2$_AW_TEMP_CUST_MEAS_MAP — This table provides information about temporary custom measures. This table is accessible to the current user.

CWM2$_AW_PERM_CUST_MEAS_MAP

The columns of the CWM2$_AW_PERM_CUST_MEAS_MAP table are described in the following table.

Column	Datatype	NULL	Description
AW_ACCESS_VIEW_NAME	VARCHAR2(61)	not null	Name of the view that contains the permanent custom measure.
CUST_ADT_COLUMN	VARCHAR2(30)	not null	Column in the view.
WORKSPACE_NAME	VARCHAR2(61)		Name of the analytic workspace that contains the measures on which the custom measure is based and the formula that defines the custom measure calculation.
AW_MEASURE_NAME	VARCHAR2(64)		Name of the derived (custom) measure.
SESSIONID	VARCHAR2(10)		ID of the session in which the custom measure was created.
USERNAME	VARCHAR2(30)		User that created the custom measure.

CWM2$_AW_TEMP_CUST_MEAS_MAP

The columns of the CWM2$_AW_TEMP_CUST_MEAS_MAP table are described in the following table.

Column	Datatype	NULL	Description
AW_ACCESS_VIEW_NAME	VARCHAR2(61)	not null	Name of the view that contains the temporary custom measure.
CUST_ADT_COLUMN	VARCHAR2(30)	not null	Column in the view.
WORKSPACE_NAME	VARCHAR2(61)		Name of the analytic workspace that contains the measures on which the custom measure is based and the formula that defines the custom measure calculation.
AW_MEASURE_NAME	VARCHAR2(64)		Name of the derived(custom) measure.
SESSIONID	VARCHAR2(10)		ID of the current session. The custom measure only exists in the current session.
USERNAME	VARCHAR2(30)		User that created the custom measure.

Example: Creating a Custom Measure

The following example creates a temporary custom measure in the analytic workspace GLOBAL_AW.GLOBAL. The measure returns the difference between Unit Price and Unit Cost for the cube PRICE_CUBE. The custom measure is returned in the view GLOBAL_AW.GLOB_GLOBA_UNITS_CU10VIEW.

To see the output of your queries, direct output to the screen.

```
SQL>set serverout on
SQL>exec cwm2_olap_manager.set_echo_on;
```

You can use the following query to obtain a list of the available analytic workspaces.

```
SQL>select * from all_olap2_aws where aw = 'GLOBAL';

OWNER                          AW                              AW_NUMBER
------------------------------ ------------------------------ ----------
GLOBAL_AW                      GLOBAL                               1005
```

The following query returns a list of the enabled views for cubes in the analytic workspaces.

```
SQL>select * from all_aw_cube_enabled_views where aw_name = 'GLOBAL';
```

```
AW_OWNER  AW_NAME CUBE_NAME  HIERCOMBO_NU HIERCOMBO_STR                  SYSTEM_VIEWNAME        USERP_VIE
--------- ------- ---------- ------------ -----------------------------   ----------------------- ---------
GLOBAL_AW GLOBAL  PRICE_CUBE   ########## DIM:PRODUCT/HIER:PRODUCT_ROLLUP;  GLOB_GLOBA_PRICE_CU4VIEW
                                          DIM:TIME/HIER:CALENDAR

GLOBAL_AW GLOBAL  UNITS_CUBE   ########## DIM:CHANNEL/HIER:CHANNEL_ROLLUP;  GLOB_GLOBA_UNITS_CU9VIEW
                                          DIM:CUSTOMER/HIER:MARKET_ROLLUP;
                                          DIM:PRODUCT/HIER:PRODUCT_ROLLUP;
                                          DIM:TIME/HIER:CALENDAR

GLOBAL_AW GLOBAL  UNITS_CUBE   ########## DIM:CHANNEL/HIER:CHANNEL_ROLLUP;   GLOB_GLOBA_UNITS_CU10VIEW
                                          DIM:CUSTOMER/HIER:SHIPMENTS_ROLLUP;
                                          DIM:PRODUCT/HIER:PRODUCT_ROLLUP;
                                          DIM:TIME/HIER:CALENDAR
```

You can query the following Active Catalog view to obtain the names of the measures in the cubes.

```
SQL>select * from all_olap2_aw_cube_measures where aw_name = 'GLOBAL';
```

```
AW_OWNER   AW_NAM  AW_CUBE_NAM   AW_MEASURE_ AW_PHYSICAL_  MEASURE_SOU  DISPLAY_NAM  DESCRIPTI   IS_AGGR
---------  ------- ----------    ----------  ------------  -----------  -----------  ----------   -----
GLOBAL_AW  GLOBAL  PRICE_CUBE    UNIT_COST    UNIT_COST    UNIT_COST    UNIT COST    Unit Cost    YES
GLOBAL_AW  GLOBAL  PRICE_CUBE    UNIT_PRICE   UNIT_PRICE   UNIT_PRICE   UNIT PRICE   Unit Price   YES
GLOBAL_AW  GLOBAL  UNITS_CUBE    UNITS        UNITS        UNITS        UNITS        Units Sold   YES
```

The following statement creates a numeric formula PRICE_COST in the analytic workspace GLOBAL in the GLOBAL_AW schema . The formula calculates the difference between unit prices and unit costs. The resulting data is returned in the view GLOBAL_AW.GLOB_GLOBA_UNITS_CU10VIEW.

```
SQL>execute dbms_aw_utilities.create_custom_measure
            ('GLOBAL_AW.GLOBAL', 'PRICE_COST',
             'UNIT_PRICE - UNIT_COST', 'temporary',
             'GLOBAL_AW.GLOB_GLOBA_UNITS_CU10VIEW');

Custom Measure 'PRICE_COST' created in Workspace 'GLOBAL_AW.GLOBAL'.
Custom Measure 'PRICE_COST' mapped to column 'CUST_MEAS_NUM1'
               in View 'GLOBAL_AW.GLOB_GLOBA_UNITS_CU10VIEW'.
```

With the following query, you can see your new custom measure listed in the CWM2$_AW_temp_CUST_MEAS_MAP table.

```
SQL>select * from olapsys.CWM2$_AW_TEMP_CUST_MEAS_MAP
                         where workspace_name = 'GLOBAL_AW.GLOBAL';

AW_ACCESS_VIEW_NAME                  CUST_ADT_COLUMN  WORKSPACE_NAME    AW_MEASURE_NAME   SESSIONID  USERNAME
-----------------------------------  ---------------  ----------------- --- ------------  ---------- -----
GLOBAL_AW.GLOB_GLOBA_UNITS_CU10VIEW  CUST_MEAS_NUM1   GLOBAL_AW.GLOBAL  PRICE_COST          325      MYUSER
```

To obtain the data resulting from the custom calculation, use the following query.

```
SQL>select CUST_MEAS_NUM1 from GLOBAL_AW.GLOB_GLOBA_UNITS_CU10VIEW;
```

Summary of DBMS_AW_UTILITIES Subprograms

Table 22–1 lists the subprograms provided in DBMS_AW_UTILITIES.

Table 22–1 DBMS_AW_UTILITIES

Subprogram	Description
CREATE_CUSTOM_MEASURE Procedure on page 22-6	Creates an OLAP formula and associates it with columns in a fact view of an analytic workspace.
DELETE_CUSTOM_MEASURE Procedure on page 22-8	Deletes a custom measure that was created by CREATE_CUSTOM_MEASURE.
UPDATE_CUSTOM_MEASURE Procedure on page 22-8	Changes the definition of an OLAP formula that was created by CREATE_CUSTOM_MEASURE.

CREATE_CUSTOM_MEASURE Procedure

The CREATE_CUSTOM_MEASURE procedure specifies a calculation to be created and stored in a formula object within an analytic workspace. The formula may be defined permanently in the analytic workspace, or it may exist temporarily until the workspace is closed.

CREATE_CUSTOM_MEASURE associates the formula with columns of a fact view. When these columns are queried, the formula calculates the custom measure and populates the columns with the result set. CREATE_CUSTOM_MEASURE assumes that the fact view was previously created by an enablement script generated by the to DBMS_AWM.CREATE_AWCUBE_ACCESS procedure. The view presents the measures of an analytic workspace cube as a set of logical fact tables. There is a separate view for each combination of hierarchies.

The views are created with empty text columns and numeric columns that may be used for custom measures. There are one hundred empty columns of each type.

The text columns are named CUST_MEAS_TEXT*n*, where *n* is a number from one to one hundred. The data type is VARCHAR2(1000).

The numeric columns are named CUST_MEAS_NUM*n*, where *n* is a number from one to one hundred. The data type is NUMBER.

Syntax

```
CREATE_CUSTOM_MEASURE(
        aw_name                  VARCHAR2,
        aw_formula_name          VARCHAR2,
        aw_formula_expr          VARCHAR2,
        aw_formula_create_mode   VARCHAR2,
        view_name                VARCHAR2);
```

Parameters

Table 22–2 CREATE_CUSTOM_MEASURE Procedure Parameters

Parameter	Description
`aw_name`	Name of the analytic workspace. The name must be specified in the form *`owner.name`*., where *`owner`* is the schema name and *`name`* is the workspace name.
`aw_formula_name`	Name of the formula to be created in the analytic workspace.
`aw_formula_expr`	A text or numeric expression to be stored in the formula.
`aw_formula_create_mode`	One of the following values: '`PERMANENT`' -- Create the formula permanently in the analytic workspace. The workspace will be opened in read/write mode, updated, and committed. '`TEMPORARY`' -- Create the formula temporarily in the analytic workspace. The workspace will be opened in read-only mode, and the formula will be discarded when the workspace is closed.
`view_name`	Name of the view that will use the `OLAP_TABLE` function to access the analytic workspace and read the custom measure data. Text data will be returned in columns named `CUST_MEAS_TEXT`*n*, where *n* is the next available sequentially numbered column. Numeric data will be returned in columns named `CUST_MEAS_NUM`*n*, where *n* is the next available sequentially numbered column.

See Also

- "CREATE_AWCUBE_ACCESS Procedure" on page 23-22

- "Enabling Relational Access to the Workspace Cube" on page 1-5
- "Creating Relational Access to the Workspace Cube" on page 1-23

DELETE_CUSTOM_MEASURE Procedure

The DELETE_CUSTOM_MEASURE procedure deletes a custom measure that was created by CREATE_CUSTOM_MEASURE. It deletes the formula that calculates the custom measure in the analytic workspace and removes the formula from the columns of the fact view.

Syntax

```
DELETE_CUSTOM_MEASURE(
          aw_name                  VARCHAR2,
          aw_formula_name          VARCHAR2,
          view_name                VARCHAR2);
```

Parameters

Table 22–3 DELETE_CUSTOM_MEASURE Procedure Parameters

Parameter	Description
aw_name	Name of the analytic workspace. The name must be specified in the form *owner.name*., where *owner* is the schema name and *name* is the workspace name.
aw_formula_name	Name of the formula to be deleted from the analytic workspace.
view_name	Name of the view specified by CREATE_CUSTOM_MEA2706345SURE. References to the custom measure will be removed from the columns of the view.

UPDATE_CUSTOM_MEASURE Procedure

This procedure updates the formula for a custom measure in an analytic workspace.

The formula was previously defined and associated with a view by the CREATE_CUSTOM_MEASURE procedure.

Syntax

```
UPDATE_CUSTOM_MEASURE(
        aw_name                  VARCHAR2,
        aw_formula_name          VARCHAR2,
        aw_formula_expr          VARCHAR2);
```

Parameters

Table 22–4 UPDATE_CUSTOM_MEASURE Procedure Parameters

Parameter	Description
`aw_name`	Name of the analytic workspace. The name must be specified in the form *`owner.name`*., where *`owner`* is the schema name and *`name`* is the workspace name.
`aw_formula_name`	Name of the formula in the analytic workspace.
`aw_formula_expr`	The new calculation to be performed by the formula.

23

DBMS_AWM

The Analytic Workspace Manager package, DBMS_AWM, provides procedures for loading data from a relational data warehouse into an analytic workspace and enabling the workspace for access by the OLAP API and BI Beans.

Note: You can access much of the functionality of the DBMS_AWM package through the graphical user interface of the Analytic Workspace Manager.

See Also:

- Chapter 1, "Creating Analytic Workspaces with DBMS_AWM"
- Chapter 2, "Creating OLAP Catalog Metadata with CWM2"

This chapter discusses the following topics:

- Parameters of DBMS_AWM Subprograms
- Summary of DBMS_AWM Subprograms

Parameters of DBMS_AWM Subprograms

The parameters cube_name, dimension_name, measure_name, and level_name refer to the metadata entities in the OLAP Catalog that map to the **relational source cube**.

The parameters aw_cube_name or aw_dimension_name refer to the **target cube** or dimension within an analytic workspace.

Parameters with the suffix _spec refer to the named specifications for loading, aggregating, and optimizing a target cube in an analytic workspace.

> **See Also:** "Overview" on page 1-2 for definitions of the terms, "relational source cube", "multidimensional target cube", and "relational target cube".

DBMS_AWM parameters are summarized in Table 23–1.

Table 23–1 Parameters of DBMS_AWM Procedures

Parameter	Description
cube_owner	Owner of the OLAP Catalog cube associated with the relational source tables (star schema).
cube_name	Name of the OLAP Catalog cube associated with the relational source tables (star schema).
dimension_owner	Owner of the OLAP Catalog dimension associated with the source dimension lookup table.
dimension_name	Name of the OLAP Catalog dimension associated with the source dimension lookup table.
aw_owner	Owner of the analytic workspace. Also the owner of cubes and dimensions within the workspace.
aw_cube_name	Name of the target cube within an analytic workspace. For information on naming requirements, see Table 23–13, " CREATE_AWCUBE Procedure Parameters".
aw_dimension_name	Name of the target dimension within an analytic workspace. For information on naming requirements, see Table 23–18, " CREATE_AWDIMENSION Procedure Parameters".
dimension_load_spec	The name of a specification for loading an OLAP Catalog source dimension into a target dimension in an analytic workspace.
cube_load_spec	The name of a specification for loading an OLAP Catalog source cube into a target cube in an analytic workspace.
aggregation_spec	The name of a specification for creating the stored summaries for a target cube in an analytic workspace.
composite_spec	The name of a specification for defining composites and dimension order for a target cube in an analytic workspace.

Summary of DBMS_AWM Subprograms

Table 23–2 lists the DBMS_AWM subprograms in alphabetical order. Each subprogram is described in detail further in this chapter.

To see the DBMS_AWM subprograms listed by function, refer to "Understanding the DBMS_AWM Procedures" on page 1-6.

Table 23–2 DBMS_AWM Subprograms

Subprogram	Description
ADD_AWCOMP_SPEC_COMP_MEMBER Procedure on page 23-6	Adds a member to a composite in a composite specification.
ADD_AWCOMP_SPEC_MEMBER Procedure on page 23-8	Adds a member to a composite specification.
ADD_AWCUBEAGG_SPEC_LEVEL Procedure on page 23-9	Adds a level to an aggregation specification.
ADD_AWCUBEAGG_SPEC_MEASURE Procedure on page 23-10	Adds a measure to an aggregation specification.
ADD_AWCUBELOAD_SPEC_COMP Procedure on page 23-11	Adds a composite specification to a cube load specification.
ADD_AWCUBELOAD_SPEC_FILTER Procedure on page 23-12	Adds a WHERE clause to a cube load specification.
ADD_AWCUBELOAD_SPEC_MEASURE Procedure on page 23-13	Adds a measure to a cube load specification.
ADD_AWDIMLOAD_SPEC_FILTER Procedure on page 23-15	Adds a WHERE clause to a dimension load specification.
AGGREGATE_AWCUBE Procedure on page 23-16	Creates stored summaries for a cube in an analytic workspace.
CREATE_AWCOMP_SPEC Procedure on page 23-18	Creates a composite specification for a cube.
CREATE_AWCUBE Procedure on page 23-19	Creates containers within an analytic workspace to hold a cube defined in the OLAP Catalog.
CREATE_AWCUBE_ACCESS Procedure on page 23-22	Creates a script to enable relational access to a cube in an analytic workspace.

Table 23–2 (Cont.) DBMS_AWM Subprograms

Subprogram	Description
CREATE_AWCUBE_ACCESS_FULL Procedure on page 23-23	Enables relational access to a cube in an analytic workspace.
CREATE_AWCUBEAGG_SPEC Procedure on page 23-25	Creates an aggregation specification for a cube.
CREATE_AWCUBELOAD_SPEC Procedure on page 23-41	Creates a load specification for a cube.
CREATE_AWDIMENSION Procedure on page 23-28	Creates containers within an analytic workspace to hold a dimension defined in the OLAP Catalog.
CREATE_AWDIMENSION_ACCESS Procedure on page 23-31	Creates a script to enable relational access to a dimension in an analytic workspace.
CREATE_AWDIMENSION_ACCESS_FULL Procedure on page 23-32	Enables relational access to a dimension in an analytic workspace.
CREATE_AWDIMLOAD_SPEC Procedure on page 23-33	Creates a load specification for a dimension.
DELETE_AWCOMP_SPEC Procedure on page 23-35	Deletes a composite specification.
DELETE_AWCOMP_SPEC_MEMBER Procedure on page 23-36	Deletes a member of a composite specification.
DELETE_AWCUBE_ACCESS Procedure on page 23-36	Creates a script that deletes the enablement views and metadata for a cube in an analytic workspace.
DELETE_AWCUBE_ACCESS_ALL Procedure on page 23-38	Deletes the enablement views and metadata for a cube in an analytic workspace.
DELETE_AWCUBEAGG_SPEC Procedure on page 23-38	Deletes an aggregation specification.
DELETE_AWCUBEAGG_SPEC_LEVEL Procedure on page 23-39	Removes a level from an aggregation specification.
DELETE_AWCUBEAGG_SPEC_MEASURE Procedure on page 23-40	Removes a measure from an aggregation specification.
DELETE_AWCUBELOAD_SPEC Procedure on page 23-41	Deletes a cube load specification.
DELETE_AWCUBELOAD_SPEC_COMP Procedure on page 23-41	Removes a composite specification from a cube load specification.

Table 23–2 (Cont.) DBMS_AWM Subprograms

Subprogram	Description
DELETE_AWCUBELOAD_SPEC_FILTER Procedure on page 23-42	Removes a WHERE clause from a cube load specification.
DELETE_AWCUBELOAD_SPEC_MEASURE Procedure on page 23-43	Removes a measure from a cube load specification.
DELETE_AWDIMENSION_ACCESS Procedure on page 23-44	Creates a script that deletes the enablement views and metadata for a dimension in an analytic workspace.
DELETE_AWDIMENSION_ACCESS_ALL Procedure on page 23-45	Deletes the enablement views and metadata for a dimension in an analytic workspace.
DELETE_AWDIMLOAD_SPEC Procedure on page 23-46	Deletes a dimension load specification.
DELETE_AWDIMLOAD_SPEC_FILTER Procedure on page 23-46	Removes a WHERE clause from a dimension load specification.
REFRESH_AWCUBE Procedure on page 23-47	Loads the data and metadata of an OLAP Catalog source cube into a target cube in an analytic workspace.
REFRESH_AWCUBE_VIEW_NAME Procedure on page 23-49	Creates metadata in the analytic workspace to support user-defined enablement view names.
REFRESH_AWDIMENSION Procedure on page 23-50	Loads the data and metadata of an OLAP Catalog source dimension into a target dimension in an analytic workspace.
REFRESH_AWDIMENSION_VIEW_NAME Procedure on page 23-52	Creates metadata in the analytic workspace to support user-defined enablement view names.
SET_AWCOMP_SPEC_CUBE Procedure on page 23-53	Changes the cube associated with a composite specification.
SET_AWCOMP_SPEC_MEMBER_NAME Procedure on page 23-54	Renames a member of a composite specification.
SET_AWCOMP_SPEC_MEMBER_POS Procedure on page 23-55	Changes the position of a member in a composite specification.
SET_AWCOMP_SPEC_MEMBER_SEG Procedure on page 23-56	Changes the segment size associated with a member of a composite specification.
SET_AWCOMP_SPEC_NAME Procedure on page 23-58	Renames a composite specification.

Table 23–2 (Cont.) DBMS_AWM Subprograms

Subprogram	Description
SET_AWCUBE_VIEW_NAME Procedure on page 23-59	Renames the relational views of an analytic workspace cube.
SET_AWCUBEAGG_SPEC_AGGOP Procedure on page 23-60	Specifies an aggregation operator for aggregating measures along a dimension of a cube.
SET_AWCUBELOAD_SPEC_CUBE Procedure on page 23-61	Changes the cube associated with a cube load specification.
SET_AWCUBELOAD_SPEC_LOADTYPE Procedure on page 23-62	Changes the type of a cube load specification.
SET_AWCUBELOAD_SPEC_NAME Procedure on page 23-63	Renames of a cube load specification.
SET_AWCUBELOAD_SPEC_PARAMETER Procedure on page 23-64	Sets parameters for a cube load specification.
SET_AWDIMENSION_VIEW_NAME Procedure on page 23-65	Renames the relational views of an analytic workspace dimension.
SET_AWDIMLOAD_SPEC_DIMENSION Procedure on page 23-66	Changes the dimension associated with a dimension load specification.
SET_AWDIMLOAD_SPEC_LOADTYPE Procedure on page 23-66	Changes the type of a dimension load specification.
SET_AWDIMLOAD_SPEC_NAME Procedure on page 23-67	Renames a dimension load specification.
SET_AWDIMLOAD_SPEC_PARAMETER Procedure on page 23-68	Sets a parameter for a dimension load specification.

ADD_AWCOMP_SPEC_COMP_MEMBER Procedure

This procedure adds a member to a composite in a composite specification. The member may be a dimension or it may be a nested composite.

Composite members must be added in order. If you want to reorder the members, you must drop and re-create the composite. Call `DELETE_AWCOMP_SPEC_MEMBER` and `ADD_AWCOMP_SPEC_MEMBER`.

Syntax

```
ADD_AWCOMP_SPEC_COMP_MEMBER (
               composite_spec         IN   VARCHAR2,
```

```
cube_owner            IN   VARCHAR2,
cube_name             IN   VARCHAR2,
composite_name        IN   VARCHAR2,
nested_member_name    IN   VARCHAR2,
nested_member_type    IN   VARCHARs,
dimension_owner       IN   VARCHAR2 DEFAULT NULL,
dimension_name        IN   VARCHAR2 DEFAULT NULL);
```

Parameters

Table 23–3 ADD_AWCOMP_SPEC_COMP_MEMBER Procedure Parameters

Parameter	Description
composite_spec	Name of a composite specification for a cube.
cube_owner	Owner of the OLAP Catalog source cube.
cube_name	Name of the OLAP Catalog source cube.
composite_name	Name of a composite in the composite specification.
nested_member_name	Name of the member to add to the composite.
nested_member_type	Type of the new member. The type can be either 'DIMENSION' or 'COMPOSITE'.
dimension_owner	Owner of the OLAP Catalog source dimension to add to the composite. If the new member is a nested composite instead of a dimension, this parameter should be NULL (default).
dimension_name	Name of the OLAP Catalog source dimension to add to the composite. If the new member is a nested composite instead of a dimension, this parameter should be NULL (default).

Example

The following statements add a composite COMP1, consisting of the PRODUCT and GEOGRAPHY dimensions, to the composite specification AC_COMPSPEC.

```
execute DBMS_AWM.Create_AWComp_spec
          ('AC_COMPSPEC' ,'XADEMO' ,'ANALYTIC_CUBE');
execute DBMS_AWM.Add_AWComp_Spec_Member
          ('AC_COMPSPEC' ,'XADEMO' ,'ANALYTIC_CUBE' ,'COMP1'  ,'COMPOSITE');
execute DBMS_AWM.Add_AWComp_Spec_Comp_Member
          ('AC_COMPSPEC','XADEMO','ANALYTIC_CUBE', 'COMP1','PROD_COMP',
           'DIMENSION','XADEMO','PRODUCT');
execute DBMS_AWM.Add_AWComp_Spec_Comp_Member
          ('AC_COMPSPEC','XADEMO','ANALYTIC_CUBE', 'COMP1','GEOG_COMP',
```

```
                'DIMENSION','XADEMO','GEOGRAPHY');
```

See Also

- "Managing Sparse Data and Optimizing the Workspace Cube" on page 1-16
- DELETE_AWCOMP_SPEC_MEMBER Procedure on page 23-36
- ADD_AWCOMP_SPEC_MEMBER Procedure on page 23-8
- CREATE_AWCOMP_SPEC Procedure on page 23-18

ADD_AWCOMP_SPEC_MEMBER Procedure

This procedure adds a member to a composite specification. The members of a composite specification are composites and dimensions.

Syntax

```
ADD_AWCOMP_SPEC_MEMBER (
               composite_spec      IN   VARCHAR2,
               cube_owner          IN   VARCHAR2,
               cube_name           IN   VARCHAR2,
               member_name         IN   VARCHAR2,
               member_type         IN   VARCHAR2,
               dimension_owner     IN   VARCHAR2 DEFAULT NULL,
               diimension_name     IN   VARCHAR2 DEFAULT NULL);
```

Parameters

Table 23–4 ADD_AWCOMP_SPEC_MEMBER Procedure Parameters

Parameter	Description
`composite_spec`	Name of a composite specification for a cube.
`cube_owner`	Owner of the OLAP Catalog source cube.
`cube_name`	Name of the OLAP Catalog source cube.
`member_name`	Name of the member of the composite specification.
`member_type`	Type of the member. The type can be either `'DIMENSION'` or `'COMPOSITE'`.
`dimension_owner`	Owner of the OLAP Catalog source dimension to add to the composite specification. If the new member is a composite instead of a dimension, this parameter should be NULL (default).

Table 23–4 (Cont.) ADD_AWCOMP_SPEC_MEMBER Procedure Parameters

Parameter	Description
dimension_name	Name of the OLAP Catalog source dimension to add to the composite specification. If the new member is a composite instead of a dimension, this parameter should be NULL (default).

Example

The following statements add the Time dimension and a composite called COMP1 to the composite specification AC_COMPSPEC.

```
execute DBMS_AWM.Add_AWComp_Spec_Member
          ('AC_COMPSPEC' ,'XADEMO' ,'ANALYTIC_CUBE' ,'TIMECOMP_MEMBER' ,
           'DIMENSION' ,'XADEMO' ,'TIME');
execute DBMS_AWM.Add_AWComp_Spec_Member
          ('AC_COMPSPEC' ,'XADEMO' ,'ANALYTIC_CUBE' ,'COMP1'  ,'COMPOSITE');
```

See Also

- "Managing Sparse Data and Optimizing the Workspace Cube" on page 1-16
- CREATE_AWCOMP_SPEC Procedure on page 23-18

ADD_AWCUBEAGG_SPEC_LEVEL Procedure

This procedure adds a level to an aggregation specification.

Syntax

```
ADD_AWCUBEAGG_SPEC_LEVEL (
               aggregation_spec      IN   VARCHAR2,
               aw_owner              IN   VARCHAR2,
               aw_name               IN   VARCHAR2,
               aw_cube_name          IN   VARCHAR2,
               aw_dimension_name     IN   VARCHAR2,
               aw_level_name         IN   VARCHAR2);
```

Parameters

Table 23–5 ADD_AWCUBEAGG_SPEC_LEVEL Procedure Parameters

Parameter	Description
`aggregation_spec`	Name of an aggregation specification for a cube in an analytic workspace.
`aw_owner`	Owner of the analytic workspace.
`aw_name`	Name of the analytic workspace.
`aw_cube_name`	Name of the cube within the analytic workspace.
`aw_dimension_name`	Name of a dimension of the cube.
`aw_level_name`	Name of a level of the dimension.

Example

The following statements add two levels of Product, one level of Channel, and one level of Time to the aggregation specification `AC_AGGSPEC`.

```
execute dbms_awm.add_awcubeagg_spec_level
          ('AC_AGGSPEC', 'MYSCHEMA', 'MYAW', 'AW_ANACUBE', 'AW_PROD', 'L3')
execute dbms_awm.add_awcubeagg_spec_level
          ('AC_AGGSPEC', 'MYSCHEMA', 'MYAW', 'AW_ANACUBE', 'AW_PROD', 'L2')
execute dbms_awm.add_awcubeagg_spec_level
          ('AC_AGGSPEC','MYSCHEMA','MYAW','AW_ANACUBE','AW_CHAN','STANDARD_2')
execute dbms_awm.add_awcubeagg_spec_level
          ('AC_AGGSPEC', 'MYSCHEMA', 'MYAW', 'AW_ANACUBE', 'AW_TIME', 'L2')
```

See Also

- "Aggregating the Data in an Analytic Workspace" on page 1-18
- CREATE_AWCUBEAGG_SPEC Procedure on page 23-25

ADD_AWCUBEAGG_SPEC_MEASURE Procedure

This procedure adds a measure to an aggregation specification.

Syntax

```
ADD_AWCUBEAGG_SPEC_MEASURE (
               aggregation_spec     IN   VARCHAR2,
               aw_owner             IN   VARCHAR2,
               aw_name              IN   VARCHAR2,
```

```
               aw_cube_name         IN   VARCHAR2,
               aw_measure_name      IN   VARCHAR2);
```

Parameters

Table 23–6 ADD_AWCUBEAGG_SPEC_MEASURE Procedure Parameters

Parameter	Description
aggregation_spec	Name of an aggregation specification for a cube in an analytic workspace.
aw_owner	Owner of the analytic workspace.
aw_name	Name of the analytic workspace.
aw_cube_name	Name of the cube within the analytic workspace.
aw_measure_name	Name of one of the measures of the cube.

Example

The following statements add the Costs and Quota measures to the aggregation specification for the cube AW_ANACUBE in the analytic workspace MYAW.

```
execute dbms_awm.add_awcubeagg_spec_measure
          ('AC_AGGSPEC', 'MYSCHEMA', 'MYAW', 'AW_ANACUBE', 'XXF.COSTS')
execute dbms_awm.add_awcubeagg_spec_measure
          ('AC_AGGSPEC', 'MYSCHEMA', 'MYAW', 'AW_ANACUBE', 'XXF.QUOTA')
```

See Also

- "Aggregating the Data in an Analytic Workspace" on page 1-18
- CREATE_AWCUBEAGG_SPEC Procedure on page 23-25

ADD_AWCUBELOAD_SPEC_COMP Procedure

This procedure adds a composite specification to a cube load specification.

Syntax

```
ADD_AWCUBELOAD_SPEC_COMP (
               cube_load_spec     IN   VARCHAR2,
               cube_owner         IN   VARCHAR2,
               cube_name          IN   VARCHAR2,
               composite_spec     IN   VARCHAR2);
```

Parameters

Table 23–7 ADD_AWCUBELOAD_SPEC_COMP Procedure Parameters

Parameter	Description
cube_load_spec	Name of a cube load specification.
cube_owner	Owner of the OLAP Catalog source cube.
cube_name	Name of the OLAP Catalog source cube.
composite_spec	Name of the composite specification to add to the cube load specification.

Example

The following statement adds the composite specification AC_COMPSPEC to the cube load specification AC_CUBELOADSPEC.

```
execute DBMS_AWM.add_AWCubeLoad_Spec_Comp
     ('AC_CUBELOADSPEC' ,'XADEMO', 'ANALYTIC_CUBE', 'AC_COMPSPEC');
```

See Also

- "Creating and Populating Workspace Cubes" on page 1-4
- CREATE_AWCUBELOAD_SPEC Procedure on page 23-26
- CREATE_AWCOMP_SPEC Procedure on page 23-18

ADD_AWCUBELOAD_SPEC_FILTER Procedure

This procedure adds a filter condition to a cube load specification. The filter is a SQL WHERE clause that will be used in the query against the source fact table.

Syntax

```
ADD_AWCUBELOAD_SPEC_FILTER (
               cube_load_spec      IN   VARCHAR2,
               cube_owner          IN   VARCHAR2,
               cube_name           IN   VARCHAR2,
               fact_table_owner    IN   VARCHAR2,
               fact_table_name     IN   VARCHAR2,
               where_clause        IN   VARCHAR2);
```

Parameters

Table 23–8 ADD_AWCUBELOAD_SPEC_FILTER Procedure Parameters

Parameter	Description
`cube_load_spec`	Name of a cube load specification.
`cube_owner`	Owner of the OLAP Catalog source cube.
`cube_name`	Name of the OLAP Catalog source cube.
`fact_table_owner`	Owner of the fact table that is mapped to the OLAP Catalog source cube.
`fact_table_name`	Name of the fact table that is mapped to the OLAP Catalog source cube
`where_clause`	A SQL `WHERE` clause that specifies which rows to load from the fact table.

Example

The following statements create a cube load specification called `AC_CUBELOADSPEC2`. When the target cube in the analytic workspace is refreshed with this specification, only sales figures less than 25 will be loaded.

```
execute dbms_awm.create_awcubeload_spec
          ('AC_CUBELOADSPEC2', 'XADEMO', 'ANALYTIC_CUBE', 'LOAD_DATA');
execute dbms_awm.add_awcubeload_spec_measure
          ('AC_CUBELOADSPEC2', 'XADEMO', 'ANALYTIC_CUBE', 'F.SALES',
           'AW_SALES', 'Sales');
execute dbms_awm.add_awcubeload_spec_filter
          ('AC_CUBELOADSPEC2', 'XADEMO', 'ANALYTIC_CUBE',
           'XADEMO', 'XADEMO_ANALYTIC_FACTS', '''SALES'' < 25');
```

See Also

- "Creating and Populating Workspace Cubes" on page 1-4
- CREATE_AWCUBELOAD_SPEC Procedure on page 23-26

ADD_AWCUBELOAD_SPEC_MEASURE Procedure

This procedure adds a measure to a cube load specification.

If you add one or more measures to a cube load specification, only those measures will be loaded. If you do not add measures to the cube load specification, then all the cube's measures will be loaded.

This procedure allows you to specify the measure name, display name, and description in the analytic workspace. If you do not specify the target names, or if you do not call this procedure at all, the source names from the OLAP Catalog are used.

Syntax

```
ADD_AWCUBELOAD_SPEC_MEASURE (
              cube_load_spec            IN   VARCHAR2,
              cube_owner                IN   VARCHAR2,
              cube_name                 IN   VARCHAR2,
              measure_name              IN   VARCHAR2,
              aw_measure_name           IN   VARCHAR2 DEFAULT NULL,
              aw_measure_display_name   IN   VARCHAR2 DEFAULT NULL,
              aw_measure_description    IN   VARCHAR2 DEFAULT NULL);
```

Parameters

Table 23–9 ADD_AWCUBELOAD_SPEC_MEASURE Procedure Parameters

Parameter	Description
cube_load_spec	Name of a cube load specification.
cube_owner	Owner of the OLAP Catalog source cube.
cube_name	Name of the OLAP Catalog source cube.
measure_name	Name of the OLAP Catalog source measure.
aw_measure_name	Name of the target measure in the analytic workspace. If you do not specify a name, the measure name from the OLAP Catalog is used.
aw_measure_display_name	Display name for the target measure in the analytic workspace. If you do not specify a display name, the display name for the measure in the OLAP Catalog is used.
aw_measure_description	Description for the target measure in the analytic workspace. If you do not specify a description, the description for the measure in the OLAP Catalog is used.

Example

The following statements create a cube load specification called AC_CUBELOADSPEC2. When the target cube in the analytic workspace is refreshed with this specification, only the sales measure will be loaded.

The target sales measure will have the logical name AW_SALES, and its description will be 'Sales'.

```
execute dbms_awm.create_awcubeload_spec
          ('AC_CUBELOADSPEC2', 'XADEMO', 'ANALYTIC_CUBE', 'LOAD_DATA');
execute dbms_awm.add_awcubeload_spec_measure
          ('AC_CUBELOADSPEC2', 'XADEMO', 'ANALYTIC_CUBE', 'F.SALES',
           'AW_SALES', 'Sales');
```

See Also

- CREATE_AWCUBELOAD_SPEC Procedure on page 23-26
- REFRESH_AWCUBE Procedure on page 23-47

ADD_AWDIMLOAD_SPEC_FILTER Procedure

This procedure adds a filter condition to a dimension load specification. The filter is a SQL WHERE clause that will be used in the query against the source dimension tables.

Syntax

```
ADD_AWDIMLOAD_SPEC_FILTER (
               dimension_load_spec       IN   VARCHAR2,
               dimension_owner           IN   VARCHAR2,
               dimension_name            IN   VARCHAR2,
               dimension_table_owner     IN   VARCHAR2,
               dimension_table_name      IN   VARCHAR2,
               where_clause              IN   VARCHAR2);
```

Parameters

Table 23–10 ADD_AWDIMLOAD_SPEC_FILTER Procedure Parameters

Parameter	Description
dimension_load_spec	Name of a dimension load specification.
dimension_owner	Owner of the OLAP Catalog source dimension.
dimension_name	Name of the OLAP Catalog source dimension.
dimension_table_owner	Owner of the dimension table that is mapped to the OLAP Catalog source dimension.
dimension_table_name	Name of the dimension table that is mapped to the OLAP Catalog source dimension.

Table 23–10 (Cont.) ADD_AWDIMLOAD_SPEC_FILTER Procedure Parameters

Parameter	Description
where_clause	A SQL WHERE clause that specifies which rows to load from the dimension table into an analytic workspace.

Example

The following statements create a load specification for the CHANNEL dimension in XADEMO. When the target dimension is refreshed with this specification, only the member DIRECT will be loaded.

```
execute dbms_awm.create_awdimload_spec
          ('CHAN_DIMLOADSPEC', 'XADEMO', 'CHANNEL', 'FULL_LOAD');
execute dbms_awm.add_awdimload_spec_filter
          ('CHAN_DIMLOADSPEC', 'XADEMO', 'CHANNEL', 'XADEMO',
          'XADEMO_CHANNEL', '''CHAN_STD_CHANNEL'' = ''DIRECT''' );
```

See Also

- "Creating and Populating Workspace Dimensions" on page 1-4
- CREATE_AWDIMLOAD_SPEC Procedure on page 23-33

AGGREGATE_AWCUBE Procedure

This procedure uses an aggregation specification to precompute and store aggregate data for a cube in an analytic workspace.

The REFRESH_AWCUBE procedure loads detail data and sets up the internal workspace structures that support dynamic aggregation. If you want to precompute and store summarized data for the cube, you must use the AGGREGATE_AWCUBE procedure.

You must rerun AGGREGATE_AWCUBE after every refresh to ensure that the stored summaries are consistent with the data.

AGGREGATE_AWCUBE executes an OLAP DML UPDATE command to save the changes in the analytic workspace. AGGREGATE_AWCUBE *does not* execute a SQL COMMIT.

Syntax

```
AGGREGATE_AWCUBE (
               aw_owner              IN    VARCHAR2,
               aw_name               IN    VARCHAR2,
```

```
                    aw_cube_name         IN    VARCHAR2,
                    aggregation_spec     IN    VARCHAR2);
```

Parameters

Table 23–11 AGGREGATE_AWCUBE Procedure Parameters

Parameter	Description
aw_owner	Owner of the analytic workspace.
aw_name	Name of the analytic workspace.
aw_cube_name	Name of the cube within the analytic workspace.
aggregation_spec	Name of an aggregation specification for the cube.

Example

The following statements create an aggregation plan AGG1 for the target cube AC2 in the analytic workspace MYSCHEMA.MYAW. The target cube was created from the source cube XADEMO.ANALYTIC_CUBE.

```
----  Create agg plan for analytic cube --------------------------------------
----   with levels 2 and 3 of product, standard_2 of channel, and 2 of time ----
----   with measures costs and quota -----------------------------------------

execute dbms_awm.create_awcubeagg_spec
          ('AGG1', 'MYSCHEMA', 'MYAW', 'AC2')
execute dbms_awm.add_awcubeagg_spec_level
          ('AGG1', 'MYSCHEMA', 'MYAW', 'AC2', 'PRODUCT', 'L3')
execute dbms_awm.add_awcubeagg_spec_level
          ('AGG1', 'MYSCHEMA', 'MYAW', 'AC2', 'PRODUCT', 'L2')
execute dbms_awm.add_awcubeagg_spec_level
          ('AGG1', 'MYSCHEMA', 'MYAW', 'AC2', 'CHANNEL', 'STANDARD_2')
execute dbms_awm.add_awcubeagg_spec_level
          ('AGG1', 'MYSCHEMA', 'MYAW', 'AC2', 'TIME', 'L2')
execute dbms_awm.add_awcubeagg_spec_measure
          ('AGG1', 'MYSCHEMA', 'MYAW', 'AC2', 'XXF.COSTS')
execute dbms_awm.add_awcubeagg_spec_measure
          ('AGG1', 'MYSCHEMA', 'MYAW', 'AC2', 'XXF.QUOTA')
execute dbms_awm.aggregate_awcube('MYSCHEMA', 'MYAW', 'AC2', 'AGG1')
```

See Also

- "Aggregating the Data in an Analytic Workspace" on page 1-18

- "CREATE_AWCUBEAGG_SPEC Procedure" on page 23-25

CREATE_AWCOMP_SPEC Procedure

This procedure creates a **composite specification** for an OLAP Catalog source cube. The composite specification determines how sparse data will be stored in the target cube in an analytic workspace. It also determines the dimension order, which affects the efficiency of data loads and queries.

A **composite** is a list of dimension value combinations that provides an index into one or more sparse measures. Composites are named objects within an analytic workspace. Composites are defined and maintained with OLAP DML commands.

Members of a composite specification are composites (whose members are dimensions) and individual dimensions.

Syntax

```
CREATE_AWCOMP_SPEC (
            composite_spec     IN   VARCHAR2,
            cube_owner         IN   VARCHAR2,
            cube_name          IN   VARCHAR2);
```

Parameters

Table 23–12 CREATE_AWCOMP_SPEC Procedure Parameters

Parameter	Description
composite_spec	Name of a composite specification for a cube.
cube_owner	Owner of the OLAP Catalog source cube.
cube_name	Name of the OLAP Catalog source cube.

Note

You can use the following procedures to modify an existing composite specification:

- SET_AWCOMP_SPEC_CUBE Procedure
- SET_AWCOMP_SPEC_MEMBER_NAME Procedure
- SET_AWCOMP_SPEC_MEMBER_POS Procedure
- SET_AWCOMP_SPEC_MEMBER_SEG Procedure
- SET_AWCOMP_SPEC_NAME Procedure

Example

The following statements create a composite specification for the ANALYTIC_CUBE in XADEMO. It consists of the Time dimension followed by a composite called COMP1.

```
execute DBMS_AWM.Create_AWComp_spec
          ('AC_COMPSPEC' ,'XADEMO' ,'ANALYTIC_CUBE');
execute DBMS_AWM.Add_AWComp_Spec_Member
          ('AC_COMPSPEC' ,'XADEMO' ,'ANALYTIC_CUBE' ,'TIMECOMP_MEMBER' ,
           'DIMENSION' ,'XADEMO' ,'TIME');
execute DBMS_AWM.Add_AWComp_Spec_Member
          ('AC_COMPSPEC' ,'XADEMO' ,'ANALYTIC_CUBE' ,'COMP1'  ,'COMPOSITE');
```

See Also

- "Managing Sparse Data and Optimizing the Workspace Cube" on page 1-16
- ADD_AWCOMP_SPEC_MEMBER Procedure on page 23-8
- ADD_AWCOMP_SPEC_COMP_MEMBER Procedure on page 23-6
- ADD_AWCUBELOAD_SPEC_COMP Procedure on page 23-11
- DEFINE COMPOSITE in the *Oracle OLAP DML Reference*

CREATE_AWCUBE Procedure

This procedure creates the multidimensional framework within an analytic workspace to hold a relational cube.

The relational cube, consisting of a star schema and OLAP Catalog metadata, is the source for the target multidimensional cube in the analytic workspace. Data and metadata are loaded from the source cube to the target cube by the REFRESH_AWCUBE procedure.

CREATE_AWCUBE executes an OLAP DML UPDATE command to save the changes in the analytic workspace. CREATE_AWCUBE *does not* execute a SQL COMMIT.

The multidimensional framework for the cube is in database standard form, ensuring its compatibility with the OLAP API enablers and with other OLAP administrative tools and utilities.

Note: Before executing CREATE_AWCUBE to create a new workspace cube, you must execute CREATE_AWDIMENSION for each of the cube's dimensions.

Syntax

```
CREATE_AWCUBE (
              cube_owner       IN   VARCHAR2,
              cube_name        IN   VARCHAR2,
              aw_owner         IN   VARCHAR2,
              aw_name          IN   VARCHAR2,
              aw_cube_name     IN   VARCHAR2   DEFAULT NULL);
```

Parameters

Table 23–13 CREATE_AWCUBE Procedure Parameters

Parameter	Description
cube_owner	Owner of the OLAP Catalog source cube.
cube_name	Name of the OLAP Catalog source cube.
aw_owner	Owner of the analytic workspace.
aw_name	Name of the analytic workspace.
aw_cube_name	Name for the target cube within the analytic workspace. If you specify a name for the cube in the analytic workspace, the name must conform to general object naming conventions for SQL, and it must be unique within the schema that owns the analytic workspace. To test uniqueness, use a statement like the following. `select owner, cube_name from all_olap2_cubes union all select aw_owner, aw_logical_name from all_olap2_aw_cubes;` Within the analytic workspace, you can generally reference the cube by its simple target cube name. However, database standard form also supports a full name for logical objects. For cubes, the full name is: *aw_owner.aw_cube_name*.CUBE

Example

The following statements create the structures for the XADEMO.ANALYTIC_CUBE in the analytic workspace MYSCHEMA.MYAW. The name of the cube in the workspace is AW_ANACUBE.

```
--- Create the dimensions in the analytic workspace ----

execute dbms_awm.create_awdimension
```

```
          ('XADEMO','CHANNEL','MYSCHEMA', 'MYAW', 'AW_CHAN');
execute dbms_awm.create_awdimension
          ('XADEMO','GEOGRAPHY','MYSCHEMA','MYAW', 'AW_GEOG');
execute dbms_awm.create_awdimension
          ('XADEMO','PRODUCT','MYSCHEMA', 'MYAW', 'AW_PROD');
execute dbms_awm.create_awdimension
          ('XADEMO','TIME','MYSCHEMA', 'MYAW', 'AW_TIME');

--- Create the cube in the analytic workspace ----

execute dbms_awm.create_awcube
          ('XADEMO', 'ANALYTIC_CUBE','MYSCHEMA', 'MYAW','AW_ANACUBE');
```

You can use statements like the following to verify that the cube has been created in the analytic workspace.

```
--- View the cube in the analytic workspace ----

execute dbms_aw.execute
          ('aw attach MYSCHEMA.MYAW');
execute dbms_aw.execute
          ('limit name to obj(property''AW$ROLE'') eq ''CUBEDEF''');
execute dbms_aw.execute
          ('report w 40 name');

NAME
----------------------------------------
AW_ANACUBE
```

Alternatively, you can query the Active Catalog to verify that the cube has been created.

```
select * from all_olap2_aw_cubes
           where owner in 'myschema' and
                 aw_name in 'myaw' and
                 aw_logical_name in 'aw_anacube';
```

See Also

- "Creating and Refreshing a Workspace Cube" on page 1-13
- CREATE_AWDIMENSION Procedure on page 23-28
- REFRESH_AWCUBE Procedure on page 23-47
- CREATE_AWCUBE_ACCESS Procedure on page 23-22

- Chapter 3, "Active Catalog Views"

CREATE_AWCUBE_ACCESS Procedure

This procedure generates a script that creates relational fact views of a cube in an analytic workspace. The views are in the embedded total format required by the OLAP API.

The script can optionally generate OLAP Catalog metadata that maps to the views of the workspace cube. This metadata is required for the OLAP API.

Both dimension views and fact views are required for relational access to the workspace cube. Use the `CREATE_AWDIMENSION_ACCESS` procedure to generate the scripts that create the dimension views.

To accomplish the cube enablement process in a single step, use the `CREATE_AWCUBE_ACCESS_FULL` procedure. This procedure both creates and runs the enablement script.

Syntax

```
CREATE_AWCUBE_ACCESS (
               aw_owner              IN   VARCHAR2,
               aw_name               IN   VARCHAR2,
               aw_cube_name          IN   VARCHAR2,
               access_type           IN   VARCHAR2,
               script_directory      IN   VARCHAR2,
               script_name           IN   VARCHAR2,
               open_mode             IN   VARCHAR2);
```

Parameters

Table 23–14 CREATE_AWCUBE_ACCESS Procedure Parameters

Parameter	Description
`aw_owner`	Owner of the analytic workspace.
`aw_name`	Name of the analytic workspace.
`aw_cube_name`	Name of the cube in the analytic workspace.
`access_type`	Controls whether or not the script generates OLAP Catalog metadata for the views. Specify one of the following values: ■ `'SQL'` does not generate metadata. ■ `'OLAP'` generates metadata

Table 23–14 (Cont.) CREATE_AWCUBE_ACCESS Procedure Parameters

Parameter	Description
`script_directory`	The directory that will contain the script. This may be either a directory object or a path set by the `UTL_FILE_DIR` parameter.
`script_name`	Name of the script file.
`open_mode`	One of the following modes for opening the script file: ■ `'W'` overwrites any existing contents of the script file ■ `'A'` appends the new script to the existing contents of the script file.

Example

The following statement creates an enablement script called `aw_anacube_enable.sql` in the `/dat1/scripts` directory. You can run the script to create fact views of the `AW_ANACUBE` cube in workspace `XADEMO.MYAW`. The script will also generate an OLAP Catalog cube called `AW_ANACUBE` that maps to the views.

```
execute dbms_awm.create_awcube_access
         ('XADEMO', 'MYAW', 'AW_ANACUBE', 'OLAP',
          '/dat1/scripts/', 'aw_anacube_enable.sql', 'w');
```

See Also

- "Creating Relational Access to the Workspace Cube" on page 1-23
- "CREATE_AWCUBE_ACCESS_FULL Procedure" on page 23-23
- "DELETE_AWCUBE_ACCESS Procedure" on page 23-36
- "SET_AWCUBE_VIEW_NAME Procedure" on page 23-59
- "CREATE_AWDIMENSION_ACCESS Procedure" on page 23-31
- "REFRESH_AWCUBE Procedure" on page 23-47
- Chapter 26, "OLAP_TABLE"

CREATE_AWCUBE_ACCESS_FULL Procedure

This procedure accomplishes the entire process of enabling a workspace cube for access by the OLAP API. Like `CREATE_AWCUBE_ACCESS` it produces an enablement script. However it does not write the script to a file. Instead it writes the script to temporary memory and runs the script.

The resulting views and metadata are identical to those created by the enablement scripts produced by CREATE_AWCUBE_ACCESS.

Syntax

```
CREATE_AWCUBE_ACCESS_FULL (
              run_id          IN   NUMBER,
              aw_owner        IN   VARCHAR2,
              aw_name         IN   VARCHAR2,
              aw_cube_name    IN   VARCHAR2,
              access_type     IN   VARCHAR2);
```

Parameters

Table 23–15 CREATE_AWCUBE_ACCESS_FULL Procedure Parameters

Parameter	Description
run_id	An assigned slot in a global temporary table for holding the record associated with this operation. In most cases, simply specify "1".
aw_owner	Owner of the analytic workspace.
aw_name	Name of the analytic workspace.
aw_cube_name	Name of the cube in the analytic workspace.
access_type	Controls whether or not to generate OLAP Catalog metadata in addition to the enablement views. Specify one of the following values: ■ 'SQL' does not generate metadata ■ 'OLAP' generates metadata

See Also

- "Creating Relational Access to the Workspace Cube" on page 1-23
- "CREATE_AWCUBE_ACCESS Procedure" on page 23-22
- "REFRESH_AWCUBE Procedure" on page 23-47
- Chapter 26, "OLAP_TABLE"

CREATE_AWCUBEAGG_SPEC Procedure

This procedure creates an **aggregation specification** for an OLAP Catalog cube. The aggregation specification determines the summary data that will be stored with the target cube in the analytic workspace.

The aggregation specification determines which of the cube's levels will be pre-summarized. You can aggregate all of the cube's measures to these levels, or you can choose individual measures. All of the measures are aggregated to the same levels.

Any levels that are not pre-aggregated will be aggregated dynamically as they are queried. Determining which data to preaggregate will involve an evaluation of storage and memory constraints and typical client queries. If you do not provide an aggregation specification, no summaries will be stored and all aggregation will be performed on demand.

An aggregation specification uses the aggregation subsystem of the OLAP DML. This includes the AGGREGATE command, aggregation maps, and related functionality.

Syntax

```
CREATE_AWCUBEAGG_SPEC (
               aggregation_spec     IN   VARCHAR2,
               aw_owner             IN   VARCHAR2,
               aw_name              IN   VARCHAR2,
               aw_cube_name          IN   VARCHAR2);
```

Parameters

Table 23–16 CREATE_AWCUBEAGG_SPEC Procedure Parameters

Parameter	Description
aggregation_spec	Name of an aggregation specification for a cube in an analytic workspace.
aw_owner	Owner of the analytic workspace.
aw_name	Name of the analytic workspace.
aw_cube_name	Name of the cube in the analytic workspace.

Note

You can use the following procedure to modify an existing aggregation specification: SET_AWCUBEAGG_SPEC_AGGOP Procedure

Example

The following statements create an aggregation specification for the target cube AW_ANACUBE in the analytic workspace MYSCHEMA.MYAW. It specifies that the Costs and Sales measures should include stored totals for the third level of PRODUCT, the STANDARD_2 level of CHANNEL, and the second level of TIME.

```
execute dbms_awm.create_awcubeagg_spec
          ('AC_AGGSPEC', 'MYSCHEMA', 'MYAW', 'AW_ANACUBE');
execute dbms_awm.add_awcubeagg_spec_level
          ('AC_AGGSPEC', 'MYSCHEMA', 'MYAW', 'AW_ANACUBE', 'AW_PROD', 'L3');
execute dbms_awm.add_awcubeagg_spec_level
          ('AC_AGGSPEC', 'MYSCHEMA', 'MYAW', 'AW_ANACUBE', 'AW_CHAN',
           'STANDARD_2');
execute dbms_awm.add_awcubeagg_spec_level
          ('AC_AGGSPEC', 'MYSCHEMA', 'MYAW', 'AW_ANACUBE', 'AW_TIME', 'L2');
execute dbms_awm.add_awcubeagg_spec_measure
          ('AC_AGGSPEC', 'MYSCHEMA', 'MYAW', 'AW_ANACUBE', 'XXF.COSTS');
execute dbms_awm.add_awcubeagg_spec_measure
          ('AC_AGGSPEC', 'MYSCHEMA', 'MYAW', 'AW_ANACUBE', 'XXF.SALES');
```

See Also

- "Aggregating the Data in an Analytic Workspace" on page 1-18
- ADD_AWCUBEAGG_SPEC_LEVEL Procedure on page 23-9
- ADD_AWCUBEAGG_SPEC_MEASURE Procedure on page 23-10
- "AGGREGATE_AWCUBE Procedure" on page 23-16
- AGGREGATE Command in the *Oracle OLAP DML Reference*

CREATE_AWCUBELOAD_SPEC Procedure

This procedure creates a **load specification** for an OLAP Catalog cube. The load specification determines how the cube's data will be loaded from the relational fact table into an analytic workspace by the REFRESH_AWCUBE procedure.

A cube load specification defines a load type, which indicates whether the data or only the load instructions should be loaded into the analytic workspace. The load instructions are OLAP DML programs. If you choose to load only the instructions, you can run these programs to perform the data load at a later time.

A separate specification created by `CREATE_AWCOMP_SPEC` can be associated with a cube load specification. This specification specifies dimension order and determines how sparse data will be stored within the analytic workspace.

Syntax

```
CREATE_AWCUBELOAD_SPEC (
              cube_load_spec     IN   VARCHAR2,
              cube_owner         IN   VARCHAR2,
              cube_name          IN   VARCHAR2,
              load_type          IN   VARCHAR2);
```

Parameters

Table 23–17 CREATE_AWCUBELOAD_SPEC Procedure Parameters

Parameter	Description
`cube_load_spec`	Name of a cube load specification.
`cube_owner`	Owner of the OLAP Catalog source cube.
`cube_name`	Name of the OLAP Catalog source cube.
`load_type`	`'LOAD_DATA'` -- Load the data from the fact table into the analytic workspace target cube. `'LOAD_PROGRAM'` -- Create the load programs in the analytic workspace but do not execute them. You can run the program manually to load the data. Cube load program names are stored in the `AW$LOADPRGS` property of the standard form cube in the analytic workspace. `->show obj(property 'aw$loadprgs' 'my_awcube_name')`

Note

You can use the following procedures to modify an existing cube load specification:

- SET_AWCUBELOAD_SPEC_CUBE Procedure
- SET_AWCUBELOAD_SPEC_LOADTYPE Procedure
- SET_AWCUBELOAD_SPEC_NAME Procedure
- SET_AWCUBELOAD_SPEC_PARAMETER Procedure

Example

The following statement creates a cube load specification for the source cube XADEMO.ANALYTIC_CUBE. The load specification is used to refresh the target cube AW_ANACUBE in MYSCHEMA.MYAW.

```
execute dbms_awm.create_awcubeload_spec
          ('AC_CUBELOADSPEC', 'XADEMO', 'ANALYTIC_CUBE', 'LOAD_DATA');
execute dbms_awm.refresh_awcube
          ('MYSCHEMA', 'MYAW', 'AW_ANACUBE', 'AC_CUBELOADSPEC');
```

See Also

- "Creating and Populating Workspace Cubes" on page 1-4
- ADD_AWCUBELOAD_SPEC_COMP Procedure on page 23-11
- REFRESH_AWCUBE Procedure on page 23-47

CREATE_AWDIMENSION Procedure

This procedure creates the multidimensional framework within an analytic workspace to hold a relational dimension.

The relational dimension, consisting of dimension lookup tables and OLAP Catalog metadata, is the source for the target dimension in the analytic workspace. Data and metadata are loaded from the source dimension to the target dimension by the REFRESH_AWDIMENSION procedure.

CREATE_AWDIMENSION executes an OLAP DML UPDATE command to save the changes in the analytic workspace. CREATE_AWDIMENSION *does not* execute a SQL COMMIT.

The multidimensional framework for the dimension is in database standard form, ensuring its compatibility with the OLAP API enablers and with other OLAP administrative tools and utilities.

Note: Before executing CREATE_AWCUBE to create a new workspace cube, you must execute CREATE_AWDIMENSION for each of the cube's dimensions.

The workspace must already exist before the first call to CREATE_AWDIMENSION.

Syntax

```
CREATE_AWDIMENSION (
              dimension_owner       IN   VARCHAR2,
              dimension_name        IN   VARCHAR2,
              aw_owner              IN   VARCHAR2,
              aw_name               IN   VARCHAR2,
              aw_dimension_name     IN   VARCHAR2   DEFAULT NULL),
```

Parameters

Table 23–18 CREATE_AWDIMENSION Procedure Parameters

Parameter	Description
`dimension_owner`	Owner of the OLAP Catalog source dimension.
`dimension_name`	Name of the OLAP Catalog source dimension.
`aw_owner`	Owner of the analytic workspace.
`aw_name`	Name of the analytic workspace.

`aw_dimension_name`	Name for the target dimension within the analytic workspace.
	If you specify a name for the dimension in the analytic workspace, the name must conform to general object naming conventions for SQL, and it must be unique within the schema that owns the analytic workspace. To test uniqueness, use a statement like the following.
	`select owner, dimension_name`
	`       from all_olap2_dimensions`
	`union all`
	`select aw_owner, aw_logical_name`
	`       from all_olap2_aw_dimensions;`
	Within the analytic workspace, you can generally reference the dimension by its simple target dimension name. However, database standard form also supports a full name for logical objects. For dimensions, the full name is:
	`aw_owner.aw_dimension_name``.DIMENSION`

Example

The following statements create analytic workspace dimensions for `CHANNEL`, `GEOGRAPHY`, `PRODUCT`, `TIME`, and `DIVISION` in the workspace `MYAW` in the `XADEMO` schema.

```
execute dbms_awm.create_awdimension
```

```
          ('XADEMO','CHANNEL','MYSCHEMA', 'MYAW', 'AW_CHAN');
execute dbms_awm.create_awdimension
          ('XADEMO','GEOGRAPHY','MYSCHEMA','MYAW', 'AW_GEOG');
execute dbms_awm.create_awdimension
          ('XADEMO','PRODUCT','MYSCHEMA', 'MYAW', 'AW_PROD');
execute dbms_awm.create_awdimension
          ('XADEMO','TIME','MYSCHEMA', 'MYAW', 'AW_TIME');
execute dbms_awm.create_awdimension
          ('XADEMO','DIVISION','MYSCHEMA', 'MYAW', 'AW_DIV');
```

You can use statements like the following to verify that the dimensions have been created in the analytic workspace.

```
execute dbms_aw.execute
          ('aw attach MYSCHEMA.MYAW');
execute dbms_aw.execute
          ('limit name to obj(property''AW$ROLE'') eq ''DIMDEF''');
execute dbms_aw.execute
          ('report w 40 name');

NAME
----------------------------------------
AW_CHAN
AW_GEOG
AW_PROD
AW_TIME
AW_DIV
```

Alternatively, you can query the Active Catalog to verify that the dimensions have been created.

```
select * from all_olap2_aw_dimensions
           where aw_owner in 'myschema' and aw_name in 'myaw';
```

See Also

- "Creating and Refreshing a Workspace Dimension" on page 1-10
- REFRESH_AWDIMENSION Procedure on page 23-50
- CREATE_AWDIMENSION_ACCESS Procedure on page 23-31
- CREATE_AWCUBE Procedure on page 23-19
- Chapter 3, "Active Catalog Views"

CREATE_AWDIMENSION_ACCESS Procedure

This procedure generates a script that creates relational views of a dimension in an analytic workspace. The views are in the embedded total format required by the OLAP API.

The script can optionally generate OLAP Catalog metadata that maps to the views of the workspace dimension. This metadata is required for the OLAP API.

Both fact views and dimension views are required for relational access to a workspace cube. Use the `CREATE_AWCUBE_ACCESS` procedure to generate the scripts that create the fact views.

To accomplish the enablement process in a single step, use the `CREATE_AWDIMENSION_ACCESS_FULL` procedure. This procedure both creates and runs the enablement script.

Syntax

```
CREATE_AWDIMENSION_ACCESS (
               aw_owner             IN   VARCHAR2,
               aw_name              IN   VARCHAR2,
               aw_dimension_name    IN   VARCHAR2,
               access_type          IN   VARCHAR2,
               script_directory     IN   VARCHAR2,
               script_name          IN   VARCHAR2,
               open_mode            IN   VARCHAR2);
```

Parameters

Table 23–19 CREATE_AWDIMENSION_ACCESS Procedure Parameters

Parameter	Description
`aw_owner`	Owner of the analytic workspace.
`aw_name`	Name of the analytic workspace.
`aw_dimension_name`	Name of the dimension in the analytic workspace.
`access_type`	Controls whether or not the script generates OLAP Catalog metadata for the views. Specify one of the following values: ■ '`SQL`' does not generate metadata. ■ '`OLAP`' generates metadata

Table 23–19 (Cont.) CREATE_AWDIMENSION_ACCESS Procedure Parameters

Parameter	Description
`script_directory`	The directory that will contain the script. This may be either a directory object or a path set by the `UTL_FILE_DIR` parameter.
`script_name`	Name of the script file.
`open_mode`	One of the following modes for opening the script file: ▪ `'W'` overwrites any existing contents of the script file ▪ `'A'` appends the new script to the existing contents of the script file.

Example

The following statement creates an enablement script called `aw_prod_enable` in the `/dat1/scripts` directory. You can run the script to create views of the `AW_PROD` dimension in workspace `XADEMO.MYAW`. The script will also generate an OLAP Catalog dimension called `AW_PROD` that maps to the view.

```
execute dbms_awm.create_awdimension_access
         ('XADEMO', 'MYAW', 'AW_PROD', 'OLAP',
          '/dat1/scripts/', 'aw_prod_enable', 'w');
```

See Also

- "Enabling Relational Access to the Workspace Cube" on page 1-5
- "DELETE_AWDIMENSION_ACCESS Procedure" on page 23-44
- "SET_AWDIMENSION_VIEW_NAME Procedure" on page 23-65
- Chapter 26, "OLAP_TABLE"

CREATE_AWDIMENSION_ACCESS_FULL Procedure

This procedure accomplishes the entire process of enabling a workspace dimension for access by the OLAP API. Like `CREATE_AWDIMENSION_ACCESS` it produces an enablement script. However it does not write the script to a file. Instead it writes the script to temporary memory and runs the script.

The resulting views and metadata are identical to those created by the enablement scripts created by `CREATE_AWDIMENSION_ACCESS`.

Syntax

```
CREATE_AWDIMENSION_ACCESS_FULL (
              run_id                 IN    NUMBER,
              aw_owner               IN    VARCHAR2,
              aw_name                IN    VARCHAR2,
              aw_dimension_name      IN    VARCHAR2,
              access_type            IN    VARCHAR2);
```

Parameters

Table 23–20   CREATE_AWDIMENSION_ACCESS_FULL Procedure Parameters

Parameter	Description
`run_id`	An assigned slot in a global temporary table for holding the record associated with this operation. In most cases, simply specify "1".
`aw_owner`	Owner of the analytic workspace.
`aw_name`	Name of the analytic workspace.
`aw_dimension_name`	Name of the dimension in the analytic workspace.
`access_type`	Controls whether or not to generate OLAP Catalog metadata in addition to the enablement views. Specify one of the following values: ■ `'SQL'` does not generate metadata ■ `'OLAP'` generates metadata

See Also

- "Creating Relational Access to the Workspace Cube" on page 1-23
- "CREATE_AWDIMENSION_ACCESS Procedure" on page 23-31
- "REFRESH_AWDIMENSION Procedure" on page 23-50
- Chapter 26, "OLAP_TABLE"

CREATE_AWDIMLOAD_SPEC Procedure

This procedure creates a **load specification** for an OLAP Catalog dimension. The load specification determines how the dimension will be loaded from relational dimension tables into an analytic workspace by the `REFRESH_AWDIMENSION` procedure.

If you refresh a dimension without a load specification, only new dimension members are loaded.

Syntax

```
CREATE_AWDIMLOAD_SPEC (
               dimension_load_spec     IN   VARCHAR2,
               dimension_owner         IN   VARCHAR2,
               dimension_name          IN   VARCHAR2,
               load_type               IN   VARCHAR2);
```

Parameters

Table 23–21 CREATE_AWDIMLOAD_SPEC Procedure Parameters

Parameter	Description
`dimension_load_spec`	Name of the load specification. You can use the `SET_AWDIMLOAD_SPEC_NAME` procedure to alter the name.
`dimension_owner`	Owner of the OLAP Catalog source dimension.
`dimension_name`	Name of the OLAP Catalog source dimension.
`load_type`	Specify one of the following: `'FULL_LOAD_ADDITIONS_ONLY'` -- Only new dimension members will be loaded when the dimension is refreshed. (Default) `'FULL_LOAD'` -- All dimension members in the workspace will be deleted, then all the members of the source dimension will be loaded.

Note

You can use the following procedures to modify an existing dimension load specification:

- SET_AWDIMLOAD_SPEC_DIMENSION Procedure
- SET_AWDIMLOAD_SPEC_LOADTYPE Procedure
- SET_AWDIMLOAD_SPEC_NAME Procedure
- SET_AWDIMLOAD_SPEC_PARAMETER Procedure

Example

The following statements create a load specification for the XADEMO.CHANNEL source dimension and use it to load the target dimension AW_CHAN in the analytic workspace MYSCHEMA.MYAW. The load specification includes a filter condition (WHERE clause) that causes only the dimension member 'DIRECT' to be loaded.

```
execute dbms_awm.create_awdimload_spec
          ('CHAN_DIMLOADSPEC', 'XADEMO', 'CHANNEL', 'FULL_LOAD');
execute dbms_awm.add_awdimload_spec_filter
          ('CHAN_DIMLOADSPEC', 'XADEMO', 'CHANNEL', 'XADEMO',
          'XADEMO_CHANNEL', '''CHAN_STD_CHANNEL'' = ''DIRECT''' );
execute dbms_awm.refresh_awdimension
          ('MYSCHEMA', 'MYAW', 'AW_CHAN', 'CHAN_DIMLOADSPEC');
```

See Also

- "Creating and Populating Workspace Dimensions" on page 1-4
- REFRESH_AWDIMENSION Procedure on page 23-50

DELETE_AWCOMP_SPEC Procedure

This procedure deletes a composite specification.

Syntax

```
DELETE_AWCOMP_SPEC (
               composite_spec     IN   VARCHAR2,
               cube_owner         IN   VARCHAR2,
               cube_name          IN   VARCHAR2);
```

Parameters

Table 23–22 DELETE_AWCOMP_SPEC Procedure Parameters

Parameter	Description
composite_spec	Name of a composite specification for a cube.
cube_ownere	Owner of the OLAP Catalog source cube.
cube_name	Name of the OLAP Catalog source cube.

See Also

CREATE_AWCOMP_SPEC Procedure on page 23-18

DELETE_AWCOMP_SPEC_MEMBER Procedure

This procedure removes a member of a composite specification. The member can be either a dimension or composite.

Syntax

```
DELETE_AWCOMP_SPEC_MEMBER (
               composite_spec     IN   VARCHAR2,
               cube_owner         IN   VARCHAR2,
               cube_name          IN   VARCHAR2,
               member_name        IN   VARCHAR2,);
```

Parameters

Table 23–23 DELETE_AWCOMP_SPEC_MEMBER Procedure Parameters

Parameter	Description
composite_spec	Name of a composite specification for a cube.
cube_owner	Owner of the OLAP Catalog source cube.
cube_name	Name of the OLAP Catalog source cube.
member_name	Name of the dimension or composite to delete.

See Also

ADD_AWCOMP_SPEC_MEMBER Procedure on page 23-8

DELETE_AWCUBE_ACCESS Procedure

This procedure generates a script that you can run to drop the views and OLAP Catalog metadata associated with a workspace cube. The script does not delete the enablement metadata that is stored in the analytic workspace.

If you drop the workspace cube or the workspace itself, you should run this procedure to clean up the associated enablement views and metadata.

You do not need to run this procedure if you are creating a new generation of enablement views and metadata. The enablement process itself drops the previous generation before creating the new views and metadata.

Syntax

```
DELETE_AWCUBE_ACCESS (
```

```
aw_owner              IN    VARCHAR2,
aw_name               IN    VARCHAR2,
aw_cube_name          IN    VARCHAR2,
access_type           IN    VARCHAR2,
script_directory      IN    VARCHAR2,
script_name           IN    VARCHAR2,
open_mode             IN    VARCHAR2);
```

Parameters

Table 23–24 DELETE_AWCUBE_ACCESS Procedure Parameters

Parameter	Description
aw_owner	Owner of the analytic workspace.
aw_name	Name of the analytic workspace.
aw_cube_name	Name of the cube in the analytic workspace.
access_type	Specifies whether or not OLAP Catalog metadata exists for the views: ▪ 'SQL' No metadata exists. ▪ 'OLAP' OLAP Catalog metadata exists
script_directory	The directory that will contain the script. This may be either a directory object or a path set by the UTL_FILE_DIR parameter.
script_name	Name of the script file.
open_mode	One of the following modes for opening the script file: ▪ 'W' overwrites any existing contents of the script file ▪ 'A' appends the new script to the existing contents of the script file.

See Also

- "Creating Relational Access to the Workspace Cube" on page 1-23
- "CREATE_AWCUBE_ACCESS Procedure" on page 23-22
- "CREATE_AWCUBE_ACCESS_FULL Procedure" on page 23-23
- "SET_AWCUBE_VIEW_NAME Procedure" on page 23-59

DELETE_AWCUBE_ACCESS_ALL Procedure

This procedure deletes all the enablement views and metadata for a cube. It writes a script to a temporary location in memory and runs the script.

Syntax

```
DELETE_AWCUBE_ACCESS_ALL (
              run_id          IN   NUMBER,
              aw_owner        IN   VARCHAR2,
              aw_name         IN   VARCHAR2,
              aw_cube_name    IN   VARCHAR2,
              access_type     IN   VARCHAR2);
```

Parameters

Table 23–25 DELETE_AWCUBE_ACCESS_ALL Procedure Parameters

Parameter	Description
`run_id`	An assigned slot in a global temporary table for holding the record associated with this operation. In most cases, simply specify "1".
`aw_owner`	Owner of the analytic workspace.
`aw_name`	Name of the analytic workspace.
`aw_cube_name`	Name of the cube in the analytic workspace.
`access_type`	Controls whether or not to generate OLAP Catalog metadata in addition to the enablement views. Specify one of the following values: ▪ `'SQL'` does not generate metadata ▪ `'OLAP'` generates metadata

See Also

- "Creating Relational Access to the Workspace Cube" on page 1-23
- "CREATE_AWCUBE_ACCESS_FULL Procedure" on page 23-23

DELETE_AWCUBEAGG_SPEC Procedure

This procedure deletes an aggregation specification.

Syntax

```
DELETE_AWCUBEAGG_SPEC (
               aggregation_spec     IN   VARCHAR2,
               aw_owner             IN   VARCHAR2,
               aw_name              IN   VARCHAR2,
               aw_cube_name         IN   VARCHAR2);
```

Parameters

Table 23–26 DELETE_AWCUBEAGG_SPEC Procedure Parameters

Parameter	Description
`aggregation_spec`	Name of an aggregation specification for a cube in an analytic workspace.
`aw_owner`	Owner of the analytic workspace.
`aw_name`	Name of the analytic workspace.
`aw_cube_name`	Name of the cube in the analytic workspace.

See Also

CREATE_AWCUBEAGG_SPEC Procedure on page 23-25

DELETE_AWCUBEAGG_SPEC_LEVEL Procedure

This procedure removes a level from an aggregation specification.

Syntax

```
DELETE_AWCUBEAGG_SPEC_LEVEL (
               aggregation_spec       IN   VARCHAR2,
               aw_owner               IN   VARCHAR2,
               aw_name                IN   VARCHAR2,
               aw_cube_name           IN   VARCHAR2,
               aw_dimension_name      IN   VARCHAR2,
               aw_level_name          IN   VARCHAR2);
```

Parameters

Table 23–27 DELETE_AWCUBEAGG_SPEC_LEVEL Procedure Parameters

Parameter	Description
aggregation_spec	Name of an aggregation specification for a cube in an analytic workspace.
aw_owner	Owner of the analytic workspace.
aw_name	Name of the analytic workspace.
aw_cube_name	Name of the cube in the analytic workspace.
aw_dimension_name	Name of a dimension of the cube.
aw_level_name	Name of a level of the dimension.

See Also

ADD_AWCUBEAGG_SPEC_LEVEL Procedure on page 23-9

DELETE_AWCUBEAGG_SPEC_MEASURE Procedure

This procedure removes a measure from an aggregation specification.

Syntax

```
DELETE_AWCUBEAGG_SPEC_MEASURE (
               aggregation_spec     IN   VARCHAR2,
               aw_owner             IN   VARCHAR2,
               aw_name              IN   VARCHAR2,
               aw_cube_name         IN   VARCHAR2,
               aw_measure_name      IN   VARCHAR2);
```

Parameters

Table 23–28 DELETE_AWCUBEAGG_SPEC_MEASURE Procedure Parameters

Parameter	Description
aggregation_spec	Name of an aggregation specification for a cube in an analytic workspace.
aw_owner	Owner of the analytic workspace.
aw_name	Name of the analytic workspace.

Table 23–28 (Cont.) DELETE_AWCUBEAGG_SPEC_MEASURE Procedure Parameters

Parameter	Description
aw_cube_name	Name of target cube in the analytic workspace.
aw_measure_name	Name of the measure to remove.

See Also

ADD_AWCUBEAGG_SPEC_MEASURE Procedure on page 23-10

DELETE_AWCUBELOAD_SPEC Procedure

This procedure deletes a cube load specification.

Syntax

```
DELETE_AWCUBELOAD_SPEC (
              cube_load_spec     IN   VARCHAR2,
              cube_owner         IN   VARCHAR2,
              cube_name          IN   VARCHAR2);
```

Parameters

Table 23–29 DELETE_AWCUBELOAD_SPEC Procedure Parameters

Parameter	Description
cube_load_spec	Name of a cube load specification.
cube_owner	Owner of the OLAP Catalog source cube.
cube_name	Name of the OLAP Catalog source cube.

See Also

CREATE_AWCUBELOAD_SPEC Procedure on page 23-26

DELETE_AWCUBELOAD_SPEC_COMP Procedure

This procedure removes a composite specification from a cube load specification.

Syntax

```
DELETE_AWCUBELOAD_SPEC_COMP (
              cube_load_spec     IN   VARCHAR2,
```

```
                cube_owner          IN    VARCHAR2,
                cube_name           IN    VARCHAR2,
                composite_spec      IN    VARCHAR2);
```

Parameters

Table 23–30 DELETE_AWCUBELOAD_SPEC_COMP Procedure Parameters

Parameter	Description
`cube_load_spec`	Name of a cube load specification.
`cube_owner`	Owner of the OLAP Catalog source cube.
`cube_name`	Name of the OLAP Catalog source cube.
`composite_spec`	Name of the composite specification to delete.

See Also

ADD_AWCUBELOAD_SPEC_COMP Procedure on page 23-11

DELETE_AWCUBELOAD_SPEC_FILTER Procedure

This procedure removes the filter condition (`WHERE` clause) from a cube load specification.

Syntax

```
DELETE_AWCUBELOAD_SPEC_FILTER (
                cube_load_spec        IN    VARCHAR2,
                cube_owner            IN    VARCHAR2,
                cube_name             IN    VARCHAR2,
                fact_table_owner      IN    VARCHAR2,
                fact_table_name       IN    VARCHAR2);
```

Parameters

Table 23–31 DELETE_AWCUBELOAD_SPEC_FILTER Procedure Parameters

Parameter	Description
`cube_load_spec`	Name of a cube load specification.
`cube_owner`	Owner of the OLAP Catalog source cube.
`cube_name`	Name of the OLAP Catalog source cube.

Table 23–31 (Cont.) DELETE_AWCUBELOAD_SPEC_FILTER Procedure Parameters

Parameter	Description
`fact_table_owner`	Owner of the fact table that is mapped to this OLAP Catalog source cube.
`fact_table_name`	Name of the fact table that is mapped to this OLAP Catalog source cube

See Also

ADD_AWCUBELOAD_SPEC_FILTER Procedure on page 23-12

DELETE_AWCUBELOAD_SPEC_MEASURE Procedure

This procedure removes a measure from a cube load specification.

Syntax

```
DELETE_AWCUBELOAD_SPEC_MEASURE (
               cube_load_spec    IN   VARCHAR2,
               cube_owner        IN   VARCHAR2,
               cube_name         IN   VARCHAR2,
               measure_name      IN   VARCHAR2);
```

Parameters

Table 23–32 DELETE_AWCUBELOAD_SPEC_MEASURE Procedure Parameters

Parameter	Description
`cube_load_spec`	Name of a cube load specification.
`cube_owner`	Owner of the OLAP Catalog source cube.
`cube_name`	Name of the OLAP Catalog source cube.
`measure_name`	Name of the measure to delete.

See Also

"ADD_AWCUBELOAD_SPEC_MEASURE Procedure" on page 23-13

DELETE_AWDIMENSION_ACCESS Procedure

This procedure generates a script that you can run to drop the views and OLAP Catalog metadata associated with a workspace dimension. The script does not delete the enablement metadata that is stored in the analytic workspace.

If you drop the workspace dimension or the workspace itself, you should run this procedure to clean up the associated enablement views and metadata.

You do not need to run this procedure if you are creating a new generation of enablement views and metadata. The enablement process itself drops the previous generation before creating the new views and metadata.

Syntax

```
DELETE_AWDIMENSION_ACCESS (
               aw_owner              IN   VARCHAR2,
               aw_name               IN   VARCHAR2,
               aw_dimension_name     IN   VARCHAR2,
               access_type           IN   VARCHAR2,
               script_directory      IN   VARCHAR2,
               script_name           IN   VARCHAR2,
               open_mode             IN   VARCHAR2);
```

Parameters

Table 23–33 DELETE_AWDIMENSION_ACCESS Procedure Parameters

Parameter	Description
`aw_owner`	Analytic workspace owner
`aw_name`	Analytic workspace name
`aw_dimension_name`	Analytic workspace dimension name.
`access_type`	Specifies whether or not OLAP Catalog metadata exists for the views: ■ `'SQL'` No metadata exists. ■ `'OLAP'` OLAP Catalog metadata exists
`script_directory`	The directory that will contain the script. This may be either a directory object or a path set by the `UTL_FILE_DIR` parameter.
`script_name`	Name of the script file.

Table 23–33 (Cont.) DELETE_AWDIMENSION_ACCESS Procedure Parameters

Parameter	Description
`open_mode`	One of the following modes for opening the script file: ■ `'W'` overwrites any existing contents of the script file ■ `'A'` appends the new script to the existing contents of the script file.

See Also

- "CREATE_AWDIMENSION_ACCESS Procedure" on page 23-31
- "CREATE_AWCUBE_ACCESS_FULL Procedure" on page 23-23
- "SET_AWDIMENSION_VIEW_NAME Procedure" on page 23-65
- "Creating and Refreshing a Workspace Dimension" on page 1-10

DELETE_AWDIMENSION_ACCESS_ALL Procedure

This procedure deletes all the enablement views and metadata for a dimension. It writes a script to a temporary location in memory and runs the script.

Syntax

```
DELETE_AWDIMENSION_ACCESS_ALL (
          run_id                IN    NUMBER,
          aw_owner              IN    VARCHAR2,
          aw_name               IN    VARCHAR2,
          aw_dimension_name     IN    VARCHAR2,
          access_type           IN    VARCHAR2);
```

Parameters

Table 23–34 DELETE_AWDIMENSION_ACCESS_ALL Procedure Parameters

Parameter	Description
`run_id`	An assigned slot in a global temporary table for holding the record associated with this operation. In most cases, simply specify "1".
`aw_owner`	Owner of the analytic workspace.
`aw_name`	Name of the analytic workspace.
`aw_dimension_name`	Name of the dimension in the analytic workspace.

Table 23–34 (Cont.) DELETE_AWDIMENSION_ACCESS_ALL Procedure Parameters

Parameter	Description
access_type	Controls whether or not to generate OLAP Catalog metadata in addition to the enablement views. Specify one of the following values: ■ 'SQL' does not generate metadata ■ 'OLAP' generates metadata

See Also

- "Creating Relational Access to the Workspace Cube" on page 1-23
- "CREATE_AWDIMENSION_ACCESS_FULL Procedure" on page 23-32

DELETE_AWDIMLOAD_SPEC Procedure

This procedure deletes a dimension load specification.

Syntax

```
DELETE_AWDIMLOAD_SPEC (
              dimension_load_spec     IN   VARCHAR2,
              dimension_owner         IN   VARCHAR2,
              dimension_name          IN   VARCHAR2);
```

Parameters

Table 23–35 DELETE_AWDIMLOAD_SPEC Procedure Parameters

Parameter	Description
dimension_load_spec	Name of a dimension load specification.
dimension_owner	Owner of the OLAP Catalog source dimension.
dimension_name	Name of the OLAP Catalog source dimension.

See Also

CREATE_AWDIMLOAD_SPEC Procedure on page 23-33

DELETE_AWDIMLOAD_SPEC_FILTER Procedure

This procedure removes the filter condition (WHERE clause) from a dimension load specification.

Syntax

```
DELETE_AWDIMLOAD_SPEC_FILTER (
               dimension_load_spec       IN   VARCHAR2,
               dimension_owner           IN   VARCHAR2,
               dimension_name            IN   VARCHAR2);
               dimension_table_owner     IN   VARCHAR2,
               dimension_table_name      IN   VARCHAR2);
```

Parameters

Table 23–36 DELETE_AWDIMLOAD_SPEC_FILTER Procedure Parameters

Parameter	Description
`dimension_load_spec`	Name of a dimension load specification.
`dimension_owner`	Owner of the OLAP Catalog source dimension.
`dimension_name`	Name of the OLAP Catalog source dimension.
`dimension_table_owner`	Owner of the dimension table that is mapped to the OLAP Catalog source dimension.
`dimension_table_name`	Name of the dimension table that is mapped to the OLAP Catalog source dimension.

See Also

ADD_AWDIMLOAD_SPEC_FILTER Procedure on page 23-15

REFRESH_AWCUBE Procedure

This procedure loads data and metadata from an OLAP Catalog source cube into a target cube in an analytic workspace.

`REFRESH_AWCUBE` executes an OLAP DML `UPDATE` command to save the changes in the analytic workspace. `REFRESH_AWCUBE` *does not* execute a SQL `COMMIT`.

You can include a cube load specification to determine how the cube's data will be refreshed. The cube load specification determines whether to load the data or only the load program for execution at a later time. The cube load specification may include a composite specification, which determines dimension order and handling of sparse data.

If you do not include a load specification, all the data is loaded. If you do not include a composite specification, the dimensions are ordered with Time as the fastest-varying followed by a composite of all the other dimensions. The dimensions in the composite are ordered in descending order according to size (number of dimension members).

Unless the load specification for the cube identifies individual measures (ADD_AWCUBELOAD_SPEC_MEASURE), all of the cube's measures are loaded into the workspace. Unless the load specification for the cube includes a filter condition (a WHERE clause on the fact table), all the measures' data is loaded into the workspace.

Before the first call to REFRESH_AWCUBE, you must call REFRESH_AWDIMENSION for each of the cube's dimensions. Before refreshing a cube that already contains data, you must refresh any of its dimensions that have changed since the last refresh.

Syntax

```
REFRESH_AWCUBE (
               aw_owner           IN   VARCHAR2,
               aw_name            IN   VARCHAR2,
               aw_cube_name       IN   VARCHAR2,
               cube_load_spec     IN   VARCHAR2 DEFAULT NULL);
```

Parameters

Table 23–37 REFRESH_AWCUBE Procedure Parameters

Parameter	Description
aw_owner	Owner of the analytic workspace.
aw_name	Name of the analytic workspace.
aw_cube_name	Name of the target cube in the analytic workspace.
cube_load_spec	Name of the cube load specification. If you do not include a load specification, all the fact data is loaded (default).

Note

All the OLAP Catalog metadata that defines the logical cube, including its dimensionality, measures, and descriptions, is refreshed whenever you refresh the workspace cube. The cube's data is refreshed according to the load specification. For more information, see "Refreshing the Cube's Metadata" on page 1-15

For information about the relationship between the refresh and enablement processes, see "Enablement Metadata in the Analytic Workspace" on page 1-26.

For information about the relationship between the refresh and aggregation processes, see "Aggregating the Data in an Analytic Workspace" on page 1-18.

Example

The following statements create the target cube AW_ANACUBE from the source cube XADEMO.ANALYTIC_CUBE. They refresh all of target cube's dimensions, then they create a load specification and refresh the target cube's data.

```
-- create cube, cube load spec, and refresh
execute dbms_awm.create_awcube
          ('XADEMO', 'ANALYTIC_CUBE','MYSCHEMA', 'MYAW','AW_ANACUBE');
execute dbms_awm.create_awcubeload_spec
          ('AC_CUBELOADSPEC', 'XADEMO', 'ANALYTIC_CUBE', 'LOAD_DATA')
execute dbms_awm.refresh_awdimension
          ('MYSCHEMA', 'MYAW', 'AW_CHAN');
execute dbms_awm.refresh_awdimension
          ('MYSCHEMA', 'MYAW', 'AW_PROD');
execute dbms_awm.refresh_awdimension
          ('MYSCHEMA', 'MYAW', 'AW_GEOG');
execute dbms_awm.refresh_awdimension
          ('MYSCHEMA', 'MYAW', 'AW_TIME');
execute dbms_awm.refresh_awcube
          ('MYSCHEMA', 'MYAW', 'AW_ANACUBE', 'AC_CUBELOADSPEC')
```

See Also

- "Creating and Refreshing a Workspace Cube" on page 1-13
- "CREATE_AWCUBE Procedure" on page 23-19
- "REFRESH_AWCUBE Procedure" on page 23-47
- "CREATE_AWCOMP_SPEC Procedure" on page 23-18
- "CREATE_AWCUBE_ACCESS Procedure" on page 23-22

REFRESH_AWCUBE_VIEW_NAME Procedure

This procedure creates metadata in the analytic workspace to support user-defined names for the enablement views. This procedure is *not used* for cubes created and maintained by the DBMS_AWM package.

This procedure is *required* if you want to specify your own names for the enablement views in analytic workspaces that were not created by DBMS_AWM. For example, if you used the OLAP Analytic Workspace Java API to refresh the cube, you must call this procedure before calling SET_AWCUBE_VIEW_NAME.

Syntax

```
REFRESH_AWCUBE_VIEW_NAME (
              aw_owner              IN    VARCHAR2,
              aw_name               IN    VARCHAR2,
              aw_cube_name          IN    VARCHAR2);
```

Parameters

Table 23–38 REFRESH_AWCUBE_VIEW_NAME Procedure Parameters

Parameter	Description
aw_owner	Analytic workspace owner.
aw_name	Analytic workspace name.
aw_cube_name	Analytic workspace cube name.

Note

For details about enablement view names, see "Default Fact View Names" on page 1-27.

See Also

- "Creating Relational Access to the Workspace Cube" on page 1-23
- "SET_AWCUBE_VIEW_NAME Procedure" on page 23-59

REFRESH_AWDIMENSION Procedure

This procedure loads data and metadata from an OLAP Catalog source dimension into a target dimension in an analytic workspace.

REFRESH_AWDIMENSION executes an OLAP DML UPDATE command to save the changes in the analytic workspace. REFRESH_AWDIMENSION *does not* execute a SQL COMMIT.

You can include a dimension load specification to determine how the dimension's members will be refreshed in the workspace. If you do not include a load specification, all dimension members are selected for loading, but only new members are actually added to the target dimension.

You can select individual dimension members to load from the source tables by specifying a filter condition (a WHERE clause on the dimension table).

Before the first call to REFRESH_AWCUBE, you must call REFRESH_AWDIMENSION for each of the cube's dimensions. On all subsequent cube refreshes, you only need to call REFRESH_AWDIMENSION if changes have been made to the source dimensions, for example if new time periods have been added to a time dimension.

Syntax

```
REFRESH_AWDIMENSION (
               aw_owner                IN    VARCHAR2,
               aw_name                 IN    VARCHAR2,
               aw_dimension_name       IN    VARCHAR2,
               dimension_load_spec     IN    VARCHAR2 DEFAULT NULL);
```

Parameters

Table 23–39   REFRESH_AWDIMENSION Procedure Parameters

Parameter	Description
aw_owner	Owner of the analytic workspace.
aw_name	Name of the analytic workspace.
aw_dimension_name	Name of the target dimension within the analytic workspace.
dimension_load_spec	Name of a dimension load specification. If you do not include a load specification, new members are appended to the target dimension (default)

Note

All the OLAP Catalog metadata that defines the logical dimension, including its levels, hierarchies, attributes, and descriptions, is refreshed whenever you refresh the workspace dimension. The dimension's data is refreshed according to the load specification. For more information, see "Refreshing the Dimension's Metadata" on page 1-12

For information about the relationship between the refresh and enablement processes, see "Enablement Metadata in the Analytic Workspace" on page 1-26.

Example

The following statements refresh the dimensions of the XADEMO.ANALYTIC_CUBE source cube in the analytic workspace MYSCHEMA.MYAW.

```
-- Create dimension load specs and refresh dimensions
```

```
-- CHANNEL dimension
execute dbms_awm.create_awdimload_spec
          ('CHAN_DIMLOADSPEC', 'XADEMO', 'CHANNEL', 'FULL_LOAD');
execute dbms_awm.add_awdimload_spec_filter
          ('CHAN_DIMLOADSPEC', 'XADEMO', 'CHANNEL', 'XADEMO',
          'XADEMO_CHANNEL', '''CHAN_STD_CHANNEL'' = ''DIRECT''' );
execute dbms_awm.refresh_awdimension
          ('MYSCHEMA', 'MYAW', 'AW_CHAN', 'CHAN_DIMLOADSPEC');

-- PRODUCT dimension
execute dbms_awm.create_awdimload_spec
          ('PROD_DIMLOADSPEC', 'XADEMO', 'PRODUCT', 'FULL_LOAD');
execute dbms_awm.Set_AWDimLoad_Spec_Parameter
          ('PROD_DIMLOADSPEC', 'XADEMO', 'PRODUCT', 'UNIQUE_RDBMS_KEY', 'YES');
execute dbms_awm.refresh_awdimension
          ('MYSCHEMA', 'MYAW', 'AW_PROD', 'PROD_DIMLOADSPEC');

-- GEOGRAPHY dimension
execute dbms_awm.create_awdimload_spec
          ('GEOG_DIMLOADSPEC', 'XADEMO', 'GEOGRAPHY', 'FULL_LOAD');
execute dbms_awm.refresh_awdimension
          ('MYSCHEMA', 'MYAW', 'AW_GEOG', 'GEOG_DIMLOADSPEC');

-- TIME dimension
execute dbms_awm.create_awdimload_spec
          ('TIME_DIMLOADSPEC', 'XADEMO', 'TIME', 'FULL_LOAD');
execute dbms_awm.refresh_awdimension
          ('MYSCHEMA', 'MYAW', 'AW_TIME', 'TIME_DIMLOADSPEC');
```

See Also

- "Creating and Refreshing a Workspace Dimension" on page 1-10
- CREATE_AWDIMENSION Procedure on page 23-28
- "CREATE_AWDIMLOAD_SPEC Procedure" on page 23-33
- "CREATE_AWDIMENSION_ACCESS Procedure" on page 23-31

REFRESH_AWDIMENSION_VIEW_NAME Procedure

This procedure creates metadata in the analytic workspace to support user-defined names for the enablement views. This procedure is *not needed* for dimensions created and maintained by the DBMS_AWM package.

This procedure is *required* if you want to specify your own names for the enablement views in analytic workspaces that were not created by DBMS_AWM. For example, if you used the OLAP Analytic Workspace Java API to refresh the dimension, you must call this procedure before calling SET_AWDIMENSION_VIEW_NAME.

Syntax

```
REFRESH_AWDIMENSION_VIEW_NAME (
              aw_owner                IN    VARCHAR2,
              aw_name                 IN    VARCHAR2,
              aw_dimension_name       IN    VARCHAR2);
```

Parameters

Table 23–40 REFRESH_AWDIMENSION_VIEW_NAME Procedure Parameters

Parameter	Description
aw_owner	Analytic workspace owner.
aw_name	Analytic workspace name.
aw_dimension_name	Analytic workspace dimension name.

Note

For details about enablement view names, see "Default Dimension View Names" on page 1-27.

See Also

- "Creating Relational Access to the Workspace Cube" on page 1-23
- "SET_AWDIMENSION_VIEW_NAME Procedure" on page 23-65

SET_AWCOMP_SPEC_CUBE Procedure

This procedure associates a composite specification with a different cube.

Syntax

```
SET_AWCOMP_SPEC_CUBE (
              composite_spec      IN    VARCHAR2,
              old_cube_owner      IN    VARCHAR2,
              old_cube_name       IN    VARCHAR2,
```

```
                 new_cube_owner      IN    VARCHAR2,
                 new_cube_name       IN    VARCHAR2);
```

Parameters

Table 23–41 SET_AWCOMP_SPEC_CUBE Procedure Parameters

Parameter	Description
`composite_spec`	Name of a composite specification.
`old_cube_owner`	Owner of the old OLAP Catalog source cube.
`old_cube_name`	Name of the old OLAP Catalog source cube.
`new_cube_owner`	Owner of the new OLAP Catalog source cube.
`new_cube_name`	Name of the new OLAP Catalog source cube.

See Also

- "Managing Sparse Data and Optimizing the Workspace Cube" on page 1-16
- CREATE_AWCOMP_SPEC Procedure on page 23-18

SET_AWCOMP_SPEC_MEMBER_NAME Procedure

This procedure changes the name of a member of a composite specification. The member may be either a dimension or a composite.

Syntax

```
SET_AWCOMP_SPEC_MEMBER_NAME (
               composite_spec      IN    VARCHAR2,
               cube_owner          IN    VARCHAR2,
               cube_name           IN    VARCHAR2,
               old_member_name     IN    VARCHAR2,
               new_member_name     IN    VARCHAR2);
```

Parameters

Table 23–42 SET_AWCOMP_SPEC_MEMBER_NAME Procedure Parameters

Parameter	Description
`composite_spec`	Name of a composite specification for a cube.
`cube_owner`	Owner of the OLAP Catalog source cube.

Table 23–42 (Cont.) SET_AWCOMP_SPEC_MEMBER_NAME Procedure Parameters

Parameter	Description
`cube_name`	Name of the OLAP Catalog source cube.
`old_member_name`	Old member name. Either a dimension or a composite.
`new_member_name`	New member name.

See Also

- "Managing Sparse Data and Optimizing the Workspace Cube" on page 1-16
- CREATE_AWCOMP_SPEC Procedure on page 23-18

SET_AWCOMP_SPEC_MEMBER_POS Procedure

This procedure sets the position of a member of a composite specification. The member can be either a dimension or a composite.

Syntax

```
SET_AWCOMP_SPEC_MEMBER_POS (
               composite_spec      IN   VARCHAR2,
               cube_owner          IN   VARCHAR2,
               cube_name           IN   VARCHAR2,
               member_name         IN   VARCHAR2,
               member_position     IN   NUMBER);
```

Parameters

Table 23–43 SET_AWCOMP_SPEC_MEMBER_POS Procedure Parameters

Parameter	Description
`composite_spec`	Name of a composite specification for a cube.
`cube_owner`	Owner of the OLAP Catalog source cube.
`cube_name`	Name of the OLAP Catalog source cube.
`member_name`	Member of the composite specification. Either a dimension or a composite.
`member_position`	Position of the member within the composite specification.

Example

The following statements create a composite specification for the `ANALYTIC_CUBE` in `XADEMO`. It includes two members: a time dimension called `TIMECOMP_MEMBER` and a composite called `COMP1`.

```
---- The logical members of the specification are:
---             <TIME COMP1<PRODUCT, GEOGRAPHY>.
---------------------------------------------------------
execute DBMS_AWM.Create_AWComp_spec
          ('AC_COMPSPEC' ,'XADEMO' ,'ANALYTIC_CUBE');
execute DBMS_AWM.Add_AWComp_Spec_Member
          ('AC_COMPSPEC' ,'XADEMO' ,'ANALYTIC_CUBE' ,'TIMECOMP_MEMBER' ,
           'DIMENSION' ,'XADEMO' ,'TIME');
execute DBMS_AWM.Add_AWComp_Spec_Member
          ('AC_COMPSPEC' ,'XADEMO' ,'ANALYTIC_CUBE' ,'COMP1'  ,'COMPOSITE');
execute DBMS_AWM.Add_AWComp_Spec_Comp_Member
          ('AC_COMPSPEC','XADEMO','ANALYTIC_CUBE', 'COMP1','PROD_COMP',
           'DIMENSION','XADEMO','PRODUCT');
execute DBMS_AWM.Add_AWComp_Spec_Comp_Member
          ('AC_COMPSPEC','XADEMO','ANALYTIC_CUBE', 'COMP1','GEOG_COMP',
           'DIMENSION','XADEMO','GEOGRAPHY');

---- With the following statement, the logical members of the specification
---- are reordered as follows.
---             <COMP1<PRODUCT, GEOGRAPHY> TIME>.
---------------------------------------------------------

execute DBMS_AWM.Set_AWComp_Spec_Member_Pos
          ('AC_COMPSPEC' ,'XADEMO' ,'ANALYTIC_CUBE' ,'COMP1'  ,1);
```

See Also

- "Managing Sparse Data and Optimizing the Workspace Cube" on page 1-16
- CREATE_AWCOMP_SPEC Procedure on page 23-18

SET_AWCOMP_SPEC_MEMBER_SEG Procedure

This procedure sets the segment size for a member of a composite specification. A member is either a dimension or a composite.

A segment is an internal buffer used by the OLAP engine for storing data. The size of segments affects the performance of data loads and queries against the data.

Syntax

```
SET_AWCOMP_SPEC_MEMBER_SEG (
               composite_spec      IN   VARCHAR2,
               cube_owner          IN   VARCHAR2,
               cube_name           IN   VARCHAR2,
               member_name         IN   VARCHAR2,
               member_segwidth     IN   NUMBER DEFAULT NULL);
```

Parameters

Table 23–44 SET_AWCOMP_SPEC_MEMBER_SEG Procedure Parameters

Parameter	Description
`composite_spec`	Name of a composite specification.
`cube_owner`	Owner of the OLAP Catalog source cube.
`cube_name`	Name of the OLAP Catalog source cube.
`member_name`	Name of the dimension or composite.
`member_segwidth`	Segment size associated with a dimension or composite. If you do not specify a segment size for a dimension, the value is the maximum size of the dimension (number of dimension members). If you do not specify a segment size for a composite, the value is 10 million.

Example

The following statements set the segment size for the time dimension to zero (the default setting in the analytic workspace) and the segment size for the `COMP1` composite to 10,000,000.

```
 execute DBMS_AWM.Create_AWComp_spec
          ('AC_COMPSPEC' ,'XADEMO' ,'ANALYTIC_CUBE');
execute DBMS_AWM.Add_AWComp_Spec_Member
          ('AC_COMPSPEC' ,'XADEMO' ,'ANALYTIC_CUBE' ,'TIME_DIM'  ,
           'DIMENSION' ,'XADEMO' ,'time');
execute DBMS_AWM.Add_AWComp_Spec_Member
          ('AC_COMPSPEC' ,'XADEMO' ,'ANALYTIC_CUBE' ,'COMP1'  ,'COMPOSITE');
execute DBMS_AWM.Add_AWComp_Spec_Comp_Member
          ('AC_COMPSPEC', 'XADEMO', 'ANALYTIC_CUBE', 'COMP1', 'COMP1_PROD',
           'DIMENSION', 'XADEMO', 'product');
execute DBMS_AWM.Add_AWComp_Spec_Comp_Member
          ('AC_COMPSPEC', 'XADEMO', 'ANALYTIC_CUBE', 'COMP1', 'COMP1_GEOG',
           'DIMENSION', 'XADEMO', 'geography');
```

```
execute DBMS_AWM.Set_AWComp_Spec_Member_Seg
          ('AC_COMPSPEC' , 'XADEMO', 'ANALYTIC_CUBE', 'TIME_DIM', 0);
execute DBMS_AWM.Set_AWComp_Spec_Member_Seg
          ('AC_COMPSPEC' , 'XADEMO', 'ANALYTIC_CUBE', 'COMP1', NULL);
```

See Also

- "Managing Sparse Data and Optimizing the Workspace Cube" on page 1-16
- In Oracle9i OLAP DML Reference help, search for "segment width"
- CREATE_AWCOMP_SPEC Procedure on page 23-18

SET_AWCOMP_SPEC_NAME Procedure

This procedure renames a composite specification.

Syntax

```
SET_AWCOMP_SPEC_NAME (
             old_composite_spec        IN   VARCHAR2,
             cube_owner                IN   VARCHAR2,
             cube_name                 IN   VARCHAR2,
             new_composite_spec        IN   VARCHAR2);
```

Parameters

Table 23–45 SET_AWCOMP_SPEC_NAME Procedure Parameters

Parameter	Description
old_composite_spec	Old name of a composite specification for a cube.
cube_owner	Owner of the OLAP Catalog source cube.
cube_name	Name of the OLAP Catalog source cube.
new_composite_spec	New name of the composite specification.

See Also

- "Managing Sparse Data and Optimizing the Workspace Cube" on page 1-16
- CREATE_AWCOMP_SPEC Procedure on page 23-18

SET_AWCUBE_VIEW_NAME Procedure

This procedure renames the relational views of an analytic workspace cube. The names are stored in the analytic workspace and instantiated when you generate and run new enablement scripts.

You can use this procedure to override the default view names established when the cube is refreshed by `REFRESH_AWCUBE`.

If the cube was refreshed by some other mechanism, such as the OLAP Analytic Workspace Java API, you must call `REFRESH_AWCUBE_VIEW_NAME` before calling this procedure.

Syntax

```
SET_AWCUBE_VIEW_NAME (
              aw_owner                 IN   VARCHAR2,
              aw_name                  IN   VARCHAR2,
              aw_cube_name             IN   VARCHAR2,
              hierarchy_combo_number   IN   NUMBER,
              view_name                IN   VARCHAR2);
```

Parameters

Table 23–46 SET_AWCUBE_VIEW_NAME Procedure Parameters

Parameter	Description
`aw_owner`	Analytic workspace owner.
`aw_name`	Analytic workspace name.
`aw_cube_name`	Analytic workspace cube name.
`hierarchy_combo_number`	Number of the hierarchy combination.
`view_name`	Name for the fact view for this hierarchy combination.

Note

For details about enablement view names, see "Default Fact View Names" on page 1-27.

See Also

- "Creating Relational Access to the Workspace Cube" on page 1-23

- "CREATE_AWCUBE_ACCESS Procedure" on page 23-22
- "DELETE_AWCUBE_ACCESS Procedure" on page 23-36
- "REFRESH_AWCUBE_VIEW_NAME Procedure" on page 23-49

SET_AWCUBEAGG_SPEC_AGGOP Procedure

This procedure sets the operator for aggregation along one of the dimensions in an aggregation specification.

You can specify any aggregation operator that can be used with the OLAP DML RELATION command. The default operator is addition (SUM). You can use this procedure to override the aggregation operator associated with the source cube in the OLAP Catalog.

> **Note:** The DBMS_AWM package currently does not support weighted aggregation operators. For example, if the OLAP Catalog specifies a weighted sum or weighted average for aggregation along one of the cube's dimensions, it is converted to the scalar equivalent (sum or average) in the analytic workspace. Weighted operators specified by SET_AWCUBEAGG_SPEC_AGGOP are similarly converted.

Syntax

```
SET_AWCUBEAGG_SPEC_AGGOP (
               aggregation_spec         IN   VARCHAR2,
               aw_owner                 IN   VARCHAR2,
               aw_name                  IN   VARCHAR2,
               aw_cube_name             IN   VARCHAR2,
               aw_measure_name          IN   VARCHAR2,
               aw_dimension_name        IN   VARCHAR2,
               aggregation_operator     IN   VARCHAR2);
```

Parameters

Table 23–47 SET_AWCUBEAGG_SPEC_AGGOP Procedure Parameters

Parameter	Description
aggregation_spec	Name of the aggregation specification in the analytic workspace.
aw_owner	Owner of the analytic workspace.

Table 23–47 (Cont.) SET_AWCUBEAGG_SPEC_AGGOP Procedure Parameters

Parameter	Description
aw_name	Name of the analytic workspace.
aw_cube_name	Name of the target cube in the analytic workspace.
aw_measure_name	Name of a measure to aggregate.
aw_dimension_name	Name of a dimension of the cube.
aggregation_operator	Aggregation operator for aggregation along this dimension. See Table 1–10, " Aggregation Operators".

Note

See "Choosing an Aggregation Method" on page 1-21 for details on aggregation methods supported in the OLAP Catalog and in the analytic workspace.

See Also

- "Aggregating the Data in an Analytic Workspace" on page 1-18
- CREATE_AWCUBEAGG_SPEC Procedure on page 23-25
- RELATION command entry in Oracle9i OLAP DML Reference help
- Chapter on Aggregation in *Oracle OLAP DML Reference*

SET_AWCUBELOAD_SPEC_CUBE Procedure

This procedure associates a cube load specification with a different cube.

Syntax

```
SET_AWCUBELOAD_SPEC_CUBE (
             cube_load_spec     IN   VARCHAR2,
             old_cube_owner     IN   VARCHAR2,
             old_cube_name      IN   VARCHAR2,
             new_cube_owner     IN   VARCHAR2,
             new_cube_name      IN   VARCHAR2);
```

Parameters

Table 23–48 SET_AWCUBELOAD_SPEC_CUBE Procedure Parameters

Parameter	Description
cube_load_spec	Name of a cube load specification.
old_cube_owner	Owner of the old OLAP Catalog source cube.
old_cube_name	Name of the old OLAP Catalog source cube.
new_cube_owner	Owner of the new OLAP Catalog source cube.
new_cube_name	Name of the new OLAP Catalog source cube.

See Also

CREATE_AWCUBELOAD_SPEC Procedure on page 23-26

SET_AWCUBELOAD_SPEC_LOADTYPE Procedure

This procedure resets the load type for a cube load specification. The load type indicates how data will be loaded into the analytic workspace.

Syntax

```
SET_AWCUBELOAD_SPEC_LOADTYPE (
               cube_load_spec     IN   VARCHAR2,
               cube_owner         IN   VARCHAR2,
               cube_name          IN   VARCHAR2,
               load_type          IN   VARCHAR2);
```

Parameters

Table 23–49 SET_AWCUBELOAD_SPEC_LOADTYPE Procedure Parameters

Parameter	Description
cube_load_spec	Name of a load specification for a cube.
cube_owner	Owner of the OLAP Catalog source cube.
cube_name	Name of the OLAP Catalog source cube.

Table 23–49 (Cont.) SET_AWCUBELOAD_SPEC_LOADTYPE Procedure Parameters

Parameter	Description
`load_type`	`'LOAD_DATA'` -- Load the data from the fact table into the analytic workspace target cube. `'LOAD_PROGRAM'` -- Create the load program in the analytic workspace but do not execute it. You can run the program manually to load the data. Cube load program names are stored in the `AW$LOADPRGS` property of the standard form cube in the analytic workspace. You can display the load program name with an OLAP DML command like the following: `->show obj(property 'aw$loadprgs' 'my_awcube_name')`

See Also

CREATE_AWCUBELOAD_SPEC Procedure on page 23-26

SET_AWCUBELOAD_SPEC_NAME Procedure

This procedure renames a cube load specification.

Syntax

```
SET_AWCUBELOAD_SPEC_NAME (
               old_cube_load_spec     IN   VARCHAR2,
               cube_owner             IN   VARCHAR2,
               cube_name              IN   VARCHAR2,
               new_cube_load_spec     IN   VARCHAR2);
```

Parameters

Table 23–50 SET_AWCUBELOAD_SPEC_NAME Procedure Parameters

Parameter	Description
`old_cube_load_spec`	Old name of a cube load specification.
`cube_owner`	Owner of the OLAP Catalog source cube.
`cube_name`	Name of the OLAP Catalog source cube.
`new_cube_load_spec`	New name of the cube load specification.

See Also

CREATE_AWCUBELOAD_SPEC Procedure on page 23-26

SET_AWCUBELOAD_SPEC_PARAMETER Procedure

This procedure sets parameters for a cube load specification.

Syntax

```
SET_AWCUBELOAD_SPEC_PARAMETER (
               cube_load_spec     IN   VARCHAR2,
               cube_owner         IN   VARCHAR2,
               cube_name          IN   VARCHAR2,
               parameter_name     IN   VARCHAR2,
               parameter_value    IN   VARCHAR2 DEFAULT NULL);
```

Parameters

Table 23–51 SET_AWCUBELOAD_SPEC_PARAMETER Procedure Parameters

Parameter	Description
`cube_load_spec`	Name of a cube load specification.
`cube_owner`	Owner of the OLAP Catalog source cube.
`cube_name`	Name of the OLAP Catalog source cube.
`parameter_name`	`'DISPLAY_NAME'` -- Whether to use the OLAP Catalog source cube name or the target cube display name as the display name for the target cube in the analytic workspace.
`parameter_value`	Value of `DISPLAY_NAME` is the display name for the target cube in the analytic workspace. If you do not specify this parameter, the display name for the source cube in the OLAP Catalog will be used as the display name for the target cube in the analytic workspace.

Example

The following statement specifies a target cube display name for the `AC_CUBELOADSPEC` cube load specification.

```
execute dbms_awm.set_awcubeload_spec_parameter
          ('AC_CUBELOADSPEC', 'XADEMO', 'ANALYTIC_CUBE',
           'DISPLAY_NAME', 'My AW Analytic Cube Display Name')
```

See Also

CREATE_AWCUBELOAD_SPEC Procedure on page 23-26

SET_AWDIMENSION_VIEW_NAME Procedure

This procedure renames the relational views of an analytic workspace dimension. The names are stored in the analytic workspace and instantiated when you generate and run new enablement scripts.

You can use this procedure to override the default view names established when the dimension was refreshed by `REFRESH_AWDIMENSION`.

If the dimension was refreshed by some other mechanism, such as the OLAP Analytic Workspace Java API, you must call `REFRESH_AWDIMENSION_VIEW_NAME` before calling this procedure.

Syntax

```
SET_AWDIMENSION_VIEW_NAME (
               aw_owner              IN   VARCHAR2,
               aw_name               IN   VARCHAR2,
               aw_dimension_name     IN   VARCHAR2,
               hierarchy_name        IN   VARCHAR2,
               view_name             IN   VARCHAR2);
```

Parameters

Table 23–52 SET_AWDIMENSION_VIEW_NAME Procedure Parameters

Parameter	Description
`aw_owner`	Analytic workspace owner
`aw_name`	Analytic workspace name
`aw_dimension_name`	Analytic workspace dimension name
`hierarchy_name`	Analytic workspace hierarchy name
`view_name`	Name for the view of the dimension hierarchy.

Note

For details about enablement view names, see "Default Dimension View Names" on page 1-27.

See Also

- "Creating Relational Access to the Workspace Cube" on page 1-23
- "CREATE_AWDIMENSION_ACCESS Procedure" on page 23-31

- "DELETE_AWDIMENSION_ACCESS Procedure" on page 23-44
- "REFRESH_AWDIMENSION_VIEW_NAME Procedure" on page 23-52

SET_AWDIMLOAD_SPEC_DIMENSION Procedure

This procedure associates a dimension load specification with a different dimension.

Syntax

```
SET_AWDIMLOAD_SPEC_DIMENSION (
              dimension_load_spec     IN   VARCHAR2,
              old_dimension_owner     IN   VARCHAR2,
              old_dimension_name      IN   VARCHAR2,
              new_dimension_owner     IN   VARCHAR2,
              new_dimension_name      IN   VARCHAR2);
```

Parameters

Table 23–53 SET_AWDIMLOAD_SPEC_DIMENSION Procedure Parameters

Parameter	Description
`dimension_load_spec`	Name of a dimension load specification.
`old_dimension_owner`	Owner of the old OLAP Catalog source dimension.
`old_dimension_name`	Name of the old OLAP Catalog source dimension.
`new_dimension_owner`	Owner of the new OLAP Catalog source dimension.
`new_dimension_name`	Name of the new OLAP Catalog source dimension.

See Also

CREATE_AWDIMLOAD_SPEC Procedure on page 23-33

SET_AWDIMLOAD_SPEC_LOADTYPE Procedure

This procedure resets the load type for a dimension load specification. The load type indicates how dimension members will be loaded into the analytic workspace.

By default only new members are loaded when the dimension is refreshed.

Syntax

```
SET_AWDIMLOAD_SPEC_LOADTYPE (
```

```
                dimension_load_spec     IN   VARCHAR2,
                dimension_owner         IN   VARCHAR2,
                dimension_name          IN   VARCHAR2,
                load_type               IN   VARCHAR2);
```

Parameters

Table 23–54 SET_AWDIMLOAD_SPEC_LOADTYPE Procedure Parameters

Parameter	Description
dimension_load_spec	Name of a dimension load specification.
dimension_owner	Owner of the OLAP Catalog source dimension.
dimension_name	Name of the OLAP Catalog source dimension.
load_type	Specify one of the following: 'FULL_LOAD_ADDITIONS_ONLY' -- Only new dimension members will be loaded when the dimension is refreshed. (Default) 'FULL_LOAD' -- When the dimension is refreshed, all dimension members in the workspace will be deleted, then all the members of the source dimension will be loaded.

See Also

CREATE_AWDIMLOAD_SPEC Procedure on page 23-33

SET_AWDIMLOAD_SPEC_NAME Procedure

This procedure renames a dimension load specification.

Syntax

```
SET_AWDIMLOAD_SPEC_NAME (
                old_dimension_load_spec     IN   VARCHAR2,
                dimension_owner             IN   VARCHAR2,
                dimension_name              IN   VARCHAR2,
                new_dimension_load_spec     IN   VARCHAR2);
```

Parameters

Table 23–55 SET_AWDIMLOAD_SPEC_NAME Procedure Parameters

Parameter	Description
old_dimension_load_spec	Old name of the dimension load specification.
dimension_owner	Owner of the OLAP Catalog source dimension.
dimension_name	Name of the OLAP Catalog source dimension.
new_dimension_load_spec	New name for the dimension load specification.

See Also

CREATE_AWDIMLOAD_SPEC Procedure on page 23-33

SET_AWDIMLOAD_SPEC_PARAMETER Procedure

This procedure sets parameters for a dimension load specification.

Syntax

```
SET_AWDIMLOAD_SPEC_PARAMETER (
               dimension_load_spec     IN   VARCHAR2,
               dimension_owner         IN   VARCHAR2,
               dimension_name          IN   VARCHAR2,
               parameter_name          IN   VARCHAR2,
               parameter_value         IN   VARCHAR2 DEFAULT NULL);
```

Parameters

Table 23–56 SET_AWDIMLOAD_SPEC_PARAMETER Procedure Parameters

Parameter	Description
dimension_load_spec	Name of a dimension load specification.
dimension_owner	Owner of the OLAP Catalog source dimension.
dimension_name	Name of the OLAP Catalog source dimension.

Table 23–56 (Cont.) SET_AWDIMLOAD_SPEC_PARAMETER Procedure Parameters

Parameter	Description
`parameter_name`	One of the following: `'UNIQUE_RDBMS_KEY'` -- Whether or not the members of this dimension are unique across all levels in the source tables. `'DISPLAY_NAME'` -- Display name for the target dimension in the analytic workspace. `'P_DISPLAY_NAME'` -- Plural display name for the target dimension in the analytic workspace.
`parameter_value`	Values of `UNIQUE_RDBMS_KEY` can be either `'YES'` or `'NO'`. The default is `'NO'`. `NO` -- Dimension member names are not unique across levels in the RDBMS tables. The corresponding dimension member names in the analytic workspace include the level name as a prefix. (Default) `YES` -- Dimension member names are unique across levels in the RDBMS tables. The corresponding dimension member names in the analytic workspace have the same names as in the source relational dimension. Value of `DISPLAY_NAME` is the display name for the target dimension in the analytic workspace. If you do not specify this parameter, the display name for the source dimension in the OLAP Catalog will be used as the display name for the target dimension in the analytic workspace. Value of `P_DISPLAY_NAME` is the plural display name for the target dimension in the analytic workspace. If you do not specify this parameter, the plural display name for the source dimension in the OLAP Catalog will be used as the plural display name for the target dimension in the analytic workspace.

Example

The following statements set parameters for the product dimension in the load specification `PROD_LOADSPEC`. These parameters prevent level prefixes on dimension member names, and they specify a display name and plural display name for the target dimension.

```
execute dbms_awm.Set_AWDimLoad_Spec_Parameter
          ('PROD_LOADSPEC', 'XADEMO', 'PRODUCT', 'UNIQUE_RDBMS_KEY', 'YES')
execute dbms_awm.Set_AWDimLoad_Spec_Parameter
          ('PROD_LOADSPEC', 'XADEMO', 'PRODUCT', 'DISPLAY_NAME',
           'My AW Product Display Name')
```

```
execute dbms_awm.Set_AWDimLoad_Spec_Parameter
          ('PROD_LOADSPEC', 'XADEMO', 'PRODUCT', 'P_DISPLAY_NAME',
           'My AW Product Plural Display Name')
```

See Also

CREATE_AWDIMLOAD_SPEC Procedure on page 23-33

24

DBMS_ODM

The OLAP Data Management package, DBMS_ODM, provides procedures for creating materialized views specific to the requirements of the OLAP API.

See Also:

- *Oracle Data Warehousing Guide* for information on creating and managing materialized views
- *Oracle OLAP Application Developer's Guide* for information on summary management for Oracle OLAP

This chapter includes the following topics:

Summary Management with Materialized Views

Summary management for relational warehouses is managed by the query rewrite facility in the database. Query rewrite enables a query to fetch aggregate data from materialized views rather than recomputing the aggregates at runtime.

When the OLAP API queries a warehouse stored in relational tables, it uses query rewrite whenever possible. However, the OLAP API can only use query rewrite when the materialized views have a specific format. The procedures in the DBMS_ODM package create materialized views that satisfy the requirements of the OLAP API.

When the source data is stored in an analytic workspace, materialized views are not needed. The native multidimensional structures within analytic workspaces support both stored summarization and run-time aggregation. You can move your data from a star schema to an analytic workspace with the `DBMS_AWM` package or with Analytic Workspace Manager.

Grouping Sets

The `DBMS_ODM` package creates a set of materialized views based on a cube defined in the OLAP Catalog. The cube must be mapped to a star schema with a single fact table containing only lowest level data.

Scripts generated by `DBMS_ODM` procedures create the following materialized views:

- A dimension materialized view for each hierarchy of each of the cube's dimensions
- A fact materialized view, created with `GROUP BY GROUPING SETS` syntax, for the cube's measures

Each grouping set generated by the `CREATE MATERIALIZED VIEW` statement identifies a unique combination of levels. With grouping sets, you can summarize your data symmetrically, for example sales at the month level across all levels of geography, or you can summarize it asymmetrically, for example sales at the month level for cities and at the quarter level for states.

Summarizing the Fact Table

`DBMS_ODM` supports several approaches to creating the grouping set materialized view for the cube's fact table. You can choose from the following options:

- Automatically generate a materialized view that defines the summaries for every level combination in the cube.

  This option may potentially generate a very large materialized view, depending on the size of the fact table. In general, you should use this option only if disk space is plentiful.

- Automatically generate a materialized view that defines minimal summarization for the cube. The materialized view will include only the most aggregate level and one level above the least aggregate level for each dimension.

  This option will generate a materialized view of moderate size, depending on the size of the fact table. The summarization will be symmetric.

- Automatically generate a materialized view that defines summarization for a percentage of the level combinations in the cube.

  This option may generate a materialized view of moderate size, depending on the size of the fact table and the percentage that you specify. The level combinations included in the materialized view will be random. The summarization will typically be asymmetric.

- Manually choose the level combinations to be included in the materialized view for the cube.

  With this option, you can finely tune both the content and the size of the materialized view. The summarization may be symmetric or asymmetric.

Note: If you have specified the same aggregation operator for each of the cube's dimensions, this operator will be used to aggregate the data for the fact materialized view. You can set an aggregation operator for a cube in Enterprise Manager, or you can use the `CWM2` procedure, SET_AGGREGATION_OPERATOR Procedure, described on page 9-6.

If you have specified an aggregation operator for some or none of the cube's dimensions, the data will by summarized by addition.

For a list of aggregation operators supported by the OLAP Catalog, see Table 1–10, " Aggregation Operators" on page 1-22.

Procedure: Automatically Generate the Materialized Views

Follow these steps to automatically create the materialized views for a cube:

1. Create a cube in the OLAP Catalog. You can use Enterprise Manager or you can use the `CWM2` procedures. If you use the `CWM2` procedures, be sure to map the cube to a star schema.
2. Configure the database to write to files. The `DBMS_ODM` procedures accept either a directory object to which your user ID has been granted the appropriate access, or a directory path specified by the `UTL_FILE_DIR` initialization parameter for the instance.
3. Log into SQL*Plus using the identity of the metadata owner.
4. Delete any materialized views that currently exist for the cube. Execute `DROP MATERIALIZED VIEW` *`mv_name`* for each materialized view you wish to delete.

5. Create scripts to generate the dimension materialized views. Execute DBMS_ODM.CREATEDIMMV_GS for each of the cube's dimensions.

6. Create a script to generate the fact materialized view. Execute DBMS_ODM.CREATESTDFACTMV and choose one of the following values for the materialization level parameter:

   - FULL — Fully materialize the cube's data. Include every level combination in the materialized view.
   - MINIMUM — Minimally materialize the cube's data. Include the level above the leaf level and the most aggregate level for each dimension in the materialized view.
   - PERCENT — Materialize the cube's data based on a percentage of the cube's level combinations.

7. Run the scripts in SQL*Plus, using commands such as the following:

```
@/users/oracle/OraHome1/olap/mvscript.sql;
```

Procedure: Manually Generate the Materialized Views

Follow these steps to create the materialized views with specific level combinations:

1. Follow the first five steps in "Procedure: Automatically Generate the Materialized Views" on page 24-3.

2. Use the following three step procedure to create a script to generate the fact materialized view:

   a. Execute DBMS_ODM.CREATEDIMLEVTUPLE to create the table sys.olaptablevels. This table lists all the dimensions of the cube and all the levels of each dimension. Edit the table to deselect any levels that you do not want to include.

   b. Execute DBMS_ODM.CREATECUBELEVELTUPLE to create the table sys.olaptableveltuples. This table lists all the possible combinations (grouping sets) of the levels you chose in the previous step. Edit the table to deselect any level combinations that you do not want to include.

   c. Execute DBMS_ODM.CREATEFACTMV_GS to create the script.

3. Run the scripts in SQL*Plus, using commands such as the following:

```
@/users/oracle/OraHome1/olap/mvscript_fact.sql;
```

Example: Create Materialized Views for a Sales Cube

Let's assume that you want to create materialized views for the PRICE_CUBE in the GLOBAL schema.

This cube contains unit costs and unit prices for different products over time. The dimensions are PRODUCT, with levels for products, families of products, classes of products, and totals, and TIME with levels for months, quarters, and years.

You want to summarize product families by month and product classes by quarter and make that data available in a materialized view.

1. First generate the scripts for the dimension materialized views. The following statements create the scripts prodmv and timemv in the directory /users/global/scripts.

```
exec dbms_odm.createdimmv_gs
     ('global', 'product','prodmv','/users/global/scripts');
exec dbms_odm.createdimmv_gs
     ('global', 'time','timemv','/users/global/scripts');
```

2. Run the scripts to create the dimension materialized views.
3. Next create the table of dimension levels for the fact materialized view.

```
exec dbms_odm.createdimlevtuple('global', 'price_cube');
```

   The table of levels, sys.olaptablevels, is a temporary table specific to your session. You can view the table as follows.

```
select * from sys.olaptablevels;

SCHEMA_NAME DIMENSION_NAME DIMENSION_OWNER CUBE_NAME    LEVEL_NAME SELECTED
----------- -------------- --------------- ----------   ---------- --------
GLOBAL      TIME           GLOBAL          PRICE_CUBE   Year              1
GLOBAL      TIME           GLOBAL          PRICE_CUBE   Quarter           1
GLOBAL      TIME           GLOBAL          PRICE_CUBE   Month             1
GLOBAL      PRODUCT        GLOBAL          PRICE_CUBE   TOTAL_PRODUCT     1
GLOBAL      PRODUCT        GLOBAL          PRICE_CUBE   CLASS             1
GLOBAL      PRODUCT        GLOBAL          PRICE_CUBE   FAMILY            1
GLOBAL      PRODUCT        GLOBAL          PRICE_CUBE   ITEM              1
```

   All the levels are initially selected with "1" in the SELECTED column.

4. Since you want the materialized view to include only product families by month and product classes by quarter, you can deselect all other levels. You could edit the table with a statement like the following.

```
update SYS.OLAPTABLEVELS set selected = 0
   where LEVEL_NAME in ('ITEM','TOTAL_PRODUCT', 'Year');
select * from sys.olaptablevels;

SCHEMA_NAME DIMENSION_NAME DIMENSION_OWNER CUBE_NAME    LEVEL_NAME SELECTED
----------- -------------- --------------- ----------   ---------- --------
GLOBAL      TIME           GLOBAL          PRICE_CUBE   Year            0
GLOBAL      TIME           GLOBAL          PRICE_CUBE   Quarter         1
GLOBAL      TIME           GLOBAL          PRICE_CUBE   Month           1
GLOBAL      PRODUCT        GLOBAL          PRICE_CUBE   TOTAL_PRODUCT   0
GLOBAL      PRODUCT        GLOBAL          PRICE_CUBE   CLASS           1
GLOBAL      PRODUCT        GLOBAL          PRICE_CUBE   FAMILY          1
GLOBAL      PRODUCT        GLOBAL          PRICE_CUBE   ITEM            0
```

5. Next create the table `sys.olaptableveltuples`. This table, which is also a session-specific temporary table, contains all the possible combinations of the levels that you selected in the previous step. Each combination of levels, or grouping set, has an identification number. All the grouping sets are initially selected with "1" in the `SELECTED` column.

```
exec dbms_odm.createcubeleveltuple('global','price_cube');
select * from sys.olaptableveltuples;

ID SCHEMA_NAME CUBE_NAME  DIMENSION_NAME DIMENSION_OWNER LEVEL_NAME SELECTED
-- ----------- ---------  -------------- --------------- ---------- --------
7   GLOBAL     PRICE_CUBE   PRODUCT      GLOBAL          CLASS        1
7   GLOBAL     PRICE_CUBE   TIME         GLOBAL          Quarter      1
6   GLOBAL     PRICE_CUBE   PRODUCT      GLOBAL          FAMILY       1
6   GLOBAL     PRICE_CUBE   TIME         GLOBAL          Quarter      1
3   GLOBAL     PRICE_CUBE   PRODUCT      GLOBAL          CLASS        1
3   GLOBAL     PRICE_CUBE   TIME         GLOBAL          Month        1
2   GLOBAL     PRICE_CUBE   PRODUCT      GLOBAL          FAMILY       1
2   GLOBAL     PRICE_CUBE   TIME         GLOBAL          Month        1
```

6. Since you want the materialized view to include only product families by month and product classes by quarter, you can deselect the other level combinations. You could edit the `sys.olaptableveltuples` table with a statement like the following.

```
update SYS.OLAPTABLEVELTUPLES set selected = 0
        where ID in ('6', '3');
select * from sys.olaptableveltuples where SELECTED = 1;

ID SCHEMA_NAME CUBE_NAME  DIMENSION_NAME DIMENSION_OWNER LEVEL_NAME SELECTED
-- ----------- ---------  -------------- --------------- ---------- --------
```

```
7   GLOBAL     PRICE_CUBE   PRODUCT      GLOBAL          CLASS        1
7   GLOBAL     PRICE_CUBE   TIME         GLOBAL          Quarter      1
2   GLOBAL     PRICE_CUBE   PRODUCT      GLOBAL          FAMILY       1
2   GLOBAL     PRICE_CUBE   TIME         GLOBAL          Month        1
```

7. To create the script that will generate the fact materialized view, run the CREATEFACTMV_GS procedure.

```
exec dbms_odm.createfactmv_gs
      ('global','price_cube',
       'price_cost_mv','/users/global/scripts',TRUE);
```

The CREATE MATERIALIZED VIEW statement in the script contains the following two grouping sets in the GROUP BY GROUPING SETS clause.

```
GROUP BY GROUPING SETS (

(TIME_DIM.YEAR_ID, TIME_DIM.QUARTER_ID, TIME_DIM.MONTH_ID,
 PRODUCT_DIM.TOTAL_PRODUCT_ID, PRODUCT_DIM.CLASS_ID, PRODUCT_DIM.FAMILY_ID),

(TIME_DIM.YEAR_ID, TIME_DIM.QUARTER_ID,
 PRODUCT_DIM.TOTAL_PRODUCT_ID, PRODUCT_DIM.CLASS_ID) )
```

The final statement in the script sets the mv_summary_code associated with the cube in the OLAP Catalog. This setting indicates that the materialized view associated with this cube is in grouping set form.

```
execute cwm2_olap_cube.set_mv_summary_code
     ('GLOBAL', 'PRICE_CUBE', 'GROUPINGSET') ;
```

8. Go to the /users/global/scripts directory and run the price_cost_mv script to create the fact materialized view.

Summary of DBMS_ODM Subprograms

Table 24–1 DBMS_ODM Subprograms

Subprogram	Description
CREATECUBELEVELTUPLE Procedure on page 24-8	Creates a table of level combinations to be included in the materialized view for a cube.
CREATEDIMLEVTUPLE Procedure on page 24-9	Creates a table of levels to be included in the materialized view for a cube.
CREATEDIMMV_GS Procedure on page 24-10	Generates a script that creates a materialized view for each hierarchy of a dimension.
CREATEFACTMV_GS Procedure on page 24-11	Generates a script that creates a materialized view for the fact table associated with a cube. The materialized view includes individual level combinations that you have previously specified.
CREATESTDFACTMV Procedure on page 24-12	Generates a script that creates a materialized view for the fact table associated with a cube. The materialized view is automatically constructed according to general instructions that you provide.

CREATECUBELEVELTUPLE Procedure

This procedure creates the table `sys.olaptableveltuples`, which lists all the level combinations to be included in the materialized view for the cube. By default, all level combinations are selected for inclusion in the materialized view. You can edit the table to deselect any level combinations that you do not want to include.

Before calling this procedure, call `CREATEDIMLEVTUPLE` to create the table of levels for the cube.

Syntax

```
CREATECUBELEVELTUPLE (
               cube_owner     IN   VARCHAR2,
               cube_name      IN   VARCHAR2);
```

Parameters

Table 24–2 CREATECUBELEVELTUPLE Procedure Parameters

Parameter	Description
cube_owner	Owner of the cube.
cube_name	Name of the cube.

See Also

"Procedure: Manually Generate the Materialized Views" on page 24-4

"Example: Create Materialized Views for a Sales Cube" on page 24-5

CREATEDIMLEVTUPLE Procedure

This procedure creates the table sys.olaptablevels, which lists all the levels of all the dimensions of the cube. By default, all levels are selected for inclusion in the materialized view. You can edit the table to deselect any levels that you do not want to include.

After calling this procedure, call CREATECUBELEVELTUPLE to create the table of level combinations (level tuples) for the cube.

Syntax

```
CREATEDIMLEVTUPLE (
              cube_owner    IN varchar2,
              cube_name     IN varchar2);
```

Parameters

Table 24–3 CREATEDIMLEVTUPLE Procedure Parameters

Parameter	Description
cube_owner	Owner of the cube.
cube_name	Name of the cube.

See Also

"Procedure: Manually Generate the Materialized Views" on page 24-4

"Example: Create Materialized Views for a Sales Cube" on page 24-5

CREATEDIMMV_GS Procedure

This procedure generates a script that creates a materialized view for each hierarchy of a dimension. You must call this procedure for each dimension of a cube.

The process of creating the dimension materialized views is the same whether you generate the fact materialized view automatically or manually.

Syntax

```
CREATEDIMMV_GS (
               dimension_owner    IN   VARCHAR2,
               dimension_name     IN   VARCHAR2,
               output_file        IN   VARCHAR2,
               output_path        IN   VARCHAR2,
               tablespace_mv      IN   VARCHAR2 DEFAULT NULL,
               tablespace_index   IN   VARCHAR2 DEFAULT NULL);
```

Parameters

Table 24–4 CREATEDIMMV_GS Procedure Parameters

Parameter	Description
dimension_owner	Owner of the dimension.
`dimension_name`	Name of the dimension.
`output_file`	File name for the output script.
`output_path`	Directory path where `output_file` will be created. This may be either a directory object or a path set by the `UTL_FILE_DIR` parameter.
`tablespace_mv`	The name of the tablespace in which the materialized view will be created. When this parameter is omitted, the materialized view is created in the user's default tablespace.
`tablespace_index`	The name of the tablespace in which the index for the materialized view will be created. When this parameter is omitted, the index is created in the user's default tablespace.

See Also

"Procedure: Automatically Generate the Materialized Views" on page 24-3

"Procedure: Manually Generate the Materialized Views" on page 24-4

"Example: Create Materialized Views for a Sales Cube" on page 24-5

CREATEFACTMV_GS Procedure

This procedure generates a script that creates a materialized view for the fact table associated with a cube.

Prior to calling this procedure, you must call CREATEDIMLEVTUPLE and CREATECUBELEVELTUPLE to create the sys.olaptableveltuples table. The materialized view will include all level combinations selected in the sys.olaptableveltuples table.

Syntax

```
CREATEFACTMV_GS (
               cube_owner              IN   VARCHAR2,
               cube_name               IN   VARCHAR2,
               outfile                 IN   VARCHAR2,
               outfile_path            IN   VARCHAR2,
               partitioning            IN   BOOLEAN,
               tablespace_mv           IN   VARCHAR2 DEFAULT NULL,
               tablespace_index        IN   VARCHAR2 DEFAULT NULL);
```

Parameters

Table 24–5 CREATEFACTMV_GS Procedure Parameters

Parameter	Description
cube_owner	Owner of the cube.
cube_name	Name of the cube.
output_file	File name for the output script.
output_path	Directory path where output_file will be created.This may be either a directory object or a path set by the UTL_FILE_DIR parameter.
partitioning	TRUE turns on index partitioning; FALSE turns it off.
tablespace_mv	The name of the tablespace in which the materialized view will be created. When this parameter is omitted, the materialized view is created in the user's default tablespace.
tablespace_index	The name of the tablespace in which the index for the materialized view will be created. When this parameter is omitted, the index is created in the user's default tablespace.

See Also

"Summarizing the Fact Table" on page 24-2

"Example: Create Materialized Views for a Sales Cube" on page 24-5

CREATESTDFACTMV Procedure

This procedure generates a script that creates a materialized view for the fact table associated with a cube.

This procedure automatically generates and updates the tables of levels and level tuples. If you want to edit these tables yourself, you must use the CREATEDIMLEVTUPLE, CREATECUBELEVELTUPLE, and CREATEFACTMV_GS procedures.

Syntax

```
CREATESTDFACTMV (
               cube_owner              IN   VARCHAR2,
               cube_name               IN   VARCHAR2,
               outfile                 IN   VARCHAR2,
               outfile_path            IN   VARCHAR2,
               partitioning            IN   BOOLEAN,
               materialization_level   IN   VARCHAR2,
               tablespace_mv           IN   VARCHAR2 DEFAULT NULL,
               tablespace_index        IN   VARCHAR2 DEFAULT NULL);
```

Parameters

Table 24–6 CREATESTDFACTMV Procedure Parameters

Parameter	Description
cube_owner	Owner of the cube.
cube_name	Name of the cube.
output_file	File name for the output script.
output_path	Directory path where output_file will be created.This may be either a directory object or a path set by the UTL_FILE_DIR parameter.
partitioning	TRUE turns on index partitioning; FALSE turns it off.
tablespace_mv	The name of the tablespace in which the materialized view will be created. When this parameter is omitted, the materialized view is created in the user's default tablespace.

Table 24–6 (Cont.) CREATESTDFACTMV Procedure Parameters

Parameter	Description
materialization_level	The level of materialization. This parameter identifies the level combinations that will be included in the materialized view. Specify one of the following values: ■ FULL — Fully materialize the cube's data. Include every level combination in the materialized view. ■ MINIMUM — Minimally materialize the cube's data. Include the level above the leaf level for each dimension and the most aggregate level for each dimension in the materialized view. ■ PERCENT — Materialize the cube's data based on a percentage of the cube's level combinations. For example, consider a cube that has two dimensions with three levels and one dimension with four levels. This cube has 36 possible level combinations (3*3*4). If you choose to materialize the cube by 30%, then 12 level combinations will be included in the materialized view.
tablespace_index	The name of the tablespace in which the index for the materialized view will be created. When this parameter is omitted, the index is created in the user's default tablespace.

See Also

"Summarizing the Fact Table" on page 24-2

25

OLAP_API_SESSION_INIT

The OLAP_API_SESSION_INIT package provides procedures for maintaining a table of initialization parameters for the OLAP API.

This chapter contains the following topics:

- Initialization Parameters for the OLAP API
- Viewing the Configuration Table
- Summary of OLAP_API_SESSION_INIT Subprograms

Initialization Parameters for the OLAP API

The OLAP_API_SESSION_INIT package contains procedures for maintaining a configuration table of initialization parameters. When the OLAP API opens a session, it executes the ALTER SESSION commands listed in the table for any user who has the specified roles. Only the OLAP API uses this table; no other type of application executes the commands stored in it.

This functionality provides an alternative to setting these parameters in the database initialization file or the init.ora file, which would alter the environment for all users.

During installation, the table is populated with ALTER SESSION commands that have been shown to enhance the performance of the OLAP API. Unless new settings prove to be more beneficial, you do not need to make changes to the table.

The information in the table can be queried through the ALL_OLAP_ALTER_SESSION view alias, which is also described in this chapter.

Note: This package is owned by the SYS user. You must explicitly be granted execution rights before you can use it.

Viewing the Configuration Table

ALL_OLAP_ALTER_SESSION is the public synonym for V$OLAP_ALTER_SESSION, which is a view for the OLAP$ALTER_SESSION table. The view and table are owned by the SYS user.

ALL_OLAP_ALTER_SESSION View

Each row of ALL_OLAP_ALTER_SESSION identifies a role and a session initialization parameter. When a user opens a session using the OLAP API, the session is initialized using the parameters for roles granted to that user. For example, if there are rows for the OLAP_DBA role and the SELECT_CATALOG_ROLE, and a user has the OLAP_DBA role, then the parameters for the OLAP_DBA role will be set, but those for the SELECT_CATALOG_ROLE will be ignored.

Table 25–1 ALL_OLAP_ALTER_SESSION Column Descriptions

Column	Datatype	NULL	Description
ROLE	VARCHAR2(30)	NOT NULL	A database role
CLAUSE_TEXT	VARCHAR2(3000)		An ALTER SESSION command

Summary of OLAP_API_SESSION_INIT Subprograms

The following table describes the subprograms provided in OLAP_API_SESSION_INIT.

Table 25–2 OLAP_API_SESSION_INIT Subprograms

Subprogram	Description
ADD_ALTER_SESSION Procedure on page 25-3	Specifies an ALTER SESSION parameter for OLAP API users with a particular database role.
CLEAN_ALTER_SESSION Procedure on page 25-4	Removes orphaned data, that is, any ALTER SESSION parameters for roles that are no longer defined in the database.
DELETE_ALTER_SESSION Procedure on page 25-4	Removes a previously defined ALTER SESSION parameter for OLAP API users with a particular database role.

ADD_ALTER_SESSION Procedure

This procedure specifies an ALTER SESSION parameter for OLAP API users with a particular database role. It adds a row to the OLAP$ALTER_SESSION table.

Syntax

```
ADD_ALTER_SESSION (
     role_name             IN     VARCHAR2,
     session_parameter     IN     VARCHAR2);
```

Parameters

The role_name and session_parameter are added as a row in OLAP$ALTER_SESSION.

Table 25–3 ADD_ALTER_SESSION Procedure Parameters

Parameter	Description
role_name	The name of a valid role in the database. Required.
session_parameter	A parameter that can be set with a SQL ALTER SESSION command. Required.

Example

The following call inserts a row in OLAP$ALTER_SESSION that turns on query rewrite for users with the OLAP_DBA role.

```
call olap_api_session_init.add_alter_session(
     'OLAP_DBA', 'SET QUERY_REWRITE_ENABLED=TRUE');
```

The ALL_OLAP_ALTER_SESSION view now contains the following row.

```
ROLE                CLAUSE TEST
-----------------   ----------------------------------------------------
OLAP_DBA            ALTER SESSION SET QUERY_REWRITE_ENABLED=TRUE
```

CLEAN_ALTER_SESSION Procedure

This procedure removes all ALTER SESSION parameters for any role that is not currently defined in the database. It removes all orphaned rows in the OLAP$ALTER_SESSION table for those roles.

Syntax

```
CLEAN_ALTER_SESSION ();
```

Examples

The following call deletes all orphaned rows.

```
call olap_api_session_init.clean_alter_session();
```

DELETE_ALTER_SESSION Procedure

This procedure removes a previously defined ALTER SESSION parameter for OLAP API users with a particular database role. It deletes a row from the OLAP$ALTER_SESSION table.

Syntax

```
DELETE_ALTER_SESSION (
     role_name           IN     VARCHAR2,
     session_parameter   IN     VARCHAR2);
```

Parameters

The role_name and session_parameter together uniquely identify a row in OLAP$ALTER_SESSION.

Table 25–4 DELETE_ALTER_SESSION Procedure Parameters

Parameter	Description
role_name	The name of a valid role in the database. Required.
session_parameter	A parameter that can be set with a SQL `ALTER SESSION` command. Required.

Examples

The following call deletes a row in `OLAP$ALTER_SESSION` that contains a value of `OLAP_DBA` in the first column and `QUERY_REWRITE_ENABLED=TRUE` in the second column.

```
call olap_api_session_init.delete_alter_session(
     'OLAP_DBA', 'SET QUERY_REWRITE_ENABLED=TRUE');
```

26

OLAP_TABLE

The OLAP_TABLE function extracts multidimensional data from an analytic workspace and presents it in the two-dimensional format of a relational table. It provides access to analytic workspace data from SQL.

This chapter contains the following topics:

- OLAP_TABLE Syntax
- OLAP_TABLE Examples

See Also:

- *Oracle OLAP Application Developer's Guide*
- *Oracle OLAP DML Reference*

OLAP_TABLE Syntax

The OLAP_TABLE function returns the table of objects which has been populated according to the mapping rules defined in *limit_map*.

The order in which OLAP_TABLE processes information specified in input parameters is described in "Order of Processing in OLAP_TABLE" on page 26-12.

Syntax

```
OLAP_TABLE(
      aw_attach          IN   VARCHAR2,
      table_name         IN   VARCHAR2,
      datamap            IN   VARCHAR2,
      limit_map          IN   VARCHAR2 );
```

Parameters

Table 26–1 OLAP_TABLE Function Parameters

Parameter	Description
aw_attach	The name of the analytic workspace with the source data. See "AW Attach Parameter" on page 26-2.
table_name	The name of a table of objects that has been defined to structure the multidimensional data in tabular form. See "Table Name Parameter" on page 26-3.
datamap	An optional OLAP DML command that controls data mapping as an alternative to the limit map. See "Datamap Parameter" on page 26-5.
limit_map	A keyword-based map that identifies the source objects in *aw_attach* and the target columns in *table_name*. See "Limit Map Parameter" on page 26-6.

AW Attach Parameter

The first parameter of the OLAP_TABLE function provides the name of the analytic workspace where the source data is stored and specifies how long the analytic workspace will be attached to your OLAP session, which opens on your first call to OLAP_TABLE. This is the full syntax of this parameter:

```
'[owner.]aw_name DURATION QUERY | SESSION'
```

For example:

```
'sys.xademo DURATION SESSION'
```

owner

Specify *owner* whenever you are creating views that will be accessed by other users. Otherwise, you can omit the *owner* if you own the analytic workspace. It is required only when you are logged in under a different user name than the owner.

QUERY

Attaches an analytic workspace for the duration of a single query. Use `QUERY` only when you need to see updates to the analytic workspace made in other sessions.

SESSION

`SESSION` attaches an analytic workspace and keeps it attached at the end of the query. It provides better performance than `QUERY` because it keeps the OLAP session open. This performance difference is significant when the function is called without either a *table_name* parameter or `AS` clauses in the limit map; in this case, the `OLAP_TABLE` function must determine the appropriate table definition.

Table Name Parameter

The second parameter identifies the name of a table of objects. The syntax of this parameter is:

```
'table_name'
```

For example:

```
'product_dim_tbl'
```

If you use *table_name*, then you cannot use `AS` clauses in the limit map.

If you omit the *table_name* parameter, then `OLAP_TABLE` converts the analytic workspace data types to SQL data types, as shown in Table 26–2. You can override these defaults by using `AS` clauses in the limit map

Table 26–2 Default Data Type Conversions

Analytic Workspace Data Type	SQL Data Type
`ID`	`CHAR(8)`
`TEXT`	`VARCHAR2(4000)`

Table 26–2 Default Data Type Conversions

Analytic Workspace Data Type	SQL Data Type
`TEXT` *(n)*	`VARCHAR2` *(n)*
`NTEXT`	`NVARCHAR2(4000)`
`NTEXT` *(n)*	`NVARCHAR2` *(n)*
`NUMBER`	`NUMBER`
`NUMBER` *(p,s)*	`NUMBER` *(p,s)*
`LONGINTEGER`	`NUMBER(19)`
`INTEGER`	`NUMBER(10)`
`SHORTINTEGER`	`NUMBER(5)`
`INTEGER WIDTH 1`	`NUMBER(3)`
`BOOLEAN`	`NUMBER(1)`
`DECIMAL`	`BINARY_DOUBLE`
`SHORTDECIMAL`	`BINARY_FLOAT`
`DATE`	`DATE`
`DAY, WEEK, MONTH, QUARTER, YEAR`	`DATE`
`DATETIME`	`TIMESTAMP`
`COMPOSITE`	`VARCHAR2(4000)`
Other	`VARCHAR2(4000)`

Creating a Table of Objects

A user-defined object type is composed of attributes, which are equivalent to the columns of a table.

This is the basic syntax for defining a row:

```
CREATE TYPE object_name AS OBJECT (
   attribute1      datatype,
   attribute2      datatype,
   attributen      datatype;
```

A table type is a collection of object types; this collection is equivalent to the rows of a table. This is the basic syntax for creating a table type:

```
CREATE TYPE table_name AS TABLE OF object_name;
```

See Also: *Oracle Database Application Developer's Guide - Object-Relational Features* for information about object types

Datamap Parameter

The third parameter of the OLAP_TABLE function is a single OLAP DML command. It is called a datamap because its primary use is to manually control the mapping of data sources, using the OLAP DML FETCH command. The *datamap* parameter is also used for selections and calculations that cannot be performed in the limit map. It is an optional parameter and is typically omitted.

The order in which OLAP_TABLE processes the *datamap* parameter is specified in "Order of Processing in OLAP_TABLE" on page 26-12.

The syntax of this parameter is:

```
'olap_command'
```

Using the FETCH Command

FETCH specifies explicitly how analytic workspace data is mapped to a table object. The basic syntax is:

```
FETCH expression...
```

Enter one expression for each target column, listing the expressions in the same order they appear in the row definition. Separate expressions with spaces or commas.You must enter the entire statement on one line, without line breaks or continuation marks of any type.

Note: Use the FETCH keyword in OLAP_TABLE only if you are migrating an Express Server application that used the FETCH command for SNAPI. In that case, note that the full syntax is the same in Oracle as in Express 6.3. You can use the same FETCH commands in OLAP_TABLE that you used previously in SNAPI.

Using a Limit Map With FETCH

When you use FETCH, the limit map is not required; if you do not provide a limit map or omit its DIMENSION clauses, then you must use the *table_name* parameter. The MEASURE and LOOP clauses of a limit map are irrelevant when used with FETCH.

Limit Map Parameter

The fourth (and last) parameter of the OLAP_TABLE function maps workspace objects to columns in the table and identifies the role of each one. It is called a limit map because it combines with the WHERE clause of a SQL SELECT statement to issue a series of LIMIT commands to the analytic workspace. The contents of the limit map populate the table specified in the *table_name* parameter.

The order in which OLAP_TABLE processes information in the limit map is specified in "Order of Processing in OLAP_TABLE" on page 26-12.

If you are using a FETCH command in the *datamap* parameter, you typically omit the limit map.

All or part of the limit map can be stored in a text variable in the analytic workspace. To insert the variable in the limit map, precede the name of the variable with an ampersand (&). This practice is called ampersand substitution in the OLAP DML.

If you supply the limit map as text in the SELECT statement, then it has a maximum length of 4000 characters, which is imposed by PL/SQL. If you store the limit map in the analytic workspace, then the limit map has no maximum length.

The syntax of the limit map has numerous clauses, primarily for defining dimension hierarchies. Pay close attention to the presence or absence of commas, since syntax errors will prevent your limit map from being parsed.

Example 26–1 Syntax of the Limit Map Parameter of OLAP_TABLE

```
'[MEASURE column [AS datatype] FROM {measure | AW_EXPR expression}]
           .
           .
           .
 DIMENSION [column [AS datatype] FROM] dimension
    [WITH
       [HIERARCHY [column [AS datatype] FROM] hierarchy_relation
            [(hierarchy_dimension 'hierarchy')]
          [INHIERARCHY inhierarchy_obj]
          [GID column [AS datatype] FROM gid_variable]
          [PARENTGID column [AS datatype] FROM gid_variable]
          [FAMILYREL col1 [AS datatype], col2 [AS datatype],
               coln [AS datatype] FROM
             {expression1, expression2, expressionn |
              family_relation USING level_dimension }
             [LABEL label_variable]]
           .
           .
           .
```

```
        ]
        [ATTRIBUTE column [AS datatype] FROM attribute_variable]
            .
            .
            .
     ]
 [ROW2CELL column]
 [LOOP composite_dimension]
 [PREDMLCMD olap_command]
 [POSTDMLCMD olap_command]
'
```

Where:

column is the name of a column in the target table.

measure is a business measure that is stored in the analytic workspace.

dimension is a dimension in the analytic workspace

expression is a formula or qualified data reference for objects in the analytic workspace

hierarchy_relation is a self-relation in the analytic workspace that defines the hierarchies for *dimension*.

hierarchy_dimension is a dimension in the analytic workspace that contains the names of the hierarchies for *dimension*.

hierarchy is a member of *hierarchy_dimension*.

inhierarchy_obj is either a valueset or a Boolean variable in the analytic workspace. It identifies whether a dimension member is in *hierarchy*. A valueset is more efficient than a Boolean variable.

gid_variable is the name of a variable in the analytic workspace that contains the grouping ID of each dimension member.

attribute_variable is the name of a variable in the analytic workspace that contains attribute values for *dimension*.

composite_dimension is the name of a composite dimension used in the definition of *measure*.

datamap is an OLAP DML command.

Table 26–3 Components of the OLAP_TABLE Limit Map

Keyword	Keyword Clause Syntax and Description

MEASURE	**MEASURE column [AS datatype] FROM {measure | AW_EXPR expression}**
	The MEASURE clause maps a variable, formula, or relation in the analytic workspace to a column in the target table.
	The AS clause specifies the data type (such as NUMBER(12) or VARCHAR2(30)) for the column. You cannot use AS if you use the OLAP_TABLE *table_name* clause. If you do not specify AS or *table_name*, then the OLAP_TABLE function determines the best data type.
	Alternatively, the AW_EXPR keyword can map a calculation performed by the OLAP engine on one or more of these objects to a column. For example, you could specify calculations such as these:
	`analytic_cube_sales - analytic_cube_cost`
	or
	`LAGDIF(analytic_cube_sales, 1, time, LEVELREL time.lvlrel)`
	You can list any number of MEASURE clauses. This clause is optional when, for example, you wish to create a dimension view.

Table 26–3 (Cont.) Components of the OLAP_TABLE Limit Map

Keyword	Keyword Clause Syntax and Description

DIMENSION	**DIMENSION [column [AS datatype] FROM] dimension...**
	The DIMENSION clause identifies a dimension or conjoint in the analytic workspace that dimensions one or more measures, attributes, or hierarchies in the limit map.
	The *column* subclause is optional when you do not want the dimension members themselves to be represented in the table. In this case, you should include a dimension attribute that can be used for data selection.
	The AS clause specifies the data type (such as NUMBER(12) or VARCHAR2(30)) for the column. You cannot use AS if you use the OLAP_TABLE *table_name* clause. If you do not specify AS or *table_name*, then the OLAP_TABLE function determines the best data type.
	Every limit map should have at least one DIMENSION clause. If the limit map contains MEASURE clauses, then it should also contain a single DIMENSION clause for each dimension of the measures, unless a dimension is being limited to a single value. If the measures are dimensioned by a composite, then you must identify each dimension in the composite with a DIMENSION clause. For the best performance when fetching a large result set, identify the composite in a LOOP clause.
	A dimension can be named in only one DIMENSION clause. Subclauses of DIMENSION identify the dimension hierarchy and attributes.
	The WITH clause introduces a HIERARCHY or ATTRIBUTE subclause. If you omit these subclauses from the limit map, then omit the WITH clause also. However, if you include one or both of these subclauses, then precede them with a single WITH clause. The syntax of the WITH clause is summarized as follows. See Table 26–4 for complete descriptions of each component.
	`[WITH`
	`    [HIERARCHY [column [AS datatype FROM]`
	`        hierarchy_relation[(hierarchy_dimension 'hierarchy')]`
	`    [INHIERARCHY inhierarchy_variable]`
	`    [GID column [AS datatype] FROM gid_variable]`
	`    [PARENTGID column [AS datatype] FROM gid_variable]`
	`    [FAMILYREL col1 [AS datatype], col2 [AS datatype],`
	`        coln [AS datatype] FROM`
	`          {expression1, expression2, expressionn |`
	`             family_relation USING level_dimension }`
	`          [LABEL label_variable]]`
	`    . . .]`
	`    [ATTRIBUTE column [AS datatype] FROM attribute_variable]`
	`    . . .]`

ROW2CELL	**ROW2CELL column** The ROW2CELL clause populates a RAW(32) column with information needed by the single-row functions in the DBMS_AW package. Use this clause when creating a view that will be used by these functions. See "OLAP_EXPRESSION Function" on page 21-26.
LOOP	**LOOP sparse_dimension** The LOOP clause identifies a single named composite that dimensions one or more measures specified in the limit map. It improves performance when fetching a large result set; however, it can slow the retrieval of a small number of values.

Table 26–3 (Cont.) Components of the OLAP_TABLE Limit Map

Keyword	Keyword Clause Syntax and Description
PREDMLCMD	**PREDMLCMD olap_command** The PREDMLCMD specifies an OLAP DML command that is executed before the data is fetched from the analytic workspace into the target table. It can be used, for example, to execute a model or forecast whose results will be fetched into the table. The results of the command are in effect during execution of the limit map, and continue into your session after execution of OLAP_TABLE is complete. See "Order of Processing in OLAP_TABLE" on page 26-12.
POSTDMLCMD	**POSTDMLCMD olap_command** The POSTDMLCMD specifies an OLAP DML command that is executed after the data is fetched from the analytic workspace into the target table. It can be used, for example, to delete objects or data that were created by commands in the PREDMLCMD clause, or to restore the dimension status that was changed in a PREDMLCMD clause. See"Order of Processing in OLAP_TABLE".

Table 26–4 WITH Subclause of DIMENSION Clause of OLAP_TABLE Limit Map

<table>
<tr><th>Keyword</th><th>Component</th></tr>
<tr><td><code>HIERARCHY</code></td><td>
<code>HIERARCHY [column [AS datatype] FROM] hierarchy_relation[(hierarchy_dimension 'hierarchy')]...</code>
The <code>HIERARCHY</code> subclause identifies the parent self-relation in the analytic workspace that defines the hierarchies for dimension.
The <code>AS</code> clause specifies the data type (such as <code>NUMBER(12)</code> or <code>VARCHAR2(30)</code>) for the column. You cannot use <code>AS</code> if you use the <code>OLAP_TABLE</code> table_name clause. If you do not specify <code>AS</code> or table_name, then the <code>OLAP_TABLE</code> function determines the best data type.
If hierarchy_dimension has more than one member, then you can specify the one that you want with a (hierarchy_dimension 'hierarchy') phrase. To include multiple hierarchies, specify a <code>HIERARCHY</code> subclause for each one. The hierarchy_dimension is limited to hierarchy for all workspace objects that are referenced in subsequent subclauses (that is, <code>INHIERARCHY</code>, <code>GID</code>, <code>PARENTGID</code>, and <code>FAMILYREL</code>).
The <code>HIERARCHY</code> subclause is optional when dimension does not have a hierarchy, or when the status of dimension has been limited to a single level of the hierarchy.

 <code>INHIERARCHY inhierarchy_variable</code>
The <code>INHIERARCHY</code> subclause identifies a boolean variable in the analytic workspace that identifies whether a dimension member is in hierarchy. It is required only when there are members of the dimension that are omitted from the hierarchy, which is typical when a dimension has multiple hierarchies.
 <code>GID column [AS datatype] FROM gid_variable</code>
The <code>GID</code> subclause maps an integer variable in the analytic workspace, which contains the grouping ID for each dimension member, to a column in the target table. It is required for Java applications that use the OLAP API.
 <code>PARENTGID column [AS datatype] FROM gid_variable</code>
The <code>PARENTGID</code> subclause calculates the grouping IDs for the parent relation using the GID variable in the analytic workspace. The parent GIDs are not stored in a workspace object. Instead, you specify the same GID variable for the <code>PARENTGID</code> clause that you used in the GID clause.
The <code>PARENTGID</code> clause is recommended for Java applications that use the OLAP API.
 <code>FAMILYREL col1 [AS datatype], col2 [AS datatype], coln [AS datatype] FROM {expression1, expression2, expressionn | family_relation USING level_dimension } [LABEL label_variable]</code>
The <code>FAMILYREL</code> subclause is used primarily to map a family relation in the analytic workspace to multiple columns in the target table. List the columns in the order of level_dimension. If you do not want a particular level included, then specify null for the target column. The resulting view is in rollup form, in which each level of the hierarchy is represented in a separate column, and the full parentage of each dimension member is identified within the row.
The <code>FAMILYREL</code> subclause can also be used to map a list of qualified data references (QDRs) to multiple columns. In this usage, the first QDR maps to the first column, the second QDR maps to the second column, and so forth.
The <code>LABEL</code> keyword identifies a text attribute that provides more meaningful names for the dimension members.
You can use multiple <code>FAMILYREL</code> clauses for each hierarchy.

</td></tr>
</table>

Table 26–4 (Cont.) WITH Subclause of DIMENSION Clause of OLAP_TABLE Limit Map

Keyword	Component
ATTRIBUTE	**ATTRIBUTE column [AS datatype] FROM attribute_variable** The ATTRIBUTE clause maps a variable in the analytic workspace to a column in the target table. If attribute_variable has multiple dimensions, then values are mapped for all members of *dimension*, but only for the first member in the current status of additional dimensions. For example, if your attributes have a language dimension, then you must set the status of that dimension to a particular language. You can set the status of dimensions in a PREDMLCMD clause.

Order of Processing in OLAP_TABLE

The following list identifies the order in which the OLAP_TABLE function processes instructions that can change the status of dimensions in the analytic workspace.

1. Execute any OLAP DML command specified in the PREDMLCMD parameter of the limit map.
2. Save the current status of all dimensions so that it can be restored later (PUSH status).
3. Keep in status only those dimension values that are in the hierarchy specified by the INHIERARCHY clause (LIMIT KEEP).
4. Keep in status only those dimension values that satisfy the WHERE clause on the SQL SELECT statement containing the OLAP_TABLE function.
5. Execute any OLAP DML command specified in the *datamap* parameter of the OLAP_TABLE function.
6. Fetch the data.
7. Restore the status of all dimensions in the limit map (POP status).
8. Execute any OLAP DML command specified in the POSTDMLCMD parameter of the limit map.

OLAP_TABLE Examples

The examples show the two basic methods of using OLAP_TABLE:

- "Creating Views for the BI Beans and OLAP API" on page 26-14 uses a limit map. This is the most common use of OLAP_TABLE.
- "Using OLAP_TABLE with the FETCH Command" on page 26-18 uses the `FETCH` command. This method is for use only by Oracle Express Server applications that are being revised for use with Oracle Database.

Creating Views for the BI Beans and OLAP API

The examples provided here define a dimension view for the PRODUCT dimension and a measure view for the ANALYTIC_CUBE cube in the XADEMO sample analytic workspace. These are the type of views created by the OLAP API enabler. The data types of the columns are specified in the limit maps in AS clauses.

Note the use of a MODEL clause in the SELECT statements. The MODEL clause, when used with OLAP_TABLE, is an optimization that enables data to be fetched much faster from an analytic workspace. Refer to *Oracle OLAP Application Developer's Guide* for information about the use of arguments in the MODEL clause.

Creating a Dimension View

Example 26–2 creates a view named XADE_XADEM_XADEM_STAND4VIEW for the PRODUCT embedded total dimension in XADEMO. The third argument to OLAP_TABLE uses ampersand substitution to reference the limit map, which is stored in a variable named OLAP_SYS_LIMITMAP in the analytic workspace.

Example 26–2 Defining a PRODUCT Dimension View

```
CREATE OR REPLACE VIEW xademo.xade_xadem_xadem_stand4view AS SELECT * FROM
   TABLE(OLAP_TABLE('xademo.xademo DURATION SESSION',
   '',
   '',
   '&(xademo.xademo!olap_sys_limitmap(xademo.xademo!olap_sys_viewdim
       ''xade_xadem_xadem_stand4view''))'))
MODEL
   DIMENSION BY (
     product_et,
     product_gid)
   MEASURES (
     product_parent,
     product_parentgid,
     r2c,
     l4_equipment_parts,
     l3_components,
     l2_divisions,
     l1_total_products,
     aw_member_order,
     color,
     size_attr,
     long_description,
     short_description)
```

```
   RULES UPDATE();
```

Example 26–3 shows the contents of the limit map for the PRODUCT dimension. This limit map specifies the data types of the columns using AS clauses, instead of using the defaults.

Example 26–3 Limit Map for PRODUCT Dimension

```
DIMENSION PRODUCT_ET AS VARCHAR2(100) FROM XADEMO.XADEMO!PRODUCT WITH
   HIERARCHY PRODUCT_PARENT AS VARCHAR2(100) FROM
     XADEMO.XADEMO!PRODUCT_PARENTREL(XADEMO.XADEMO!PRODUCT_HIERLIST 1)
     INHIERARCHY XADEMO.XADEMO!PRODUCT_INHIER
     GID PRODUCT_GID AS NUMBER(12) FROM XADEMO.XADEMO!PRODUCT_GID
     PARENTGID PRODUCT_PARENTGID AS NUMBER(12) FROM XADEMO.XADEMO!PRODUCT_GID
     LEVELREL L4_Equipment_Parts AS VARCHAR2(100),
       L3_Components AS VARCHAR2(100),
       L2_Divisions AS VARCHAR2(100),
       L1_Total_Products AS VARCHAR2(100)
       FROM XADEMO.XADEMO!PRODUCT_FAMILYREL USING
         XADEMO.XADEMO!PRODUCT_LEVELLIST
   ATTRIBUTE AW_MEMBER_ORDER AS NUMBER FROM XADEMO.XADEMO!PRODUCT_ORDER
   ATTRIBUTE COLOR AS VARCHAR2(1000) FROM XADEMO.XADEMO!PRODUCT_COLOR
   ATTRIBUTE SIZE_ATTR AS VARCHAR2(1000) FROM XADEMO.XADEMO!PRODUCT_SIZE
   ATTRIBUTE LONG_DESCRIPTION AS VARCHAR2(1000) FROM
   XADEMO.XADEMO!PRODUCT_LONG_DESCRIPTION
   ATTRIBUTE SHORT_DESCRIPTION AS VARCHAR2(1000) FROM
     XADEMO.XADEMO!PRODUCT_SHORT_DESCRIPTION
   ROW2CELL R2C
   PREDMLCMD 'limit XADEMO.XADEMO!PRODUCT_HIERLIST to 1'
```

Creating a Measure View

Example 26–4 creates a view named XADEMO.XADE_XADEM_ANALY11VIEW for the measures in ANALYTIC_CUBE in XADEMO. The third argument to OLAP_TABLE uses ampersand substitution to reference the limit map, which is stored in a variable named OLAP_SYS_LIMITMAP in the analytic workspace. The OLAP API enabler stores all limit maps in this variable, which is dimensioned by OLAP_SYS_VIEWDIM so that the limit map for each view can be stored in a separate cell.

Note also how the MODEL clause is used in a measure view.

Example 26–4 Defining a Cube View

```
CREATE OR REPLACE VIEW xademo.xade_xadem_analy11view AS
   SELECT * FROM TABLE(OLAP_TABLE('xademo.xademo DURATION SESSION',
```

```
    '',
    '',
    '&(xademo.xademo!olap_sys_limitmap(xademo.xademo!olap_sys_viewdim
    ''xade_xadem_analy11view''))'))
  MODEL
    DIMENSION BY (
      channel_et,
      channel_gid,
      geography_et,
      geography_gid,
      product_et,
      product_gid,
      time_et,
      time_gid)
    MEASURES (
      analytic_cube_f_sales,
      analytic_cube_f_costs,
      analytic_cube_f_units,
      analytic_cube_f_quota,
      analytic_cube_f_promo,
      r2c)
    RULES UPDATE();
```

Example 26–5 shows the contents of the limit map for the measures in `ANALYTIC_CUBE`. This limit map specifies the data types of the columns using `AS` clauses, instead of using the defaults.

Example 26–5 Limit Map for ANALYTIC_CUBE

```
MEASURE ANALYTIC_CUBE_F_SALES AS NUMBER FROM
   XADEMO.XADEMO!ANALYTIC_CUBE_F_SALES
MEASURE ANALYTIC_CUBE_F_COSTS AS NUMBER FROM
   XADEMO.XADEMO!ANALYTIC_CUBE_F_COSTS
MEASURE ANALYTIC_CUBE_F_UNITS AS NUMBER FROM
   XADEMO.XADEMO!ANALYTIC_CUBE_F_UNITS
MEASURE ANALYTIC_CUBE_F_QUOTA AS NUMBER FROM
   XADEMO.XADEMO!ANALYTIC_CUBE_F_QUOTA
MEASURE ANALYTIC_CUBE_F_PROMO AS NUMBER FROM
   XADEMO.XADEMO!ANALYTIC_CUBE_F_PROMO
ROW2CELL R2C
DIMENSION CHANNEL_ET AS VARCHAR2(100) FROM XADEMO.XADEMO!CHANNEL
WITH HIERARCHY XADEMO.XADEMO!CHANNEL_PARENTREL(CHANNEL_HIERLIST 1)
INHIERARCHY XADEMO.XADEMO!CHANNEL_INHIER
GID CHANNEL_GID AS NUMBER(12) FROM XADEMO.XADEMO!CHANNEL_GID
DIMENSION GEOGRAPHY_ET AS VARCHAR2(100) FROM XADEMO.XADEMO!GEOGRAPHY
```

```
WITH HIERARCHY XADEMO.XADEMO!GEOGRAPHY_PARENTREL(GEOGRAPHY_HIERLIST 1)
INHIERARCHY XADEMO.XADEMO!GEOGRAPHY_INHIER
GID GEOGRAPHY_GID AS NUMBER(12) FROM XADEMO.XADEMO!GEOGRAPHY_GID
DIMENSION PRODUCT_ET AS VARCHAR2(100) FROM XADEMO.XADEMO!PRODUCT
WITH HIERARCHY XADEMO.XADEMO!PRODUCT_PARENTREL(PRODUCT_HIERLIST 1)
INHIERARCHY XADEMO.XADEMO!PRODUCT_INHIER
GID PRODUCT_GID AS NUMBER(12) FROM XADEMO.XADEMO!PRODUCT_GID
DIMENSION TIME_ET AS VARCHAR2(100) FROM XADEMO.XADEMO!TIME
WITH HIERARCHY XADEMO.XADEMO!TIME_PARENTREL(TIME_HIERLIST 2)
INHIERARCHY XADEMO.XADEMO!TIME_INHIER
GID TIME_GID AS NUMBER(12) FROM XADEMO.XADEMO!TIME_GID
```

Using OLAP_TABLE with the FETCH Command

The following example fetches data from two variables (SALES and COST) in the GLOBAL analytic workspace, and calculates two custom measures (COST_PRIOR_PERIOD and PROFIT). This example also shows the use of OLAP_TABLE directly by an application, without creating a view.

The data types of the columns are defined explicitly with CREATE TYPE statements. These user types can be saved permanently and used by multiple calls to OLAP_TABLE.

Example 26–6 Script Using FETCH with OLAP_TABLE

```
CREATE TYPE measure_row AS OBJECT (
   time                      VARCHAR2(20),
   geography                 VARCHAR2(30),
   product                   VARCHAR2(30),
   channel                   VARCHAR2(30),
   sales                     NUMBER(16),
   cost                      NUMBER(16),
   cost_prior_period         NUMBER(16),
   profit                    NUMBER(16));
/

CREATE TYPE measure_table AS TABLE OF measure_row;
/

SELECT time, geography, product, channel,
   sales, cost, cost_prior_period, profit
     FROM TABLE(OLAP_TABLE(
     'xademo DURATION SESSION',
     'measure_table',
     'FETCH time, geography, product, channel, analytic_cube_f.sales,
analytic_cube_f.costs, LAG(analytic_cube_f.costs, 1, time, LEVELREL time_member_levelrel),
analytic_cube_f.sales - analytic_cube_f.costs',
     ''))
   WHERE channel = 'STANDARD_2.TOTALCHANNEL' AND
         product = 'L1.TOTALPROD' AND
         geography = 'L1.WORLD'
   ORDER BY time;
```

This SQL SELECT statement returns the following result set:

```
TIME      GEOGRAPHY PRODUCT      CHANNEL                  SALES       COST    COST_PRIOR_PERIOD    PROFIT
--------- --------- ------------ ------------------------ --------- --------- ------------------ --------
L1.1996   L1.WORLD  L1.TOTALPROD STANDARD_2.TOTALCHANNEL 118247112   2490243                     115756869
L1.1997   L1.WORLD  L1.TOTALPROD STANDARD_2.TOTALCHANNEL  46412113   1078031            2490243   45334082
```

```
L2.Q1.96  L1.WORLD  L1.TOTALPROD STANDARD_2.TOTALCHANNEL  26084848    560379                   25524469
L2.Q1.97  L1.WORLD  L1.TOTALPROD STANDARD_2.TOTALCHANNEL  26501765    615399          560379   25886367
L2.Q2.96  L1.WORLD  L1.TOTALPROD STANDARD_2.TOTALCHANNEL  30468054    649004          615399   29819049
L2.Q2.97  L1.WORLD  L1.TOTALPROD STANDARD_2.TOTALCHANNEL  19910347    462632          649004   19447715
L2.Q3.96  L1.WORLD  L1.TOTALPROD STANDARD_2.TOTALCHANNEL  27781702    582693          462632   27199009
L2.Q4.96  L1.WORLD  L1.TOTALPROD STANDARD_2.TOTALCHANNEL  33912508    698166          582693   33214342
L3.APR96  L1.WORLD  L1.TOTALPROD STANDARD_2.TOTALCHANNEL   8859808    188851                    8670957
                                              .
                                              .
                                              .
27 rows selected.
```

Index

A

C

T

U

V

W

Y