MANAGING OLAP CATALOG METADATA IN ORACLE9I RELEASE 2

Shinji Matsumoto, IAF Software, Inc.
smatsumo@iafsoft.com

INTRODUCTION

Oracle9i R2 provides fully integrated multidimensional technology in the relational database. Oracle OLAP is the next generation of analytic servers and provides an upgrade path for Oracle Express users. This white paper explains how to manage OLAP Catalog Metadata, one of the important components in Oracle9i OLAP, from the perspective of the database administrator.

ORACLE9I R2 OLAP ARCHITECTURE

ORACLE9I OLAP COMPONENTS

Oracle OLAP consists of the following components:

- OLAP processing engine

  The OLAP calculation engine supports the selection and rapid calculation of multidimensional data. The OLAP engine runs within the Oracle kernel.

- Analytic Workspace

  An analytic workspace stores multidimensional data objects and procedures written in the OLAP DML. An analytic workspace can be temporary (that is, for the life of the session) or it can be persistent, that is, saved from one session to the next. When an analytic workspace is persistent, the data is stored as BLOBs in relational database tables. Analytic workspaces also provide an alternative to materialized views as a means of storing aggregate data.

- OLAP DML

  The OLAP DML is a data manipulation language that is understood by the Oracle OLAP calculation engine. The OLAP DML extends the analytical capabilities of querying languages such as SQL and the OLAP API to include forecasting, modeling, and what-if scenarios.

  OLAP DML commands and functions include the following categories:

  - Aggregation
  - Allocation
  - Data Selection
  - Date and Time Operations
  - File Reading and Writing
  - Financial Operations
  - Forecasts and Regressions
  - Numeric Manipulation
  - Models
  - Statistical Operations
  - Text Manipulation

  - Time Series Manipulation

- PL/SQL Table Functions

  SQL table functions can take a set of rows as input and produce a set of rows as output that can be queried like a physical database table. Application developers who use SQL can access SQL packages that use table functions to create views of multidimensional data. SQL applications can then access these views.

- OLAP API (Oracle's Java API for OLAP)

  The OLAP API is a Java-based application programming interface to Oracle OLAP that provides access to multidimensional data for analytical business application The OLAP API is the technology underlying the Oracle BI Beans for access to relational and multidimensional data.

- OLAP Catalog (OLAP metadata repositories)

  OLAP Catalog metadata is designed specifically for use with Oracle OLAP. OLAP applications can query this metadata repository to find out what data is available for analyses and display. The metadata contains information about the physical location of the data, that is, whether it is stored in a relational table or in an analytic workspace. The OLAP Catalog Metadata API (CWM2 PL/SQL packages) provides stored procedures for creating, dropping, and updating OLAP Catalog metadata.

- Tools within Oracle Enterprise Manager

  Oracle Enterprise Manager (OEM) provides enhanced OLAP management functions. OLAP Management is a tool in OEM that provides an easy-to-use graphical interface to the OLAP Catalog (release 1).

ORACLE9I OLAP ARCHITECTURE

Oracle9i R2 fully integrates multidimensional technology, including the former Oracle Express cubes, into the relational database. You can choose either relational table storage or multidimensional storage. Different types of applications can access both relational data and multidimensional data in different ways.

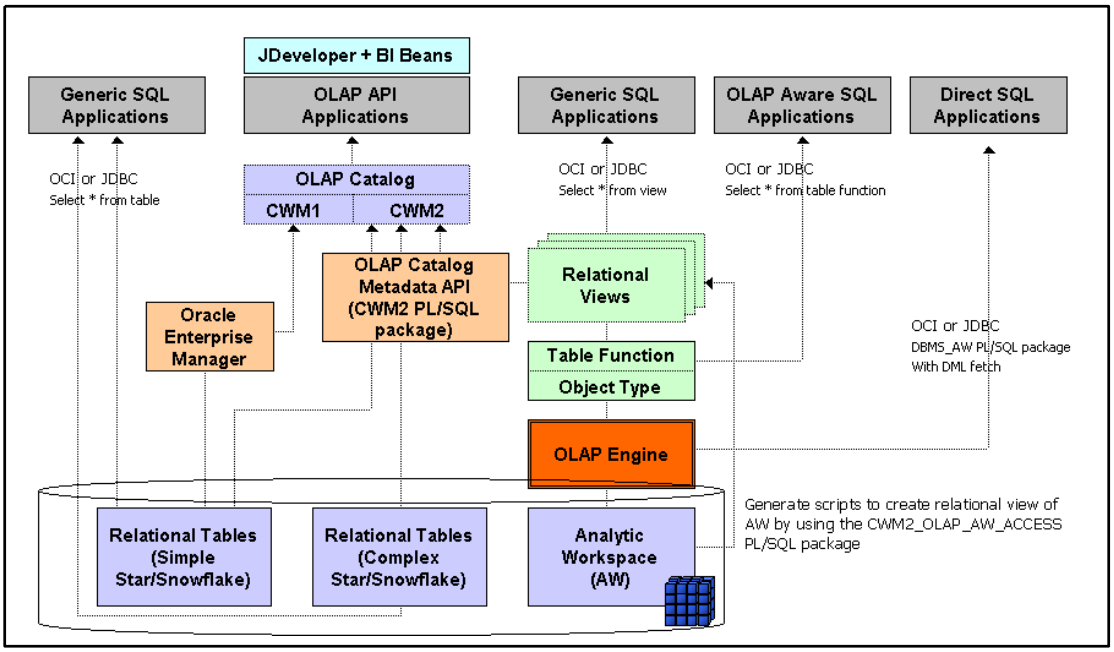

Figure 1: Oracle9i OLAP Architecture

OLAP METADATA MODEL AND OLAP CATALOG

OLAP METADATA MODEL

The basic data model in a relational database is a table composed of one or more columns of data. All of the data is stored in columns. In contrast, the basic data model for multidimensional analysis is a **cube**, which is composed of Measures, Dimensions, and Attributes.

Within the OLAP Catalog, you identify whether the data will function as a measure, a dimension, or an attribute. Once these decisions are stored in the OLAP Catalog metadata, the OLAP API can access warehouse data without regard to its underlying storage format. Whether the data is stored in relational tables, analytic workspaces, or some combination of relational and multidimensional schemas, the OLAP Catalog presents the same logical model to applications that use the OLAP API.

OLAP CATALOG

OLAP Catalog metadata is designed specifically for use with Oracle OLAP. It is required by the Java-based Oracle OLAP API, and can also be used by SQL-based applications to query the database. This repository is included by default with Enterprise Edition of the database when installed with either the General Purpose or Data Warehouse configuration.

The OLAP Catalog tables are owned by OLAPSYS. To create OLAP metadata stored in these tables, the user must have the OLAP_DBA role. A set of views, identified by the `ALL_OLAP2` prefix, presents the metadata in the

OLAP Catalog. OLAP Catalog includes two versions of metadata repository. CWM, first released with Oracle9i R1, is used by Oracle Enterprise Manager's OLAP Management future. The CWM2, newly available with Release 2(9.2), provides support for additional warehouse configurations. You can create and view CWM2 metadata by using the CWM2 PL/SQL packages and views. In other words, CWM2 metadata cannot be accessed from Oracle Enterprise Manager.

The metadata may have been created with the CWM2 PL/SQL packages or with Enterprise Manager. The tool that you can use depends on the structure of your data warehouse.

CREATING OLAP CATALOG METADATA

CHOOSING THE RIGHT METADATA CREATION METHOD

If you are using a simple star or snowflake schema for the data warehouse, you can use either the OEM OLAP Management tool or OLAP Catalog API (CWM2 PL/SQL packages) to create OLAP Catalog metadata. However, if you are using complex schema or multidimensional data (Analytic Workspaces), you have to use CWM2 PL/SQL packages.

DW Storage/Design	OEM	CWM2 API
CASE1: Basic Star/snowflake schema - Unsolved, Level-based	Y	Y
CASE2: Complex Schema - Solved, Level-based - Dimension w/ Complex Hierarchy - Parent-child Dimension	N	Y
CASE3: Multidimensional data - Analytic Workspace	N	Y

Figure 2: Choosing the right metadata creation method

CASE 1: SIMPLE STAR/SNOWFLAKE SCHEMA

There are restrictions in using OEM to create the OLAP Catalog. To use OEM, the source schema must be comprised of the following:

- Fact table
  - Single fact table
  - Fact data must be unsolved
- Dimension table
  - Star or snow-flake
  - Level-based
  - Level columns cannot be NULLs
  - Multiple hierarchies have the same base level

The fact table must be a single fact table and the data must be unsolved. That means it is stored only at the lowest level of the hierarchy, and all the data for a cube must be stored in a single fact table.

The dimension table must be a star or a snowflake schema and all hierarchies must be level-based; the schema cannot use parent-child dimension tables. The level columns cannot contain NULLs. If the dimension has multiple hierarchies, they must have the same base level.

If your data warehouse complies with all of these requirements, then you can use either Enterprise Manager or the CWM2 APIs for defining OLAP metadata. No preprocessing steps are required. The OLAP tool in Oracle Enterprise Manager provides a wizard-driven GUI interface that allows the DBA to define the logical multidimensional models and map the logical model to tables in the data warehouse.

STEP 1: CREATING DIMENSIONS

First, you have to create dimensions including hierarchies, levels and level attributes and map them to appropriate columns in dimensional tables.

STEP 2: CREATING CUBES

Second, you have to create cubes including measures and map them to columns in the fact table.

STEP 3: CREATING MATERIALIZED VIEWS (OPTIONAL)

Next, you can create materialized views to improve the access performance.

STEP 4: CREATING MEASURE FOLDERS (OPTIONAL)

Finally, you can create measure folders that facilitate the browsing of data by business area.

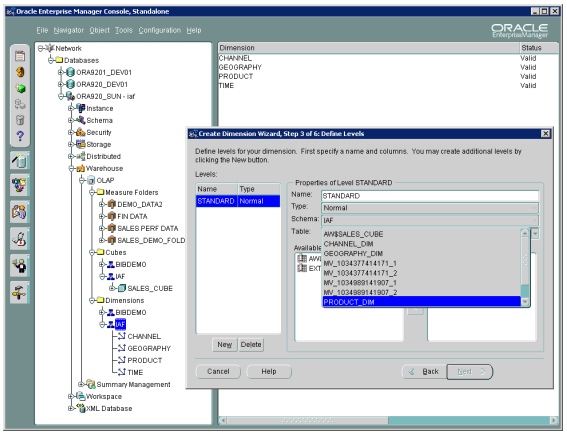

Figure 3: OLAP Management Tool in Oracle Enterprise Manager

CASE 2: COMPLEX STAR/SNOWFLAKE SCHEMA

If your data warehouse schema has any of following characteristics, you have to use CWM2 PL/SQL packages instead of OEM.

- Fact table
  - Multi fact table
  - Solved fact table (embedded total)
- Dimension table
  - Parent-child
  - Level-based dimension with NULL columns (Skip-level)
  - Multiple hierarchies have different base levels (Ragged hierarchies)

The fact table can be a multi-fact table and the data can be either completely solved or completely unsolved. For parent/child dimensions, fact data must be completely unsolved.

Dimension tables can be either level-based or parent/child dimensions. If they are level-based, the level columns can contain NULLs. This condition is referred to as "skip-level." If the dimension has multiple hierarchies, they are able to have the different base levels. This is sometimes called "ragged hierarchies."

STEP 1: CREATING DIMENSIONS

Creating dimensions is the first step in creating the OLAP metadata for a dimension. Typically you will create hierarchies and dimension attributes after creating the dimension and before creating the dimension levels and level attributes.

Once you have created a dimension, you will need to call procedures in the following packages to fully define the dimension's metadata:

- CWM2_OLAP_DIMENSION
- CWM2_OLAP_DIMENSION_ATTRIBUTE
- CWM2_OLAP_HIERARCHY
- CWM2_OLAP_LEVEL
- CWM2_OLAP_LEVEL_ATTRIBUTE

STEP 2: CREATING CUBES AND MEASURES

Second, you have to create the cube that will provide its context. To create the cube, use procedures in the CWM2_OLAP_CUBE package. Then you must create the measure entity by using CWM2_OLAP_MEASURE package. Measures represent data stored in fact tables.

- CWM2_OLAP_CUBE
- CWM2_OLAP_MEASURE

STEP 3: MAPPING

Once the dimension and cube metadata objects are defined in OLAP Catalog, you can map them to columns in warehouse dimension tables and fact tables. CWM2_OLAP_TABLE_MAP contains procedures that map metadata entities to relational fact tables and dimension tables.

- CWM2_OLAP_TABLE_MAP

STEP 4: CREATING MEASURE FOLDERS (OPTIONAL)

The CWM2_OLAP_CATALOG package is used primarily to manipulate OLAP measure folders.

- CWM2_OLAP_CATALOG

STEP 5: VALIDATE AND REFRESH

To test the validity of OLAP metadata, use the VALIDATE_CUBE and VALIDATE_DIMENSION procedures in the CWM2_OLAP_VALIDATE package. The validation process checks the structural integrity of the metadata and verifies that it is properly mapped to columns in tables or views.

You can determine whether or not a cube is valid by checking the INVALID column of the ALL_OLAP2_CUBES view. You can determine whether or not a dimension is valid by checking the INVALID column of the ALL_OLAP2_DIMENSIONS view.

- CWM2_OLAP_VALIDATE

After the validation, you finally have to commit the new metadata. The CWM2_OLAP_METADATA_REFRESH package provides a procedure that refreshes a set of metadata tables for the OLAP API. (The procedure is new from Oracle9i R2 9.2.0.1a and is included in the 9.2.0.1a patch's README file.)

- CWM2_OLAP_METADATA_REFRESH

Figure 4 is the sample script of CWM2 PL/SQL packages used to create OLAP Catalog metadata. You can download the complete script from www.iafsoft.com/downloads/ for your reference.

```
-- +-------------------------------------------------+
-- + IOUG-Live! 2003 technical presentation          +
-- +  #12327 managing OLAP Catalog Metadata          +
-- +        Case 2: CWM2 PL/SQL API sample script    +
-- +        IAF Software, Inc. (www.iafsof.com)      +
-- +-------------------------------------------------+
--
BEGIN
-- +-------------------------------------------------+
-- + Defining Channel dimension                      +
-- +-------------------------------------------------+
-- +---- Creating Dimension -------------------------+
cwm2_olap_dimension.create_dimension('iaf', 'chan_dim', 'chan_dim disp',
'chan_dim pl', 'chan_dim short', 'chan_dim long');

-- +----- Completing Dimension Metadata -------------+
cwm2_olap_dimension_attribute.create_dimension_attribute('iaf',
'chan_dim', 'Long Description', 'Long Description', 'Long Description',
'Long Description', true);
cwm2_olap_dimension_attribute.create_dimension_attribute('iaf',
'chan_dim', 'Short Description', 'Short Description', 'Short Description',
'Short Description', true);

cwm2_olap_hierarchy.create_hierarchy('iaf', 'chan_dim', 'chan_hier',
'chan_hier disp', 'chan_hier disp short', 'chan_hier long', 'UNSOLVED
LEVEL-BASED');
   cwm2_olap_dimension.set_default_display_hierarchy('iaf', 'chan_dim',
'chan_hier');

cwm2_olap_level.create_level('iaf', 'chan_dim', 'chan_chan', 'chan_cahn
disp', 'chan_chan pl', 'chan_chan short', 'chan_chan long');
cwm2_olap_level.create_level('iaf', 'chan_dim', 'chan_top', 'chan_top
disp', 'chan_top pl', 'chan_top short','chan_top long');
cwm2_olap_level_attribute.create_level_attribute('iaf', 'chan_dim', 'Long
Description', 'chan_chan', 'Long Description', 'Chan_chan_LongDesc', 'Chan
chan Long Desc', 'Long Description', true);
                                                    (Continued)
```

(Continued)

```
cwm2_olap_level_attribute.create_level_attribute('iaf', 'chan_dim',
'Short Description', 'chan_chan', 'Short Description',
'Chan_chan_ShortDesc', 'Chan chan Short Desc', 'Short Description',
true);

(Abbreviated for length.)

-- +-----------------------------------------------+
-- + Defining Cube and Measure                     +
-- +-----------------------------------------------+
cwm2_olap_cube.create_cube('iaf', 'iaf_sales', 'iaf_sales disp',
'iaf_sales short', 'iaf_sales long');
cwm2_olap_cube.add_dimension_to_cube('iaf', 'iaf_sales', 'iaf',
'chan_dim');
cwm2_olap_cube.add_dimension_to_cube('iaf', 'iaf_sales', 'iaf',
'prod_dim');
cwm2_olap_cube.add_dimension_to_cube('iaf', 'iaf_sales', 'iaf',
'geog_dim');
cwm2_olap_cube.add_dimension_to_cube('iaf', 'iaf_sales', 'iaf',
'time_dim');

cwm2_olap_measure.create_measure('iaf', 'iaf_sales', 'sales_meas',
'sales_meas disp', 'sales_meas short', 'sales_meas long');

(Abbreviated for length.)

-- +-----------------------------------------------+
-- + Mapping Facts                                 +
-- +-----------------------------------------------+
-- +----- OLAP Metadata Mapping for facts ----------+
cwm2_olap_table_map.map_facttbl_levelkey('iaf', 'iaf_sales', 'iaf',
'sales_facts', 'LOWESTLEVEL',
'DIM:iaf.chan_dim/HIER:chan_hier/LVL:chan_chan/COL:chan_leaves;DIM:iaf.p
rod_dim/HIER:prod_hier/LVL:prod_prd/COL:prod_leaves;DIM:iaf.geog_dim/HIE
R:geog_hier/LVL:geog_city/COL:geog_leaves;DIM:iaf.time_dim/HIER:time_hie
r/LVL:Time_mon/COL:time_leaves;');

cwm2_olap_table_map.map_facttbl_measure('iaf', 'iaf_sales',
'sales_meas', 'iaf', 'sales_facts', 'sales',
'DIM:iaf.chan_dim/HIER:chan_hier/LVL:chan_chan/COL:chan_leaves;DIM:iaf.p
rod_dim/HIER:prod_hier/LVL:prod_prd/COL:prod_leaves;DIM:iaf.geog_dim/HIE
R:geog_hier/LVL:geog_city/COL:geog_leaves;DIM:iaf.time_dim/HIER:time_hie
r/LVL:Time_mon/COL:time_leaves;');

-- +-----------------------------------------------+
-- + Create Measure Folder                         +
-- +-----------------------------------------------+
cwm2_olap_catalog.create_catalog('iaf_sales_folder', 'IAF Sales Data');
cwm2_olap_catalog.add_catalog_entity('iaf_sales_folder', 'iaf',
'iaf_sales', 'Sales');

-- +-----------------------------------------------+
-- + Validate and Refresh                          +
-- +-----------------------------------------------+
cwm2_olap_validate.validate_dimension('iaf', 'chan_dim');
cwm2_olap_validate.validate_dimension('iaf', 'prod_dim');
cwm2_olap_validate.validate_dimension('iaf', 'prod_dim');
cwm2_olap_validate.validate_dimension('iaf', 'time_dim');
cwm2_olap_validate.validate_cube('iaf', 'iaf_sales');
cwm2_olap_metadata_refresh.mr_refresh;
END;
/
```

Figure 4: Sample script of CWM2 PL/SQL packages used to create OLAP Catalog metadata

TIPS: PARENT-CHILD DIMENSION TABLE

If the dimensions of your data are stored in parent-child dimension tables, then you must convert them to solved/level-based dimensions before creating OLAP metadata. Oracle9i OLAP provides a CWM2_OLAP_PC_TRANSFORM package that generates the script to convert a parent-child dimension into a solved/level-based dimension table.

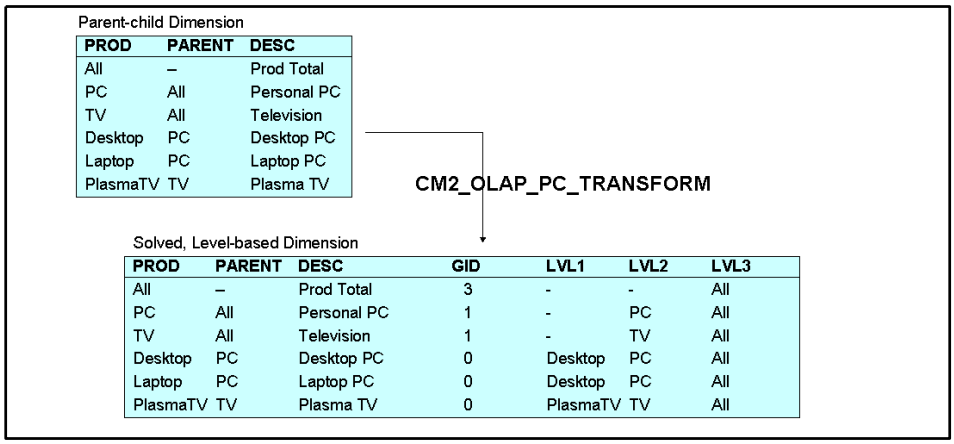

Figure 5: Converting from a parent-child to solved/level-based dimension

TIPS: UNSOLVED AND SOLVED

Fact data is unsolved when it is stored at the lowest level of aggregation. Fact data is solved when it is stored with embedded totals. Which kind of fact table you are using will affect your OLAP Catalog metadata registration. If you are using an unsolved fact table, you should specify UNSOLVED LEVEL-BASED in the SOLVED_CODE parameter when you create hierarchy objects by using the CWM2_OLAP_HIERARCHY.CREATE_HIERARCHY subprogram, and LOWEST in STORETYPE parameter when you map source fact table to the metadata object using CWM2_OLAP_TABLE_MAP.MAP_FACTTBL_LEVELKEY subprogram. If you are using solved fact tables, you should specify SOLVED_LEVEL-BASED in the SOLVED_CODE parameter when you create hierarchy object using CWM2_OLAP_HIERARCHY.CREATE_HIERARCHY subprogram, and ET (which stands for Embedded Total) in the STORETYPE parameter when you map source fact table to the metadata object using CWM2_OLAP_TABLE_MAP.MAP_FACTTBL_LEVELKEY subprogram.

TIPS: MEASURE FOLDER

The CWM2_OLAP_CATALOG package, which can manage measure folders in CWM2 metadata, is available in Oracle9i OLAP 9.2.0.1.0a or later, although the Oracle9i R2 OLAP User's Guide (9.2) does not describe its usage.

On the other hand, the CWM_CLASSIFY package, which is part of CWM1, cannot be used for CWM2 metadata, although the User's Guide says this package is used not only by CWM1 but also by CWM2.

CASE 3: MULTIDIMENSIONAL DATA (ANALYTIC WORKSPACES)

If your data warehouse stores data in analytic workspaces, then you must use the CWM2 APIs to create OLAP Catalog metadata.

ANALYTIC WORKSPACES

Analytic Workspaces (AW) store multidimensional data objects. They are stored in Oracle relational tables as a Binary Large Objects (BLOB).

You can create an Analytic Workspace with OLAP DML. After creating multidimensional objects, such as dimension and hierarchy, with the DEFINE command, you may load data from relational tables with the SQL FETCH command or text data file. Current Express users can convert the .DB file to an AW by using export/import commands through an EIF file.

In addition, Oracle provides DBMS_AWM package * (a new package with R9.2.0.3) to create AW from relational tables. You have to create OLAP Catalog metadata and register the source relational star/snow-flake schema in advance to use the program.

* CWM2_OLAP_AW_CREATE of 9.2.0.2 has been replaced with DBMS_AWM for 9.2.0.3.

STEP 1: PREPARING AW OBJECTS

Analytic Workspace has basic objects, like measure, dimension, hierarchy, parent-child relation and level dimension. Oracle Express users will find these are the same objects as Express. Before you create OLAP Catalog metadata from AW, you have to generate the additional AW objects, such as *Grouping ID/Parent Grouping ID* and *Family relations* that are needed by the OLAP API. The OLAP DML provides two commands for generating the objects, GROUPINGID and HIERHEIGHT.

STEP 2: CREATING RELATIONAL VIEWS

As I described above, the Analytic Workspace data is stored in a relational table as a BLOB. That means you cannot access the data directly by select statements.

One way to access multidimensional data is by using the OLAP_TABLE function. The OLAP_TABLE function is a predefined table function that returns a table of objects that map to analytic workspace data. Once you create a relational view using the Table Function, you can access Analytic Workspace data transparently by using standard SQL statements.

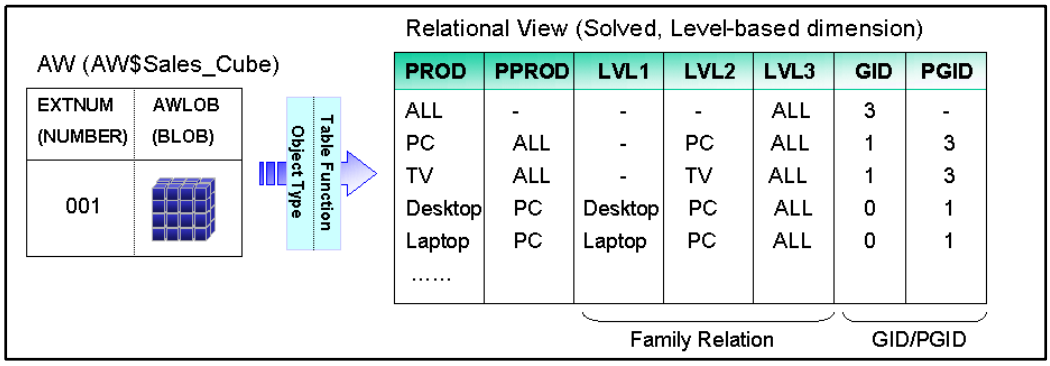

Figure 6: Creating a relational view by using the table function

Figure 7 is an example of the script used to create a relational view. You can download the complete script from www.iafsoft.com/downloads/ for your reference.

```
create or replace type prod as object(
  p0             varchar2(35),
  p0parent       varchar2(35),
  p0item         varchar2(35),
  p0category     varchar2(35),
  p0all          varchar2(35),
  p0gid          number(10),
  p0pgid         number(10),
  p0longdesc     varchar2(35),
  p0shortdesc    varchar2(35));

create or replace type prod_tbl as table of prod;

create or replace view prod_view
 as select * from table(cast(olap_table(
  'sales_cube duration session', 'prod_tbl', '',
  'dimension p0 from prod
  with hierarchy p0parent from p0.parent
  gid p0gid from p0.groupingid
  parentgid p0pgid from p0.groupingid
  levelrel p0item, p0category, p0all
  from p0.hierheight using p0.leveldim
  attribute p0longdesc from p0.longlabel
  attribute p0shortdesc from p0.shortlabel') as prod_tbl));
```

Figure 7: Creating relational views by table function

STEP 3: USING THE CWM2 PACKAGES TO GENERATE THE OLAP CATALOG METADATA

Once you create relational views of the Analytic Workspace, you can create the OLAP Catalog with the OLAP Catalog Metadata API as shown in steps 1 to 5 of Case 2 (above).

*TIPS: CWM2_OLAP_AW_ACCESS PACKAGE **

The CWM2_OLAP_AW_ACCESS package contains procedures for generating scripts that create views of analytic workspace objects. The utl_file_dir parameter in the init.ora file must be set to a valid directory to use the package. Otherwise, the procedures in CWM2_OLAP_AW_ACCESS will not be able to write the SQL scripts to a file.

* This function has been dropped in 9.2.0.3.

TIPS: AW MANAGER

A new tool called “AW Manager” may be included in the Oracle Enterprise Manager. AW Manager is designhned to create and manage Analytic Workspaces more easily. After you create OLAP Catalog (CWM1) metadata entities and map them between source relational schemas, you would create Analytic Workspaces and the OLAP Catalog (CWM2) metadata using AW Manager without writing code.

* At this time of publication for this white paper, AW Manager has not been available.

SUMMARY

I have discussed managing OLAP Catalog metadata in this white paper. Fully integrating multidimensional and relational database in Oracle9i R2 and a rich environment for managing OLAP Catalog metadata means that database administrators can manage the analytic models and source data in the corporate warehouse together.

USEFUL REFERENCE AND LINK

Oracle9i OLAP product information (http://www.oracle.com/ip/index.html?olap_home.html)

- Product white papers

Oracle Technology Network (http://otn.oracle.com/)

- Product downloads
- Product manuals (Oracle9i OLAP User's Guide Release2 and others)
- Technical white papers

IAF Software, Inc. homepage (http://www.iafsoft.com/)

- Complete sample script of CWM2 PL/SQL packages
- Demonstration (Viewlet)
- Product white papers