97
DBMS_HIERARCHY

`DBMS_HIERARCHY` contains subprograms for validating the data in tables used by hierarchies and analytic views.

This chapter contains the following topics:

- DBMS_HIERARCHY Overview
- DBMS_HIERARCHY Security Model
- Summary of DBMS_HIERARCHY Subprograms

DBMS_HIERARCHY Overview

The `DBMS_HIERARCHY` package contains functions for validating that the contents of a database table are suitable for use by an analytic view or a hierarchy, a function for verifying the success of the validation, and a procedure for creating a table for logging validation operations.

> **Note:**
>
> Names specified by parameters of the `DBMS_HIERARCHY` subprograms are case-sensitive.

For information about using analytic views, see *Oracle Database Data Warehousing Guide*.

DBMS_HIERARCHY Security Model

Summarizes security considerations for the validation of analytic view and hierarchy objects.

All procedures in this package validate that the current user has the necessary privileges on the specified objects and return an error if those privileges are not found.

> **Note:**
>
> To ensure that the user has enough tablespace to log validation operations, do one of the following:
>
> - `GRANT UNLIMITED TABLESPACE TO` *username*;
> - `ALTER USERNAME` *username* `QUOTA` *size* `ON` *tablespace_name*;

The following system privileges are required to use this package:

To validate objects in the user's own schema:

- `CREATE TABLE` privilege for `CREATE_VALIDATE_LOG_TABLE` or to have `VALIDATE_ANALTYIC_VIEW` or `VALIDATE_HIERARCHY` automatically create a table
- `SELECT` privilege on the tables or views used by the analytic view or hierarchy
- `INSERT` privilege on the tables used by the attribute dimensions of the hierarchy or the fact table used by the analytic view

To validate objects in different schemas:

- `CREATE ANY TABLE` privilege for `CREATE_VALIDATE_LOG_TABLE` or to have the `VALIDATE_ANALTYIC_VIEW` or `VALIDATE_HIERARCHY` automatically create a table
- `INSERT ANY TABLE` privilege on the tables used by the attribute dimensions of the hierarchy or the fact table used by the analytic view

Summary of DBMS_HIERARCHY Subprograms

This table lists the `DBMS_HIERARCHY` subprograms and briefly describes them.

Subprogram	Description
CREATE_VALIDATE_LOG_TABLE Procedure	Creates a table that you can use for logging messages generated by the `VALIDATE_HIERARCHY` and `VALIDATE_ANALYTIC_VIEW` functions.
CREATE_VIEW_FOR_FACT_ROWS Procedure	This procedure creates a view for fact rows.
CREATE_VIEW_FOR_STAR_ROWS Procedure	This procedure creates a view for star rows.
GET_MV_SQL_FOR_AV_CACHE Function	This function either returns the SQL for a cache that is defined in the AV or the SQL for a given set of measures and levels. These measures and levels do not need to be part of a cache in the AV.
GET_MV_SQL_FOR_STAR_CACHE Function	This function returns the SQL in text form for the star cache.
UPGRADE_VALIDATE_LOG_TABLE Procedure	This procedure takes a log table and upgrades it to the newest format. Returns an error if the table cannot be upgraded due to other errors.
VALIDATE_ANALYTIC_VIEW Function	Validates that the data in a table is suitable for use by an analytic view.
VALIDATE_CHECK_SUCCESS Function	Indicates whether a prior call to `VALIDATE_HIERARCHY` or `VALIDATE_ANALYTIC_VIEW` was successful or produced validation errors.
VALIDATE_HIERARCHY Function	Validates that the data in a table is suitable for use by a hierarchy.

CREATE_VALIDATE_LOG_TABLE Procedure

This procedure creates a table that you can use for logging messages generated by the `VALIDATE_ANALYTIC_VIEW` or `VALIDATE_HIERARCHY` function, which validate data used by an analytic view or hierarchy.

The table that this procedure creates has the following structure.

```
NAME                 NULL?    DATATYPE
-------------------- -------- --------
LOG_NUMBER           NOT NULL NUMBER
ACTION_ORDER         NOT NULL NUMBER
OBJECT_OWNER         NOT NULL VARCHAR2(128 BYTE)
OBJECT_NAME          NOT NULL VARCHAR2(128 BYTE)
ACTION               NOT NULL VARCHAR2(10 BYTE)
TIME                 NOT NULL TIMESTAMP(6)
ERROR_NUMBER                  NUMBER
ERROR_MESSAGE                 VARCHAR2(4000)
```

Syntax

```
DBMS_HIERARCHY.CREATE_VALIDATE_LOG_TABLE (
    table_name         IN  VARCHAR2,
    owner_name         IN  VARCHAR2          DEFAULT NULL
    IGNORE_IF_EXISTS   IN  PL/SQL BOOLEAN    DEFAULT FALSE);
```

Parameters

Parameter	Description
`table_name`	The name of the table to create.
`owner_name`	The name of the schema in which to create the table. If `owner_name` is `NULL`, then the table is created in the current user's schema.
`IGNORE_IF_EXISTS`	A Boolean that indicates whether to create the table if a table by the same name exists. If you specify a table, it must have the same structure as the table that this procedure creates.

Examples

Example 97-1 Creating a Validation Log Table

```
BEGIN
  DBMS_HIERARCHY.CREATE_VALIDATE_LOG_TABLE (
    'VAL_AV_HIERARCHY_LOG',
    'AV_USER',
    FALSE
  );
END;
/
```

CREATE_VIEW_FOR_FACT_ROWS Procedure

This procedure creates a view over the analytic view that exposes only the rows of the fact table. This allows standard SQL group functions to be used to aggregate those rows. The advantage to querying such a view rather than the fact table directly is that the analytic view may be able to perform query optimizations otherwise unavailable. When `include_meas` is `TRUE`, the `AV_AGGREGATE` group function may be used to expose measure columns defined in the analytic view.

Syntax

```
DBMS_HIERARCHY.CREATE_VIEW_FOR_FACT_ROWS (
      analytic_view_name         IN VARCHAR2,
      view__name                 IN VARCHAR2,
      dim_hier_seq               IN ID2_SEQUENCE  DEFAULT NULL,
      analytic_view_owner_name   IN VARCHAR2      DEFAULT
SYS_CONTEXT('USERENV', 'CURRENT_SCHEMA'),
      view_owner_name            IN VARCHAR2      DEFAULT
SYS_CONTEXT('USERENV', 'CURRENT_SCHEMA'),
      dim_qual_sep               IN VARCHAR2      DEFAULT '_',
      all_join_keys              IN BOOLEAN       DEFAULT TRUE,
      include_meas               IN BOOLEAN       DEFAULT FALSE,
      include_hier_attr          IN BOOLEAN       DEFAULT FALSE);
```

Parameters

Parameter	Description
`analytic_view_name`	The name of the analytic view to be created.
`view__name`	The name of the view.
`dim_hier_seq`	A sequence of `ID2` records. An ID2 record contains two `VARCHAR2(128)` for the two parallel components, providing (`dim, hier`) name pairs. Default value is `NULL` which indicates that all hierarchies should be considered.
`analytic_view_owner_name`	The name of the analytic view owner. The default value is `SYS_CONTEXT('USERENV', 'CURRENT_SCHEMA')`.
`view_owner_name`	The owner of the view to be created. The default value is `SYS_CONTEXT('USERENV', 'CURRENT_SCHEMA')`.
`dim_qual_sep`	The separator to be used on the columns of the created view. For example, • for `fact` columns - `"FACT" || dim_qual_sep || {name of fact column alias}` • for `measures` - `"MEASURE" || dim_qual_sep || {measure name alias}` The default value is `_`.
`all_join_keys`	If `TRUE`, adds all of the join key columns in the AV to the view.

Parameter	Description
include_meas	If TRUE, include all of the base measures and calculated measures defined in the AV as columns in the created view.
include_hier_attr	If TRUE, include all of the hierarchical attributes defined in the AV as columns in the created view.

CREATE_VIEW_FOR_STAR_ROWS Procedure

This procedure creates a view over the analytic view that exposes only rows for a particular dimension star table. Such a view is intended to be used in conjunction with a corresponding fact rows view created using CREATE_VIEW_FOR_FACT_ROWS, joined using their corresponding keys as defined in the analytic view.

Syntax

```
DBMS_HIERARCHY.CREATE_VIEW_FOR_STAR_ROWS (
      analytic_view_name         IN VARCHAR2,
      dimension_alias            IN VARCHAR2,
      view__name                 IN VARCHAR2,
      analytic_view_owner_name   IN VARCHAR2      DEFAULT
SYS_CONTEXT('USERENV', 'CURRENT_SCHEMA'),
      view_owner_name            IN VARCHAR2      DEFAULT
SYS_CONTEXT('USERENV', 'CURRENT_SCHEMA'),
      hier_qual_sep              IN VARCHAR2      DEFAULT '_',
      include_hier_attr          IN BOOLEAN       DEFAULT FALSE);
```

Parameters

Parameter	Description
analytic_view_name	The name of the analytic view to be created.
dimension_alias	The alias of the attribute dimension that you wish to create a star rows view for.
view__name	The name of the view.
analytic_view_owner_name	The name of the analytic view owner. The default value is SYS_CONTEXT('USERENV', 'CURRENT_SCHEMA').
view_owner_name	The owner of the view to be created. The default value is SYS_CONTEXT('USERENV', 'CURRENT_SCHEMA').
hier_qual_sep	The separator to be used. The separator is between the attribute dimension name and the built-in hierarchical attributes, such as MEMBER_NAME, LEVEL_NAME, DEPTH, and so on. The default value is _.
include_hier_attr	If TRUE, include all of the hierarchical attributes defined in the AV as columns in the created view.

GET_MV_SQL_FOR_AV_CACHE Function

This function has two signatures. The first version of this method returns the SQL for a cache that is defined in the AV. The second version of this method returns SQL for a given set of measures and levels. These measures and levels do not need to be part of a cache in the AV.

Syntax

```
DBMS_HIERARCHY.GET_MV_SQL_FOR_AV_CACHE (
    analytic_view_name          IN    VARCHAR2,
    cache_idx                   IN    NUMBER,
    analytic_view_owner_name    IN    VARCHAR2  DEFAULT SYS_CONTEXT('USERENV',
'CURRENT_SCHEMA'))
  RETURN CLOB;



DBMS_HIERARCHY.GET_MV_SQL_FOR_AV_CACHE (
    analytic_view_name          IN    VARCHAR2,
    lvl_seq                     IN    ID3_SEQUENCE  DEFAULT NULL,
    meas_seq                    IN    ID_SEQUENCE   DEFAULT NULL,
    analytic_view_owner_name    IN    VARCHAR2  DEFAULT SYS_CONTEXT('USERENV',
'CURRENT_SCHEMA'))
  RETURN CLOB;
```

Parameters

Table 97-1 GET_MV_SQL_FOR_AV_CACHE Function Parameters

Parameter	Description
analytic_view_name	The name of the analytic view to be created.
cache_idx	Zero-based index for the caches defined on the AV.
analytic_view_owner_name	The name of the analytic view owner. The default value is SYS_CONTEXT('USERENV', 'CURRENT_SCHEMA').
lvl_seq	The list of levels that the cache should be based on.
meas_seq	The list of measures that the cache should be based on.

GET_MV_SQL_FOR_STAR_CACHE Function

This function returns the SQL in text form for the star cache.

Syntax

```
DBMS_HIERARCHY.GET_MV_SQL_FOR_STAR_CACHE (
    attr_dim_name           IN  VARCHAR2,
    attr_dim_owner_name     IN  VARCHAR2 DEFAULT SYS_CONTEXT('USERENV',
'CURRENT_SCHEMA'))
  RETURN CLOB;
```

Parameters

Table 97-2 GET_MV_SQL_FOR_STAR_CACHE Function Parameters

Parameter	Description
attr_dim_name	The name of the attribute dimension on which to create a cache.
attr_dim_owner_name	The schema that owns the attribute dimension. The default value is SYS_CONTEXT('USERENV', 'CURRENT_SCHEMA').

UPGRADE_VALIDATE_LOG_TABLE Procedure

This procedure takes a log table and upgrades it to the newest format. Returns an error if the table cannot be upgraded due to other errors.

Syntax

```
DBMS_HIERARCHY.UPGRADE_VALIDATE_LOG_TABLE (
    table_name        IN  VARCHAR2,
    owner_name        IN  VARCHAR2         DEFAULT NULL);
```

Parameters

Parameter	Description
table_name	The name of the table to upgrade.
owner_name	The name of the schema in which to find the table. If owner_name is NULL, then the table is created in the current user's schema.

VALIDATE_ANALYTIC_VIEW Function

This function validates that the data in a table or view conforms to the logical constraints inherent in the definition of an analytic view.

Syntax

```
DBMS_HIERARCHY.VALIDATE_ANALYTIC_VIEW (
      analytic_view_name          IN VARCHAR2   DEFAULT NULL,
      analytic_view_owner_name    IN VARCHAR2   DEFAULT NULL,
      log_table_name              IN VARCHAR2   DEFAULT NULL,
      log_table_owner_name        IN VARCHAR2   DEFAULT NULL)
      RETURN NUMBER;
```

Parameters

Parameter	Description
analytic_view_name	The name of the analytic view to validate.
analytic_view_owner_name	The name of the owner of the schema that contains the analytic view.
log_table_name	The name of the validation log table in which to put the results of the validation operation.

Parameter	Description
log_table_owner_name	The name of the owner of the schema in which the validation log table exists or in which to create the table.

Returns

The number of the entry in the validation log table for the validation results.

Usage Notes

If the log_table_name parameter is NULL, then the VALIDATE_ANALYTIC_VIEW function creates a validation log table. The name of the table it creates is DBMS_HIERARCHY_LOG.

When the validation operation begins, a row is inserted into the log table with the action of START. When the operation completes, a row is inserted into the log table with the action of END. When an error is detected, a row is inserted into the log table with the action of ERROR, and the associated error_number and error_message columns are populated. All rows inserted into the validation log table include a log number and the time of the insert.

The VALIDATE_ANALYTIC_VIEW function verifies that the following conditions are true for each attribute dimension the analytic view is dimensioned by:

- The key values found in the fact table for the attribute dimension must exist in the star schema dimension table for that attribute dimension.
- The referenced attribute values for the attribute dimension must be unique across all rows of the star schema dimension table for that dimension.

Also, for every hierarchy in the analytic view, the function verifies that the following conditions are true:

- The primary key of a level determines a unique value for each attribute of the level.
- For each row of the table or view used by the attribute dimension of the hierarchy, the value for every level key column (including alternate keys) of a NOT NULL level is non-NULL.
- For each row of the table or view, either all level key columns and alternate key columns of a SKIP WHEN NULL level must be NULL or they must all be non-NULL. This verifies that the alternate level key is determined by the level key.
- For each group of rows that have the same alternate key column values for a level, the key column values must have the same column values. This verifies that the level key is determined by the alternate level key, which is required for an alternate key.

Examples

Example 97-2 Validating an Analytic View

```
DECLARE
  log_num NUMBER;
  obj_name VARCHAR2(8) := 'SALES_AV';
BEGIN
   log_num := DBMS_HIERARCHY.VALIDATE_ANALYTIC_VIEW(obj_name);
END;
/
```

VALIDATE_CHECK_SUCCESS Function

This function indicates whether a prior call to `VALIDATE_HIERARCHY` or `VALIDATE_ANALYTIC_VIEW` was successful or produced validation errors.

Syntax

```
DBMS_HIERARCHY.VALIDATE_CHECK_SUCCESS (
    TOPOBJ_NAME           IN  VARCHAR2,
    TOPOBJ_OWNER          IN  VARCHAR2,
    LOG_NUMBER            IN  VARCHAR2
    LOG_TABLE_NAME        IN  VARCHAR2
    LOG_TABLE_OWNER_NAME  IN  VARCHAR2 )
    RETURN VARCHAR2;
```

Parameters

Parameter	Description
`TOPOBJ_NAME`	The name of the hierarchy or analytic view.
`TOPOBJ_OWNDER`	The owner of the hierarchy or analytic view.
`LOG_NUMBER`	The number of the log entry.
`LOG_TABLE_NAME`	The name of the log table.
`LOG_TABLE_OWNER_NAME`	The name of the schema in which the table exits.

Returns

A `VARCHAR2` that is SUCCESS if no errors occurred or ERROR if errors did occur.

Examples

Example 97-3 Using VALIDATE_CHECK_SUCCESS

This example finds out whether the prior call to `VALIDATE_ANALTYIC_VIEW` encountered errors.

```
DECLARE
  log_num NUMBER;
  succ VARCHAR2(7);
  obj_name VARCHAR2(8) := 'SALES_AV';
BEGIN
   log_num := dbms_hierarchy.validate_analytic_view(obj_name);
   succ := dbms_hierarchy.validate_check_success(
      topobj_name => obj_name, log_number => log_num);
  IF (succ != 'SUCCESS') THEN
     RAISE_APPLICATION_ERROR(
       num => -20000,
       msg => 'Validate failed!');
  END IF;
END;
/
```

VALIDATE_HIERARCHY Function

This function validates that the data in a table or view conforms to the logical constraints inherent in the definitions of an attribute dimension that uses the table or view and a hierarchy that uses the attribute dimension.

Syntax

```
DBMS_HIERARCHY.VALIDATE_HIERARCHY (
      hier_name                IN VARCHAR2,
      hier_owner_name          IN VARCHAR2    DEFAULT NULL,
      log_table_name           IN VARCHAR2    DEFAULT NULL,
      log_table_owner_name     IN VARCHAR2    DEFAULT NULL)
      RETURN NUMBER;
```

Parameters

Parameter	Description
hier_name	The name of the hierarchy to validate.
hier_owner_name	The name of the owner of the schema that contains the hierarchy.
log_table_name	The name of the validation log table in which to put the results of the validation operation.
log_table_owner_name	The name of the owner of the schema in which the validation log table exists or in which to create the table.

Returns

The number of the entry in the validation log table for the validation results.

Usage Notes

If the log_table_name parameter is NULL, then the VALIDATE_HIERARCHY function creates a validation log table. The name of the table it creates is DBMS_HIERARCHY_LOG.

When the validation operation begins, a row is inserted into the log table with the action of START. When the operation completes, a row is inserted into the log table with the action of END. When an error is detected, a row is inserted into the log table with the action of ERROR, and the associated error_number and error_message columns are populated. All rows inserted into the validation log table include a log number and the time of the insert.

The VALIDATE_HIERARCHY function verifies that the following conditions are true for the hierarchy:

- The primary key of a level determines a unique value for each attribute of the level.
- For each row of the table or view used by the attribute dimension of the hierarchy, the value for every level key column (including alternate keys) of a NOT NULL level is non-NULL.
- For each row of the table or view, either all level key columns and alternate key columns of a SKIP WHEN NULL level must be NULL or they must all be non-NULL. This verifies that the alternate level key is determined by the level key.
- For each group of rows that have the same alternate key column values for a level, the key column values must have the same column values. This verifies that the level key is determined by the alternate level key, which is required for an alternate key.

Examples

Example 97-4 Validating a Hierarchy and Specifying a Table Name

This example validates the `PRODUCT_HIER` hierarchy and specifies that the results be inserted in the table named `VAL_AV_HIERARCHY_LOG`. The owner of the hierarchy and of the schema that contains the table is AV_USER.

```
-- Create a log table.
BEGIN
  DBMS_HIERARCHY.CREATE_VALIDATE_LOG_TABLE (
    'VAL_AV_HIERARCHY_LOG',
    'AV_USER',
    FALSE
  );
END;
/
-- Validate the hierarchy.
DECLARE
  log_num NUMBER;
  obj_name VARCHAR2(12) := 'PRODUCT_HIER';
  table_name  VARCHAR2(28) := 'VAL_AV_HIERARCHY_LOG';
BEGIN
   log_num := DBMS_HIERARCHY.VALIDATE_HIERARCHY(obj_name, 'AV_USER',
table_name);
END;
/
```

Query the log table.

```
SELECT LOG_NUMBER, ACTION, OBJECT_NAME, ERROR_NUMBER, ERROR_MESSAGE
  FROM AV_USER.VAL_AV_HIERARCHY_LOG;
  WHERE OBJECT_NAME = 'PRODUCT_HIER';

LOG_NUMBER ACTION     OBJECT_NAME          ERROR_NUMBER ERROR_MESSAGE
---------- ---------- -------------------- ------------ -------------
         1 START      PRODUCT_HIER
         1 END        PRODUCT_HIER
```

Example 97-5 Validating a Hierarchy Without Specifying a Table Name

This example shows that if you do not specify a validation log table, then the `VALIDATE_HIERARCHY` function creates one named DBMS_HIERARCHY_LOG.

```
DECLARE
  log_num NUMBER;
  obj_name VARCHAR2(12) := 'PRODUCT_HIER';
BEGIN
   log_num := DBMS_HIERARCHY.VALIDATE_HIERARCHY(obj_name);
END;
```

Query the log table.

```
SELECT LOG_NUMBER, ACTION, OBJECT_NAME, ERROR_NUMBER, ERROR_MESSAGE
  FROM DBMS_HIERARCHY_LOG
  WHERE OBJECT_NAME = 'PRODUCT_HIER';

LOG_NUMBER ACTION     OBJECT_NAME         ERROR_NUMBER ERROR_MESSAGE
---------- ---------- ------------------- ------------ -------------
         1 START      PRODUCT_HIER
         1 END        PRODUCT_HIER
```