135
DBMS_MVIEW

DBMS_MVIEW enables you to understand capabilities for materialized views and potential materialized views, including their rewrite availability. It also enables you to refresh materialized views that are not part of the same refresh group and purge logs.

> **Note:**
>
> DBMS_MVIEW is a synonym for DBMS_SNAPSHOT.

> **See Also:**
>
> *Oracle Database Data Warehousing Guide* for more information about using materialized views in a data warehousing environment

This chapter contains the following topics:

DBMS_MVIEW Operational Notes

If a query is less than 256 characters long, you can invoke EXPLAIN_REWRITE using the EXECUTE command from SQL*Plus. Otherwise, the recommended method is to use a PL/SQL BEGIN..END block, as shown in the examples in /rdbms/demo/smxrw.sql.

DBMS_MVIEW Security Model

The DBMS_MVIEW package consists of a number of materialized view-related subprograms, each of which has different functionality and privilege requirements.

The privilege model is generally based on the invoker's right. Each package subprogram is executed by first checking the privileges against the invoker. If all the required privileges are met, the subprogram will be executed. Otherwise, an insufficient privileges error will be thrown.

DBMS_MVIEW Rules and Limits

The DBMS_MVIEW.EXPLAIN_REWRITE procedure cannot accept queries longer than 32627 characters. These restrictions also apply when passing the defining query of a materialized view to the DBMS_MVIEW.EXPLAIN_MVIEW procedure.

Summary of DBMS_MVIEW Subprograms

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

Table 135-1 DBMS_MVIEW Package Subprograms

Subprogram	Description
BEGIN_TABLE_REORGANIZATION Procedure	Performs a process to preserve materialized view data needed for refresh
END_TABLE_REORGANIZATION Procedure	Ensures that the materialized view data for the master table is valid and that the master table is in the proper state
ESTIMATE_MVIEW_SIZE Procedure	Estimates the size of a materialized view that you might create, in bytes and rows
EXPLAIN_MVIEW Procedure	Explains what is possible with a materialized view or potential materialized view
EXPLAIN_REWRITE Procedure	Explains why a query failed to rewrite or why the optimizer chose to rewrite a query with a particular materialized view or materialized views
I_AM_A_REFRESH Function	Returns the value of the `I_AM_REFRESH` package state
PMARKER Function	Returns a partition marker from a rowid, and is used for Partition Change Tracking (PCT)
PURGE_DIRECT_LOAD_LOG Procedure	Purges rows from the direct loader log after they are no longer needed by any materialized views (used with data warehousing)
PURGE_LOG Procedure	Purges rows from the materialized view log
PURGE_MVIEW_FROM_LOG Procedure	Purges rows from the materialized view log
REFRESH Procedures	Refreshes one or more materialized views that are not members of the same refresh group
REFRESH_ALL_MVIEWS Procedure	Refreshes all materialized views that do not reflect changes to their master table or master materialized view
REFRESH_DEPENDENT Procedures	Refreshes all table-based materialized views that depend on a specified master table or master materialized view, or list of master tables or master materialized views
REGISTER_MVIEW Procedure	Enables the administration of individual materialized views
UNREGISTER_MVIEW Procedure	Enables the administration of individual materialized views once invoked at a master site or master materialized view site to unregister a materialized view

BEGIN_TABLE_REORGANIZATION Procedure

This procedure performs a process to preserve materialized view data needed for refresh. It must be called before a master table is reorganized.

Syntax

```
DBMS_MVIEW.BEGIN_TABLE_REORGANIZATION (
   tabowner     IN   VARCHAR2,
   tabname      IN   VARCHAR2);
```

Parameters

Table 135-2 BEGIN_TABLE_REORGANIZATION Procedure Parameters

Parameter	Description
tabowner	Owner of the table being reorganized
tabname	Name of the table being reorganized

END_TABLE_REORGANIZATION Procedure

This procedure ensures that the materialized view data for the master table is valid and that the master table is in the proper state. It must be called after a master table is reorganized.

Syntax

```
DBMS_MVIEW.END_TABLE_REORGANIZATION (
   tabowner    IN   VARCHAR2,
   tabname     IN   VARCHAR2);
```

Parameters

Table 135-3 END_TABLE_REORGANIZATION Procedure Parameters

Parameter	Description
tabowner	Owner of the table being reorganized
tabname	Name of the table being reorganized

ESTIMATE_MVIEW_SIZE Procedure

This procedure estimates the size of a materialized view that you might create, in bytes and number of rows.

Syntax

```
DBMS_MVIEW.ESTIMATE_MVIEW_SIZE (
   stmt_id       IN  VARCHAR2,
   select_clause IN  VARCHAR2,
   num_rows      OUT NUMBER,
   num_bytes     OUT NUMBER);
```

Parameters

Table 135-4 ESTIMATE_MVIEW_SIZE Procedure Parameters

Parameter	Description
stmt_id	Arbitrary string used to identify the statement in an EXPLAIN PLAN
select_clause	The SELECT statement to be analyzed
num_rows	Estimated cardinality
num_bytes	Estimated number of bytes

EXPLAIN_MVIEW Procedure

This procedure enables you to learn what is possible with a materialized view or potential materialized view. For example, you can determine if a materialized view is fast refreshable and what types of query rewrite you can perform with a particular materialized view.

Using this procedure is straightforward. You simply call `DBMS_MVIEW.EXPLAIN_MVIEW`, passing in as parameters the schema and materialized view name for an existing materialized view. Alternatively, you can specify the `SELECT` string or `CREATE MATERIALIZED VIEW` statement for a potential materialized view. The materialized view or potential materialized view is then analyzed and the results are written into either a table called `MV_CAPABILITIES_TABLE`, which is the default, or to an array called `MSG_ARRAY`.

The procedure is overloaded:

- The first version is for explaining an existing or potential materialized view with output to `MV_CAPABILITIES_TABLE`.
- The second version is for explaining an existing or potential materialized view with output to a `VARRAY`.

Syntax

```
DBMS_MVIEW.EXPLAIN_MVIEW (
   mv            IN VARCHAR2,
   statement_id  IN VARCHAR2:= NULL);

DBMS_MVIEW.EXPLAIN_MVIEW (
   mv          IN VARCHAR2,
   msg_array   OUT SYS.ExplainMVArrayType);
```

Parameters

Table 135-5 EXPLAIN_MVIEW Procedure Parameters

Parameter	Description
`mv`	The name of an existing materialized view (optionally qualified with the owner name separated by a ".") or a `SELECT` statement or a `CREATE MATERIALIZED VIEW` statement for a potential materialized view.
`statement_id`	A client-supplied unique identifier to associate output rows with specific invocations of `EXPLAIN_MVIEW`
`msg_array`	The PL/SQL `VARRAY` that receives the output. Use this parameter to direct `EXPLAIN_MVIEW`'s output to a PL/SQL VARRAY rather than `MV_CAPABILITIES_TABLE`.

Usage Notes

- You must run the `utlxmv.sql` script to create `MV_CAPABILITIES_TABLE` in the current schema prior to calling `EXPLAIN_MVIEW` except when you direct output to a `VARRAY`. The script is found in the `ADMIN` directory.

- In Oracle database version 19.18, the following capability is added and is displayed while explaining a materialized view using `dbms_mview.explain_mview`. It indicates that the materialized view is eligible for LPCT fast refresh:

  ```
  REFRESH_FAST_LPT
  ```

> **See Also:**
>
> *Oracle Database Data Warehousing Guide* for more information about refresh methods for materialized view .

EXPLAIN_REWRITE Procedure

This procedure enables you to learn why a query failed to rewrite, or, if it rewrites, which materialized views will be used.

Using the results from the procedure, you can take the appropriate action needed to make a query rewrite if at all possible. The query specified in the `EXPLAIN_REWRITE` statement is never actually executed.

A demo file, `xrwutl.sql`, is available to help format the output from `EXPLAIN_REWRITE`.

Syntax

You can obtain the output from `DBMS_MVIEW.EXPLAIN_REWRITE` in two ways. The first is to use a table, while the second is to create a `VARRAY`. The following shows the basic syntax for using an output table:

```
DBMS_MVIEW.EXPLAIN_REWRITE (
    query           VARCHAR2,
    mv              VARCHAR2(30),
    statement_id    VARCHAR2(30));
```

You can create an output table called `REWRITE_TABLE` by executing the `utlxrw.sql` script.

The `query` parameter is a text string representing the SQL query. The parameter, `mv`, is a fully qualified materialized view name in the form of `schema.mv`. This is an optional parameter. When it is not specified, `EXPLAIN_REWRITE` returns any relevant messages regarding all the materialized views considered for rewriting the given query. When `schema` is omitted and only `mv` is specified, `EXPLAIN_REWRITE` looks for the materialized view in the current schema.

If you want to direct the output of `EXPLAIN_REWRITE` to a `VARRAY` instead of a table, you should call the procedure as follows:

```
DBMS_MVIEW.EXPLAIN_REWRITE (
    query           [VARCHAR2 | CLOB],
    mv               VARCHAR2(30),
    output_array     SYS.RewriteArrayType);
```

Note that if the query is less than 256 characters long, `EXPLAIN_REWRITE` can be easily invoked with the `EXECUTE` command from SQL*Plus. Otherwise, the recommended method is to use a PL/SQL `BEGIN... END` block, as shown in the examples in `/rdbms/demo/smxrw*`.

You can also use `EXPLAIN_REWRITE` with multiple materialized views, in which case the syntax will be the same as with a single materialized view, except that the materialized views are

specified by a comma-delimited string. For example, to find out whether a given set of materialized views `mv1`, `mv2`, and `mv3` could be used to rewrite the query, `query_txt`, and, if not, why not, use `EXPLAIN_REWRITE` as follows:

```
DBMS_MVIEW.EXPLAIN_REWRITE(query_txt, 'mv1, mv2, mv3')
```

See *Oracle Database Data Warehousing Guide* for more information on using the `EXPLAIN_REWRITE` procedure.

Parameters

Table 135-6 EXPLAIN_REWRITE Procedure Parameters

Parameter	Description
`query`	SQL `SELECT` statement to be explained
`mv`	The fully qualified name of an existing materialized view in the form of `SCHEMA.MV`. For multiple materialized views, you can provide a comma-delimited list of names.
`statement_id`	A client-supplied unique identifier to distinguish output messages
`msg_array`	The PL/SQL `VARRAY` that receives the output. Use this parameter to direct `EXPLAIN_REWRITE`'s output to a PL/SQL `VARRAY`.

Usage Notes

To obtain the output into a table, you must run the `utlxrw.sql` script before calling `EXPLAIN_REWRITE`. This script creates a table named `REWRITE_TABLE` in the current schema.

I_AM_A_REFRESH Function

This function returns the value of the `I_AM_REFRESH` package state.

Syntax

```
DBMS_MVIEW.I_AM_A_REFRESH
   RETURN BOOLEAN;
```

Return Values

A return value of `true` indicates that all local replication triggers for materialized views are effectively disabled in this session because each replication trigger first checks this state. A return value of `false` indicates that these triggers are enabled.

PMARKER Function

This function returns a partition marker from a rowid. It is used for Partition Change Tracking (PCT).

Syntax

```
DBMS_MVIEW.PMARKER(
   rid IN ROWID)
 RETURN NUMBER;
```

Parameters

Table 135-7 PMARKER Function Parameters

Parameter	Description
rid	The rowid of a row entry in a master table

PURGE_DIRECT_LOAD_LOG Procedure

This procedure removes entries from the direct loader log after they are no longer needed for any known materialized view. This procedure usually is used in environments using Oracle's data warehousing technology.

Syntax

```
DBMS_MVIEW.PURGE_DIRECT_LOAD_LOG();
```

PURGE_LOG Procedure

This procedure purges rows from the materialized view log.

Syntax

```
DBMS_MVIEW.PURGE_LOG (
   master        IN   VARCHAR2,
   num           IN   BINARY_INTEGER := 1,
   flag          IN   VARCHAR2       := 'NOP');
```

Parameters

Table 135-8 PURGE_LOG Procedure Parameters

Parameter	Description
master	Name of the master table or master materialized view.

num	Number of least recently refreshed materialized views whose rows you want to remove from materialized view log. For example, the following statement deletes rows needed to refresh the two least recently refreshed materialized views:
	`DBMS_MVIEW.PURGE_LOG('master_table', 2);`
	To delete all rows in the materialized view log, indicate a high number of materialized views to disregard, as in this example:
	`DBMS_MVIEW.PURGE_LOG('master_table',9999);`
	This statement completely purges the materialized view log that corresponds to `master_table` if fewer than 9999 materialized views are based on `master_table`. A simple materialized view whose rows have been purged from the materialized view log must be completely refreshed the next time it is refreshed.

flag	Specify `delete` to guarantee that rows are deleted from the materialized view log for at least one materialized view. This parameter can override the setting for the parameter `num`. For example, the following statement deletes rows from the materialized view log that has dependency rows in the least recently refreshed materialized view: `DBMS_MVIEW.PURGE_LOG('master_table',1,'delete');`

PURGE_MVIEW_FROM_LOG Procedure

This procedure is called on the master site or master materialized view site to delete the rows in materialized view refresh related data dictionary tables maintained at the master for the specified materialized view identified by `mview_id` or the combination of `mviewowner`, `mviewname`, and `mviewsite`.

If the materialized view specified is the oldest materialized view to have refreshed from any of the master tables or master materialized views, then the materialized view log is also purged. This procedure does not unregister the materialized view.

Syntax

```
DBMS_MVIEW.PURGE_MVIEW_FROM_LOG (
   mview_id       IN   BINARY_INTEGER);

DBMS_MVIEW.PURGE_MVIEW_FROM_LOG (
   mviewowner     IN   VARCHAR2,
   mviewname      IN   VARCHAR2,
   mviewsite      IN   VARCHAR2);
```

> **Note:**
>
> This procedure is overloaded. The parameter `mview_id` is mutually exclusive with the three remaining parameters: `mviewowner`, `mviewname`, and `mviewsite`.

Parameters

Table 135-9 PURGE_MVIEW_FROM_LOG Procedure Parameters

Parameter	Description
`mview_id`	If you want to execute this procedure based on the identification of the target materialized view, specify the materialized view identification using the `mview_id` parameter. Query the `DBA_BASE_TABLE_MVIEWS` view at the materialized view log site for a listing of materialized view IDs. Executing this procedure based on the materialized view identification is useful if the target materialized view is not listed in the list of registered materialized views (`DBA_REGISTERED_MVIEWS`).
`mviewowner`	If you do not specify an `mview_id`, enter the owner of the target materialized view using the `mviewowner` parameter. Query the `DBA_REGISTERED_MVIEWS` view at the materialized view log site to view the materialized view owners.
`mviewname`	If you do not specify an `mview_id`, enter the name of the target materialized view using the `mviewname` parameter. Query the `DBA_REGISTERED_MVIEWS` view at the materialized view log site to view the materialized view names.
`mviewsite`	If you do not specify an `mview_id`, enter the site of the target materialized view using the `mviewsite` parameter. Query the `DBA_REGISTERED_MVIEWS` view at the materialized view log site to view the materialized view sites.

Usage Notes

If there is an error while purging one of the materialized view logs, the successful purge operations of the previous materialized view logs are not rolled back. This is to minimize the

size of the materialized view logs. In case of an error, this procedure can be invoked again until all the materialized view logs are purged.

REFRESH Procedures

This procedure refreshes a list of materialized views.

Syntax

```
DBMS_MVIEW.REFRESH (
   { list                 IN     VARCHAR2,
   | tab                  IN     DBMS_UTILITY.UNCL_ARRAY,}
   method                 IN     VARCHAR2       := NULL,
   rollback_seg           IN     VARCHAR2       := NULL,
   push_deferred_rpc      IN     BOOLEAN        := true,
   refresh_after_errors   IN     BOOLEAN        := false,
   purge_option           IN     BINARY_INTEGER := 1,
   parallelism            IN     BINARY_INTEGER := 0,
   heap_size              IN     BINARY_INTEGER := 0,
   atomic_refresh         IN     BOOLEAN        := true,
   nested                 IN     BOOLEAN        := false,
   out_of_place           IN     BOOLEAN        := false,
   skip_ext_data             IN     BOOLEAN        := false);
```

> **Note:**
>
> This procedure is overloaded. The `list` and `tab` parameters are mutually exclusive.

Parameters

Table 135-10 REFRESH Procedure Parameters

Parameter	Description
`list | tab`	Comma-delimited list of materialized views that you want to refresh. (Synonyms are not supported.) These materialized views can be located in different schemas and have different master tables or master materialized views. However, all of the listed materialized views must be in your local database. Alternatively, you may pass in a PL/SQL index-by table of type `DBMS_UTILITY.UNCL_ARRAY`, where each element is the name of a materialized view.

Table 135-10 (Cont.) REFRESH Procedure Parameters

Parameter	Description

method	A string of refresh methods indicating how to refresh the listed materialized views. An f indicates fast refresh, ? indicates force refresh, C or c indicates complete refresh, and A or a indicates always refresh. A and C are equivalent. P or p refreshes by recomputing the rows in the materialized view affected by changed partitions in the detail tables. L or l indicates LPCT refresh, that is, recomputing the rows in the materialized view as per the logical partition of the base tables.
	
	Note:
	"L" and "?" refreshes attempt a combined PCT and LCPT refresh. (physical and logical partitions).
	
	If a materialized view does not have a corresponding refresh method (that is, if more materialized views are specified than refresh methods), then that materialized view is refreshed according to its default refresh method. For example, consider the following EXECUTE statement within SQL*Plus:
	
	`DBMS_MVIEW.REFRESH`
	`('countries_mv,regions_mv,hr.employees_mv','cf');`
	
	This statement performs a complete refresh of the countries_mv materialized view, a fast refresh of the regions_mv materialized view, and a default refresh of the hr.employees materialized view.

rollback_seg	Name of the materialized view site rollback segment to use while refreshing materialized views
push_deferred_rpc	Used by updatable materialized views only. Set this parameter to true if you want to push changes from the materialized view to its associated master tables or master materialized views before refreshing the materialized view. Otherwise, these changes may appear to be temporarily lost.
refresh_after_errors	If this parameter is true, an updatable materialized view continues to refresh even if there are outstanding conflicts logged in the DEFERROR view for the materialized view's master table or master materialized view. If this parameter is true and atomic_refresh is false, this procedure continues to refresh other materialized views if it fails while refreshing a materialized view.
purge_option	If you are using the parallel propagation mechanism (in other words, parallelism is set to 1 or greater), 0 means do not purge, 1 means lazy purge, and 2 means aggressive purge. In most cases, lazy purge is the optimal setting. Set purge to aggressive to trim the queue if multiple master replication groups are pushed to different target sites, and updates to one or more replication groups are infrequent and infrequently pushed. If all replication groups are infrequently updated and pushed, then set this parameter to 0 and occasionally execute PUSH with this parameter set to 2 to reduce the queue.
parallelism	0 specifies serial propagation. *n* > 1 specifies parallel propagation with *n* parallel processes. 1 specifies parallel propagation using only one parallel process.

Table 135-10 (Cont.) REFRESH Procedure Parameters

Parameter	Description
heap_size	Maximum number of transactions to be examined simultaneously for parallel propagation scheduling. Oracle automatically calculates the default setting for optimal performance. Note: Do not set this parameter unless directed to do so by Oracle Support Services.
atomic_refresh	If this parameter is set to true, then the list of materialized views is refreshed in a single transaction. All of the refreshed materialized views are updated to a single point in time. If the refresh fails for any of the materialized views, none of the materialized views are updated. If this parameter is set to false, then each of the materialized views is refreshed non-atomically in separate transactions. As part of complete refresh, if truncate is used (non-atomic refresh), unique index rebuild is executed. INDEX REBUILD automatically computes statistics. Thus, statistics are updated for truncated tables.
nested	If true, then perform nested refresh operations for the specified set of materialized views. Nested refresh operations refresh all the depending materialized views and the specified set of materialized views based on a dependency order to ensure the nested materialized views are truly fresh with respect to the underlying base tables.
out_of_place	If true, then it performs an out-of-place refresh. The default is false. This parameter uses the four methods of refresh (F, P, C, ?). So, for example, if you specify F and out_of_place = true, then an out-of-place fast refresh will be attempted. Similarly, if you specify P and out_of_place = true, then out-of-place PCT refresh will be attempted.
skip_ext_data	Provides you an option to skip the MV data refresh corresponding to the external partitions.

Usage Notes

- The preference of LPCT refresh on the materialized view is:
  1. If the materialized view is both PCT and LPCT enabled, a combined LPCT+PCT refresh is performed.
  2. If the materialized view is only LPCT enabled, LPCT refresh is performed.

> **See Also:**
>
> *Oracle Database Data Warehousing Guide* for more information about refresh methods for materialized view .

REFRESH_ALL_MVIEWS Procedure

This procedure refreshes all materialized views that have certain properties

All materialized views with the following properties are refreshed:

- The materialized view has not been refreshed since the most recent change to a master table or master materialized view on which it depends.

- The materialized view and all of the master tables or master materialized views on which it depends are local.
- The materialized view is in the view `DBA_MVIEWS`.

This procedure is intended for use with data warehouses.

Syntax

```
DBMS_MVIEW.REFRESH_ALL_MVIEWS (
   number_of_failures     OUT   BINARY_INTEGER,
   method                 IN    VARCHAR2         := NULL,
   rollback_seg           IN    VARCHAR2         := NULL,
   refresh_after_errors   IN    BOOLEAN          := false,
   atomic_refresh         IN    BOOLEAN          := true,
   out_of_place           IN    BOOLEAN          := false);
```

Parameters

Table 135-11 REFRESH_ALL_MVIEWS Procedure Parameters

Parameter	Description
`number_of_failures`	Returns the number of failures that occurred during processing
`method`	A single refresh method indicating the type of refresh to perform for each materialized view that is refreshed. `F` or `f` indicates fast refresh, `?` indicates force refresh, `C` or c indicates complete refresh, and `A` or a indicates always refresh. `A` and `C` are equivalent. If no method is specified, a materialized view is refreshed according to its default refresh method. `P` or `p` refreshes by recomputing the rows in the materialized view affected by changed partitions in the detail tables. `L` or `l` indicates LPCT refresh, that is, recomputing the rows in the materialized view as per the logical partition of the base tables. **Note:** "L" and "?" refreshes attempt a combined PCT and LCPT refresh. (physical and logical partitions).
`rollback_seg`	Name of the materialized view site rollback segment to use while refreshing materialized views
`refresh_after_errors`	If this parameter is `true`, an updatable materialized view continues to refresh even if there are outstanding conflicts logged in the `DEFERROR` view for the materialized view's master table or master materialized view. If this parameter is `true` and `atomic_refresh` is `false`, this procedure continues to refresh other materialized views if it fails while refreshing a materialized view.
`atomic_refresh`	If this parameter is set to `true`, then the refreshed materialized views are refreshed in a single transaction. All of the refreshed materialized views are updated to a single point in time. If the refresh fails for any of the materialized views, none of the materialized views are updated. If this parameter is set to `false`, then each of the materialized views is refreshed non-atomically in separate transactions.

Table 135-11 (Cont.) REFRESH_ALL_MVIEWS Procedure Parameters

Parameter	Description
out_of_place	If true, then it performs an out-of-place refresh. The default is false. This parameter uses the four methods of refresh (F, P, C, ?). So, for example, if you specify F and out_of_place = true, then an out-of-place fast refresh will be attempted. Similarly, if you specify P and out_of_place = true, then out-of-place PCT refresh will be attempted.

REFRESH_DEPENDENT Procedures

This procedure refreshes all materialized views that have certain properties.

Materialized views with the following properties are refreshed:

- The materialized view depends on a master table or master materialized view in the list of specified masters.
- The materialized view has not been refreshed since the most recent change to a master table or master materialized view on which it depends.
- The materialized view and all of the master tables or master materialized views on which it depends are local.
- The materialized view is in the view DBA_MVIEWS.

This procedure is intended for use with data warehouses.

Syntax

```
DBMS_MVIEW.REFRESH_DEPENDENT (
   number_of_failures     OUT    BINARY_INTEGER,
   { list                 IN     VARCHAR2,
   | tab                  IN     DBMS_UTILITY.UNCL_ARRAY,}
   method                 IN     VARCHAR2    := NULL,
   rollback_seg           IN     VARCHAR2    := NULL,
   refresh_after_errors   IN     BOOLEAN     := false,
   atomic_refresh         IN     BOOLEAN     := true,
   nested                 IN     BOOLEAN     := false,
   out_of_place           IN     BOOLEAN     := false);
```

> **Note:**
>
> This procedure is overloaded. The list and tab parameters are mutually exclusive.

Parameters

Table 135-12 REFRESH_DEPENDENT Procedure Parameters

Parameter	Description
number_of_failures	Returns the number of failures that occurred during processing

Table 135-12 (Cont.) REFRESH_DEPENDENT Procedure Parameters

Parameter	Description
`list | tab`	Comma-delimited list of master tables or master materialized views on which materialized views can depend. (Synonyms are not supported.) These tables and the materialized views that depend on them can be located in different schemas. However, all of the tables and materialized views must be in your local database. Alternatively, you may pass in a PL/SQL index-by table of type `DBMS_UTILITY.UNCL_ARRAY`, where each element is the name of a table.
`method`	A string of refresh methods indicating how to refresh the dependent materialized views. All of the materialized views that depend on a particular table are refreshed according to the refresh method associated with that table. `F` or `f` indicates fast refresh, ? indicates force refresh, `C` or `c` indicates complete refresh, and `A` or `a` indicates always refresh. `A` and `C` are equivalent. `P` or `p` refreshes by recomputing the rows in the materialized view affected by changed partitions in the detail tables. `L` or `l` indicates LPCT refresh, that is, recomputing the rows in the materialized view as per the logical partition of the base tables. **Note:** "L" and "?" refreshes attempt a combined PCT and LCPT refresh. (physical and logical partitions). If a table does not have a corresponding refresh method (that is, if more tables are specified than refresh methods), then any materialized view that depends on that table is refreshed according to its default refresh method. For example, the following `EXECUTE` statement within SQL*Plus: `DBMS_MVIEW.REFRESH_DEPENDENT ('employees,deptartments,hr.regions','cf');` performs a complete refresh of the materialized views that depend on the `employees` table, a fast refresh of the materialized views that depend on the `departments` table, and a default refresh of the materialized views that depend on the `hr.regions` table.
`rollback_seg`	Name of the materialized view site rollback segment to use while refreshing materialized views
`refresh_after_errors`	If this parameter is `true`, an updatable materialized view continues to refresh even if there are outstanding conflicts logged in the `DEFERROR` view for the materialized view's master table or master materialized view. If this parameter is `true` and `atomic_refresh` is `false`, this procedure continues to refresh other materialized views if it fails while refreshing a materialized view.
`atomic_refresh`	If this parameter is set to `true`, then the refreshed materialized views are refreshed in a single transaction. All of the refreshed materialized views are updated to a single point in time. If the refresh fails for any of the materialized views, none of the materialized views are updated. If this parameter is set to `false`, then each of the materialized views is refreshed non-atomically in separate transactions.
`nested`	If `true`, then perform nested refresh operations for the specified set of tables. Nested refresh operations refresh all the depending materialized views of the specified set of tables based on a dependency order to ensure the nested materialized views are truly fresh with respect to the underlying base tables.

Table 135-12 (Cont.) REFRESH_DEPENDENT Procedure Parameters

Parameter	Description
out_of_place	If true, then it performs an out-of-place refresh. The default is false. This parameter uses the four methods of refresh (F, P, C, ?). So, for example, if you specify F and out_of_place = true, then an out-of-place fast refresh will be attempted. Similarly, if you specify P and out_of_place = true, then out-of-place PCT refresh will be attempted.

REGISTER_MVIEW Procedure

This procedure enables the administration of individual materialized views. It is invoked at a master site or master materialized view site to register a materialized view.

Note that, typically, a materialized view is registered automatically during materialized view creation. You should only run this procedure to manually register a materialized view if the automatic registration failed or if the registration information was deleted.

Syntax

```
DBMS_MVIEW.REGISTER_MVIEW (
   mviewowner  IN   VARCHAR2,
   mviewname   IN   VARCHAR2,
   mviewsite   IN   VARCHAR2,
   mview_id    IN   DATE | BINARY_INTEGER,
   flag        IN   BINARY_INTEGER,
   qry_txt     IN   VARCHAR2,
   rep_type    IN   BINARY_INTEGER := DBMS_MVIEW.REG_UNKNOWN);
```

Parameters

Table 135-13 REGISTER_MVIEW Procedure Parameters

Parameter	Description
mviewowner	Owner of the materialized view.
mviewname	Name of the materialized view.
mviewsite	Name of the materialized view site for a materialized view registering at an Oracle database version 8.x and higher master site or master materialized view site. This name should not contain any double quotes.
mview_id	The identification number of the materialized view. Specify an Oracle database version 8.x and higher materialized view as a BINARY_INTEGER. Specify an Oracle database version 7 materialized view registering at an Oracle database version 8.x and higher master sites or master materialized view sites as a DATE.

Table 135-13 (Cont.) REGISTER_MVIEW Procedure Parameters

Parameter	Description
flag	A constant that describes the properties of the materialized view being registered. Valid constants that can be assigned include the following: DBMS_MVIEW.REG_ROWID_MVIEW for a rowid materialized view DBMS_MVIEW.REG_PRIMARY_KEY_MVIEW for a primary key materialized view DBMS_MVIEW.REG_OBJECT_ID_MVIEW for an object id materialized view DBMS_MVIEW.REG_FAST_REFRESHABLE_MVIEW for a materialized view that can be fast refreshed DBMS_MVIEW.REG_UPDATABLE_MVIEW for a materialized view that is updatable A materialized view can have more than one of these properties. In this case, use the plus sign (+) to specify more than one property. For example, if a primary key materialized view can be fast refreshed, you can enter the following for this parameter: DBMS_MVIEW.REG_PRIMARY_KEY_MVIEW + DBMS_MVIEW.REG_FAST_REFRESHABLE_MVIEW You can determine the properties of a materialized view by querying the ALL_MVIEWS data dictionary view.
qry_txt	The first 32,000 bytes of the materialized view definition query.
rep_type	Version of the materialized view. Valid constants that can be assigned include the following: DBMS_MVIEW.REG_V7_SNAPSHOT if the materialized view is at an Oracle database version 7 site • DBMS_MVIEW.REG_V8_SNAPSHOT reg_repapi_snapshot if the materialized view is at an Oracle database version 8.x or higher site DBMS_MVIEW.REG_UNKNOWN (the default) if you do not know whether the materialized view is at an Oracle database version 7 site or an Oracle database version 8.x (or higher) site

Usage Notes

This procedure is invoked at the master site or master materialized view site by a remote materialized view site using a remote procedure call. If REGISTER_MVIEW is called multiple times with the same mviewowner, mviewname, and mviewsite, then the most recent values for mview_id, flag, and qry_txt are stored. If a query exceeds the maximum VARCHAR2 size, then qry_txt contains the first 32000 characters of the query and the remainder is truncated. When invoked manually, the value of mview_id must be looked up in the materialized view data dictionary views by the person who calls the procedure.

UNREGISTER_MVIEW Procedure

This procedure enables the administration of individual materialized views. It is invoked at a master site or master materialized view site to unregister a materialized view.

Syntax

```
DBMS_MVIEW.UNREGISTER_MVIEW (
   mviewowner      IN   VARCHAR2,
   mviewname       IN   VARCHAR2,
   mviewsite       IN   VARCHAR2);
```

Parameters

Table 135-14 UNREGISTER_MVIEW Procedure Parameters

Parameters	Description
`mviewowner`	Owner of the materialized view
`mviewname`	Name of the materialized view
`mviewsite`	Name of the materialized view site