199
DBMS_SYNC_REFRESH

The `DBMS_SYNC_REFRESH` package provides an interface to perform a synchronous refresh of materialized views.

> **See Also:**
>
> *Oracle Database Data Warehousing Guide* for more information on using `DBMS_SYNC_REFRESH`

This chapter contains the following topics:

DBMS_SYNC_REFRESH Overview

Synchronous refresh is a refresh method introduced in Oracle Database Release 12*c*, which enables you to keep a set of tables and the materialized views defined on them to be always in sync.

> **See Also:**
>
> *Oracle Database Data Warehousing Guide* for more information about using synchronous refresh

DBMS_SYNC_REFRESH Security Model

The execute privilege for this package is granted to `PUBLIC`, so all users can execute the procedures in this package to perform synchronous refresh on objects owned by them. The database administrator can perform synchronous refresh operations on all tables and materialized views in the database.

In general, if a user without the `DBA` privilege wants to use synchronous refresh on another user's table, he must complete privileges to read from and write to that table, that it, the user must have the `SELECT` or `READ`, `INSERT`, `UPDATE`, and `DELETE` privileges on that table or materialized view. A couple of exceptions are:

- `PURGE_REFRESH_STATS` and `ALTER_REFRESH_STATS_RETENTION` Functions

  These two functions implement the purge policy and can be used to change the default retention period. These functions can be only be executed by the database administrator.

- The `CAN_SYNCREF_TABLE` Function

  This is an advisory function which examines the eligibility for sync refresh of all the materialized views associated with a specified table. Hence, this function requires the `SELECT` or `READ` privilege on all materialized views associated with the specified table.

Summary of DBMS_SYNC_REFRESH Subprograms

This table lists and briefly describes the `DBMS_SYNC_REFRESH` package subprograms.

Table 199-1 DBMS_SYNC_REFRESH Package Subprograms

Subprogram	Description
ABORT_REFRESH Procedure	Aborts a refresh.
ALTER_REFRESH_STATS_RETENTION Procedure	Alters the refresh history retention value, specified in days.
CAN_SYNCREF_TABLE Procedure	Advises on whether a table and its dependent materialized views are eligible for synchronous refresh.
EXECUTE_REFRESH Procedure	Executes synchronous refresh on the synchronous refresh groups.
GET_ALL_GROUP_IDS Function	Returns the group IDs of all the synchronous refresh groups in the database.
GET_GROUP_ID Function	Returns the group ID of a table or materialized view.
GET_GROUP_ID_LIST Function	Returns the group IDs of the tables in a given list of objects (tables or materialized views).
PREPARE_REFRESH Procedure	Prepares the sync refresh groups for refresh.
PREPARE_STAGING_LOG Procedure	Validates and collects statistics on the data in the staging log.
PURGE_REFRESH_STATS Procedure	Purges the refresh history of sync refreshes that took place within a time specified by a timestamp parameter.
REGISTER_MVIEWS	Registers materialized views for synchronous refresh.
REGISTER_PARTITION_OPERATION Procedure	Registers a partition maintenance operation on a partition of a base table.
UNREGISTER_MVIEWS	Unregisters materialized views from synchronous refresh.
UNREGISTER_PARTITION_OPERATION Procedure	Unregisters a partition maintenance operation on a partition of a base table.

ABORT_REFRESH Procedure

This procedure undoes all the changes made by `PREPARE_REFRESH` or `EXECUTE_REFRESH` for the specified sync refresh groups. It helps you to recover to a state where the tables and materialized views are usable and consistent in case they encounter unexpected errors.

This procedure is overloaded.

Syntax

```
DBMS_SYNC_REFRESH.ABORT_REFRESH (
   group_id      IN NUMBER);

DBMS_SYNC_REFRESH.ABORT_REFRESH (
   group_id_list IN DBMS_UTILITY.NUMBER_ARRAY);
```

Parameters

Table 199-2 ABORT_REFRESH Procedure Parameters

Parameter	Description
group_id	The group ID of a sync refresh group.
group_id_list	An array of group IDs of the sync refresh groups to be aborted for sync refresh.

Usage Notes

If called after PREPARE_REFRESH, this procedure drops the outside tables created by it and unlocks the tables and materialized views in the sync refresh group.

If called after EXECUTE_REFRESH fails, this procedure restores the state of tables to before EXECUTE_REFRESH by undoing any partition exchanges which successfully finished.

This procedure releases the locks placed on the tables in the sync refresh group which were placed on them by the PREPARE_REFRESH procedure. See "PREPARE_REFRESH Procedure" for a description of these locks.

ABORT_REFRESH will work only if a PREPARE_REFRESH or EXECUTE_REFRESH statement has failed. It cannot be used after successful runs of those commands, and throws an error in such cases.

ALTER_REFRESH_STATS_RETENTION Procedure

This procedure alters the refresh history retention value, specified in days. It is intended for use in conjunction with PURGE_REFRESH_HISTORY. It also requires the SYSDBA privilege in addition to the privilege to execute it.

Syntax

```
DBMS_SYNC_REFRESH.ALTER_REFRESH_STATS_RETENTION (
   retention   IN NUMBER);
```

Parameters

Table 199-3 ALTER_REFRESH_STATS_RETENTION Procedure Parameters

Parameter	Description
retention	The retention time in days. The refresh history will be retained for at least these many number of days. The valid range is 1 to 365,000. You can use the following values for special purposes: • -1 - Refresh history is never purged by PREPARE_REFRESH. • 0 - Old refresh history is never saved. PREPARE_REFRESH will delete all refresh history. • NULL - Change refresh history retention to default value.

CAN_SYNCREF_TABLE Procedure

This procedure advises on whether a table and its dependent materialized views are eligible for sync refresh. It provides an explanation of its analysis. If not eligible, you can examine the reasons and take appropriate action if possible.

This procedure lists all of the table's dependent materialized views and whether they qualify for sync refresh. Note that a materialized view may qualify for sync refresh even though the base table may not.

The eligibility rules for materialized views for synchronous refresh are discussed in detail in *Oracle Database Data Warehousing Guide.*

You can invoke `CAN_SYNCREF_TABLE` in two ways. The first is to use a table, while the second is to create a `VARRAY`.

Syntax

```
DBMS_SYNC_REFRESH.CAN_SYNCREF_TABLE (
    schema_name     IN VARCHAR2,
    table_name      IN VARCHAR2,
    statement_id    IN VARCHAR2);

DBMS_SYNC_REFRESH.CAN_SYNCREF_TABLE (
    schema_name     IN VARCHAR2,
    table_name      IN VARCHAR2,
    output_array    IN OUT Sys.CanSyncRefTypeArray);
```

Note that only one of `statement_id` or `output_array` need be provided to `CAN_SYNCREF_TABLE`.

Parameters

Table 199-4 CAN_SYNCREF_TABLE Procedure Parameters

Parameter	Description
`schema_name`	The name of the schema of the base table.
`table_name`	The name of the base table.
`statement_id`	A string (`VARCHAR2(30)`) to identify the rows pertaining to an invocation of `CAN_SYNCREF_TABLE` when the output is directed to a table named `SYNCREF_TABLE` in the user's schema.
`output_array`	The output array into which `CAN_SYNCREF_TABLE` records the information on the eligibility of the base table and its dependent materialized views for synchronous refresh.

Using SYNCREF_TABLE

The output of `CAN_SYNCREF_TABLE` can be directed to a table named `SYNCREF_TABLE`. The user is responsible for creating the `SYNCREF_TABLE`; it can be dropped when it is no longer needed. Its structure is as follows:

```
CREATE TABLE SYNCREF_TABLE (
   statement_id    VARCHAR2(30),
   schema_name     VARCHAR2(30),
   table_name      VARCHAR2(30),
   mv_schema_name  VARCHAR2(30),
```

```
   mv_name         VARCHAR2(30),
   eligible        VARCHAR2(1),
   seq_num         NUMBER,
   msg_number      NUMBER,
   message         VARCHAR2(4000));
```

Using a VARRAY

You can save the output of `CAN_SYNCREF_TABLE` in a PL/SQL `VARRAY`. The elements of this array are of type `CanSyncRefMessage`, which is predefined in the `SYS` schema, as shown in the following:

```
TYPE CanSyncRefMessage IS OBJECT (
   schema_name     VARCHAR2(30),
   table_name      VARCHAR2(30),
   mv_schema_name  VARCHAR2(30),
   mv_name         VARCHAR2(30),
   eligible        VARCHAR2(1),
   seq_num         NUMBER,
   msg_number      NUMBER,
   message         VARCHAR2(4000));
```

The array type `CanSyncRefArrayType`, which is a varray of `CanSyncRefMessage` objects, is predefined in the `SYS` schema as follows:

```
TYPE CanSyncRefArrayType AS VARRAY(256) OF CanSyncRefMessage;
```

Each `CanSyncRefMessage` record provides a message concerning the eligibility of the base table or a dependent materialized view for synchronous refresh. The semantics of the fields is the same as that of the corresponding fields in the `SYNCREF_TABLE`. However, the `SYNCREF_TABLE` has a `statement_id` field which is absent in `CanSyncRefMessage` because no `statement_id` is supplied (because it is not required) when `CAN_SYNCREF_TABLE` is called with a `VARRAY` parameter.

EXECUTE_REFRESH Procedure

This procedure executes sync refresh on the sync refresh groups prepared by `DBMS_SYNC_REFRESH.PREPARE_REFRESH`. These groups are identified by their group IDs.

Note this procedure will only perform the refresh on those materialized views that have been registered for synch refresh; any other materialized views will become stale once this procedure completes.

For more information on how to monitor the status of the two synchronous refresh operations, `PREPARE_REFRESH` and `EXECUTE_REFRESH` and how to troubleshoot errors that might occur using the information in the catalog views, refer to "Trouble-Shooting Synchronous Refresh Operations" in *Oracle Database Data Warehousing Guide.*

This procedure is overloaded.

Syntax

```
DBMS_SYNC_REFRESH.EXECUTE_REFRESH (
   group_id   IN NUMBER);

DBMS_SYNC_REFRESH.EXECUTE_REFRESH (
   group_id_list  IN DBMS_UTILITY.NUMBER_ARRAY);
```

Parameters

Table 199-5 EXECUTE_REFRESH Procedure Parameters

Parameter	Description
group_id	The group ID of a sync refresh group.
group_id_list	An array of group IDs of the sync refresh groups to be executed for sync refresh.

Usage Notes

This procedure also releases the locks placed on the tables in the sync refresh group that were placed on them by the PREPARE_REFRESH procedure. See "PREPARE_REFRESH Procedure" for a description of these locks and *Oracle Database Reference* for information regarding the status of the refresh operation after DBMS_SYNC_REFRESH.EXECUTE_REFRESH.

GET_ALL_GROUP_IDS Function

This function returns the group IDs of all the sync refresh groups in the database.

Syntax

```
FUNCTION DBMS_SYNC_REFRESH.GET_ALL_GROUP_IDS
           RETURN DBMS_UTILITY.NUMBER_ARRAY;
```

Parameters

Table 199-6 GET_ALL_GROUP_IDS Function Parameter

Parameter	Description
get_all_group_ids	Returns the group IDs of all the sync refresh groups in the database.

GET_GROUP_ID Function

This function returns the group ID of a materialized view. The group ID identifies the sync refresh group the table belongs to. A sync refresh group is a group of related tables and their dependent materialized views which must be all refreshed together jointly to ensure consistency and correctness.

Syntax

```
DBMS_SYNC_REFRESH.GET_GROUP_ID (
   object_name_list    IN VARCHAR2)
RETURN DBMS_UTILITY.NUMBER_ARRAY;
```

Parameters

Table 199-7 GET_GROUP_ID Function Parameter

Parameter	Description
object_name_list	The name of the materialized view. The name can be schema-qualified.

GET_GROUP_ID_LIST Function

This function returns the group IDs of the tables in a given list of objects (materialized views).

Syntax

```
DBMS_SYNC_REFRESH.GET_GROUP_ID_LIST (
   object_name_list    IN VARCHAR2)
RETURN DBMS_UTILITY.NUMBER_ARRAY;
```

Parameters

Table 199-8 GET_GROUP_ID_LIST Function Parameter

Parameter	Description
object_name_list	A comma-separated list of object names (materialized views). Each name can be schema-qualified.

PREPARE_REFRESH Procedure

This procedure prepares for refresh the sync refresh groups identified by the group ID in the input.

A sync refresh group consists of a set of related tables and all materialized views dependent on those base tables. Note this procedure will only prepare for refresh those dependent materialized views that have been registered for synchronous refresh.

For more information on how to monitor the status of the two synchronous refresh operations, PREPARE_REFRESH and EXECUTE_REFRESH and how to troubleshoot errors that might occur using the information in the catalog views, refer to "Trouble-Shooting Synchronous Refresh Operations" in *Oracle Database Data Warehousing Guide.*

Syntax

```
DBMS_SYNC_REFRESH.PREPARE_REFRESH (
   group_id    IN NUMBER)
RETURN DBMS_UTILITY.NUMBER_ARRAY;
```

Parameters

Table 199-9 PREPARE_REFRESH Procedure Parameters

Parameter	Description
group_id	The group ID of the sync refresh group to be prepared for sync refresh.

Usage Notes

This procedure plans the three phases of the sync refresh operation and executes the steps associated with the prepare phase itself. These steps include identifying the partitions of the fact tables and materialized views that have been changed, and computing their new values as a result of the changes. The new values of the partitions are stored in tables called outside tables that are exchanged into their corresponding partitions at the time of the EXECUTE_REFRESH.

Before running this procedure, the user must run PREPARE_STAGING_LOG on all tables in the group. This is required even for staging logs that do not have changes in them. The user must also register any partition operations on the tables in the group using the REGISTER_PARTITION_OPERATION.

One of the side effects of this procedure is that the tables being prepared are locked in this sense: the staging logs of the tables will be locked to prevent any DMLs from occurring and the registration of partition operations will be disabled. These locks will be in effect until you issue an EXECUTE_REFRESH statement. Alternatively, you can issue an ABORT_REFRESH operation to release these locks. Another side effect of this procedure is that it purges from the catalog records of earlier sync refresh operations; if they are older than the retention period, they are purged.

The degree of parallelism of the prepare refresh job is inherited from the session parameters which you can control with an ALTER SESSION statement.

The group ID of a table can be found using GET_GROUP_ID(table_name). The group IDs of a list of tables can be found with GET_GROUP_ID_LIST(table_name_list). The group IDs of all the lists of tables can be retrieved with GET_ALL_GROUP_IDS.

By default, synchronous refresh does not maintain global indexes belonging to the tables and materialized views in the sync refresh group. If you wish to do so, you can set event 31904, level 64 before executing PREPARE_REFRESH. This will cause the partition exchange DDL statements generated by PREPARE_REFRESH to have the UPDATE INDEXES clause appended to them, and when they are executed by EXECUTE_REFRESH, the global indexes will be maintained.

PREPARE_STAGING_LOG Procedure

This procedure collects statistics on the data in the staging log of the base table and validates the data in the log.

It can be run in several different modes ranging from the enforced mode in which strict checking of the data is done to trusted mode in which no checking is done. You should run this procedure after loading the staging log and before running PREPARE_REFRESH.

In the enforced mode, which is the default, this procedure will fill in the missing values of the columns of the rows being deleted or updated. An error is thrown if any violations of the staging-log rules are found. You can query the view USER_SR_STLOG_EXCEPTIONS to get details on the exceptions.

The notion of the staging log key is described in *Oracle Database Data Warehousing Guide.*

In the enforced mode, this procedure processes each delete/update row in the staging log as follows:

- It verifies the existence of the row in the base table using the key.
- For the rows being deleted (DMLTYPE$$ is 'D'), it verifies a row with this key exists in the base table; if non-null non-key values are supplied in the staging log, it verifies the values match the corresponding columns in the base table; else an exception is logged in the exceptions table. If the values of any of the non-key columns are missing, it fills in those values from the row in the base table.
- For the rows being updated (DMLTYPE$$ is 'UO' or 'UN'), it verifies a row with this key exists in the base table. In the old values row (DMLTYPE$$ is 'UO'), it makes the same check and does the same processing as with rows being deleted. In the new values row (DMLTYPE$$ is 'UN'), it checks that at least the value of one the columns differs from its old value; else an exception is logged.

- In the new values row (DMLTYPE$$ is 'UN'), a null value in a column is interpreted as having the same value as the old value of the column except if the old value is non-null and the new value is null in which case, the new value of the column is interpreted as being null. This requires that the user must provide the old value of columns which are being updated to NULL.

In the default enforced mode, this procedure verifies that each key is specified for at most once for a delete or update operation. This means that the user, when doing the change consolidation, must consolidate delete-insert of the same row into an update operation with rows 'UO' and 'UN;' multiple updates must be consolidated into a single update; and null changes such as an insert-update-delete of the same row must not appear in the staging log.

The checking done in the enforced mode can be time-consuming. If you are confident in the integrity of the data, you can choose a lower level of checking. You can choose to:

- trust all the insert rows (DMLTYPE$$ is 'I') by choosing the psl_mode of DBMS_SYNC_REFRESH.INSERT_TRUSTED
- trust all the delete rows (DMLTYPE$$ is 'D') by choosing the psl_mode of DBMS_SYNC_REFRESH.DELETE_TRUSTED
- trust all the update rows (DMLTYPE$$ is 'UO' or 'UN') by choosing the psl_mode of DBMS_SYNC_REFRESH.UPDATE_TRUSTED
- trust all three types of DMLs by choosing the psl_mode of DBMS_SYNC_REFRESH.TRUSTED.

In addition, you can specify the psl_mode as a bitmask of the flags described above. For example, DBMS_SYNC_REFRESH.INSERT_TRUSTED + DBMS_SYNC_REFRESH_DELETE_TRUSTED will treat inserts and deletes to be trusted but not updates.

Syntax

```
DBMS_SYNC_REFRESH.PREPARE_STAGING_LOG (
   schema_name      IN VARCHAR2,
   base_table_name  IN VARCHAR2,
   psl_mode         IN NUMBER DEFAULT
   DBMS_SYNC_REFRESH.ENFORCED);
```

Parameters

Table 199-10 PREPARE_STAGING_LOG Procedure Parameters

Parameter	Description
schema_name	The name of the schema of the base table.
base_table_name	The name of the base table.
psl_mode	The mode in which staging log preparation should be done. The possible values are: • DBMS_SYNC_REFRESH.ENFORCED (the default) • DBMS_SYNC_REFRESH.INSERT_TRUSTED • DBMS_SYNC_REFRESH.DELETE_TRUSTED • DBMS_SYNC_REFRESH.UPDATE_TRUSTED • DBMS_SYNC_REFRESH.TRUSTED

PURGE_REFRESH_STATS Procedure

This procedure purges the refresh history of sync refreshes that took place before the value specified by the `BEFORE_TIMESTAMP` parameter.

This procedure requires the `SYSDBA` privilege in addition to the privilege to execute it.

Syntax

```
DBMS_SYNC_REFRESH.PURGE_REFRESH_STATS (
   before_timestamp IN TIMESTAMP WITH TIME ZONE);
```

Parameters

Table 199-11 PURGE_REFRESH_STATS Procedure Parameter

Parameter	Description
`before_timestamp`	Records of sync refreshes saved before this timestamp are purged. If `NULL`, it uses the purging policy used by automatic purge. The automatic purge deletes all history older than (current time - refresh - history retention). The refresh history retention value can be changed using `ALTER_REFRESH_STATS_RETENTION`. The default is 31 days.

REGISTER_MVIEWS

This procedure registers a list of materialized views for synchronous refresh.

It checks each materialized view in the list for eligibility and places it in the sync refresh group it belongs to. A sync refresh group is a set of related tables and materialized views defined on top of them. Two tables are considered related if there is a referential constraint between them.

The eligibility rules of materialized views for synchronous refresh are described in detail in *Oracle Database Data Warehousing Guide.* The principal requirements are that the materialized view must be partitioned and its partition key must be derivable from the partition key of its fact table. The materialized view definition must specify the `USING TRUSTED CONSTRAINTS` clause because sync refresh trusts the foreign key and primary key relationships to perform various refresh optimizations. The materialized view's refresh policy must be specified as `ON DEMAND`.

You have an option to register only some of the materialized views associated with a table, and leave some unregistered. Oracle Corporation does not recommend this, and in such a case, the user has to maintain the unregistered ones using the PCT or complete refresh methods.

A staging table must have been created for each base table of each materialized view in the materialized view list (`mv_list`), or else an error is thrown.

If any of the materialized views are not eligible for sync refresh, an error is thrown and the registration of all materialized views in the materialized view list fails.

Syntax

```
DBMS_SYNC_REFRESH.REGISTER_MVIEWS (
   mv_list   IN VARCHAR2);
```

Parameter

Table 199-12 REGISTER_MVIEWS Procedure Parameter

Parameter	Description
mv_list	A comma-delimited list of materialized views to register. These names are optionally schema-qualified.

REGISTER_PARTITION_OPERATION Procedure

This procedure registers a partition-maintenance operation (PMOP) on a partition of a base table.

Syntax

```
DBMS_SYNC_REFRESH.REGISTER_PARTITION_OPERATION (
   partition_op                IN VARCHAR2,
   schema_name                 IN VARCHAR2,
   base_table_name             IN VARCHAR2,
   partition_name              IN VARCHAR2,
   outside_partn_table_schema  IN VARCHAR2,
   outside_partn_table_name    IN VARCHAR2);
```

Parameters

Table 199-13 REGISTER_PARTITION_OPERATION Procedure Parameters

Parameter	Description
partition_op	The name of the partition operation (DROP, EXCHANGE, or TRUNCATE).
schema_name	The name of the schema of the base table.
base_table_name	The name of the base table.
partition_name	The name of the partition to be changed; either exchanged with the outside partition table or dropped or truncated.
outside_partn_table_schema	The name of the schema of the outside partition table (required for EXCHANGE only).
outside_partn_table_name	The name of the outside partition table (required for EXCHANGE only).

Usage Notes

The three kinds of change operations that may be specified on partitions are DROP, TRUNCATE, and EXCHANGE.

If DROP is specified, then the partition will be dropped from the base table at the time of EXECUTE_REFRESH. If TRUNCATE is specified, then the data from the partition will be deleted but the partition itself will not be dropped. These operations provide a more efficient way of specifying the deletes of all the rows in a partition than specifying them individually in the staging log.

If EXCHANGE is specified, then the contents of the outside table is exchanged with contents of the specified partition of EXECUTE_REFRESH. This provides an alternative method to the user of providing the changes to the base tables instead of populating the staging log.

UNREGISTER_MVIEWS

This procedure unregisters a list of materialized views from synchronous refresh. Once a materialized view is unregistered, it can be maintained by the user with any of the traditional refresh methods, such as complete or PCT, refresh.

Syntax

```
DBMS_SYNC_REFRESH.UNREGISTER_MVIEWS (
   mv_list   IN VARCHAR20;
```

Parameter

Table 199-14 UNREGISTER_MVIEWS Parameter

Parameter	Description
mv_list	A comma-delimited list of materialized views to unregister. These names are optionally schema-qualified.

UNREGISTER_PARTITION_OPERATION Procedure

This procedure unregisters a partition-maintenance operation (PMOP) that had been previously registered with REGISTER_PARTITION_OPERATION on a base table. The three kinds of change operations that can be specified on partitions are DROP, TRUNCATE, and EXCHANGE.

Syntax

```
DBMS_SYNC_REFRESH.UNREGISTER_PARTITION_OPERATION (
   partition_op     IN VARCHAR2,
   schema_name      IN VARCHAR2,
   base_table_name  IN VARCHAR2,
   partition_name   IN VARCHAR2);
```

Parameters

Table 199-15 UNREGISTER_PARTITION_OPERATION Procedure Parameters

Parameter	Description
partition_op	The name of the partition operation (DROP, EXCHANGE, or TRUNCATE).
schema_name	The name of the schema of the base table.
base_table_name	The name of the base table.
partition_name	The name of the partition to be changed; either exchanged with the outside partition table or dropped or truncated.