160 DBMS_REDEFINITION

The `DBMS_REDEFINITION` package provides an interface to perform an online redefinition of tables.

This chapter contains the following topics:

See Also:

Oracle Database Administrator's Guide for more information about online redefinition of tables

DBMS_REDEFINITION Overview

To achieve online redefinition, incrementally maintainable local materialized views are used. These logs keep track of the changes to the master tables and are used by the materialized views during refresh synchronization.

DBMS_REDEFINITION Security Model

Subprograms in the `DBMS_REDEFINITION` package are run with invokers' rights (with the privileges of the current user).

There are two modes:

- In `USER` mode, the user who has the `CREATE TABLE` and `CREATE MVIEW` privileges may redefine a table residing in his own schema.
- In `FULL` mode, the user who has the `ANY` privilege may redefine tables in any schema.

DBMS_REDEFINITION Constants

The `DBMS_REDEFINITION` package defines several constants for specifying parameter values.

Table 160-1 DBMS_REDEFINITION Constants

Constant	Type	Value	Description
`CONS_CONSTRAINT`	`PLS_INTEGER`	`3`	Used to specify that dependent object type is a constraint
`CONS_INDEX`	`PLS_INTEGER`	`2`	Used to specify that dependent object type is a index
`CONS_MVLOG`	`PLS_INTEGER`	`10`	Used to (un)register a materialized view log, as a dependent object of the table, through the REGISTER_DEPENDENT_OBJECT Procedureand the UNREGISTER_DEPENDENT_OBJECT Procedure.
`CONS_ORIG_PARAMS`	`PLS_INTEGER`	`1`	Used to specify that indexes should be cloned with their original storage parameters
`CONS_TRIGGER`	`PLS_INTEGER`	`4`	Used to specify that dependent object type is a trigger
`CONS_USE_PK`	`BINARY_INTEGER`	`1`	Used to indicate that the redefinition should be done using primary keys or pseudo-primary keys (unique keys with all component columns having not-`NULL` constraints)
`CONS_USE_ROWID`	`BINARY_INTEGER`	`2`	Used to indicate that the redefinition should be done using rowids
`CONS_VPD_AUTO`	`BINARY_INTEGER`	`2`	Used to indicate to copy VPD policies automatically
`CONS_VPD_MANUAL`	`BINARY_INTEGER`	`4`	Used to indicate to copy VPD policies manually
`CONS_VPD_NONE`	`BINARY_INTEGER`	1	Used to indicate that there are no VPD policies on the original table

DBMS_REDEFINITION Operational Notes

The following operational notes apply to DBMS_REDEFINITION.

- `CONS_USE_PK` and `CONS_USE_ROWID` are constants used as input to the `OPTIONS_FLAG` parameter in both the START_REDEF_TABLE Procedure and CAN_REDEF_TABLE Procedure. `CONS_USE_ROWID` is used to indicate that the redefinition should be done using rowids while `CONS_USE_PK` implies that the redefinition should be done using primary keys or pseudo-primary keys (which are unique keys with all component columns having `NOT NULL` constraints).
- `CONS_INDEX`, `CONS_MVLOG`,`CONS_TRIGGER` and `CONS_CONSTRAINT` are used to specify the type of the dependent object being (un)registered in REGISTER_DEPENDENT_OBJECT Procedure and UNREGISTER_DEPENDENT_OBJECT Procedure (parameter `DEP_TYPE`).

  `CONS_INDEX` ==> dependent object is of type `INDEX`

  `CONS_TRIGGER` ==> dependent object is of type `TRIGGER`

  `CONS_CONSTRAINT` ==> dependent object type is of type `CONSTRAINT`

  `CONS_MVLOG` ==> dependent object is of type `MATERIALIZED VIEW LOG`

- CONS_ORIG_PARAMS as used as input to the COPY_INDEXES parameter in COPY_TABLE_DEPENDENTS Procedure. Using this parameter implies that the indexes on the original table be copied onto the interim table using the same storage parameters as that of the original index.
- After a table redefinition is complete, the interim table will have the object ID of the original table. That is, the object IDs are essentially swapped during the redefinition operation. If there are any audit policies on the original table, they will now audit the interim table instead. This happens because audit policies are created on object IDs, not object names. Check any audit policies on tables involved in a redefinition operation and alter them as needed to audit the desired table.

DBMS_REDEFINITION Rules and Limits

Various rules and limits apply to implementation of the DBMS_REDEFINITION package.

For more information about these, see the *Oracle Database Administrator's Guide*.

DBMS_REDEFINITION Examples

The following examples demonstrate DBMS_REDEFINITION functionality.

We create two tables EMP and EMP_INT as the original and the interim tables, respectively:

```
CREATE TABLE emp
( empno     NUMBER(4,0) PRIMARY KEY,
  ename     VARCHAR2(10),
  job       VARCHAR2(9),
  mgr       NUMBER(4,0),
  hiredate  DATE,
  sal       NUMBER(7,2),
  comm      NUMBER(7,2),
  deptno    NUMBER(2,0)
)
TABLESPACE myts;

CREATE TABLE emp_int
( empno     NUMBER(4,0) PRIMARY KEY,
  ename     VARCHAR2(10),
  job       VARCHAR2(9),
  mgr       NUMBER(4,0),
  hiredate  DATE,
  sal       NUMBER(7,2),
  comm      NUMBER(7,2),
  deptno    NUMBER(2,0)
)
TABLESPACE compressed_ts;
```

Regular Multi-Step Redefinition

```
DBMS_REDEFINITION.START_REDEF_TABLE('SCOTT', 'EMP', 'EMP_INT',
ENABLE_ROLLBACK => TRUE);
DBMS_REDEFINITION.FINISH_REDEF_TABLE('SCOTT', 'EMP', 'EMP_INT');
```

Assume theDBA wants to evaluate the performance of the application for 2 days, after moving the table `EMP` from tablespace `myts` to `compressed_ts`. One can run sync_interim_table SYNC_INTERIM_TABLE Procedureto keep both the tables in sync (say, every hour).

```
DBMS_REDEFINITION.SYNC_INTERIM_TABLE('SCOTT', 'EMP', 'EMP_INT');
```

Case 1 — DBA is not happy with the performance, so decides to rollback.

```
DBMS_REDEFINITION.ROLLBACK('SCOTT', 'EMP', 'EMP_INT');
```

Case 2 — DBA is happy with the performance, so decides not to rollback.

```
DBMS_REDEFINITION.ABORT_ROLLBACK('SCOTT', 'EMP', 'EMP_INT');
```

This terminates the possibility of rollback.

Single-Step Redefinition

```
DBMS_REDEFINITION.REDEF_TABLE('SCOTT','EMP','ROW STORE COMPRESS ADVANCED',
enable_rollback => TRUE);
```

Note:

Online table redefinition rollback is not supported when the `REDEF_TABLE` procedure is used to redefine a table.

Summary of DBMS_REDEFINITION Subprograms

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

Table 160-2 DBMS_REDEFINITION Package Subprograms

Subprogram	Description
ABORT_REDEF_TABLE Procedure	Cleans up errors that occur during the redefinition process and removes all temporary objects created by the reorganization process
ABORT_ROLLBACK Procedure	Aborts rollback
ABORT_UPDATE Procedure	Aborts an update started with the `EXECUTE_UPDATE` procedure
CAN_REDEF_TABLE Procedure	Determines if a given table can be redefined online
COPY_TABLE_DEPENDENTS Procedure	Copies the dependent objects of the original table onto the interim table
EXECUTE_UPDATE Procedure	Optimizes the performance of bulk updates to a table
FINISH_REDEF_TABLE Procedure	Completes the redefinition process
REDEF_TABLE Procedure	Provides a single push-button interface that integrates several redefinition steps

Table 160-2 (Cont.) DBMS_REDEFINITION Package Subprograms

Subprogram	Description
REGISTER_DEPENDENT_OBJECT Procedure	Registers a dependent object (index, trigger, constraint or materialized view log) on the table being redefined and the corresponding dependent object on the interim table
ROLLBACK Procedure	Performs rollback
SET_PARAM Procedure	Sets a new value for a specified parameter used by the redefinition process identified by a redefinition ID
START_REDEF_TABLE Procedure	Initiates the redefinition process
SYNC_INTERIM_TABLE Procedure	Keeps the interim table synchronized with the original table
UNREGISTER_DEPENDENT_OBJECT Procedure	Unregisters a dependent object (index, trigger, constraint or materialized view log) on the table being redefined and the corresponding dependent object on the interim table

ABORT_REDEF_TABLE Procedure

This procedure cleans up errors that occur during the redefinition process.

This procedure can also be used to terminate the redefinition process any time after the START_REDEF_TABLE Procedure has been called and before the FINISH_REDEF_TABLE Procedure is called. This process will remove the temporary objects that are created by the redefinition process such as materialized view logs.

Syntax

```
DBMS_REDEFINITION.ABORT_REDEF_TABLE (
   uname                    IN  VARCHAR2,
   orig_table               IN  VARCHAR2,
   int_table                IN  VARCHAR2,
   part_name                IN  VARCHAR2 := NULL);
```

Parameters

Table 160-3 ABORT_REDEF_TABLE Procedure Parameters

Parameter	Description
`uname`	Schema name of the tables
`orig_table`	Name of the table to be redefined
`int_table`	Name of the interim table. Can take a comma-delimited list of interim table names.
`part_name`	Name of the partition being redefined. If redefining only a single partition of a table, specify the partition name in this parameter. `NULL` implies the entire table is being redefined. Can take a comma-delimited list of partition names to be redefined.

ABORT_ROLLBACK Procedure

This procedure aborts rollback for a table that was redefined.

When online redefinition of a table is started with the `START_REDEF_TABLE` procedure, rollback can be enabled for the changes performed by online redefinition of a table by setting the `enable_rollback` parameter to `TRUE`. If you want to retain the changes made by online redefinition, you can abort the rollback to clean up the database objects that enable rollback.

Syntax

```
DBMS_REDEFINITION.ABORT_ROLLBACK (
   uname            IN  VARCHAR2,
   orig_table       IN  VARCHAR2,
   int_table        IN  VARCHAR2 := NULL,
   part_name        IN  VARCHAR2 := NULL);
```

Parameters

Table 160-4 ABORT_ROLLBACK Procedure Parameters

Parameter	Description
`uname`	Schema name of the tables
`orig_table`	Name of the table to be redefined
`int_table`	Name of the interim table
`part_name`	Name of the partition being redefined

ABORT_UPDATE Procedure

This procedure can aborts an update started with the `EXECUTE_UPDATE` procedure in the `RDBMS_REDEFINITION` package.

Syntax

```
DBMS_REDEFINITION.ABORT_UPDATE (
   update_stmt   IN  CLOB);
```

Parameters

Table 160-5 ABORT_UPDATE Procedure Parameters

Parameter	Description
`update_stmt`	The SQL `UPDATE` statement to be aborted The SQL statement must exactly match the SQL statement in the `EXECUTE_UPDATE` procedure.

See Also:

Oracle Database Administrator's Guide

CAN_REDEF_TABLE Procedure

This procedure determines if a given table can be redefined online. This is the first step of the online redefinition process. If the table is not a candidate for online redefinition, an error message is raised.

Syntax

```
DBMS_REDEFINITION.CAN_REDEF_TABLE (
   uname          IN  VARCHAR2,
   tname          IN  VARCHAR2,
   options_flag   IN  PLS_INTEGER := 1,
   part_name      IN  VARCHAR2 := NULL);
```

Parameters

Table 160-6 CAN_REDEF_TABLE Procedure Parameters

Parameter	Description
`uname`	Schema name of the table
`tname`	Name of the table to be re-organized
`options_flag`	Indicates the type of redefinition method to use. • If `dbms_redefinition.cons_use_pk`, the redefinition is done using primary keys or pseudo-primary keys (unique keys with all component columns having `NOT NULL` constraints). The default method of redefinition is using primary keys. • If `dbms_redefinition.cons_use_rowid`, the redefinition is done using rowids.
`part_name`	Name of the partition being redefined. If redefining only a single partition of a table, specify the partition name in this parameter. `NULL` implies the entire table is being redefined.

Exceptions

If the table is not a candidate for online redefinition, an error message is raised.

COPY_TABLE_DEPENDENTS Procedure

This procedure clones the dependent objects of the table being redefined onto the interim table and registers the dependent objects. This procedure does not clone the already registered dependent objects.

This subprogram is used to clone the dependent objects like grants, triggers, constraints and privileges from the table being redefined to the interim table (which represents the post-redefinition table).

Syntax

```
DBMS_REDEFINITION.COPY_TABLE_DEPENDENTS(
   uname                    IN  VARCHAR2,
   orig_table               IN  VARCHAR2,
   int_table                IN  VARCHAR2,
   copy_indexes             IN  PLS_INTEGER := 1,
   copy_triggers            IN  BOOLEAN     := TRUE,
   copy_constraints         IN  BOOLEAN     := TRUE,
   copy_privileges          IN  BOOLEAN     := TRUE,
   ignore_errors            IN  BOOLEAN     := FALSE,
   num_errors               OUT PLS_INTEGER,
   copy_statistics          IN  BOOLEAN     := FALSE,
   copy_mvlog               IN  BOOLEAN     := FALSE);
```

Parameters

Table 160-7 COPY_TABLE_DEPENDENTS Procedure Parameters

Parameter	Description
`uname`	Schema name of the tables
`orig_table`	Name of the table being redefined
`int_table`	Name of the interim table
`copy_indexes`	Flag indicating whether to copy the indexes • `0` - do not copy any index • `dbms_redefinition.cons_orig_params` – copy the indexes using the physical parameters of the source indexes
`copy_triggers`	`TRUE` = clone triggers, `FALSE` = do nothing
`copy_constraints`	`TRUE` = clone constraints, `FALSE` = do nothing. If compatibility setting is 10.2 or higher, then clone `CHECK` and `NOT NULL` constraints
`copy_privileges`	`TRUE` = clone privileges, `FALSE` = do nothing
`ignore_errors`	`TRUE` = if an error occurs while cloning a particular dependent object, then skip that object and continue cloning other dependent objects. `FALSE` = that the cloning process should stop upon encountering an error.
`num_errors`	Number of errors that occurred while cloning dependent objects
`copy_statistics`	`TRUE` = copy statistics, `FALSE` = do nothing
`copy_mvlog`	`TRUE` = copy materialized view log, `FALSE` = do nothing

Usage Notes

- The user must check the column `num_errors` before proceeding to ensure that no errors occurred during the cloning of the objects.
- In case of an error, the user should fix the cause of the error and call the COPY_TABLE_DEPENDENTS Procedure again to clone the dependent object. Alternatively the user can manually clone the dependent object and then register the manually cloned dependent object using the REGISTER_DEPENDENT_OBJECT Procedure.
- All cloned referential constraints involving the interim tables will be created disabled (they will be automatically enabled after the redefinition) and all triggers on interim tables will not fire till the redefinition is completed. After the redefinition is complete, the cloned objects will be renamed to the corresponding pre-redefinition names of the objects (from which they were cloned from).
- It is the user's responsibility that the cloned dependent objects are unaffected by the redefinition. All the triggers will be cloned and it is the user's responsibility that the cloned triggers are unaffected by the redefinition.

EXECUTE_UPDATE Procedure

This procedure can optimize the performance of bulk updates to a table. Performance is optimized because the updates are not logged in the redo log.

The `EXECUTE_UPDATE` procedure automatically uses the components of online table redefinition, such an interim table, a materialized view, and a materialized view log, to enable optimized bulk updates to a table. The `EXECUTE_UPDATE` procedure also removes fragmentation of the affected rows and ensures that the update is atomic. If the bulk updates raise any errors, then you can use the `ABORT_UPDATE` procedure to undo the changes made by the `EXECUTE_UPDATE` procedure.

Syntax

```
DBMS_REDEFINITION.EXECUTE_UPDATE (
   update_stmt   IN   CLOB);
```

Parameters

Table 160-8   EXECUTE_UPDATE Procedure Parameters

Parameter	Description
`update_stmt`	The SQL `UPDATE` statement

See Also:

Oracle Database Administrator's Guide

FINISH_REDEF_TABLE Procedure

This procedure completes the redefinition process.

Before this step, you can create new indexes, triggers, grants, and constraints on the interim table. The referential constraints involving the interim table must be disabled. After completing this step, the original table is redefined with the attributes and data of the interim table. The original table is locked briefly during this procedure.

Syntax

```
DBMS_REDEFINITION.FINISH_REDEF_TABLE (
   uname                    IN   VARCHAR2,
   orig_table               IN   VARCHAR2,
   int_table                IN   VARCHAR2,
   part_name                IN   VARCHAR2 := NULL,
   dml_lock_timeout         IN   PLS_INTEGER := NULL,
   continue_after_errors    IN   BOOLEAN := FALSE,
   disable_rollback         IN   PLS_INTEGER := FALSE);
```

Parameters

Table 160-9 FINISH_REDEF_TABLE Procedure Parameters

Parameters	Description
`uname`	Schema name of the tables
`orig_table`	Name of the table to be redefined
`int_table`	Name of the interim table. Can take a comma-delimited list of interim table names.
`part_name`	Name of the partition being redefined. If redefining only a single partition of a table, specify the partition name in this parameter. `NULL` implies the entire table is being redefined. Can take a comma-delimited list of partition names to be redefined.
`dml_lock_timeout`	Specifies the number of seconds the procedure waits for its required locks before failing. The permissible range of values for timeout is 0 to 1,000,000. The default is `NULL` (wait mode).
`continue_after_errors`	When redefining multiple partitions allows operation execution to continue on the next partition (applies only to batched partition redefinition).
`disable_rollback`	When set to `TRUE`, disables the rollback option if it was enabled in the `START_REDEF_TABLE` procedure. Specifying `TRUE` cleans up the database objects that enable rollback.

Examples

Wait up to 600 seconds for required locks on `SH.SALES`:

```
EXECUTE DBMS_REDEFINITION.FINISH_REDEF_TABLE (
  'SH', 'SALES', 'INT_SALES', 600);
```

REDEF_TABLE Procedure

This procedure provides a single interface that integrates several redefinition steps including the CAN_REDEF_TABLE Procedure, the START_REDEF_TABLE Procedure, the COPY_TABLE_DEPENDENTS Procedure and the FINISH_REDEF_TABLE Procedure.

This procedure can change data storage properties including tablespaces (for table, partition, subpartition, index, LOB column), compress type (for table, partition, subpartition, index, LOB column) and `STORE_AS` clause for the LOB column.

Syntax

```
DBMS_REDEFINITION.REDEF_TABLE (
   uname                        IN  VARCHAR2,
   tname                        IN  VARCHAR2,
   table_compression_type       IN  VARCHAR2 := NULL,
   table_part_tablespace        IN  VARCHAR2 := NULL,
   index_key_compression_type   IN  VARCHAR2 := NULL,
   index_tablespace             IN  VARCHAR2 := NULL,
   lob_compression_type         IN  VARCHAR2 := NULL,
   lob_tablespace               IN  VARCHAR2 := NULL,
   lob_store_as                 IN  VARCHAR2 := NULL,
   refresh_dep_mviews           IN  VARCHAR2 := 'N',
   dml_lock_timeout             IN  PLS_INTEGER := NULL);
```

Parameters

Table 160-10 REDEF_TABLE Procedure Parameters

Parameter	Description
`uname`	Schema name of the table
`tname`	Name of the table to be redefined
`table_compression_type`	Text string of the table compression clause. `NULL` means there is no change.
`table_part_tablespace`	Tablespace name for the entire table or partitions. `NULL` means there is no change.
`index_key_compression_type`	Text string of the compression clause for all indexes on the table. `NULL` means there is no change.
`index_tablespace`	Tablespace name for all indexes on the table. `NULL` means there is no change.
`lob_compression_type`	Text string of the compression clause for all LOBs in the entire table. `NULL` means there is no change.
`lob_tablespace`	Tablespace name for all LOBs in the table. `NULL` means there is no change.
`lob_store_as`	Specifies LOB store as `'SECUREFILE'` or `'BASICFILE'`. `NULL` means there is no change.
`refresh_dep_mviews`	When set to `'Y'`, fast refresh of dependent materialized views is performed once at the end of the redefinition operation.
`dml_lock_timeout`	Specifies the number of seconds the procedure waits for its required locks before failing. The permissible range of values for timeout is 0 to 1,000,000. The default is `NULL` (wait mode).

Examples

```
BEGIN
   DBMS_REDEFINITION.REDEF_TABLE(
     uname                      => 'TABOWNER2',
     tname                      => 'EMP2',
     table_compression_type     => 'ROW STORE COMPRESS ADVANCED',
     table_part_tablespace      => 'NEWTBS',
     index_key_compression_type => 'COMPRESS 1',
     index_tablespace           => 'NEWIDXTBS',
     lob_compression_type       => 'COMPRESS HIGH',
     lob_tablespace             => 'SLOBTBS',
     lob_store_as               => 'SECUREFILE');
END;
```

Related Topics

- CAN_REDEF_TABLE Procedure
  This procedure determines if a given table can be redefined online. This is the first step of the online redefinition process. If the table is not a candidate for online redefinition, an error message is raised.
- START_REDEF_TABLE Procedure
  This procedure starts a table redefinition.

- COPY_TABLE_DEPENDENTS Procedure
  This procedure clones the dependent objects of the table being redefined onto the interim table and registers the dependent objects. This procedure does not clone the already registered dependent objects.
- FINISH_REDEF_TABLE Procedure
  This procedure completes the redefinition process.

See Also:

Oracle Database Administrator's Guide regarding "Performing Online Redefinition with the `REDEF_TABLE` Procedure"

REGISTER_DEPENDENT_OBJECT Procedure

This procedure registers a dependent object (index, trigger, constraint or materialized view log) on the table being redefined and the corresponding dependent object on the interim table.

This can be used to have the same object on each table but with different attributes. For example: for an index, the storage and tablespace attributes could be different but the columns indexed remain the same

Syntax

```
DBMS_REDEFINITION.REGISTER_DEPENDENT_OBJECT(
   uname                   IN    VARCHAR2,
   orig_table              IN    VARCHAR2,
   int_table               IN    VARCHAR2,
   dep_type                IN    PLS_INTEGER,
   dep_owner               IN    VARCHAR2,
   dep_orig_name           IN    VARCHAR2,
   dep_int_name            IN    VARCHAR2);
```

Parameters

Table 160-11 REGISTER_DEPENDENT_OBJECT Procedure Parameters

Parameters	Description
`uname`	Schema name of the tables
`orig_table`	Name of the table to be redefined
`int_table`	Name of the interim table
`dep_type`	Type of the dependent object (see Constants and Operational Notes)
`dep_owner`	Owner of the dependent object
`dep_orig_name`	Name of the original dependent object
`dep_int_name`	Name of the interim dependent object

Usage Notes

- Attempting to register an already registered object will raise an error.
- Registering a dependent object will automatically remove that object from `DBA_REDEFINITION_ERRORS` if an entry exists for that object.

ROLLBACK Procedure

This procedure rolls back changes to a table after online table redefinition to return the table to its original definition and preserve DML changes made to the table.

Syntax

```
DBMS_REDEFINITION.ROLLBACK (
   uname                     IN  VARCHAR2,
   orig_table                IN  VARCHAR2,
   int_table                 IN  VARCHAR2 := NULL,
   part_name                 IN  VARCHAR2 := NULL,
   dml_lock_timeout          IN  PLS_INTEGER := NULL,
   continue_after_errors     IN  BOOLEAN := FALSE);
```

Parameters

Table 160-12 ROLLBACK Procedure Parameters

Parameter	Description
`uname`	Schema name of the table to be redefined
`orig_table`	Name of the table to be redefined
`int_table`	Name of the interim table.
`part_name`	Name of the partition being redefined.
`dml_lock_timeout`	Specifies the number of seconds the procedure waits for its required locks before failing. The permissible range of values for timeout is 0 to 1,000,000. The default is `NULL` (wait mode).
`continue_after_errors`	When rolling back redefinition changes on multiple partitions, allows operation execution to continue on the next partition (applies only to batched partition redefinition).

SET_PARAM Procedure

This procedure sets a new value for a specified parameter used by the redefinition process identified by a redefinition ID.

Note:

Currently, the only value that can be changed by this procedure is the value for the of the `refresh_dep_mviews` parameter that is specified in the `REDEF_TABLE` procedure or the `START_REDEF_TABLE` procedure. You can determine the redefinition ID and check the value of the `refresh_dep_mviews` parameter for an online table redefinition operation by querying the `DBA_REDEFINITION_STATUS` view.

Syntax

```
DBMS_REDEFINITION.SET_PARAM (
   redefinition_id  IN  VARCHAR2,
   param_name       IN  VARCHAR2,
   param_value      IN  VARCHAR2);
```

Parameters

Table 160-13 SET_PARAM Procedure Parameters

Parameter	Description
`redefinition_id`	The redefinition ID that identifies the redefinition process
`param_name`	The parameter name
`param_value`	The new parameter value

See Also:

Oracle Database Administrator's Guide

START_REDEF_TABLE Procedure

This procedure starts a table redefinition.

Prior to calling this procedure, you must manually create an empty interim table (in the same schema as the table to be redefined) with the desired attributes of the post-redefinition table, and then call this procedure to initiate the redefinition.

Syntax

```
DBMS_REDEFINITION.START_REDEF_TABLE (
   uname                    IN  VARCHAR2,
   orig_table               IN  VARCHAR2,
   int_table                IN  VARCHAR2,
   col_mapping              IN  VARCHAR2 := NULL,
   options_flag             IN  BINARY_INTEGER := 1,
   orderby_cols             IN  VARCHAR2 := NULL,
   part_name                IN  VARCHAR2 := NULL,
   continue_after_errors    IN  BOOLEAN := FALSE,
   copy_vpd_opt             IN  BINARY_INTEGER := CONS_VPD_NONE,
   refresh_dep_mviews       IN  VARCHAR2 := 'N',
   enable_rollback          IN  BOOLEAN := FALSE);
```

Parameters

Table 160-14 START_REDEF_TABLE Procedure Parameters

Parameter	Description
`uname`	Schema name of the tables
`orig_table`	Name of the table to be redefined
`int_table`	Name of the interim table. Can take a comma-delimited list of interim table names.
`col_mapping`	Mapping information from the columns in the original table to the columns in the interim table. (This is similar to the column list on the `SELECT` clause of a query.) If `NULL,` all the columns in the original table are selected and have the same name after redefinition.

Table 160-14 (Cont.) START_REDEF_TABLE Procedure Parameters

Parameter	Description
`options_flag`	Indicates the type of redefinition method to use: • If `dbms_redefinition.cons_use_pk`, the redefinition is done using primary keys or pseudo-primary keys (unique keys with all component columns having `NOT NULL` constraints). The default method of redefinition is using primary keys. • If `dbms_redefinition.cons_use_rowid`, the redefinition is done using rowids.
`orderby_cols`	This optional parameter accepts the list of columns (along with the optional keyword(s) ascending/descending) with which to order by the rows during the initial instantiation of the interim table (the order by is only done for the initial instantiation and not for subsequent synchronizations)
`part_name`	Name of the partition being redefined. If redefining only a single partition of a table, specify the partition name in this parameter. `NULL` implies the entire table is being redefined. Can take a comma-delimited list of partition names to be redefined.
`continue_after_errors`	When redefining multiple partitions allows operation execution to continue on the next partition (applies only to batched partition redefinition)
`copy_vpd_opt`	Specifies how VPD policies are handled in online redefinition
`refresh_dep_mviews`	When set to `'Y'`, fast refresh of dependent materialized views is performed when the `START_REDEF_TABLE` procedure is run, each time the `SYNC_INTERIM_TABLE` procedure is run, and when the `FINISH_REDEF_TABLE` procedure is run.
`enable_rollback`	When set to `TRUE`, enables the rollback option. When this parameter is set to true, Oracle Database maintains the interim table created during redefinition after redefinition is complete. You can run the `SYNC_INTERIM_TABLE` procedure to synchronize the interim table periodically to apply DML changes made to the redefined table to the interim table. An internal materialized view and materialized view log enables maintenance of the interim table. If you decide to roll back the online table redefinition with the `ROLLBACK` procedure, then the interim table is synchronized, and Oracle Database switches back to it so that the table has its original definition.

Examples

Start redefinition of three partitions (`sal03q1,sal03q2,sal03q3`) in table `'STEVE.salestable'` using three interim tables of `int_salestable1`, `int_salestable2` and `int_salestable3`, respectively. The operation will continue on `sal03q3` even if it fails on `sal03q1`.

```
DBMS_REDEFINITION.START_REDEF_TABLE(
   uname                 => 'STEVE',
   orig_table            => 'salestable',
   int_table             => 'int_salestable1, int_salestable2, int_salestable3',
   col_mapping           => NULL,
   options_flag          => DBMS_REDEFINITION.CONS_USE_ROWID,
   part_name             => 'sal03q1,sal03q2,sal03q3',
   continue_after_errors => TRUE);
```

Specify to copy VPD policies automatically:

```
EXECUTE DBMS_REDEFINITION.START_REDEF_TABLE (
  uname                     => 'SCOTT',
  orig_table                => 'T',
  int_table                 => 'INT_T',
  copy_vpd_opt              => DBMS_REDEFINITION.CONS_VPD_AUTO);
```

SYNC_INTERIM_TABLE Procedure

This procedure keeps the interim table synchronized with the original table.

Syntax

```
DBMS_REDEFINITION.SYNC_INTERIM_TABLE (
   uname                    IN  VARCHAR2,
   orig_table               IN  VARCHAR2,
   int_table                IN  VARCHAR2,
   part_name                IN  VARCHAR2 := NULL,
   continue_after_errors    IN  BOOLEAN := FALSE);
```

Parameters

Table 160-15 SYNC_INTERIM_TABLE Procedure Parameters

Parameter	Description
`uname`	Schema name of the table
`orig_table`	Name of the table to be redefined
`int_table`	Name of the interim table. Can take a comma-delimited list of interim table names.
`part_name`	Name of the partition being redefined. If redefining only a single partition of a table, specify the partition name in this parameter. `NULL` implies the entire table is being redefined. Can take a comma-delimited list of partition names to be redefined.
`continue_after_errors`	When redefining multiple partitions allows operation execution to continue on the next partition (applies only to batched partition redefinition)

Usage Notes

- This step is useful in minimizing the amount of synchronization needed to be done by the FINISH_REDEF_TABLE Procedure before completing the online redefinition.
- This procedure can be called between long running operations (such as `CREATE INDEX`) on the interim table to sync it up with the data in the original table and speed up subsequent operations.

UNREGISTER_DEPENDENT_OBJECT Procedure

This procedure unregisters a dependent object (index, trigger, constraint or materialized view log) on the table being redefined and the corresponding dependent object on the interim table.

Syntax

```
DBMS_REDEFINITION.UNREGISTER_DEPENDENT_OBJECT(
   uname              IN VARCHAR2,
   orig_table         IN VARCHAR2,
   int_table          IN VARCHAR2,
   dep_type          IN PLS_INTEGER,
   dep_owner         IN VARCHAR2,
   dep_orig_name     IN VARCHAR2,
   dep_int_name      IN VARCHAR2);
```

Parameters

Table 160-16 UNREGISTER_DEPENDENT_OBJECT Procedure Parameters

Parameters	Description
`uname`	Schema name of the tables
`orig_table`	Name of the table to be redefined
`int_table`	Name of the interim table
`dep_type`	Type of the dependent object
`dep_owner`	Owner of the dependent object
`dep_orig_name`	Name of the original dependent object
`dep_int_name`	Name of the interim dependent object