How To Partition Existing Table Using DBMS_Redefinition (Doc ID 472449.1)

Modified: 01-Mar-2013 Type: HOWTO

In this Document

This document is being delivered to you via Oracle Support's Rapid Visibility (RaV) process and therefore has not been subject to an independent technical review.

APPLIES TO:

Oracle Server - Enterprise Edition - Version 9.2.0.4 and later
Information in this document applies to any platform.
"Checked for relevance on 29-Sep-2010"

GOAL

The purpose of this document is to provide step by step instructions on how to convert unpartitioned table to partitioned one using dbms_redefinition package.

FIX

1) Create unpartitioned table with the name unpar_table

```
SQL> CREATE TABLE unpar_table (
id NUMBER(10),
create_date DATE,
name VARCHAR2(100)
);
```

2) Apply some constraints to the table:

```
SQL> ALTER TABLE unpar_table ADD (
CONSTRAINT unpar_table_pk PRIMARY KEY (id)
);

SQL> CREATE INDEX create_date_ind ON unpar_table(create_date);
```

3) Gather statistics on the table:

```
SQL> EXEC DBMS_STATS.gather_table_stats(USER, 'unpar_table', cascade => TRUE);
```

4) Create a Partitioned Interim Table:

```
SQL> CREATE TABLE par_table (
id NUMBER(10),
create_date DATE,
name VARCHAR2(100)
```

```
)
PARTITION BY RANGE (create_date)
(PARTITION unpar_table_2005 VALUES LESS THAN (TO_DATE('01/01/2005', 'DD/MM/YYYY')),
PARTITION unpar_table_2006 VALUES LESS THAN (TO_DATE('01/01/2006', 'DD/MM/YYYY')),
PARTITION unpar_table_2007 VALUES LESS THAN (MAXVALUE));
```

5) Start the Redefinition Process:

a) Check the redefinition is possible using the following command:

```
SQL> EXEC Dbms_Redefinition.can_redef_table(USER, 'unpar_table');
```

b)If no errors are reported, start the redefintion using the following command:

```
SQL> BEGIN
DBMS_REDEFINITION.start_redef_table(
uname => USER,
orig_table => 'unpar_table',
int_table => 'par_table');
END;
/
```

Note: This operation can take quite some time to complete.

c) Optionally synchronize new table with interim name before index creation:

```
SQL> BEGIN
dbms_redefinition.sync_interim_table(
uname => USER,
orig_table => 'unpar_table',
int_table => 'par_table');
END;
/
```

d) Create Constraints and Indexes:

```
SQL> ALTER TABLE par_table ADD (
CONSTRAINT unpar_table_pk2 PRIMARY KEY (id)
);

SQL> CREATE INDEX create_date_ind2 ON par_table(create_date);
```

e) Gather statistics on the new table:

```
SQL> EXEC DBMS_STATS.gather_table_stats(USER, 'par_table', cascade => TRUE);
```

f) Complete the Redefintion Process:

```
SQL> BEGIN
dbms_redefinition.finish_redef_table(
uname => USER,
orig_table => 'unpar_table',
int_table => 'par_table');
END;
/
```

At this point the interim table has become the "real" table and their names have been switched in the name dictionary.

g) Remove original table which now has the name of the interim table:

```
SQL> DROP TABLE par_table;
```

h)Rename all the constraints and indexes to match the original names.

```
ALTER TABLE unpar_table RENAME CONSTRAINT unpar_table_pk2 TO unpar_table_pk;
ALTER INDEX create_date_ind2 RENAME TO create_date_ind;
```

i) Check whether partitioning is successful or not:

```
SQL> SELECT partitioned
FROM user_tables
WHERE table_name = 'unpar_table';

PAR
---
YES

1 row selected.

SQL> SELECT partition_name
FROM user_tab_partitions
WHERE table_name = 'unpar_table';

PARTITION_NAME
------------------------------
unpar_table_2005
unpar_table_2006
unpar_table_2007

3 rows selected.
```

Please not that the 9i redefinition procedures has some restrictions:

* There must be enough space to hold two copies of the table.
* Primary key columns cannot be modified.
* Tables must have primary keys.
* Redefinition must be done within the same schema.
* New columns added cannot be made NOT NULL until after the redefinition operation.
* Tables cannot contain LONGs, BFILEs or User Defined Types.
* Clustered tables cannot be redefined.
* Tables in the SYS or SYSTEM schema cannot be redefined.
* Tables with materialized view logs or materialized views defined on them cannot be redefined.
* Horizontal sub setting of data cannot be performed during the redefinition.

Restrictions for Online Redefinition of Tables in 11.2

- If the table is to be redefined using primary key or pseudo-primary keys (unique keys or constraints with all component columns having not null constraints), then the post-redefinition table must have the same primary key or pseudo-primary key columns. If the table is to be redefined using rowids, then the table must not be an index-organized table.
- After redefining a table that has a materialized view log, the subsequent refresh of any dependent materialized view must be a complete refresh.
- Tables that are replicated in an n-way master configuration can be redefined, but horizontal subsetting (subset of rows in the table), vertical subsetting (subset of columns in the table), and column transformations are not allowed.
- The overflow table of an index-organized table cannot be redefined online independently.
- Tables with fine-grained access control (row-level security) cannot be redefined online.
- Tables for which Flashback Data Archive is enabled cannot be redefined online. You cannot enable Flashback

Data Archive for the interim table.

- Tables with `BFILE` columns cannot be redefined online.
- Tables with `LONG` columns can be redefined online, but those columns must be converted to `CLOBS`. Also, `LONG RAW` columns must be converted to`BLOBS`. Tables with `LOB` columns are acceptable.
- On a system with sufficient resources for parallel execution, and in the case where the interim table is not partitioned, redefinition of a `LONG` column to a `LOB` column can be executed in parallel, provided that:
    - The segment used to store the `LOB` column in the interim table belongs to a locally managed tablespace with Automatic Segment Space Management (ASSM) enabled.
    - There is a simple mapping from one `LONG` column to one `LOB` column, and the interim table has only one`LOB`column.

  In the case where the interim table is partitioned, the normal methods for parallel execution for partitioning apply.
- Tables in the `SYS` and `SYSTEM` schema cannot be redefined online.
- Temporary tables cannot be redefined.
- A subset of rows in the table cannot be redefined.
- Only simple deterministic expressions, sequences, and `SYSDATE`can be used when mapping the columns in the interim table to those of the original table. For example, subqueries are not allowed.
- If new columns are being added as part of the redefinition and there are no column mappings for these columns, then they must not be declared `NOTNULL` until the redefinition is complete.
- There cannot be any referential constraints between the table being redefined and the interim table.
- Table redefinition cannot be done `NOLOGGING`.
- For materialized view logs and queue tables, online redefinition is restricted to changes in physical properties. No horizontal or vertical subsetting is permitted, nor are any column transformations. The only valid value for the column mapping string is`NULL`.
- You can convert a `VARRAY` to a nested table with the `CAST` operator in the column mapping. However, you cannot convert a nested table to a `VARRAY`.

REFERENCES

NOTE:1481558.1 - DBMS_REDEFINITION: Case Study for a Large Non-Partition Table to a Partition Table with Online Transactions occuring