DBMS_REDEFINITION: Case Study for a Large Non-Partition Table to a Partition Table with Online Transactions occuring (Doc ID 1481558.1)

Modified: 02-Sep-2012 Type: BULLETIN

In this Document

APPLIES TO:

Oracle Server - Enterprise Edition - Version 11.2.0.2.0 and later
Information in this document applies to any platform.

PURPOSE

This Note has been written to provide some volume details and timings for the procedure to convert a normal table to a partition table using the DBMS_REDEFINITION package.

The note will also include details of objects used by the process (Fast Refresh) and how these become populated during the procedure.

The procedure is broken down into the following sections:

1. Create the Partition Table structure required, known as the Interim table.
2. Execute DBMS_REDEFINITION.can_redef_table...
3. Execute DBMS_REDEFINITION.start_redef_table...
4. Execute DBMS_REDEFINITION.sync_interim_table...
5. Execute DBMS_REDEFINITION.finish_redef_table...
6. Drop the interim table.

DETAILS

Worked example under 11.2.0.3

-- Initial setup of table to be partitioned.

```
CREATE TABLE unpar_table (
  a NUMBER, y number,
  name VARCHAR2(100), date_used date);

alter table unpar_table ADD (CONSTRAINT unpar_table_pk PRIMARY KEY (a,y));

--  load table with 1,000,000 rows

begin
        for i in 1 .. 1000
        loop
            for j in 1 .. 1000
            loop
insert into unpar_table values ( i, j, dbms_random.random, sysdate-j );
            end loop;
        end loop;
end;
   /
commit;

PL/SQL procedure successfully completed.
```

```
Elapsed: 00:01:56.90

Commit complete.

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

SELECT   num_rows FROM user_tables WHERE table_name = 'UNPAR_TABLE';
  NUM_ROWS
----------
   1000000

Elapsed: 00:00:00.01

SQL> CREATE TABLE par_table (
    a NUMBER, y number,
    name VARCHAR2(100),date_used DATE)
   PARTITION BY RANGE (date_used)
    (PARTITION unpar_table_12 VALUES LESS THAN (TO_DATE('10/07/2012', 'DD/MM/YYYY')),
     PARTITION unpar_table_15 VALUES LESS THAN (TO_DATE('15/07/2012', 'DD/MM/YYYY')),
     PARTITION unpar_table_MX VALUES LESS THAN (MAXVALUE));

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

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.07

--  This procedure (DBMS_REDEFINITION.start_redef_table) creates a materialized view based
on a CTAS, as we can see below with
--  the PREBUILT container table.

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

PL/SQL procedure successfully completed.

Elapsed: 00:00:06.69

select mview_name,container_name, build_mode from user_mviews;
MVIEW_NAME                     CONTAINER_NAME                 BUILD_MOD
------------------------------ ------------------------------ ---------
PAR_TABLE                      PAR_TABLE                      PREBUILT

Elapsed: 00:00:00.13
```

-- Insert 1000 rows into the master table while the DBMS_REDEF is active.
-- This will use the mview log created by the DBMS_REDEFINITION.start_redef_table.
-- Check the MLOG$_<table name> table to confirm these online updates have been recorded.

```
SQL> begin
        for i in 1001 .. 1010
        loop
            for j in 1001 .. 1100
            loop
insert into unpar_table values ( i, j, dbms_random.random, sysdate-j );
            end loop;
        end loop;
    end;
   /
commit;

Elapsed: 00:00:00.24
```

```
SQL> select count(*) from MLOG$_UNPAR_TABLE;

  COUNT(*)
----------
      1000
```

-- Run the dbms_redefinition.sync_interim_table to populate the new table structure (implements a MVIEW FAST REFRESH)
-- with the online updates. This will purge the mview log of each record applied.
-- This can be run many times and should be, before we do the Finish_REDEF_TABLE.

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

PL/SQL procedure successfully completed.

Elapsed: 00:00:02.01

ALTER TABLE par_table ADD (CONSTRAINT par_table_pk2 PRIMARY KEY (a,y));


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

-- Finish_redef_table swaps the table names so the interim table becomes the original table name.
-- 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.

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

-- Note, both tables will now be synchronised.

```
select count(*) from par_table ;
  COUNT(*)
----------
   1001000

select count(*) from unpar_table ;
  COUNT(*)
----------
   1001000
```

-- Dictionary views to confirm the change of strructure of our original table "UNPAR_TABLE".

```
SELECT partitioned FROM user_tables WHERE table_name = 'UNPAR_TABLE';

PAR
---
YES
```

```
SELECT partition_name, num_rows FROM user_tab_partitions WHERE table_name = 'UNPAR_TABLE';


PARTITION_NAME                   NUM_ROWS
------------------------------ ----------
UNPAR_TABLE_12                     980000
UNPAR_TABLE_15                       5000
UNPAR_TABLE_MX                      16000
```

-- At this point the Interim table can be dropped.

```
drop TABLE par_table  cascade constraints;
```

Note for previous versions.

For tests done under 11.1.0.7 the timings for the sync_interim_table where the interim table is large (1,000,000 in our case) the timings for this are considerably longer. This was not reviewed in depth, but initial investigation looks like the mview refresh for DBMS_REDEFINITION has been updated during the MERGE cycle.

example:

11.1.0.7
SYNC of 1000 rows --   Elapsed: 00:01:36.30   (1.5 minutes)