Online Table Redefinition

Author	**박대성**
Creation Date	**2019-08-12**
Last Updated	
Version	

Version	변경일자	변경자(작성자)	주요내용
1			
2			
3			

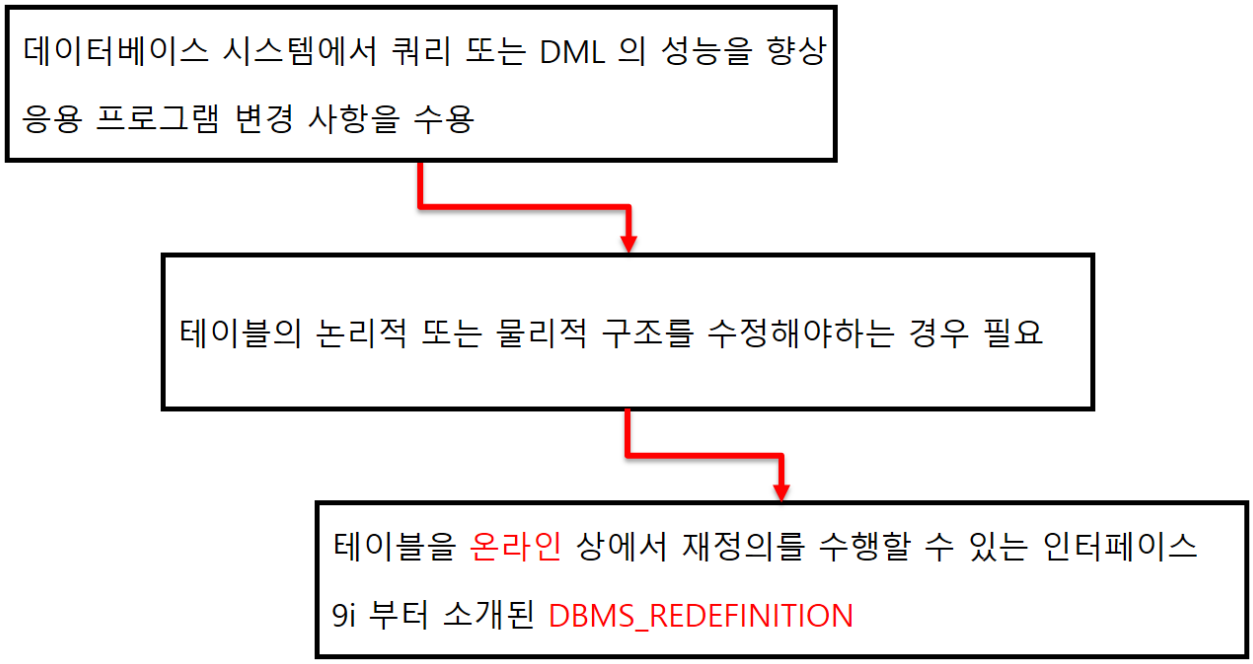
CONTENTS

1. About Redefining Tables Online

모든 데이터베이스 시스템에서 쿼리 또는 DML 의 성능을 향상 시키거나 응용 프로그램 변경 사항을 수용하거나 스토리지 관리를 위해 테이블의 논리적 또는 물리적 구조를 수정해야하는 경우가 있습니다. 테이블을 온라인 상에서 재정의를 수행할 수 있는 인터페이스를 제공해주는 9i 부터 소개된 DBMS_REDEFINITION 패키지를 소개합니다.

ORACLE DATABASE 는 테이블의 가용성에 큰 영향을 미치지 않으면서 테이블 구조를 수정하는 메커니즘을 제공합니다. 이 메커니즘을 온라인 테이블 재정의라고 합니다. 온라인으로 테이블을 재정의하면 테이블을 다시 정의하는 기존의 방법과 비교하여 가용성이 크게 향상됩니다.

테이블을 온라인으로 재정의하면 재정의 프로세스가 진행되는 동안 쿼리와 DML 모두에서 테이블에 액세스 할 수 있습니다. 일반적으로 테이블은 테이블의 크기 및 재정의의 복잡성과 무관한 매우 작은 창에서만 배타 모드로 잠겨 있으며 사용자에게는 완전히 투명합니다.



2. Features of Online Table Redefinition

온라인 테이블 재정의를 통해 테이블이 온라인 상태인 동안 여러 가지 다른 방법으로 테이블을 수정할 수 있습니다.

- 테이블이나 클러스터의 **스토리지 옵션**을 변경
- 테이블이나 클러스터를 **다른 테이블스페이스로 MOVE** (In the same schema)
- 테이블의 **컬럼을 추가, 변경, 삭제**
- **파티션 테이블 구조** 변경
- ORACLE STREAMS ADVANCED QUEUING QUEUE 테이블이나 MVIEW 로그의 물리적 속성을 변경
- **PARALLEL QUERIES** 지원
- 단편화를 줄이면서 테이블이나 클러스터를 재생성
- IOT 테이블에서 NORMAL 테이블 (HEAP ORGANIZATION) 구조를 변경 (REVERSE 도 가능)

이 외에도 공식 문서에서 수정할 수 있는 다양한 방법이 더 서술되어 있습니다.

3. Privileges Required

패키지의 서브 프로그램을 실행하려면 DBMS_REDEFINITION 패키지에 대한 실행 권한이 필요합니다. DBMS_REDEFINITION 패키지에 대한 실행 권한은 EXECUTE_CATALOG_ROLE 에 부여됩니다.

< 자기 자신에 대한 테이블의 REDEFINITION >

사용자가 패키지를 사용하여 사용자의 스키마에서 테이블을 REDIFINE 하려면 사용자에게 다음 권한이 부여되어야 합니다.

- **CREATE TABLE**
- **CREATE MATERIALIZED VIEW**

CREATE TRIGGER 권한은 COPY_TABLE_DEPENDENTS 프로시저를 실행하는 데 필요합니다.

< 다른 스키마에 대한 테이블의 REDIFINITION >

사용자가 패키지를 사용하여 다른 스키마의 테이블을 다시 정의하려면 사용자에게 다음 권한이 부여되어야 합니다.

- **CREATE ANY TABLE**
- **ALTER ANY TABLE**
- **DROP ANY TABLE**
- **LOCK ANY TABLE**
- **SELECT ANY TABLE**
- **CREATE ANY TRIGGER**
- **CREATE ANY INDEX**

4. Restrictions

DBMS_REDEFINITION Package 는 Enterprise Edition 만 가능하며 해당 테이블의 및 관련된 세그먼트에 대한 2 배의 사이즈만큼의 공간이 필요합니다.

또한 테이블의 온라인 재정의에는 하기의 몇 가지 제한 사항이 적용됩니다.

1. primary key or pseudo-primary keys (not-null 제약 조건이 가진 모든 구성 요소 열이 있는 고유 키 또는 제약 조건) 를 사용하여 테이블을 재정의하는 경우 재정의 후 테이블은 동일한 primary key or pseudo-primary keys 열을 가져야 하며, ROWID 를 사용하여 테이블을 재정의 할 경우, 테이블은 index-organized table 이 아니어야 합니다.
2. index-organized table 의 overflow 테이블은 온라인으로 독립적으로 재정의 할 수 없습니다.
3. 플래시백 데이터 아카이브가 활성화 된 테이블은 온라인으로 재정의 할 수 없습니다. 임시 테이블에 대해 플래시백 데이터 아카이브를 활성화 할 수 없습니다.
4. LONG 열이 있는 테이블은 온라인으로 다시 정의 할 수 있지만 이러한 열은 CLOBS 로 변환해야합니다. 또한 LONG RAW 열은 BLOB 로 변환되어야합니다. LOB 컬럼이 있는 테이블은 사용할 수 있습니다.
5. SYS & SYSTEM schema 소유의 테이블은 재정의 할 수 없습니다.
6. Temporary tables 은 재정의 할 수 없습니다.
7. A subset of rows in the table 은 재정의 할 수 없습니다.
8. interim 테이블의 열을 original 테이블의 열로 매핑 할 때 간단한 결정적 표현식, 시퀀스 및 SYSDATE 의 행 부분 집합을 사용할 수 있습니다. 예를 들어 subqueries 는 허용되지 않습니다.
9. 재정의의 일부로 새 열이 추가되고 이 열에 대해 열 매핑이 없는 경우 재정의가 완료 될 때까지 NOT NULL 로 선언해서는 안됩니다.
10. 테이블 재정의는 NOLOGGING 을 수행 할 수 없습니다.
11. 하나 이상의 중첩 테이블을 포함하는 파티션에서 온라인 재정의를 수행 할 수 없습니다.
12. 테이블이 reference partitioning 과 관련되어 있으면 별도의 DBMS_REDEFINITION 세션에서 여러 테이블에 대해 온라인 재정의를 동시에 실행할 수 없습니다.
13. Oracle Label Security (OLS)를 사용하는 테이블은 온라인으로 재정의 할 수 없습니다.
14. fine-grained access control 가 있는 테이블은 온라인으로 재정의 할 수 없습니다.
15. Oracle Real Application Security 를 사용하는 테이블은 온라인으로 재정의 할 수 없습니다.

5. REDEF_TABLE SINGLE

REDEF_TABLE 프로시저를 사용하여 다음 PROPERTIES 를 변경하려는 경우 **SINGLE STEP** 으로 테이블의 STORAGE PROPERTIES 를 온라인으로 재정의 할 수 있습니다.

- Tablespace changes, including a tablespace change for a table, partition, index, or LOB columns
- Compression type changes, including a compression type change for a table, partition, index key, or LOB columns
- For LOB columns, a change to SECUREFILE or BASICFILE storage

상기의 변경인 경우 REDEF_TABLE 프로시저를 통한 Single Step 으로 변경 가능

Single-Step Redefinition

```
EXEC DBMS_REDEFINITION.REDEF_TABLE('SCOTT','EMP','ROW STORE COMPRESS ADVANCED');
```

6. DBMS_REDEFINITION MULTIPLE

온라인 재정의 작업이 이러한 변경에만 국한되지 않으면 **MULTIPLE STEP** 를 사용하여 테이블을 온라인으로 재정의해야합니다. 이 단계에는 CAN_REDEF_TABLE, START_REDEF_TABLE, COPY_TABLE_DEPENDENTS 및 FINISH_REDEF_TABLE 프로 시저를 포함하여 DBMS_REDEFINITION 패키지에서 여러 프로 시저를 호출하는 것이 포함됩니다.

GENERAL STEPS TO ONLINE REDEFINE A TABLE

1) Move 하려는 테이블이 온라인에서 재정의할 수 있는지 확인합니다
(DBMS_REDEFINITION.**CAN_REDEF_TABLE**)

2) INTERIM 테이블을 **CREATE** 합니다.
INTERIM 테이블은 ORIGINAL 테이블과 동일한 '모양' (구조가 유사함) 일 필요는 없습니다.

3) 테이블 재정의를 시작합니다.
(DBMS_REDEFINITION.**START_REDEF_TABLE**)

4. ORIGINAL 테이블에서 INTERIM 테이블로 종속된 OBJECT 를 COPY 합니다.
(DBMS_REDEFINITION.**COPY_TABLE_DEPENDENTS**)

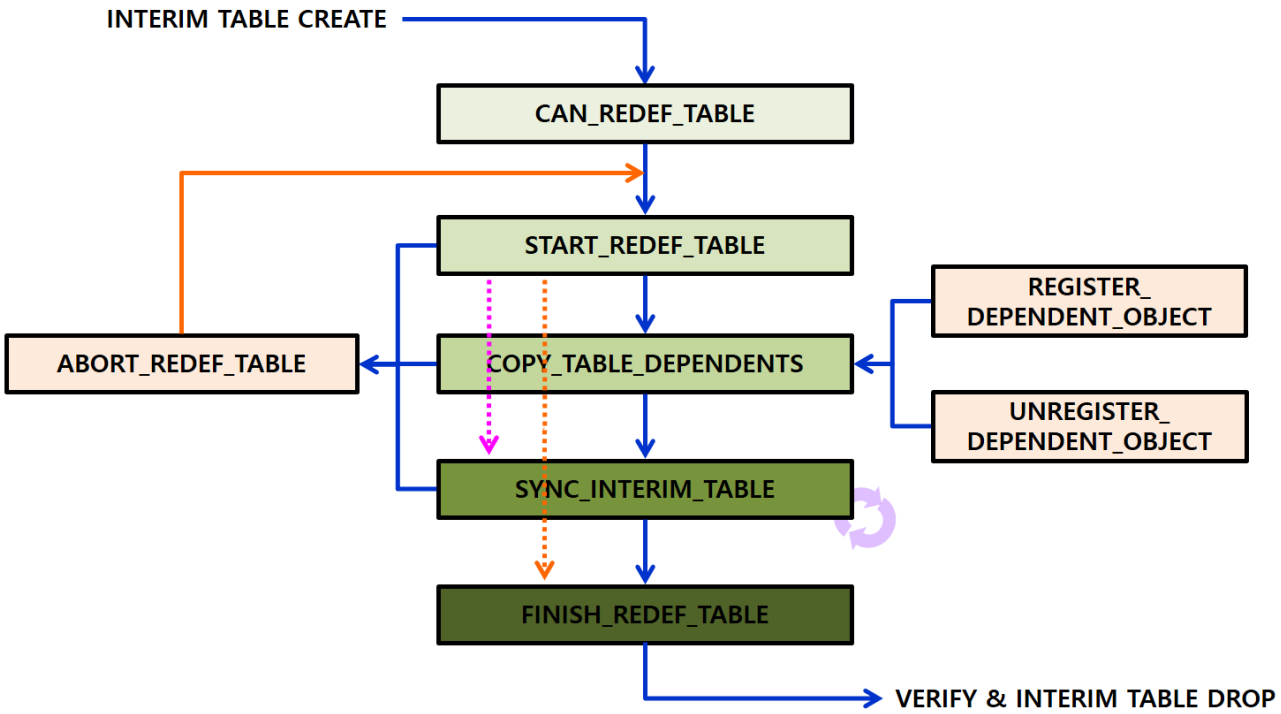
5. ORIGINAL 테이블과 INTERIM 테이블 간의 최종 동기화를 실행합니다.
(DBMS_REDEFINITION.**SYNC_INTERIM_TABLE**)
이 단계는 재정의 완료를 실행하는 데 필요한 시간을 최소화합니다.

6. 재정의를 완료합니다.
(DBMS_REDEFINITION.**FINISH_REDEF_TABLE**)

7. ORIGINAL 테이블 (이전의 INTERIM) 과 INTERIM 테이블 (이전의 ORIGINAL)을 비교하여 재정의가 성공했는지 **VERIFY** 합니다.

8. INTERIM 테이블을 **DROP** 합니다.

DBMS_REDEFINITION 절차를 요약하자면 다음과 같습니다.



DBMS_REDEFINITION 패키지의 SUBPROGRAM 은 다음과 같으며 12C NF 로 ABORT_ROLLBACK 등 6 개의 SUBPROGRAM 이 더 추가되었습니다.

구 분	설 명
CAN_REDEF_TABLE	해당되는 Table 의 Online Redifine 가능 여부 확인
START_REDEF_TABLE	Online Redifine 시작
COPY_TABLE_DEPENDENTS	Original Table 에 있는 Dependent Objects 를 Interim Table 로 Copy
SYNC_INTERIM_TABLE	Interim Table 을 Original Table 과 동기화
FINISH_REDEF_TABLE	Online Redifine 완료
REGISTER_DEPENDENT_OBJECT	Original Table 에 있는 Dependent Objects (index, trigger or constraint) 를 Interim Table 에 Dependent Objects 로 등록
UINREGISTER_DEPENDENT_OBJECT	Original Table 에 있는 Dependent Objects (index, trigger or constraint) 를 Interim Table 에 Dependent Objects 로 등록 해제
ABORT_REDEF_TABLE	Online Redifine 을 진행시 오류가 발생하여 관련된 모든 Temporary Object 정리
ABORT_ROLLBACK	Aborts rollback
ABORT_UPDATE	Aborts an update started with the EXECUTE_UPDATE procedure
EXECUTE_UPDATE	Optimizes the performance of bulk updates to a table 12c NF
REDEF_TABLE	Provides a single push-button interface that integrates several redefinition steps
ROLLBACK	Performs rollback
SET_PARM	Sets a new value for a specified parameter used by the redefinition process identified by a redefinition ID

DBMS_REDEFINITION MULTIPLE 을 이용하여 REDEF_ORG SINGLE 테이블을 PARTITION 테이블로 재정의 해보겠습니다. (REDEF_ORG (ORIGINAL TABLE) / REDEF_INT (INTERIM TABLE))

6.1. CAN_REDEF_TABLE

이 프로시저는 주어진 테이블을 온라인으로 REDEFINED 할 수 있는지 여부를 결정하며 ONLINE REDEFINITION 프로세스의 첫 번째 단계입니다.

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

Parameter	설 명
uname	The schema name of the table
tname	The name of the table to be re-organized
options_flag	Indicates the type of redefinition method to use • dbms_redefinition.cons_use_pk **(default method)** • dbms_redefinition.cons_use_rowid
part_name	The name of the partition being redefined. If redefining only a single partition of a table, specify the partition name in this parameter.

CAN_REDEF_TABLE SUBPROGRAM 을 사용하여 REDEF_ORG 테이블이 온라인으로 재정의 될 수 있는지 확인합니다. 하기와 같이 두가지 방법으로 실행할 수 있습니다.

```
EXAMPLE
BEGIN
DBMS_REDEFINITION.CAN_REDEF_TABLE (
   uname => 'YOUNHA',
   tname =>  'REDEF_ORG',
   options_flag => dbms_redefinition.cons_use_pk,
   part_name => NULL);
END;
/

PL/SQL procedure successfully completed.

OR

EXEC DBMS_REDEFINITION.CAN_REDEF_TABLE ( 'YOUNHA' , 'REDEF_ORG' );
```

6.2. START_REDEF_TABLE

INTERIM 테이블 생성 후 실행해야 하며, ORIGINAL <-> INTERIM 테이블 간의 Data Copy 단계입니다.
ORIGINAL 테이블의 ONLINE DML 로 인한 DATA 변경사항을 INTERIM TABLE 과의 동기화 하기 위해 MATERIALIZED VIEW 를 생성합니다. (DBA_MVIEWS, DBA_MVIEW_LOGS 로 확인 가능)
COLUMN MAPPING PARAMETER 를 통한 컬럼 변형이 가능하며 INTERIM 테이블이 ORIGINAL 테이블과 동일한 구조인 경우 COL_MAPPING 값에 [NULL, '*'] 을 입력하면 됩니다.
암시적인 DATA TYPE 의 변형이 가능하며, ORDERBY_COLS PARAMETER 를 사용하여 해당 컬럼 순으로 정렬하여 데이터를 COPY 할 수 있습니다.

```
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);
```

Parameter	설 명
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.
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)
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.
orig_table	Name of the table to be redefined

REDEF_ORG (SINGLE 테이블) 을 생성하고 PK 를 추가합니다. <ORIGINAL>
REDEF_ING (PARTITION 테이블) 을 생성합니다. <INTERIM>

```
EXAMPLE
CREATE TABLE REDEF_ORG AS SELECT * FROM DBA_OBJECTS;
ALTER TABLE REDEF_ORG ADD CONSTRAINT REDEF_ORG_PK PRIMARY KEY (OBJECT_ID);

CREATE TABLE YOUNHA.REDEF_INT
(
  OWNER                           VARCHAR2(128),
  OBJECT_NAME                     VARCHAR2(128),
  SUBOBJECT_NAME                  VARCHAR2(128),
  OBJECT_ID                       NUMBER,
  DATA_OBJECT_ID                  NUMBER,
  OBJECT_TYPE                     VARCHAR2(23),
  CREATED                         DATE,
  LAST_DDL_TIME                   DATE,
  TIMESTAMP                       VARCHAR2(19),
  STATUS                          VARCHAR2(7),
  TEMPORARY                       VARCHAR2(1),
  GENERATED                       VARCHAR2(1),
  SECONDARY                       VARCHAR2(1),
  NAMESPACE                       NUMBER,
  EDITION_NAME                    VARCHAR2(128),
  SHARING                         VARCHAR2(18),
  EDITIONABLE                     VARCHAR2(1),
  ORACLE_MAINTAINED               VARCHAR2(1),
  APPLICATION                     VARCHAR2(1),
  DEFAULT_COLLATION               VARCHAR2(100),
  DUPLICATED                      VARCHAR2(1),
  SHARDED                         VARCHAR2(1),
  CREATED_APPID                   NUMBER,
  CREATED_VSNID                   NUMBER,
  MODIFIED_APPID                  NUMBER,
  MODIFIED_VSNID                  NUMBER
)
PARTITION BY RANGE(OBJECT_ID)
(PARTITION REDEF_INT_P001 VALUES LESS THAN (40000) TABLESPACE REDEFINITION,
 PARTITION REDEF_INT_P002 VALUES LESS THAN (80000) TABLESPACE REDEFINITION);
```

START_REDEF_TABLE SUBPROGRAM 을 사용하여 REDEF_ORG -> REDEF_INT 테이블 데이터 COPY 를 시작합니다.
조건은 기존 DATA_OBJECT_ID + 100 으로 DATA_OBJECT_ID 컬럼의 데이터 값 변경 및 OBJECT_ID 로 정렬하여 COPY 합니다.

```
BEGIN
DBMS_REDEFINITION.START_REDEF_TABLE (
   uname => 'YOUNHA',
   orig_table => 'REDEF_ORG',
   int_table =>  'REDEF_INT',
   col_mapping =>                'OWNER                OWNER,
                                 OBJECT_NAME           OBJECT_NAME,
                                 SUBOBJECT_NAME        SUBOBJECT_NAME,
                                 OBJECT_ID             OBJECT_ID,
                                 DATA_OBJECT_ID+100    DATA_OBJECT_ID,
                                 OBJECT_TYPE           OBJECT_TYPE,
                                 CREATED               CREATED,
                                 LAST_DDL_TIME         LAST_DDL_TIME,
                                 TIMESTAMP             TIMESTAMP,
                                 STATUS                STATUS,
                                 TEMPORARY             TEMPORARY,
                                 GENERATED             GENERATED,
                                 SECONDARY             SECONDARY,
                                 NAMESPACE             NAMESPACE,
                                 EDITION_NAME          EDITION_NAME,
                                 SHARING               SHARING,
                                 EDITIONABLE           EDITIONABLE,
                                 ORACLE_MAINTAINED     ORACLE_MAINTAINED,
                                 APPLICATION           APPLICATION,
                                 DEFAULT_COLLATION     DEFAULT_COLLATION,
                                 DUPLICATED            DUPLICATED,
                                 SHARDED               SHARDED,
                                 CREATED_APPID         CREATED_APPID,
                                 CREATED_VSNID         CREATED_VSNID,
                                 MODIFIED_APPID        MODIFIED_APPID,
                                 MODIFIED_VSNID        MODIFIED_VSNID',
   options_flag => DBMS_REDEFINITION.CONS_USE_PK,
   orderby_cols => 'OBJECT_ID',
   part_name => NULL);
END;
/
```

USER_OBJECTS 에서 확인하면 REDEF_ORG 테이블에 PK 인덱스가 존재하며, REDEF_INT 테이블에는 인덱스가 없습니다.

MLOG$,RUPD$, I_MLOG$ 의 테이블 및 인덱스가 생성되어 내부적으로 REDEF_ORG 테이블에서의 START 이후 데이터 변경 분을 저장합니다.

```
SELECT OBJECT_NAME, SUBOBJECT_NAME, OBJECT_ID, OBJECT_TYPE, STATUS
FROM USER_OBJECTS ORDER BY OBJECT_ID;

OBJECT_NAME          SUBOBJECT_NAME        OBJECT_ID OBJECT_TYPE             STATUS
-------------------- -------------------- ---------- ----------------------- -------
REDEF_ORG                                      72781 TABLE                   VALID
REDEF_ORG_PK                                   72782 INDEX                   VALID
REDEF_INT                                      72783 TABLE                   VALID
REDEF_INT            REDEF_INT_P001            72784 TABLE PARTITION         VALID
REDEF_INT            REDEF_INT_P002            72785 TABLE PARTITION         VALID
MLOG$_REDEF_ORG                                72786 TABLE                   VALID
RUPD$_REDEF_ORG                                72787 TABLE                   VALID
I_MLOG$_REDEF_ORG                              72788 INDEX                   VALID

SELECT OWNER,OBJECT_NAME,OBJECT_ID,DATA_OBJECT_ID
FROM YOUNHA.REDEF_ORG A WHERE ROWNUM < 5;

OWNER      OBJECT_NAME          OBJECT_ID DATA_OBJECT_ID
---------- -------------------- ---------- --------------
SYS        OBJ$                         18             18
SYS        CCOL$                        32             29
SYS        CLU$                          5              2
SYS        SEG$                         14              8

SELECT OWNER,OBJECT_NAME,OBJECT_ID,DATA_OBJECT_ID
FROM YOUNHA.REDEF_INT A WHERE ROWNUM < 5;

OWNER      OBJECT_NAME          OBJECT_ID DATA_OBJECT_ID
---------- -------------------- ---------- --------------
SYS        C_OBJ#                        2            102
SYS        I_OBJ#                        3            103
SYS        TAB$                          4            102
SYS        CLU$                          5            102
```

OBJECT_ID 정렬 확인

DATA_OBJECT_ID 값 +100 확인

6.3. COPY_TABLE_DEPENDENTS

ORIGINAL 테이블의 DEPENDENT OBJECTS 를 INTERIM 테이블로 COPY 합니다.
EX) GRANTS, TRIGGERS, CONSTRAINTS, PRIVILEGES (ORIGINAL 테이블 DEPENDENT OBJECT -> INTERIM)
이미 등록한 OBJECT 는 복제하지 않습니다. (REGISTER_TABLE_DEPENDENTS 로 등록한 OBJECT)

```
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);
```

Parameter	설 명
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

COPY_TABLE_DEPENDENTS SUBPROGRAM 을 사용하여 REDEF_ORG 테이블의 종속된 오브젝트를 REDEF_INT 테이블로 COPY 합니다.
하기에서는 REDEF_ORG_PK INDEX 가 COPY 되어 TMP$$_REDEF_ORG_PK0 로 생성된 것을 확인할 수 있습니다.

```
EXAMPLE
SET SERVEROUTPUT ON
DECLARE
  l_num_errors PLS_INTEGER;
BEGIN
  DBMS_REDEFINITION.COPY_TABLE_DEPENDENTS(
        uname =>'YOUNHA',
        orig_table => 'REDEF_ORG',
        int_table  => 'REDEF_INT',
        copy_indexes => dbms_redefinition.cons_orig_params,
        copy_triggers => TRUE,                    --Default
        copy_constraints => TRUE,                 --Default
        copy_privileges  => TRUE,                 --Default
        ignore_errors => FALSE,                   --Default
        num_errors => l_num_errors);
        DBMS_OUTPUT.put_line('No of errors during copy of dependents ' || l_num_errors);
END;
/

No of errors during copy of dependents 0

PL/SQL procedure successfully completed.

SELECT OBJECT_NAME, SUBOBJECT_NAME, OBJECT_ID, OBJECT_TYPE, STATUS
FROM USER_OBJECTS ORDER BY OBJECT_ID;

OBJECT_NAME          SUBOBJECT_NAME        OBJECT_ID OBJECT_TYPE             STATUS
-------------------- -------------------- ---------- ----------------------- -------
REDEF_ORG                                      72781 TABLE                   VALID
REDEF_ORG_PK                                   72782 INDEX                   VALID
REDEF_INT                                      72783 TABLE                   VALID
REDEF_INT            REDEF_INT_P001            72784 TABLE PARTITION         VALID
REDEF_INT            REDEF_INT_P002            72785 TABLE PARTITION         VALID
MLOG$_REDEF_ORG                                72786 TABLE                   VALID
RUPD$_REDEF_ORG                                72787 TABLE                   VALID
I_MLOG$_REDEF_ORG                              72788 INDEX                   VALID
TMP$$_REDEF_ORG_PK0                            72791 INDEX                   VALID
```

6.4. REGISTER_DEPENDENT_OBJECT

COPY_TABLE_DEPENDENTS 에서 오류가 발생한 경우 수동으로 등록 가능합니다.

```
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);
```

Parameter	설 명
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

6.5. SYNC_INTERIM_TABLE

ORIGINAL 과 INTERIM 테이블 간의 ONLINE DML 에 대한 DATA 동기화 작업입니다.
사전 동기화 작업이기 때문에 FINISH_REDEF_TABLE 실행 시 시간을 단축시킬 수 있습니다.
MATERIALIZED VIEW 에 생성된 자료 참고하며 SYNC 작업으로 동기화가 완료된 건은 MVIEW 에서 삭제됩니다.

```
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);
```

Parameter	설 명
uname	The schema name of the table
orig_table	Name of the table to be redefined
int_table	Name of the interim table.
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)

SYNC_INTERIM_TABLE SUBPROGRAM 을 사용하여 REDEF_ORG -> REDEF_INT 테이블간의 동기화 작업을 실시합니다.

```
EXAMPLE
BEGIN
DBMS_REDEFINITION.SYNC_INTERIM_TABLE (
   uname  => 'YOUNHA',
   orig_table => 'REDEF_ORG',
   int_table => 'REDEF_INT',
   part_name => NULL);
END;
/

PL/SQL procedure successfully completed.
```

6.6. FINISH_REDEF_TABLE

REDEFINITION 처리 과정 완료 작업입니다. (ORIGINAL <-> INTERIM)
ORIGINAL 테이블이 INTERIM 테이블과 재정의됨으로써 순간 TABLE EXCLUSIVE LOCK 이 발생합니다.
ORIGINAL TABLE 에 DML 이 많이 발생하면 처리 시간 길어져 SYNC 를 자주 수행하여 동기화 시간을 짧게 가져가야 좋습니다.

```
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);
```

Parameter	설 명
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_TABLEprocedure. Specifying TRUE cleans up the database objects that enable rollback.

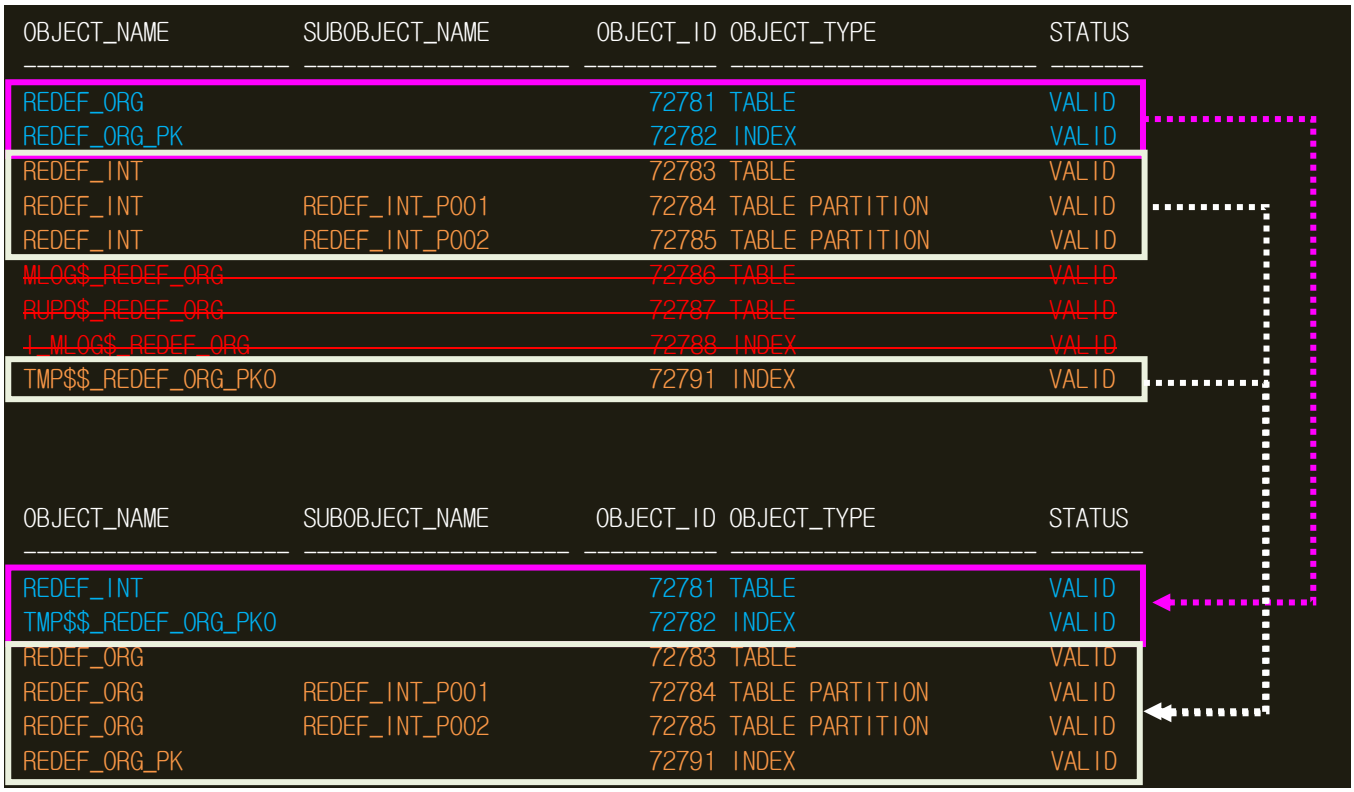
FINISH_REDEF_TABLE SUBPROGRAM 을 사용하여 REDEF_ORG <-> REDEF_INT 테이블간의 재정의를 완료합니다.

```
EXAMPLE
BEGIN
DBMS_REDEFINITION.FINISH_REDEF_TABLE (
   uname  => 'YOUNHA',
   orig_table => 'REDEF_ORG',
   int_table => 'REDEF_INT',
   part_name => NULL);
END;
/

PL/SQL procedure successfully completed.

SELECT OBJECT_NAME, SUBOBJECT_NAME, OBJECT_ID, OBJECT_TYPE, STATUS
FROM USER_OBJECTS ORDER BY OBJECT_ID;

OBJECT_NAME          SUBOBJECT_NAME        OBJECT_ID OBJECT_TYPE             STATUS
-------------------- -------------------- ---------- ----------------------- -------
REDEF_INT                                      72781 TABLE                   VALID
TMP$$_REDEF_ORG_PK0                            72782 INDEX                   VALID
REDEF_ORG                                      72783 TABLE                   VALID
REDEF_ORG            REDEF_INT_P001            72784 TABLE PARTITION         VALID
REDEF_ORG            REDEF_INT_P002            72785 TABLE PARTITION         VALID
REDEF_ORG_PK                                   72791 INDEX                   VALID

SELECT OBJECT_NAME, OBJECT_ID, DATA_OBJECT_ID, OBJECT_TYPE
FROM YOUNHA.REDEF_ORG WHERE OBJECT_NAME = 'EMP';

OBJECT_NAME          OBJECT_ID DATA_OBJECT_ID OBJECT_TYPE
-------------------- ---------- -------------- -----------------------
EMP                       72712          72812 TABLE

SELECT OBJECT_NAME, OBJECT_ID, DATA_OBJECT_ID, OBJECT_TYPE
FROM YOUNHA.REDEF_INT WHERE OBJECT_NAME = 'EMP';

OBJECT_NAME          OBJECT_ID DATA_OBJECT_ID OBJECT_TYPE
-------------------- ---------- -------------- -----------------------
EMP                       72712          72712 TABLE
SELECT OWNER, TABLE_NAME, PARTITIONING_TYPE, PARTITION_COUNT
FROM DBA_PART_TABLES WHERE OWNER='YOUNHA';

OWNER      TABLE_NAME           PARTITION PARTITION_COUNT
---------- -------------------- --------- ---------------
YOUNHA     REDEF_ORG            RANGE                   2
```

6.7. VERIFY

하기 쿼리의 결과값은 위는 COPY_TABLE_DEPENDNETS 단계, 아래는 FINISH_REDEF_TABLE 을 수행 완료 단계입니다.

OBJECT_ID 72781 & 72782 인 REDEF_ORG 테이블 관련 OBJECT 가 REDEF_INT 테이블 OBJECT 로 변경된 것을 확인 할 수 있으며, 반면 OBJECT_ID 72783 & 72784 & 72785 & 72791 인 REDEF_INT 테이블 관련 OBJECT 들이 REDEF_ORG 로 변경되었습니다.



	OBJECT_ID REDEFINITION 전	OBJECT_ID REDEFINITION 후
REDEF_ORG	72781	72783
REDEF_INT	72783	72781

주의 할 점은 OBJECT_ID 가 변경됨으로써 기존 해당 테이블의 조회하던 SQL PLAN 들이 무효화됩니다. 따라서 HARD 파싱이 일어나게 되고 만약 운영 중 대량의 조회 작업시 하기와 같은 Library cache load lock & Library cache: mutex X, Row cache mutex 등의 이벤트가 발생하여 전체적인 성능 저하 및 서비스 장애가 일어날 수 있습니다.

SID	Serial#	SPID	User Name	O/S User	Command	Status	Event
393	24494	5635		oracle	UNKNOWN	ACTIVE	library cache load lock
1350	25816	5553		oracle	UNKNOWN	ACTIVE	library cache: mutex X
1159	37631	5574		oracle	UNKNOWN	ACTIVE	library cache load lock
1349	24696	5547		oracle	UNKNOWN	ACTIVE	library cache load lock
11	34976	5648		oracle	UNKNOWN	ACTIVE	library cache: mutex X
13	16762	5731		oracle	UNKNOWN	ACTIVE	library cache load lock
207	19040	5717		oracle	UNKNOWN	ACTIVE	library cache load lock
209	63141	5736		oracle	UNKNOWN	ACTIVE	library cache load lock
394	61561	5611		oracle	UNKNOWN	ACTIVE	library cache load lock
588	21035	5588		oracle	UNKNOWN	ACTIVE	library cache: mutex X
776	60590	5538		oracle	UNKNOWN	ACTIVE	library cache load lock
779	64536	5596		oracle	UNKNOWN	ACTIVE	library cache: mutex X
966	17395	5670		oracle	UNKNOWN	ACTIVE	library cache: mutex X
968	44521	5666		oracle	UNKNOWN	ACTIVE	library cache load lock
1158	16586	5702		oracle	UNKNOWN	ACTIVE	library cache load lock
1160	40875	5570		oracle	UNKNOWN	ACTIVE	library cache load lock

SID	Serial#	SPID	User Name	O/S User	Command	Status	Event
22	65038	5669	REORG	oracle	SELECT	ACTIVE	row cache mutex
25	1546	5542	REORG	oracle	SELECT	ACTIVE	row cache mutex
30	4328	5720	REORG	oracle	SELECT	ACTIVE	row cache mutex
34	15690	5687	REORG	oracle	UNKNOWN	ACTIVE	row cache mutex
227	47338	5685	REORG	oracle	SELECT	ACTIVE	row cache mutex
393	24494	5635	REORG	oracle	SELECT	ACTIVE	row cache mutex
399	27613	5733	REORG	oracle	SELECT	ACTIVE	row cache mutex
400	34123	5592	REORG	oracle	SELECT	ACTIVE	row cache mutex
403	11389	5692	REORG	oracle	UNKNOWN	ACTIVE	row cache mutex
407	47307	5580	REORG	oracle	SELECT	ACTIVE	row cache mutex
408	54099	5558	REORG	oracle	SELECT	ACTIVE	row cache mutex
604	33961	5537	REORG	oracle	SELECT	ACTIVE	row cache mutex
606	1938	5577	REORG	oracle	SELECT	ACTIVE	row cache mutex
784	31140	5630	REORG	oracle	SELECT	ACTIVE	row cache mutex
785	60149	5606	REORG	oracle	SELECT	ACTIVE	row cache mutex
786	21399	5564	REORG	oracle	SELECT	ACTIVE	row cache mutex

7. REORG

단편화가 심한 테이블의 크기를 줄이기 위해 온라인 재정의 (DBMS_REDEFINITION) 를 사용할 수 있습니다. 단 INTERIM 테이블뿐만 아니라 ORIGINAL 테이블을 저장하는 데 필요한 공간이 일시적 필요합니다. (ORIGINAL 테이블의 SIZE 2 배)

```
TEST CONFIGURATION
create table ORIGINAL (COL1 NUMBER, COL2 VARCHAR2(1000), COL3 VARCHAR2(1000),
COL4 VARCHAR2(1000));

ORIGINAL 테이블에 대량의 데이터 INSERT

declare
   v_out varchar2(1000);
begin
   v_out := null;
   for i in 1..1000 loop
      v_out := v_out||'A';
   end loop;
   for i in 1..10000 loop
      insert into ORIGINAL values (i,v_out,v_out,v_out);
      if i/10000 = trunc(i/1000) then
         commit;
      end if;
   end loop;
   commit;
end;

ORIGINAL 테이블에서 2/3 데이터 DELETE -> 단편화 발생

DELETE FROM ORIGINAL where (COL1/3) <> trunc(COL1/3);

6667 rows deleted.

COMMIT;
```

DBMS_REDEFINITON 을 통한 REORG 수행

1. CHECK !

```
BEGIN
   DBMS_REDEFINITION.CAN_REDEF_TABLE('TEST','ORIGINAL', DBMS_REDEFINITION.CONS_USE_ROWID);
END;
/
```

2. INTERIM TABLE CREATE !

```
CREATE TABLE INTERIM (COL1 NUMBER, COL2 VARCHAR2(1000),COL3 VARCHAR2(1000), COL4 VARCHAR2(1000));
```

3. START !

```
BEGIN
   DBMS_REDEFINITION.START_REDEF_TABLE(
                uname => 'TEST',
                orig_table => 'ORIGINAL',
                int_table => 'INTERIM',
                options_flag => DBMS_REDEFINITION.CONS_USE_ROWID);
END;
/
```

4. DEPENDENT OBJECT COPY !

```
DECLARE
   error_count pls_integer := 0;
BEGIN
   DBMS_REDEFINITION.COPY_TABLE_DEPENDENTS('TEST', 'ORIGINAL', 'INTERIM '
, dbms_redefinition.cons_orig_params, TRUE,TRUE,TRUE,FALSE, error_count);
   DBMS_OUTPUT.PUT_LINE('errors := ' || TO_CHAR(error_count));
END;
/
```

5. SYNC !

```
BEGIN
   DBMS_REDEFINITION.SYNC_INTERIM_TABLE('TEST', 'ORIGINAL', 'INTERIM');
END;
/
```

6. FINISH !

```
exec DBMS_REDEFINITION.FINISH_REDEF_TABLE('TEST','ORIGINAL','INTERIM');
```

```
REORG 확인 후 INTRIM 테이블 DROP

SELECT COUNT(*), SEGMENT_NAME FROM USER_EXTENTS GROUP BY SEGMENT_NAME;
  COUNT(*) SEGMENT_NAME
---------- --------------------
        40 INTERIM
        14 ORIGINAL

DROP TABLE INTERIM PURGE;
```

8. REFERENCE

- https://docs.oracle.com/en/database/oracle/oracle-database/12.2/admin/managing-tables.html#GUID-070A3AEC-4226-4E65-BDB0-A5B2B5B48B1F (20.8 Redefining Tables Online)
- https://docs.oracle.com/en/database/oracle/oracle-database/12.2/arpls/DBMS_REDEFINITION.html (129 DBMS_REDEFINITION)
- DBMS_redefinition Word 문서 (굿어스)
- Master Note:Overview of Online Redefinition of Tables (DBMS_REDEFINITION) (Doc ID 1357825.1)
- IF: Online Redefinition Tips And Tricks (Doc ID 2359350.1)
- How to Shrink a Table Using Online Redefinition (Doc ID 1357878.1)
- DBMS_REDEFINITION ONLINE REORGANIZATION OF TABLES (Doc ID 149564.1)
- Online Redefintion using DBMS_REDEFINITION.REDEF_TABLE (Doc ID 2412059.1)
- HOW TO USE DBMS_REDEFINITION.REGISTER_DEPENDENT_OBJECT (Doc ID 1304838.1)
- How To Partition Existing Table Using DBMS_REDEFINITION (Doc ID 472449.1)