5주차 3. 파일과 데이터 관리 & 4. 스키마 객체 관리

제출일 : 2022년 12월 2일 금요일

작성자 : 박민영

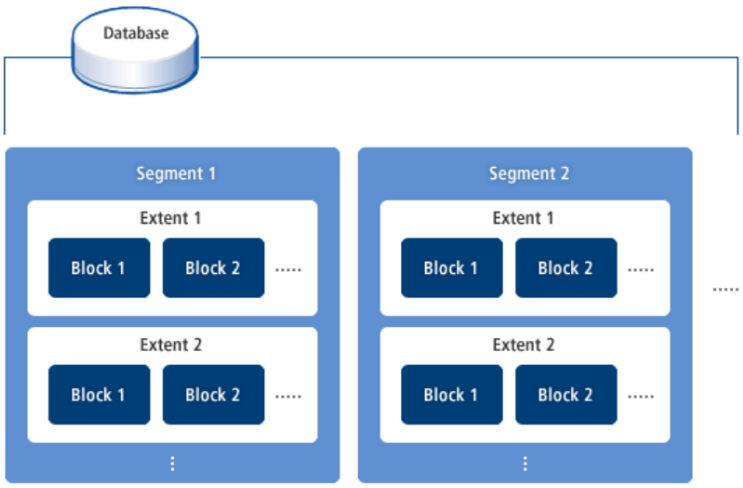
---

1. 테이블 스페이스 생성, 제거

1.1 설명

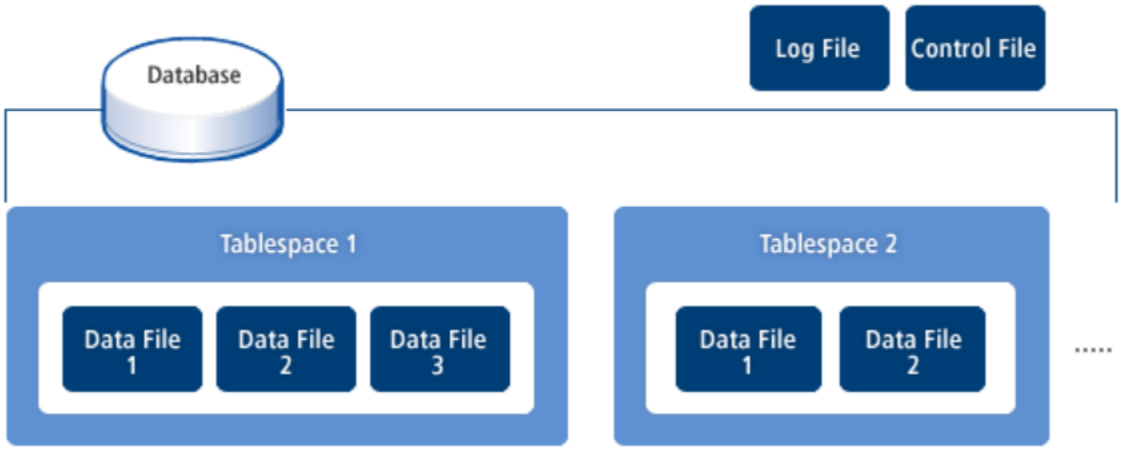
테이블 스페이스

- 논리적 저장 영역과 물리적 저장 영역에 공통적으로 포함된다.
- 논리적 저장 영역에는 Tibero의 모든 데이터가 저장되며, 물리적 저장 영역에는 데이터 파일이 하나 이상 저장된다.
- 테이블 스페이스는 논리적 저장 영역과 물리적 저장 영역을 연관시키기 위한 단위이다.
- 테이블 스페이스의 논리적 구성
  - 데이터 블록 : 데이터베이스에서 사용하는 데이터의 최소 단위.
  - Extent : 연속된 데이터 블록의 집합
  - Segment : Extent의 집합, 하나의 테이블, 인덱스 등에 대응되는 것



- 테이블 스페이스의 물리적 구성

- 물리적으로 여러 개의 데이터 파일로 구성된다.
- 빈번하게 사용되는 두 테이블 스페이스(예: 테이블과 인덱스)는 물리적으로 서로 다른 디스크에 저장하는 것이 좋다.



1.2 시나리오 수행

시나리오 내용
1. /home/tibero/dtf 폴더 생성
2. 테이블 스페이스 생성
3. 테이블 스페이스 데이터 파일 추가
4. 테이블 스페이스 데이터 파일 크기 변경
5. 테이블 스페이스 제거

1.3 시나리오 수행 결과

1. /home/tibero/dtf 폴더 생성

```
-- 1. /home/tibero/dtf 폴더 생성
cd /home/tibero
mkdir dtf
```

```
[tibero@T1:/home/tibero]$ ls
a  audit  start_db.sh  stop_db.sh  tool_tbsql  week2  week3  week4
[tibero@T1:/home/tibero]$ mkdir dtf
```

2. 테이블 스페이스 생성

```
-- 2. 테이블 스페이스 생성
CREATE TABLESPACE my_space
    DATAFILE '/home/tibero/dtf/my_file1.dtf' SIZE 20M,
             '/home/tibero/dtf/my_file2.dtf' SIZE 30M
    AUTOEXTEND ON NEXT 1M
    EXTENT MANAGEMENT LOCAL UNIFORM SIZE 64K;
```

```
CREATE TABLESPACE my_space
    DATAFILE '/home/tibero/dtf/my_file1.dtf' SIZE 20M,
             '/home/tibero/dtf/my_file2.dtf' SIZE 30M
AUTOEXTEND ON NEXT 1M
   5     EXTENT MANAGEMENT LOCAL UNIFORM SIZE 64K;

Tablespace 'MY_SPACE' created.
```

```
[tibero@T1:/home/tibero/dtf]$ ls -l
total 51200
-rw------- 1 tibero dba 20971520 Nov 29 17:14 my_file1.dtf
-rw------- 1 tibero dba 31457280 Nov 29 17:14 my_file2.dtf
```

- 데이터 파일 my_file1.dtf, my_file2.dtf는 SQL 문장을 실행함과 동시에 생성된다.
- `AUTOEXTEND ON NEXT 1M` : 저장 공간이 더 필요할 것에 대비하여 1MB씩 확장하도록 설정
- 테이블 스페이스 my_space의 전체 크기 : 20MB + 30MB = 50MB

3. 테이블 스페이스 데이터 파일 추가

```
-- 3. 테이블 스페이스 데이터 파일 추가
ALTER TABLESPACE my_space
  ADD DATAFILE '/home/tibero/dtf/my_file3.dtf' SIZE 20M;
```

```
ALTER TABLESPACE my_space
   2 ADD DATAFILE '/home/tibero/dtf/my_file3.dtf' SIZE 20M;

Tablespace 'MY_SPACE' altered.
```

```
[tibero@T1:/home/tibero/dtf]$ ls -l
total 71680
-rw------- 1 tibero dba 20971520 Nov 29 17:14 my_file1.dtf
-rw------- 1 tibero dba 31457280 Nov 29 17:14 my_file2.dtf
-rw------- 1 tibero dba 20971520 Nov 29 17:30 my_file3.dtf
```

my_file3.dtf 파일도 생성된 것을 확인해볼 수 있다.

4. 테이블 스페이스 데이터 파일 크기 변경

```
-- 4. 테이블 스페이스 데이터 파일 크기 변경
ALTER DATABASE DATAFILE '/home/tibero/dtf/my_file1.dtf' RESIZE 100M;
```

```
SQL> ALTER DATABASE DATAFILE '/home/tibero/dtf/my_file1.dtf' RESIZE 1
00M;

Database altered.
```

이렇게 변경하고 나서 다시 조회해보면 아래와 같이 100MB 크기로 변경된 것을 확인해볼 수 있다.

```
[tibero@T1:/home/tibero/dtf]$ ls -l my_file1.dtf
-rw------- 1 tibero dba 104857600 Nov 29 17:31 my_file1.dtf
```

5. 테이블 스페이스 제거

```
-- 5. 테이블 스페이스 제거
DROP TABLESPACE my_space
  INCLUDING CONTENTS AND DATAFILES;
```

```
DROP TABLESPACE my_space
   2 INCLUDING CONTENTS AND DATAFILES;

Tablespace 'MY_SPACE' dropped.
```

`INCLUDING CONTENTS AND DATAFILES` : 데이터 파일까지 제거하려면 다음과 같이 **INCLUDING** 절을 삽입하여 DROP TABLESPACE 문을 실행해야 한다.

1.sh

```
-- 1. /home/tibero/dtf 폴더 생성
-- cd /home/tibero
-- mkdir dtf

-- 2. 테이블 스페이스 생성
CREATE TABLESPACE my_space
    DATAFILE '/home/tibero/dtf/my_file1.dtf' SIZE 20M,
             '/home/tibero/dtf/my_file2.dtf' SIZE 30M
    AUTOEXTEND ON NEXT 1M
    EXTENT MANAGEMENT LOCAL UNIFORM SIZE 64K;

-- 3. 테이블 스페이스 데이터 파일 추가
ALTER TABLESPACE my_space
  ADD DATAFILE '/home/tibero/dtf/my_file3.dtf' SIZE 20M;

-- 4. 테이블 스페이스 데이터 파일 크기 변경
ALTER DATABASE DATAFILE '/home/tibero/dtf/my_file1.dtf' RESIZE 100M;

-- 5. 테이블 스페이스 제거
DROP TABLESPACE my_space
  INCLUDING CONTENTS AND DATAFILES;
```

---

2. 테이블 스페이스 정보 조회

2.1 설명

아래의 표에 해당하는 뷰에서는 테이블 스페이스 내의 익스텐트의 크기 및 개수, 할당된 서버, 포함된 데이터 파일의 이름 및 크기, 세그먼트의 이름 및 종류, 크기 등의 정보를 제공한다.

뷰	설명
DBA_TABLESPACES	Tibero 내의 **모든 테이블 스페이스의 정보**를 조회하는 뷰
USER_TABLESPACES	**현재 사용자에 속한 테이블 스페이스의 정보**를 조회하는 뷰

뷰	설명
V$TABLESPACE	Tibero 내의 **모든** 테이블 스페이스에 대한 **간략한 정보**를 조회하는 뷰

2.2 시나리오 수행

시나리오 내용
1. DBA_TABLESPACES 조회
2. USER_TABLESPACES 조회
3. V$TABLESPACE 조회

2.3 시나리오 수행 결과

1. DBA_TABLESPACES 조회

```
SQL> SELECT systimestamp FROM dual;

SYSTIMESTAMP
--------------------------------------------------------------------------------
2022/11/30 16:46:47.299175 Asia/Seoul

1 row selected.



SQL> SELECT * FROM DBA_TABLESPACES;

TABLESPACE_NAME
--------------------------------------------------------------------------------
     TS_ID DATAFILE_COUNT BLOCK_SIZE NEXT_EXTENT STATUS    CONTENTS  LOGGING
---------- -------------- ---------- ----------- --------- --------- ---------
FORCE_LOGGING ALLOCATION_TYPE ENCRYPTED INITIAL_EXTENT MIN_EXTENTS MAX_EXTENTS
------------- --------------- --------- -------------- ----------- -----------
EXTENT_MANAGEMENT SEGMENT_SPACE_MANAGEMENT
----------------- ------------------------
SYSTEM
         0              1       8192             ONLINE    PERMANENT LOGGING
NO             SYSTEM          NO                131072           1  2147483645
LOCAL              AUTO

UNDO
         1              1       8192      131072 ONLINE    UNDO      LOGGING
NO             UNIFORM         NO                131072           1  2147483645
LOCAL              AUTO

TEMP
         2              1       8192             ONLINE    TEMPORARY NOLOGGING
NO             SYSTEM          NO                131072           1  2147483645
```

```
LOCAL             AUTO

USR
         3              1       8192             ONLINE    PERMANENT LOGGING
NO             SYSTEM          NO                131072           1  2147483645
LOCAL             AUTO


TABLESPACE_NAME
--------------------------------------------------------------------------------
     TS_ID DATAFILE_COUNT BLOCK_SIZE NEXT_EXTENT STATUS    CONTENTS  LOGGING
---------- -------------- ---------- ----------- --------- --------- ---------
FORCE_LOGGING ALLOCATION_TYPE ENCRYPTED INITIAL_EXTENT MIN_EXTENTS MAX_EXTENTS
------------- --------------- --------- -------------- ----------- -----------
EXTENT_MANAGEMENT SEGMENT_SPACE_MANAGEMENT
----------------- ------------------------
SYSSUB
         4              1       8192             ONLINE    PERMANENT LOGGING
NO             SYSTEM          NO                131072           1  2147483645
LOCAL             AUTO


5 rows selected.
```

2. USER_TABLESPACES 조회

```
SQL> SELECT * FROM USER_TABLESPACES;

TABLESPACE_NAME
--------------------------------------------------------------------------------
     TS_ID DATAFILE_COUNT BLOCK_SIZE NEXT_EXTENT STATUS    CONTENTS  LOGGING
---------- -------------- ---------- ----------- --------- --------- ---------
FORCE_LOGGING ALLOCATION_TYPE ENCRYPTED INITIAL_EXTENT MIN_EXTENTS MAX_EXTENTS
------------- --------------- --------- -------------- ----------- -----------
EXTENT_MANAGEMENT SEGMENT_SPACE_MANAGEMENT
----------------- ------------------------
SYSTEM
         0              1       8192             ONLINE    PERMANENT LOGGING
NO             SYSTEM          NO                131072           1  2147483645
LOCAL             AUTO

UNDO
         1              1       8192      131072 ONLINE    UNDO      LOGGING
NO             UNIFORM         NO                131072           1  2147483645
LOCAL             AUTO

TEMP
         2              1       8192             ONLINE    TEMPORARY NOLOGGING
NO             SYSTEM          NO                131072           1  2147483645
LOCAL             AUTO

USR
         3              1       8192             ONLINE    PERMANENT LOGGING
```

```
NO            SYSTEM          NO                131072           1  2147483645
LOCAL             AUTO


TABLESPACE_NAME
------------------------------------------------------------------------------
     TS_ID DATAFILE_COUNT BLOCK_SIZE NEXT_EXTENT STATUS    CONTENTS  LOGGING
---------- -------------- ---------- ----------- --------- --------- ---------
FORCE_LOGGING ALLOCATION_TYPE ENCRYPTED INITIAL_EXTENT MIN_EXTENTS MAX_EXTENTS
------------- --------------- --------- -------------- ----------- -----------
EXTENT_MANAGEMENT SEGMENT_SPACE_MANAGEMENT
----------------- ------------------------
SYSSUB
         4              1       8192             ONLINE    PERMANENT LOGGING
NO            SYSTEM          NO                131072           1  2147483645
LOCAL             AUTO


5 rows selected.
```

3. V$TABLESPACE 조회

```
COL NAME FOR A20;
SELECT * FROM V$TABLESPACE;
```

```
SQL> col NAME for a20;
SQL> SELECT * FROM V$TABLESPACE;

       TS# NAME                 TYPE BIGFILE FLASHBACK_ON
---------- -------------------- ---- ------- ------------
         0 SYSTEM               DATA NO      NO
         1 UNDO                 UNDO NO      NO
         2 TEMP                 TEMP NO      NO
         3 USR                  DATA NO      NO
         4 SYSSUB               DATA NO      NO

5 rows selected.
```

2.sh

```
-- 1. DBA_TABLESPACES 조회
SELECT * FROM DBA_TABLESPACES;

-- 2. USER_TABLESPACES 조회
```

```
SELECT * FROM USER_TABLESPACES;

-- 3. V$TABLESPACE 조회
COL NAME FOR A20;
SELECT * FROM V$TABLESPACE;
```

---

3. 로그 파일 정보 조회, 생성, 제거

3.1 설명

새로운 로그 그룹 또는 로그 그룹에 포함되어 있는 로그 멤버를 생성하거나 제거하려면 **ALTER DATABASE** 문을 사용해야 한다.

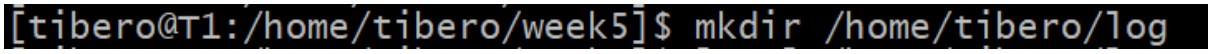
3.2 시나리오 수행

시나리오 내용
1. /home/tibero/log 폴더 생성
2. 로그 그룹 정보 조회
3. 로그 파일 정보 조회
4. 로그 파일 생성
5. 기존의 로그 그룹에 새로운 로그 멤버 추가
6. 로그 그룹 내의 하나의 로그 멤버 제거
7. 로그 그룹 제거

3.3 시나리오 수행 결과

1. /home/tibero/log 폴더 생성

```
-- 1. /home/tibero/log 폴더 생성
-- mkdir /home/tibero/log
-- ls -l /home/tibero
```



```
[tibero@T1:/home/tibero/week5]$ ls -l /home/tibero
total 16
-rw-r--r-- 1 tibero dba    4 Aug 12 18:24 a
drwxr-xr-x 2 tibero dba   23 Nov  4 14:43 audit
drwxr-xr-x 2 tibero dba    6 Nov 29 17:33 dtf
drwxr-xr-x 2 tibero dba    6 Dec  1 11:08 log
```

2. 로그 그룹 정보 조회

```
-- 2. 로그 그룹 정보 조회
SELECT * FROM V$LOG;
```

```
SQL> SELECT * FROM V$LOG;

   THREAD#     GROUP#  SEQUENCE#      BYTES    MEMBERS ARCHIVED STATUS
---------- ---------- ---------- ---------- ---------- -------- --------
FIRST_CHANGE# FIRST_TIME
------------- --------------------------------------------------------------
         0          0        250    1048576          2 NO       CURRENT
       183333 2022/12/01

         0          1        248    1048576          2 YES      INACTIVE
       181861 2022/12/01

         0          2        249    1048576          2 YES      INACTIVE
       181923 2022/12/01

3 rows selected.
```

그룹이 0,1,2만 있는 것을 확인해볼 수 있다.

3. 로그 파일 정보 조회

```
-- 3. 로그 파일 정보 조회
COL MEMBER FOR A40
SELECT * FROM V$LOGFILE;
```

```
COL MEMBER FOR A40
SQL> SELECT * FROM V$LOGFILE;

    GROUP# STATUS  TYPE    MEMBER
---------- ------- ------ ----------------------------------------
         0         ONLINE /tibero/tbdata/tibero/redo01.redo
         0         ONLINE /tibero/tbdata/tibero/redo02.redo
         1         ONLINE /tibero/tbdata/tibero/redo11.redo
         1         ONLINE /tibero/tbdata/tibero/redo12.redo
         2         ONLINE /tibero/tbdata/tibero/redo21.redo
         2         ONLINE /tibero/tbdata/tibero/redo22.redo

6 rows selected.
```

그룹 0,1,2에 있는 파일들을 확인해볼 수 있다.

4. 로그 파일 생성

```
-- 4. 로그 파일 생성
ALTER DATABASE ADD LOGFILE GROUP 3 (
        '/home/tibero/log/my_log3_1.log',
        '/home/tibero/log/my_log3_2.log') SIZE 512K;

SELECT * FROM V$LOG;
SELECT * FROM V$LOGFILE;
```

```
-- 4. 로 그  파 일  생 성
ALTER DATABASE ADD LOGFILE GROUP 3 (
        '/home/tibero/log/my_log3_1.log',
        '/home/tibero/log/my_log3_2.log') SIZE 512K;

SELECT * FROM V$LOG;

Database altered.
```

```
SELECT * FROM V$LOG;

   THREAD#     GROUP#  SEQUENCE#      BYTES     MEMBERS ARCHIVED STATUS
---------- ---------- ---------- ---------- ---------- -------- --------
FIRST_CHANGE# FIRST_TIME
------------- -------------------------------------------------------------
         0          0        250    1048576          2 YES      INACTIVE
       183333 2022/12/01

         0          1        251    1048576          2 NO       CURRENT
       183526 2022/12/01

         0          2        249    1048576          2 YES      INACTIVE
       181923 2022/12/01

         0          3         -1     524288          2 NO       UNUSED
            0

4 rows selected.
```

```
COL MEMBER FOR A40
SQL> SELECT * FROM V$LOGFILE;

    GROUP# STATUS  TYPE    MEMBER
---------- ------- ------ ----------------------------------------
         0         ONLINE /tibero/tbdata/tibero/redo01.redo
         0         ONLINE /tibero/tbdata/tibero/redo02.redo
         1         ONLINE /tibero/tbdata/tibero/redo11.redo
         1         ONLINE /tibero/tbdata/tibero/redo12.redo
         2         ONLINE /tibero/tbdata/tibero/redo21.redo
         2         ONLINE /tibero/tbdata/tibero/redo22.redo
         3         ONLINE /home/tibero/log/my_log3_1.log
         3         ONLINE /home/tibero/log/my_log3_2.log

8 rows selected.
```

GROUP 3에 로그 파일을 생성한 것을 확인할 수 있다.

5. 기존의 로그 그룹에 새로운 로그 멤버 추가

```
-- 5. 기존의 로그 그룹에 새로운 로그 멤버 추가
ALTER DATABASE ADD LOGFILE MEMBER
        '/home/tibero/log/my_log3_3.log' TO GROUP 3;

SELECT * FROM V$LOG;
SELECT * FROM V$LOGFILE;
```

```
-- 5. 기존의 로그 그룹에 새로운 로그 멤버 추가
ALTER DATABASE ADD LOGFILE MEMBER
        '/home/tibero/log/my_log3_3.log' TO GROUP 3;


Database altered.
```

```
SELECT * FROM V$LOG;

   THREAD#     GROUP#  SEQUENCE#      BYTES     MEMBERS ARCHIVED STATUS
---------- ---------- ---------- ---------- ---------- -------- --------
FIRST_CHANGE# FIRST_TIME
------------- -------------------------------------------------------------
         0          0        250    1048576          2 YES      INACTIVE
       183333 2022/12/01

         0          1        251    1048576          2 NO       CURRENT
       183526 2022/12/01

         0          2        249    1048576          2 YES      INACTIVE
       181923 2022/12/01

         0          3         -1     524288          3 NO       UNUSED
            0

4 rows selected.
```

```
SQL> SELECT * FROM V$LOGFILE;

    GROUP# STATUS  TYPE    MEMBER
---------- ------- ------ ------------------------------------
         0         ONLINE /tibero/tbdata/tibero/redo01.redo
         0         ONLINE /tibero/tbdata/tibero/redo02.redo
         1         ONLINE /tibero/tbdata/tibero/redo11.redo
         1         ONLINE /tibero/tbdata/tibero/redo12.redo
         2         ONLINE /tibero/tbdata/tibero/redo21.redo
         2         ONLINE /tibero/tbdata/tibero/redo22.redo
         3         ONLINE /home/tibero/log/my_log3_1.log
         3         ONLINE /home/tibero/log/my_log3_2.log
         3         ONLINE /home/tibero/log/my_log3_3.log

9 rows selected.
```

그룹 3에 멤버 파일이 하나 더 생긴 것을 확인해볼 수 있다.

6. 로그 그룹 내의 하나의 로그 멤버 제거

```
-- 6. 로그 그룹 내의 하나의 로그 멤버 제거
ALTER DATABASE DROP LOGFILE MEMBER '/home/tibero/log/my_log3_3.log';

SELECT * FROM V$LOG;
SELECT * FROM V$LOGFILE;
```

```
-- 6. 로그 그룹 내의 하나의 로그 멤버 제거
ALTER DATABASE DROP LOGFILE MEMBER '/home/tibero/log/my_log3_3.log';

SELECT * FROM V$LOG;

Database altered.

SQL> SQL>
   THREAD#     GROUP#  SEQUENCE#      BYTES    MEMBERS ARCHIVED STATUS
---------- ---------- ---------- ---------- ---------- -------- --------
FIRST_CHANGE# FIRST_TIME
------------- --------------------------------------------------------------
         0          0        250    1048576          2 YES      INACTIVE
    183333 2022/12/01

         0          1        251    1048576          2 NO       CURRENT
    183526 2022/12/01

         0          2        249    1048576          2 YES      INACTIVE
    181923 2022/12/01

         0          3         -1     524288          2 NO       UNUSED
            0


4 rows selected.
```

```
SQL> SELECT * FROM V$LOGFILE;

    GROUP# STATUS  TYPE    MEMBER
---------- ------- ------ ----------------------------------------
         0         ONLINE /tibero/tbdata/tibero/redo01.redo
         0         ONLINE /tibero/tbdata/tibero/redo02.redo
         1         ONLINE /tibero/tbdata/tibero/redo11.redo
         1         ONLINE /tibero/tbdata/tibero/redo12.redo
         2         ONLINE /tibero/tbdata/tibero/redo21.redo
         2         ONLINE /tibero/tbdata/tibero/redo22.redo
         3         ONLINE /home/tibero/log/my_log3_1.log
         3         ONLINE /home/tibero/log/my_log3_2.log

8 rows selected.
```

그룹 3의 멤버 my_log3_3이 삭제된 것을 확인해볼 수 있다.

7. 로그 그룹 제거

```
-- 7. 로그 그룹 제거
ALTER DATABASE DROP LOGFILE GROUP 3;

SELECT * FROM V$LOG;
SELECT * FROM V$LOGFILE;
```

```
-- 7. 로그 그룹 제거
ALTER DATABASE DROP LOGFILE GROUP 3;

SELECT * FROM V$LOG;

Database altered.

SQL> SQL>
   THREAD#     GROUP#  SEQUENCE#      BYTES    MEMBERS ARCHIVED STATUS
---------- ---------- ---------- ---------- ---------- -------- --------
FIRST_CHANGE# FIRST_TIME
------------- --------------------------------------------------------------
         0          0        250    1048576          2 YES      INACTIVE
       183333 2022/12/01

         0          1        251    1048576          2 NO       CURRENT
       183526 2022/12/01

         0          2        249    1048576          2 YES      INACTIVE
       181923 2022/12/01


3 rows selected.
```

```
SQL> SELECT * FROM V$LOGFILE;

    GROUP# STATUS  TYPE    MEMBER
---------- ------- ------- ----------------------------------------
         0         ONLINE /tibero/tbdata/tibero/redo01.redo
         0         ONLINE /tibero/tbdata/tibero/redo02.redo
         1         ONLINE /tibero/tbdata/tibero/redo11.redo
         1         ONLINE /tibero/tbdata/tibero/redo12.redo
         2         ONLINE /tibero/tbdata/tibero/redo21.redo
         2         ONLINE /tibero/tbdata/tibero/redo22.redo

6 rows selected.
```

그룹 3이 삭제되어서 이제 조회되지 않는 것을 확인해볼 수 있다.

하지만 /home/tibero/log 에서는 파일들이 그대로 남아 있다.

```
[tibero@T1:/home/tibero/log]$ ls -l /home/tibero/log
total 1536
-rw------- 1 tibero dba 524288 Dec  1 14:04 my_log3_1.log
-rw------- 1 tibero dba 524288 Dec  1 14:04 my_log3_2.log
-rw------- 1 tibero dba 524288 Dec  1 14:08 my_log3_3.log
```

3.sh

```
-- 1. /home/tibero/log 폴더 생성
-- mkdir /home/tibero/log
-- ls -l /home/tibero


-- 2. 로그 그룹 정보 조회
SELECT * FROM V$LOG;


-- 3. 로그 파일 정보 조회
COL MEMBER FOR A40
SELECT * FROM V$LOGFILE;


-- 4. 로그 파일 생성
ALTER DATABASE ADD LOGFILE GROUP 3 (
        '/home/tibero/log/my_log3_1.log',
        '/home/tibero/log/my_log3_2.log') SIZE 512K;

SELECT * FROM V$LOG;
SELECT * FROM V$LOGFILE;


-- 5. 기존의 로그 그룹에 새로운 로그 멤버 추가
ALTER DATABASE ADD LOGFILE MEMBER
        '/home/tibero/log/my_log3_3.log' TO GROUP 3;

SELECT * FROM V$LOG;
SELECT * FROM V$LOGFILE;


-- 6. 로그 그룹 내의 하나의 로그 멤버 제거
ALTER DATABASE DROP LOGFILE MEMBER '/home/tibero/log/my_log3_3.log';

SELECT * FROM V$LOG;
SELECT * FROM V$LOGFILE;


-- 7. 로그 그룹 제거
ALTER DATABASE DROP LOGFILE GROUP 3;

SELECT * FROM V$LOG;
SELECT * FROM V$LOGFILE;
```

▼ 4. CREATE TABLE

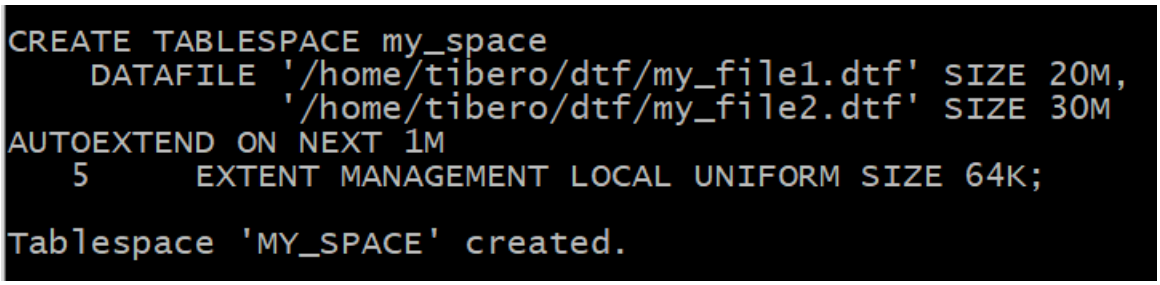
4.1 시나리오 수행

시나리오 내용
0. my_space 테이블 스페이스 생성
1. dept2 테이블 생성
2. emp2 테이블 생성
3. 제약조건에 맞지 않게 설정했을 때 오류가 뜨는 경우들
3.1. dept2에 아무 값이 없는데 emp 값을 넣으려는 경우
3.2. CHECK (salary >= 5000) 조건을 만족하지 않는 경우
3.3. 정상적으로 값이 insert되는 경우

4.2 시나리오 수행 결과

0. my_space 테이블 스페이스 생성

```
-- 0. my_space 테이블 스페이스 생성
CREATE TABLESPACE my_space
    DATAFILE '/home/tibero/dtf/my_file1.dtf' SIZE 20M,
             '/home/tibero/dtf/my_file2.dtf' SIZE 30M
    AUTOEXTEND ON NEXT 1M
    EXTENT MANAGEMENT LOCAL UNIFORM SIZE 64K;
```



1. dept2 테이블 생성

```
-- 1. dept2 테이블 생성
CREATE TABLE DEPT2
  (
      deptno    NUMBER PRIMARY KEY,
      deptname  VARCHAR(20)
  )
  TABLESPACE my_space
  PCTFREE 5 INITRANS 3;
```

```
CREATE TABLE DEPT2
  (
      deptno     NUMBER PRIMARY KEY,
      deptname   VARCHAR(20)
  )
  TABLESPACE my_space
   7   PCTFREE 5 INITRANS 3;

Table 'DEPT2' created.
```

2. emp2 테이블 생성

```
-- 2. emp2 테이블 생성
CREATE TABLE EMP2
  (
      empno     NUMBER PRIMARY KEY,
      ename     VARCHAR(16) NOT NULL,
      addr      VARCHAR(24),
      salary    NUMBER,
      deptno    NUMBER,
      CHECK (salary >= 5000),
      FOREIGN KEY (deptno) REFERENCES DEPT2(deptno)
  )
  TABLESPACE my_space
  PCTFREE 5 INITRANS 3;
```

```
CREATE TABLE EMP2
  (
      empno     NUMBER PRIMARY KEY,
      ename     VARCHAR(16) NOT NULL,
      addr      VARCHAR(24),
      salary    NUMBER,
      deptno    NUMBER,
      CHECK (salary >= 5000),
      FOREIGN KEY (deptno) REFERENCES DEPT2(deptno)
  )
  TABLESPACE my_space
  12   PCTFREE 5 INITRANS 3;

Table 'EMP2' created.
```

3. 제약조건에 맞지 않게 설정했을 때 오류가 뜨는 경우들

3.1. dept2에 아무 값이 없는데 emp 값을 넣으려는 경우

```
-- 3. 제약조건에 맞지 않게 설정했을 때 오류가 뜨는 경우들
--- 3.1. dept2에 아무 값이 없는데 emp 값을 넣으려는 경우
INSERT INTO emp2 VALUES(1, 'emp1', 'seoul', 5000, 10);
```

```
SQL> INSERT INTO emp2 VALUES(1, 'emp1', 'seoul', 5000, 10);
TBR-10008: INTEGRITY constraint violation ('SYS'.'_SYS_CON35700970'): primary key not found.
```

3.2. CHECK (salary >= 5000) 조건을 만족하지 않는 경우

```
--- 3.2. CHECK (salary >= 5000) 조건을 만족하지 않는 경우
INSERT INTO dept2 VALUES(10, 'dept1');
INSERT INTO emp2 VALUES(1, 'emp1', 'seoul', 3000, 10);
```

```
SQL> INSERT INTO dept2 VALUES(10, 'dept1');

1 row inserted.

SQL> INSERT INTO emp2 VALUES(1, 'emp1', 'seoul', 3000, 10);
TBR-10006: CHECK constraint violation ('SYS'.'_SYS_CON35600721').
```

3.3. 정상적으로 값이 insert되는 경우

```
--- 3.3. 정상적으로 값이 insert되는 경우
INSERT INTO emp2 VALUES(1, 'emp1', 'seoul', 5000, 10);
```

```
SQL> INSERT INTO emp2 VALUES(1, 'emp1', 'seoul', 5000, 10);

1 row inserted.
```

4.sh

```
-- 0. my_space 테이블 스페이스 생성
CREATE TABLESPACE my_space
    DATAFILE '/home/tibero/dtf/my_file1.dtf' SIZE 20M,
             '/home/tibero/dtf/my_file2.dtf' SIZE 30M
```

```
    AUTOEXTEND ON NEXT 1M
    EXTENT MANAGEMENT LOCAL UNIFORM SIZE 64K;

-- 1. dept2 테이블 생성
CREATE TABLE DEPT2
  (
      deptno    NUMBER PRIMARY KEY,
      deptname  VARCHAR(20)
  )
  TABLESPACE my_space
  PCTFREE 5 INITRANS 3;


-- 2. emp2 테이블 생성
CREATE TABLE EMP2
  (
      empno     NUMBER PRIMARY KEY,
      ename     VARCHAR(16) NOT NULL,
      addr      VARCHAR(24),
      salary    NUMBER,
      deptno    NUMBER,
      CHECK (salary >= 5000),
      FOREIGN KEY (deptno) REFERENCES DEPT2(deptno)
  )
  TABLESPACE my_space
  PCTFREE 5 INITRANS 3;


-- 3. 제약조건에 맞지 않게 설정했을 때 오류가 뜨는 경우들
--- 3.1. dept2에 아무 값이 없는데 emp 값을 넣으려는 경우
INSERT INTO emp2 VALUES(1, 'emp1', 'seoul', 5000, 10);

--- 3.2. CHECK (salary >= 5000) 조건을 만족하지 않는 경우
INSERT INTO dept2 VALUES(10, 'dept1');
INSERT INTO emp2 VALUES(1, 'emp1', 'seoul', 3000, 10);

--- 3.3. 정상적으로 값이 insert되는 경우
INSERT INTO emp2 VALUES(1, 'emp1', 'seoul', 5000, 10);
```

5. ALTER TABLE

5.1 설명

ALTER TABLE 문에서 포함되는 구성 요소

구성요소	설명
테이블의 이름	• 테이블의 이름은 최대 128자로 변경할 수 있다.

구성요소	설명
컬럼의 정의 변경	• 컬럼에 정의된 속성(디폴트 값, 제약조건 등)을 변경한다. • MODIFY 절을 이용하여 변경한다.
컬럼의 이름	• 컬럼 이름은 최대 30자로 변경할 수 있으며 RENAME COLUMN 절을 사용하여 변경한다.
디스크 블록의 파라미터	• 파라미터의 이름과 값을 지정한다.
제약조건	• 제약조건의 이름을 변경한다. • 제약조건을 추가하거나 제거한다. • 제약조건의 상태를 변경한다.
테이블 스페이스	• 테이블에 할당된 테이블 스페이스는 변경할 수 없다.
파티션	• 파티션을 추가하거나 제거한다.

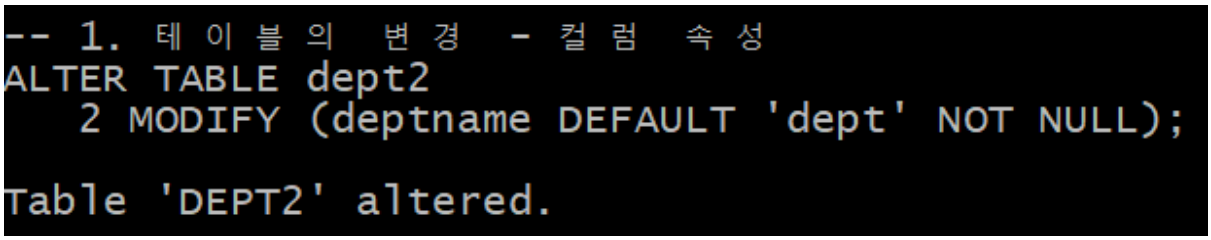
5.2 시나리오 수행

시나리오 내용
1. 테이블의 변경 - 컬럼 속성
2. 테이블의 변경 - 컬럼 이름
3. 테이블의 변경 - 디스크 블록의 파라미터
4. 테이블 제거 (참조 무결성 제약조건 제거 옵션으로)

5.3 시나리오 수행 결과

1. 테이블의 변경 - 컬럼 속성

```
-- 1. 테이블의 변경 - 컬럼 속성
ALTER TABLE dept2
  MODIFY (deptname DEFAULT 'dept' NOT NULL);
```



2. 테이블의 변경 - 컬럼 이름

```
-- 2. 테이블의 변경 - 컬럼 이름
ALTER TABLE dept2 RENAME COLUMN deptname TO dept_name;
```

```
-- 2. 테 이 블 의  변 경  - 컬 럼  이 름
SQL> ALTER TABLE dept2 RENAME COLUMN deptname TO dept_name;

Table 'DEPT2' altered.
```

3. 테이블의 변경 - 디스크 블록의 파라미터

```
-- 3. 테이블의 변경 - 디스크 블록의 파라미터
ALTER TABLE dept2 PCTFREE 10;
```

```
-- 3. 테 이 블 의  변 경  - 디 스 크  블 록 의  파 라 미 터
SQL> ALTER TABLE dept2 PCTFREE 10;

Table 'DEPT2' altered.
```

4. 테이블 제거 (참조 무결성 제약조건 제거 옵션으로)

```
-- 4. 테이블 제거 (참조 무결성 제약조건 제거 옵션으로)
DROP TABLE dept2 CASCADE CONSTRAINTS;
DROP TABLE emp2 CASCADE CONSTRAINTS;
```

```
-- 4. 테 이 블  제 거  (참 조  무 결 성  제 약 조 건  제 거  옵 션 으 로 )
DROP TABLE dept2 CASCADE CONSTRAINTS;

Table 'DEPT2' dropped.

SQL> DROP TABLE emp2 CASCADE CONSTRAINTS;

Table 'EMP2' dropped.
```

5.sh

```
-- 1. 테이블의 변경 - 컬럼 속성
ALTER TABLE dept2
  MODIFY (deptname DEFAULT 'dept' NOT NULL);

-- 2. 테이블의 변경 - 컬럼 이름
ALTER TABLE dept2 RENAME COLUMN deptname TO dept_name;

-- 3. 테이블의 변경 - 디스크 블록의 파라미터
ALTER TABLE dept2 PCTFREE 10;

-- 4. 테이블 제거 (참조 무결성 제약조건 제거 옵션으로)
DROP TABLE dept2 CASCADE CONSTRAINTS;
DROP TABLE emp2 CASCADE CONSTRAINTS;
```

---

6. 레인지 파티션

6.1 설명

레인지 파티션

- 테이블의 특정 컬럼의 범위를 기준으로 분할하는 방법
- 주로 일자로 관리하는 테이블에 적합
- 파티션 키를 중심으로 데이터가 어느 파티션에 존재하는지 파악할 수 있어 데이터 관리 측면에서 유리
- 하지만 테이블 파티션에 대한 데이터 분산은 파티션 키에 의존하므로 특정 파티션에 데이터가 편중되는 현상이 발생할 수 있음
- 레인지 파티션 테이블 생성 방법

```
CREATE TABLE <테이블명> (
    <컬럼1>,
    <컬럼2>,
    <컬럼3>
)
PARTITION BY RANGE(<컬럼>) (
    PARTITION <파티션명> VALUES LESS THAN (<값>),
    PARTITION <파티션명> VALUES LESS THAN (<값>),
```

```
    PARTITION <파티션명> VALUES LESS THAN (MAXVALUE))
ENABLE ROW MOVEMENT;
```

6.2 시나리오 수행

시나리오 내용
1. 레인지 파티션 테이블 생성
2. 데이터 60개 생성
3. DEPT_PART_R 테이블에 대해 통계 정보 수집
4. 데이터 적재 건수 및 파티션 테이블 생성 확인

6.3 시나리오 수행 결과

1. 레인지 파티션 테이블 생성

```
-- 1. 레인지 파티션 테이블 생성
CREATE TABLE DEPT_PART_R (
  deptno NUMBER(4),
  deptName VARCHAR(30),
  mgrNo NUMBER(6),
  locNo NUMBER(4)
  )
  PARTITION BY RANGE(deptno) (
    PARTITION part1 VALUES LESS THAN (11) ,
    PARTITION part2 VALUES LESS THAN (21) ,
    PARTITION part3 VALUES LESS THAN (31) ,
    PARTITION part4 VALUES LESS THAN (41) ,
    PARTITION partMAX VALUES LESS THAN (MAXVALUE))
  ENABLE ROW MOVEMENT;
```

```
-- 1. 레인지 파티션 테이블 생성
CREATE TABLE DEPT_PART_R (
deptno NUMBER(4),
deptName VARCHAR(30),
mgrNo NUMBER(6),
locNo NUMBER(4)
)
PARTITION BY RANGE(deptno) (
PARTITION part1 VALUES LESS THAN (11) ,
PARTITION part2 VALUES LESS THAN (21) ,
PARTITION part3 VALUES LESS THAN (31) ,
PARTITION part4 VALUES LESS THAN (41) ,
PARTITION partMAX VALUES LESS THAN (MAXVALUE))
  13 ENABLE ROW MOVEMENT;

Table 'DEPT_PART_R' created.
```

ENABLE ROW MOVEMENT : enable row movement 옵션이 걸린 상태에서 파티션 키 값에 대해 Update 를 수행하게 되면, 단순히 데이터블록에 들어있는 값을 변경하는게 아니라, 해당 블록에서 데이터를 Delete 하고, 적당한 다른 파티션에 위치한 적절한 블록에 새롭게 Insert 하는 방식으로 처리된다.

2. 데이터 60개 생성

```
-- 2. 데이터 60개 생성
INSERT INTO DEPT_PART_R
  SELECT LEVEL, CHR(65+MOD(LEVEL,26)), LEVEL, LEVEL FROM DUAL
  CONNECT BY LEVEL <= 60;
```

```
-- 2. 데이터 60개 생성
INSERT INTO DEPT_PART_R
SELECT LEVEL, CHR(65+MOD(LEVEL,26)), LEVEL, LEVEL FROM DUAL
   3 CONNECT BY LEVEL <= 60;

60 rows inserted.
```

3. DEPT_PART_R 테이블에 대해 통계 정보 수집

```
-- 3. DEPT_PART_R 테이블에 대해 통계 정보 수집
begin
```

```
  dbms_stats.gather_table_stats(OWNNAME=>'sys',
                                TABNAME=>'DEPT_PART_R',
                                ESTIMATE_PERCENT=>100);
end;
/
```

```
-- 3. DEPT_PART_R 테이블에 대해 통계 정보 수집
begin
dbms_stats.gather_table_stats(OWNNAME=>'sys',
TABNAME=>'DEPT_PART_R',
ESTIMATE_PERCENT=>100);
end;
   6 /

PSM completed.
```

- **OWNNAME** : 분석할 테이블 소유자
- **TABNAME** : 테이블 이름
- **ESTIMATE_PERCENT** : 분석할 row의 percentage 옵션. 100으로 지정하면 100%를 샘플링해서 통계정보를 생성하겠다는 의미.

4. 데이터 적재 건수 및 파티션 테이블 생성 확인

```
-- 4. 데이터 적재 건수 및 파티션 테이블 생성 확인
col TABLE_NAME format a15
col PARTITION_NAME format a15
col BOUND format a10
col NUM_ROWS format 999,999,999
SELECT TABLE_NAME, PARTITION_NAME, BOUND, NUM_ROWS
  FROM USER_TAB_PARTITIONS
  WHERE table_name = 'DEPT_PART_R';
```

```
-- 4. 데 이 터  적 재  건 수  및  파 티 션  테 이 블  생 성  확 인
col TABLE_NAME format a15
col PARTITION_NAME format a15
col BOUND format a10
col NUM_ROWS format 999,999,999
SELECT TABLE_NAME, PARTITION_NAME, BOUND, NUM_ROWS
FROM USER_TAB_PARTITIONS
   3 WHERE table_name = 'DEPT_PART_R';

TABLE_NAME      PARTITION_NAME  BOUND          NUM_ROWS
--------------- --------------- ---------- ------------
DEPT_PART_R     PART1           11                   10
DEPT_PART_R     PART2           21                   10
DEPT_PART_R     PART3           31                   10
DEPT_PART_R     PART4           41                   10
DEPT_PART_R     PARTMAX         MAXVALUE             20

5 rows selected.
```

USER_TAB_PARTITIONS : 접속계정의 테이블 파티션을 조회할 수 있는 뷰

part1 : 1 ~ 10 >> 10개 rows

part2 : 11 ~ 20 >> 10개 rows

part3 : 21 ~ 30 >> 10개 rows

part4 : 31 ~ 40 >> 10개 rows

partmax : 41 ~ 60 >> 20개 rows

6.sh

```
-- 1. 레인지 파티션 테이블 생성
CREATE TABLE DEPT_PART_R (
  deptno NUMBER(4),
  deptName VARCHAR(30),
  mgrNo NUMBER(6),
  locNo NUMBER(4)
  )
  PARTITION BY RANGE(deptno) (
    PARTITION part1 VALUES LESS THAN (11) ,
    PARTITION part2 VALUES LESS THAN (21) ,
    PARTITION part3 VALUES LESS THAN (31) ,
    PARTITION part4 VALUES LESS THAN (41) ,
    PARTITION partMAX VALUES LESS THAN (MAXVALUE))
  ENABLE ROW MOVEMENT;


-- 2. 데이터 60개 생성
INSERT INTO DEPT_PART_R
  SELECT LEVEL, CHR(65+MOD(LEVEL,26)), LEVEL, LEVEL FROM DUAL
```

```
  CONNECT BY LEVEL <= 60;

COMMIT;


-- 3. DEPT_PART_R 테이블에 대해 통계 정보 수집
begin
  dbms_stats.gather_table_stats(OWNNAME=>'sys',
                                TABNAME=>'DEPT_PART_R',
                                ESTIMATE_PERCENT=>100);
end;
/

-- 4. 데이터 적재 건수 및 파티션 테이블 생성 확인
col TABLE_NAME format a15
col PARTITION_NAME format a15
col BOUND format a10
col NUM_ROWS format 999,999,999
SELECT TABLE_NAME, PARTITION_NAME, BOUND, NUM_ROWS
  FROM USER_TAB_PARTITIONS
  WHERE table_name = 'DEPT_PART_R';
```

---

7. 해시 파티션

7.1 설명

해시 파티션

- 파티션 키 값에 해시 함수를 적용하여 데이터를 분할하는 방식
- 데이터 관리 목적보다는 여러 위치에 분산배치해서 Disk I/O 성능을 개선하기 위한 용도로 사용
- 해시 키를 중심으로 데이터가 분산되므로 내 데이터가 어느 파티션으로 들어갈 지, 어느 파티션에 어떤 데이터가 존재하는지 알 수 없으나 파티션별 데이터는 균등한 분포도를 보임
- 해시 파티션 생성 방법

```
CREATE TABLE <테이블명> (
  <컬럼1>,
  <컬럼2>,
  <컬럼3>
  )
  PARTITION BY HASH(<컬럼>) (
```

```
    PARTITION <파티션명>,
    PARTITION <파티션명>,
    PARTITION <파티션명>,
    PARTITION <파티션명>)
  ENABLE ROW MOVEMENT;
```

7.2 시나리오 수행

시나리오 내용
1. 해시 파티션 테이블 생성
2. 데이터 48개 생성
3. DEPT_PART_H 테이블에 대해 통계 정보 수집
4. 데이터 적재 건수 및 파티션 테이블 생성 확인

7.3 시나리오 수행 결과

1. 해시 파티션 테이블 생성

```
-- 1. 해시 파티션 테이블 생성
CREATE TABLE DEPT_PART_H (
    deptno NUMBER(4),
    deptname VARCHAR(30),
    mgrNO NUMBER(6),
    locNO NUMBER(4)
)
PARTITION BY HASH(deptno) (
    PARTITION HASH1,
    PARTITION HASH2,
    PARTITION HASH3,
    PARTITION HASH4)
  ENABLE ROW MOVEMENT;
```

```
CREATE TABLE DEPT_PART_H (
deptno NUMBER(4),
deptname VARCHAR(30),
mgrNO NUMBER(6),
locNO NUMBER(4)
)
PARTITION BY HASH(deptno) (
PARTITION HASH1,
PARTITION HASH2,
PARTITION HASH3,
PARTITION HASH4)
  12 ENABLE ROW MOVEMENT;

Table 'DEPT_PART_H' created.
```

2. 데이터 48개 생성

```
-- 2. 데이터 48개 생성
INSERT INTO DEPT_PART_H
  SELECT LEVEL, CHR(65+MOD(LEVEL,26)), LEVEL, LEVEL FROM DUAL
  CONNECT BY LEVEL <= 48;

COMMIT;
```

```
-- 2. 데이터 48개 생성
INSERT INTO DEPT_PART_H
SELECT LEVEL, CHR(65+MOD(LEVEL,26)), LEVEL, LEVEL FROM DUAL
CONNECT BY LEVEL <= 48;

48 rows inserted.

SQL>
SQL> COMMIT;

Commit completed.
```

3. 통계 정보 수집

```
-- 3. 통계 정보 수집
begin
  dbms_stats.gather_table_stats(OWNNAME=>'sys',
```

```
                                TABNAME=>'DEPT_PART_H',
                                ESTIMATE_PERCENT=>100);
end;
/
```

```
-- 3. 통 계  정 보  수 집
begin
dbms_stats.gather_table_stats(OWNNAME=>'sys',
TABNAME=>'DEPT_PART_H',
ESTIMATE_PERCENT=>100);
end;
   6 /

PSM completed.
```

4. 데이터 적재 건수 및 파티션 테이블 생성 확인

```
-- 4. 데이터 적재 건수 및 파티션 테이블 생성 확인
col TABLE_NAME format a15
col PARTITION_NAME format a15
col BOUND format a10
col NUM_ROWS format 999,999,999
SELECT TABLE_NAME, PARTITION_NAME, BOUND, NUM_ROWS
  FROM USER_TAB_PARTITIONS
  WHERE TABLE_NAME = 'DEPT_PART_H';
```

```
-- 4. 데 이 터  적 재  건 수  및  파 티 션  테 이 블  생 성  확 인
col TABLE_NAME format a15
col PARTITION_NAME format a15
col BOUND format a10
col NUM_ROWS format 999,999,999
SELECT TABLE_NAME, PARTITION_NAME, BOUND, NUM_ROWS
FROM USER_TAB_PARTITIONS
   3 WHERE TABLE_NAME = 'DEPT_PART_H';

TABLE_NAME      PARTITION_NAME  BOUND          NUM_ROWS
--------------- --------------- ---------- ------------
DEPT_PART_H     HASH1                              14
DEPT_PART_H     HASH2                              13
DEPT_PART_H     HASH3                              10
DEPT_PART_H     HASH4                              11

4 rows selected.
```

7.sh

```
-- 1. 해시 테이블 생성
CREATE TABLE DEPT_PART_H (
    deptno NUMBER(4),
    deptname VARCHAR(30),
    mgrNO NUMBER(6),
    locNO NUMBER(4)
)
PARTITION BY HASH(deptno) (
    PARTITION HASH1,
    PARTITION HASH2,
    PARTITION HASH3,
    PARTITION HASH4)
  ENABLE ROW MOVEMENT;

-- 2. 데이터 48개 생성
INSERT INTO DEPT_PART_H
  SELECT LEVEL, CHR(65+MOD(LEVEL,26)), LEVEL, LEVEL FROM DUAL
  CONNECT BY LEVEL <= 48;

COMMIT;


-- 3. 통계 정보 수집
begin
  dbms_stats.gather_table_stats(OWNNAME=>'sys',
                                TABNAME=>'DEPT_PART_H',
                                ESTIMATE_PERCENT=>100);
end;
/


-- 4. 데이터 적재 건수 및 파티션 테이블 생성 확인
col TABLE_NAME format a15
col PARTITION_NAME format a15
col BOUND format a10
col NUM_ROWS format 999,999,999
SELECT TABLE_NAME, PARTITION_NAME, BOUND, NUM_ROWS
  FROM USER_TAB_PARTITIONS
  WHERE TABLE_NAME = 'DEPT_PART_H';
```

---

8. 리스트 파티션

8.1 설명

리스트 파티션

- 파티션 키의 Value List를 기준으로 분할하는 방법
- 여러 개의 컬럼으로 파티션할 수 있는 레인지 파티션과 달리
  하나의 컬럼으로만 파티션 할 수 있음
- 별개의 각 값들이 어떤 파티션에 속해야 할지 명시적으로 지정해야 함
- 순차적이지 않고 서로 연관성이 없는 데이터를 그룹핑 지어 관리하기에 용이함
- 리스트 파티션 생성

```
CREATE TABLE <테이블명> (
  <컬럼1>,
  <컬럼2>,
  <컬럼3>
)
PARTITION BY LIST(<컬럼>) (
  PARTITION <파티션명> VALUES (<값>),
  PARTITION <파티션명> VALUES (<값>),
  PARTITION <파티션명> VALUES (<값>),
  PARTITION <파티션명> VALUES (<값>))
ENABLE ROW MOVEMENT;
```

8.2 시나리오 수행

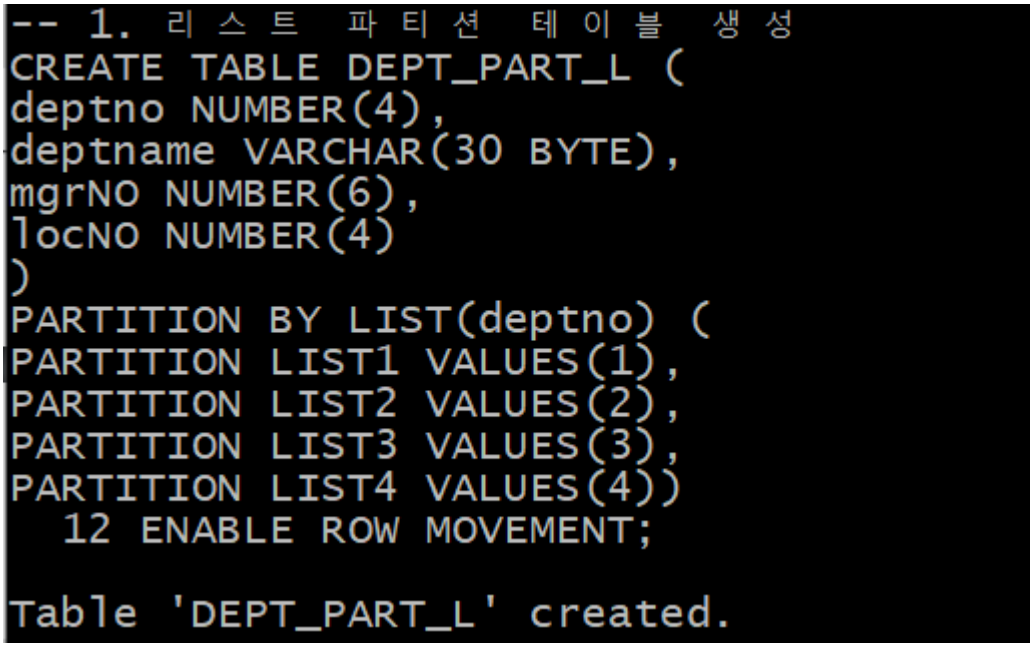
시나리오 내용
1. 리스트 파티션 테이블 생성
2. 데이터 4개 생성
3. DEPT_PART_L 테이블에 대해 통계 정보 수집
4. 데이터 적재 건수 및 파티션 테이블 생성 확인

8.3 시나리오 수행 결과

1. 리스트 파티션 테이블 생성

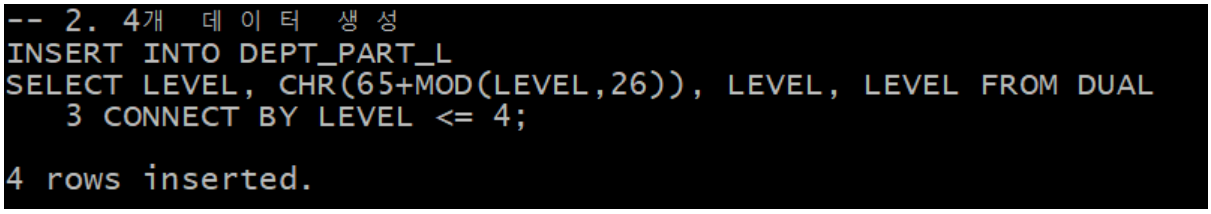
```
-- 1. 리스트 파티션 테이블 생성
CREATE TABLE DEPT_PART_L (
  deptno NUMBER(4),
  deptname VARCHAR(30 BYTE),
  mgrNO NUMBER(6),
  locNO NUMBER(4)
```

```
)
PARTITION BY LIST(deptno) (
  PARTITION LIST1 VALUES(1),
  PARTITION LIST2 VALUES(2),
  PARTITION LIST3 VALUES(3),
  PARTITION LIST4 VALUES(4))
  ENABLE ROW MOVEMENT;
```



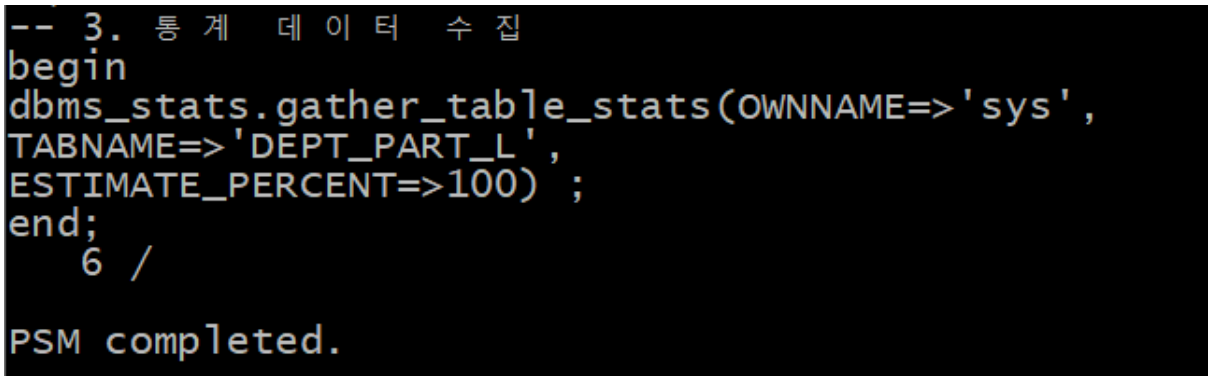
2. 데이터 4개 생성

```
-- 2. 4개 데이터 생성
INSERT INTO DEPT_PART_L
  SELECT LEVEL, CHR(65+MOD(LEVEL,26)), LEVEL, LEVEL FROM DUAL
  CONNECT BY LEVEL <= 4;
```



3. DEPT_PART_L 테이블에 대해 통계 정보 수집

```
-- 3. 통계 데이터 수집
begin
  dbms_stats.gather_table_stats(OWNNAME=>'sys',
                                TABNAME=>'DEPT_PART_L',
                                ESTIMATE_PERCENT=>100) ;
end;
/
```



4. 데이터 적재 건수 및 파티션 테이블 생성 확인

```
-- 4. 데이터 적재 건수 및 파티션 테이블 생성 확인
col TABLE_NAME format a15
col PARTITION_NAME format a15
col BOUND format a10
col NUM_ROWS format 999,999,999
SELECT TABLE_NAME, PARTITION_NAME, BOUND,
  NUM_ROWS FROM USER_TAB_PARTITIONS
  WHERE TABLE_NAME = 'DEPT_PART_L';
```

```
-- 4. 데 이 터  적 재  건 수  및  파 티 션  테 이 블  생 성  확 인
col TABLE_NAME format a15
col PARTITION_NAME format a15
col BOUND format a10
col NUM_ROWS format 999,999,999
SELECT TABLE_NAME, PARTITION_NAME, BOUND,
NUM_ROWS FROM USER_TAB_PARTITIONS
   3 WHERE TABLE_NAME = 'DEPT_PART_L';

TABLE_NAME      PARTITION_NAME  BOUND          NUM_ROWS
--------------- --------------- ---------- ------------
DEPT_PART_L     LIST1           1                     1
DEPT_PART_L     LIST2           2                     1
DEPT_PART_L     LIST3           3                     1
DEPT_PART_L     LIST4           4                     1

4 rows selected.
```

8.sh

```
-- 1. 리스트 파티션 테이블 생성
CREATE TABLE DEPT_PART_L (
  deptno NUMBER(4),
  deptname VARCHAR(30 BYTE),
  mgrNO NUMBER(6),
  locNO NUMBER(4)
)
PARTITION BY LIST(deptno) (
  PARTITION LIST1 VALUES(1),
  PARTITION LIST2 VALUES(2),
  PARTITION LIST3 VALUES(3),
  PARTITION LIST4 VALUES(4))
  ENABLE ROW MOVEMENT;


-- 2. 4개 데이터 생성
INSERT INTO DEPT_PART_L
  SELECT LEVEL, CHR(65+MOD(LEVEL,26)), LEVEL, LEVEL FROM DUAL
  CONNECT BY LEVEL <= 4;


-- 3. 통계 데이터 수집
begin
  dbms_stats.gather_table_stats(OWNNAME=>'sys',
                                TABNAME=>'DEPT_PART_L',
                                ESTIMATE_PERCENT=>100) ;
end;
/
```

```
-- 4. 데이터 적재 건수 및 파티션 테이블 생성 확인
col TABLE_NAME format a15
col PARTITION_NAME format a15
col BOUND format a10
col NUM_ROWS format 999,999,999
SELECT TABLE_NAME, PARTITION_NAME, BOUND,
  NUM_ROWS FROM USER_TAB_PARTITIONS
  WHERE TABLE_NAME = 'DEPT_PART_L';
```

---

9. 복합 파티션

9.1 설명

복합 파티션

- 분할한 각각의 파티션을 해시 또는 리스트 방식을 적용하여 다시 한번 서브 파티션으로 분할하는 방법
- 레인지-해시 방식을 사용할 경우 데이터 관리가 용이한 레인지 파티션과 성능 향상에 유리한 해시 파티션의 장점을 동시에 활용

9.2 시나리오 수행

시나리오 내용
1. 레인지 - 해시 파티션 테이블 생성
2. 생성 확인

9.3 시나리오 수행 결과

1. 레인지 - 해시 파티션 테이블 생성

```
-- 1. 레인지 - 해시 파티션 테이블 생성
CREATE TABLE DEPT_PART_RH (
    deptno NUMBER(4),
    deptname VARCHAR(30),
    mgrNO NUMBER(6),
    locNO NUMBER(4)
  )
  PCTFREE 10
  INITRANS 2
  PARTITION BY RANGE(deptno)
```

```
   SUBPARTITION BY HASH(mgrNO) (
       PARTITION R_HASH1 VALUES LESS THAN (10) SUBPARTITIONS 4,
       PARTITION R_HASH2 VALUES LESS THAN (20) SUBPARTITIONS 4,
       PARTITION R_HASH3 VALUES LESS THAN (30) SUBPARTITIONS 4,
       PARTITION R_HASH4 VALUES LESS THAN (40) SUBPARTITIONS 4,
       PARTITION R_HASHMAX VALUES LESS THAN (MAXVALUE) SUBPARTITIONS 4
   )
   ENABLE ROW MOVEMENT;
```

```
-- 1. 레 인 지  - 해 시  파 티 션  테 이 블  생 성
CREATE TABLE DEPT_PART_RH (
deptno NUMBER(4),
deptname VARCHAR(30),
mgrNO NUMBER(6),
locNO NUMBER(4)
)
PCTFREE 10
INITRANS 2
PARTITION BY RANGE(deptno)
SUBPARTITION BY HASH(mgrNO) (
PARTITION R_HASH1 VALUES LESS THAN (10) SUBPARTITIONS 4,
PARTITION R_HASH2 VALUES LESS THAN (20) SUBPARTITIONS 4,
PARTITION R_HASH3 VALUES LESS THAN (30) SUBPARTITIONS 4,
PARTITION R_HASH4 VALUES LESS THAN (40) SUBPARTITIONS 4,
PARTITION R_HASHMAX VALUES LESS THAN (MAXVALUE) SUBPARTITIONS 4
)
  17 ENABLE ROW MOVEMENT;

Table 'DEPT_PART_RH' created.
```

deptno 컬럼을 파티션 키로 하는 레인지 파티션을 메인 파티션으로 만들고,

그 파티션들을 다시 mgrNO 컬럼을 서브파티션 키로 하는 해시 서브 파티션으로 만들어 복합파티션을 구성하였다.

2. 생성 확인

```
-- 2. 생성 확인
col TABLE_NAME format a15
col PARTITION_NAME format a15
col SUBPARTITION_NO format 999,999,999
SELECT TABLE_NAME, PARTITION_NAME, SUBPARTITION_NO
  FROM USER_TAB_SUBPARTITIONS
  WHERE TABLE_NAME = 'DEPT_PART_RH';
```

```
-- 2. 생 성  확 인
col TABLE_NAME format a15
col PARTITION_NAME format a15
col SUBPARTITION_NO format 999,999,999
SELECT TABLE_NAME, PARTITION_NAME, SUBPARTITION_NO
FROM USER_TAB_SUBPARTITIONS
   3 WHERE TABLE_NAME = 'DEPT_PART_RH';

TABLE_NAME      PARTITION_NAME  SUBPARTITION
--------------- --------------- ------------
DEPT_PART_RH    R_HASH1                    1
DEPT_PART_RH    R_HASH1                    2
DEPT_PART_RH    R_HASH1                    3
DEPT_PART_RH    R_HASH1                    4
DEPT_PART_RH    R_HASH2                    1
DEPT_PART_RH    R_HASH2                    2
DEPT_PART_RH    R_HASH2                    3
DEPT_PART_RH    R_HASH2                    4
DEPT_PART_RH    R_HASH3                    1
DEPT_PART_RH    R_HASH3                    2
DEPT_PART_RH    R_HASH3                    3
DEPT_PART_RH    R_HASH3                    4
DEPT_PART_RH    R_HASH4                    1
DEPT_PART_RH    R_HASH4                    2
DEPT_PART_RH    R_HASH4                    3
DEPT_PART_RH    R_HASH4                    4
DEPT_PART_RH    R_HASHMAX                  1
DEPT_PART_RH    R_HASHMAX                  2
DEPT_PART_RH    R_HASHMAX                  3
DEPT_PART_RH    R_HASHMAX                  4

20 rows selected.
```

9.sh

```
-- 1. 레인지 - 해시 파티션 테이블 생성
CREATE TABLE DEPT_PART_RH (
    deptno NUMBER(4),
    deptname VARCHAR(30),
    mrgNO NUMBER(6),
    locNO NUMBER(4)
  )
  PCTFREE 10
  INITRANS 2
  PARTITION BY RANGE(deptno)
  SUBPARTITION BY HASH(mgrNO) (
      PARTITION R_HASH1 VALUES LESS THAN (10) SUBPARTITIONS 4,
      PARTITION R_HASH2 VALUES LESS THAN (20) SUBPARTITIONS 4,
```

```
      PARTITION R_HASH3 VALUES LESS THAN (30) SUBPARTITIONS 4,
      PARTITION R_HASH4 VALUES LESS THAN (40) SUBPARTITIONS 4,
      PARTITION R_HASHMAX VALUES LESS THAN (MAXVALUE) SUBPARTITIONS 4
  )
  ENABLE ROW MOVEMENT;


-- 2. 생성 확인
col TABLE_NAME format a15
col PARTITION_NAME format a15
col SUBPARTITION_NO format 999,999,999
SELECT TABLE_NAME, PARTITION_NAME, SUBPARTITION_NO
  FROM USER_TAB_SUBPARTITIONS
  WHERE TABLE_NAME = 'DEPT_PART_RH';
```

---

10. 테이블 압축

10.1 설명

테이블 압축

- 테이블에 대해 중복된 컬럼 값을 압축하여 저장공간을 절약
- 블록에 존재하는 중복된 컬럼 값을 한번만 저장함으로써 압축을 수행
- 압축을 수행하면 디스크 공간을 절약 할 수 있지만 압축을 위해 CPU를 더 많이 소모함. 테이블에 DML이 많은 경우 점점 더 압축 효율이 낮아지게 됨.

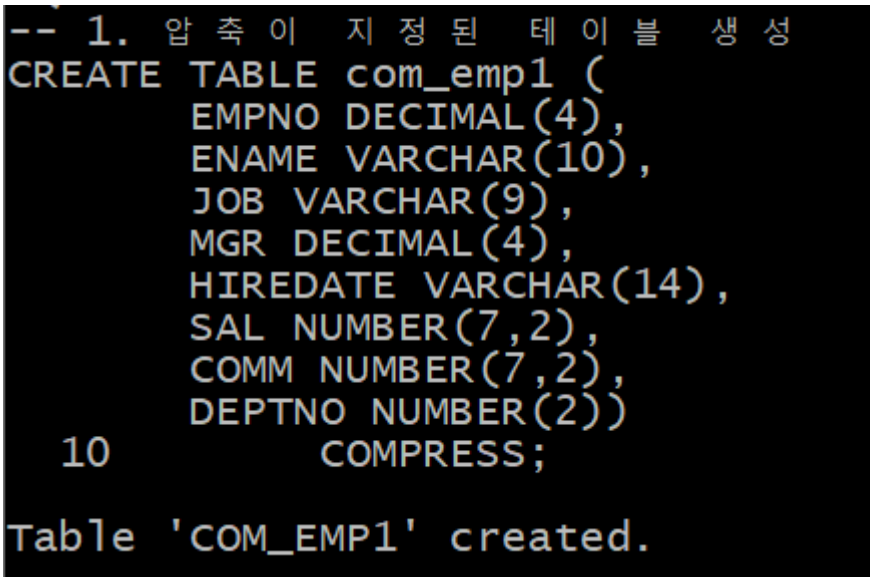
10.2 시나리오 수행

시나리오 내용
1. 압축이 지정된 테이블 생성
2. 파티션 별 압축을 지정하는 테이블 생성
3. 테이블 압축 상태 확인
4. 기존 파티션 압축 해제
5. 기존 테이블 압축 해제
6. 기존 테이블 압축

10.3 시나리오 수행 결과

1. 압축이 지정된 테이블 생성

```
-- 1. 압축이 지정된 테이블 생성
CREATE TABLE com_emp1 (
       EMPNO DECIMAL(4),
       ENAME VARCHAR(10),
       JOB VARCHAR(9),
       MGR DECIMAL(4),
       HIREDATE VARCHAR(14),
       SAL NUMBER(7,2),
       COMM NUMBER(7,2),
       DEPTNO NUMBER(2))
       COMPRESS;
```



2. 파티션 별 압축을 지정하는 테이블 생성

```
-- 2. 파티션 별 압축을 지정하는 테이블 생성
CREATE TABLE com_emp2 (
       empno DECIMAL(4),
       ename VARCHAR(10),
       job VARCHAR(9),
       mgr DECIMAL(4),
       hiredate VARCHAR(14),
       sal NUMBER(7,2),
       comm NUMBER(7,2),
       deptno NUMBER(2))
       COMPRESS
  PARTITION BY RANGE(empno)
     ( PARTITION EMP_PART1 VALUES LESS THAN(500),
       PARTITION EMP_PART2 VALUES LESS THAN(1000) NOCOMPRESS,
       PARTITION EMP_PART3 VALUES LESS THAN(1500),
       PARTITION EMP_PART4 VALUES LESS THAN(2000) NOCOMPRESS,
       PARTITION EMP_PART5 VALUES LESS THAN(MAXVALUE));
```

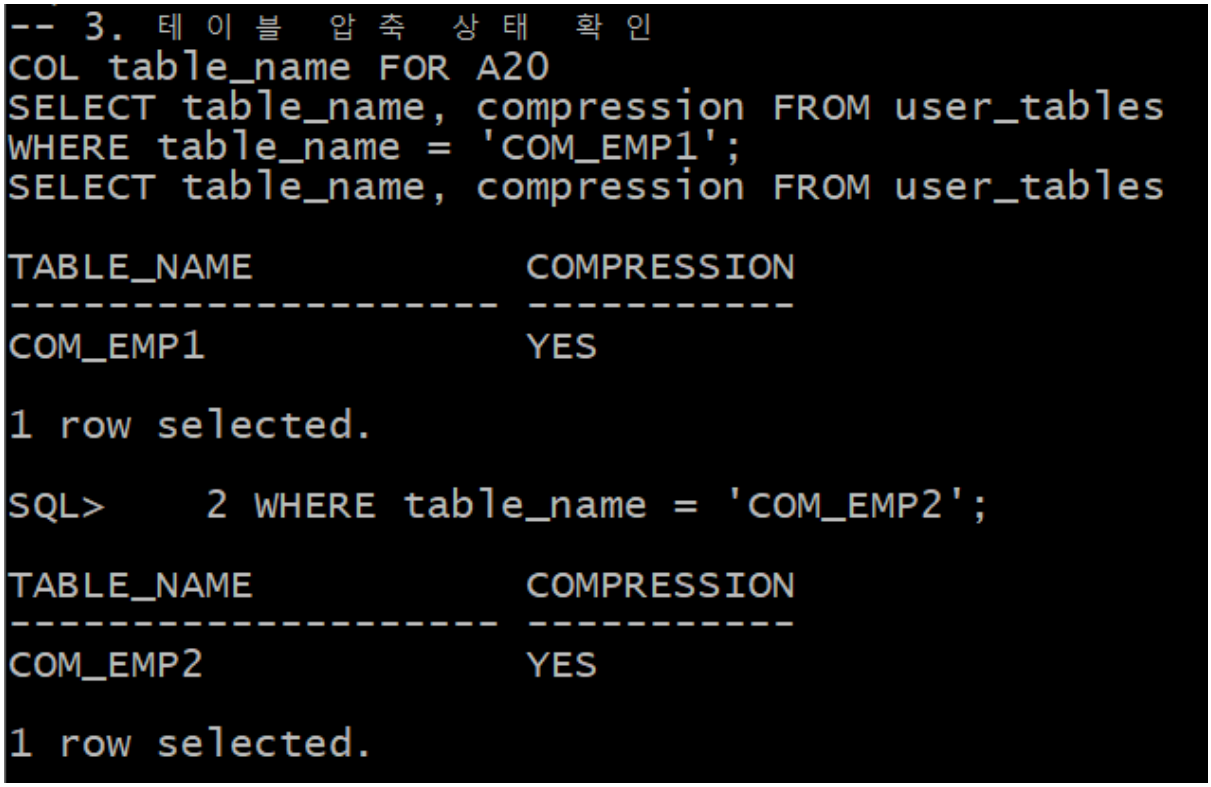
```
-- 2. 파 티 션  별  압 축 을  지 정 하 는  테 이 블  생 성
CREATE TABLE com_emp2 (
       empno DECIMAL(4),
       ename VARCHAR(10),
       job VARCHAR(9),
       mgr DECIMAL(4),
       hiredate VARCHAR(14),
       sal NUMBER(7,2),
       comm NUMBER(7,2),
       deptno NUMBER(2))
       COMPRESS
PARTITION BY RANGE(empno)
   ( PARTITION EMP_PART1 VALUES LESS THAN(500),
     PARTITION EMP_PART2 VALUES LESS THAN(1000) NOCOMPRESS,
     PARTITION EMP_PART3 VALUES LESS THAN(1500),
     PARTITION EMP_PART4 VALUES LESS THAN(2000) NOCOMPRESS,
  16       PARTITION EMP_PART5 VALUES LESS THAN(MAXVALUE));

Table 'COM_EMP2' created.
```

파티션 EMP_PART1, 3, 5에 대해서는 압축을 수행하고,

EMP_PART2, 4에 대해서는 압축을 수행하지 않도록 설정하여 테이블을 생성한다.

3. 테이블 압축 상태 확인

```
-- 3. 테이블 압축 상태 확인
COL table_name FOR A20
SELECT table_name, compression FROM user_tables
  WHERE table_name = 'COM_EMP1';
SELECT table_name, compression FROM user_tables
  WHERE table_name = 'COM_EMP2';
```



compression 컬럼의 값이 'YES'인 경우 추가적인 DML에 대해 압축을 수행하는 것이다.

4. 기존 파티션 압축 해제

```
-- 4. 기존 파티션 압축 해제
ALTER TABLE com_emp2 MOVE PARTITION EMP_PART1 NOCOMPRESS;
ALTER TABLE com_emp2 MOVE PARTITION EMP_PART3 NOCOMPRESS;
ALTER TABLE com_emp2 MOVE PARTITION EMP_PART5 NOCOMPRESS;

SELECT table_name, compression FROM user_tables
  WHERE table_name = 'COM_EMP2';
```

```
-- 4. 기존 파티션 압축 해제
ALTER TABLE com_emp2 MOVE PARTITION EMP_PART1 NOCOMPRESS;
ALTER TABLE com_emp2 MOVE PARTITION EMP_PART3 NOCOMPRESS;

Table 'COM_EMP2' altered.

SQL> ALTER TABLE com_emp2 MOVE PARTITION EMP_PART5 NOCOMPRESS;

Table 'COM_EMP2' altered.

SQL> SELECT table_name, compression FROM user_tables

Table 'COM_EMP2' altered.

SQL>    2 WHERE table_name = 'COM_EMP2';

TABLE_NAME           COMPRESSION
-------------------- -----------
COM_EMP2             YES

1 row selected.
```

모든 파티션들의 압축을 해제했는데도 com_emp2 테이블의 compression이 YES로 조회된다.

5. 기존 테이블 압축 해제

```
-- 5. 기존 테이블 압축 해제
ALTER TABLE com_emp1 MOVE NOCOMPRESS;
SELECT table_name, compression FROM user_tables
  WHERE table_name = 'COM_EMP1';
```

```
SQL> ALTER TABLE com_emp1 MOVE NOCOMPRESS;

Table 'COM_EMP1' altered.

SELECT table_name, compression FROM user_tables
   2 WHERE table_name = 'COM_EMP1';

TABLE_NAME           COMPRESSION
-------------------- -----------
COM_EMP1             YES

1 row selected.
```

압축을 해제했는데도 com_emp1 테이블의 compression이 YES로 조회된다.

10.sh

```
-- 1. 압축이 지정된 테이블 생성
CREATE TABLE com_emp1 (
       EMPNO DECIMAL(4),
       ENAME VARCHAR(10),
       JOB VARCHAR(9),
       MGR DECIMAL(4),
       HIREDATE VARCHAR(14),
       SAL NUMBER(7,2),
       COMM NUMBER(7,2),
       DEPTNO NUMBER(2))
       COMPRESS;

-- 2. 파티션 별 압축을 지정하는 테이블 생성
CREATE TABLE com_emp2 (
       empno DECIMAL(4),
       ename VARCHAR(10),
       job VARCHAR(9),
       mgr DECIMAL(4),
       hiredate VARCHAR(14),
       sal NUMBER(7,2),
       comm NUMBER(7,2),
       deptno NUMBER(2))
       COMPRESS
  PARTITION BY RANGE(empno)
     ( PARTITION EMP_PART1 VALUES LESS THAN(500),
       PARTITION EMP_PART2 VALUES LESS THAN(1000) NOCOMPRESS,
       PARTITION EMP_PART3 VALUES LESS THAN(1500),
       PARTITION EMP_PART4 VALUES LESS THAN(2000) NOCOMPRESS,
       PARTITION EMP_PART5 VALUES LESS THAN(MAXVALUE));

-- 3. 테이블 압축 상태 확인
COL table_name FOR A20
SELECT table_name, compression FROM user_tables
  WHERE table_name = 'COM_EMP1';
SELECT table_name, compression FROM user_tables
  WHERE table_name = 'COM_EMP2';


-- 4. 기존 파티션 압축 해제
ALTER TABLE com_emp2 MOVE PARTITION EMP_PART1 NOCOMPRESS;
ALTER TABLE com_emp2 MOVE PARTITION EMP_PART3 NOCOMPRESS;
ALTER TABLE com_emp2 MOVE PARTITION EMP_PART5 NOCOMPRESS;

SELECT table_name, compression FROM user_tables
  WHERE table_name = 'COM_EMP2';


-- 5. 기존 테이블 압축 해제
ALTER TABLE com_emp1 MOVE NOCOMPRESS;
SELECT table_name, compression FROM user_tables
  WHERE table_name = 'COM_EMP1';
```

```
-- 6. 기존 테이블 압축
ALTER TABLE com_emp1 MOVE COMPRESS;
SELECT table_name, compression FROM user_tables
  WHERE table_name = 'COM_EMP1';
```