프로젝트 보고서_1주차

1. 사용자관리

1.1 사용자 계정 생성, 패스워드 생성, 권한 설정

1.1.1 설명

USER

- USER는 현재 세션을 생성한 사용자의 이름을 반환하는 함수
- 문법

- 예제

```
SQL> select user from dual;

USER
--------------------------------------------------------------------------------
SYS

1 row selected.
```

CREATE USER

- 특권
  - 사용자를 생성하기 위해서는 CREATE USER 시스템 특권이 필요
  - CREATE USER로 생성한 사용자는 처음에 아무런 특권이 없는 상태이다. 따라서 생성한 사용자로 접속하기 위해서는 CREATE SESSION 시스템 특권이 필요하고, 그 밖에도 해당 시스템 특권이 필요
- 구성요소
  - create_user

구성요소	설명
username	- 생성할 사용자의 이름 - 사용자의 이름은 VARCHAR 데이터 타입으로 저장되고 길이는 최대 30자까지 가능 - 사용자의 이름은 전체 데이터베이스의 다른 사용자 또는 역할의 이름과 중복되면 안 된다.
IDENTIFIED BY password	- 생성한 사용자를 인증하기 위한 패스워드 - 패스워드는 문자열로서 임의의 문자를 사용가능 - 특수 문자를 쓰기 위해서는 따옴표(' ' 또는 " ")를 사용해서 패스워드를 감싸야 함 - 대소문자가 구분되는 패스워드를 이용하고 싶다면 작은따옴표(' ')로 패스워드를 감싸야 함
create_user_clause	- 생성한 사용자의 기본값을 설정하기 위한 부분으로 생략 가능

  - create_user_clause

구성요소	설명

DEFAULT TABLESPACE	- 생성한 사용자가 사용할 디폴트 테이블 스페이스를 지정 - 해당 테이블 스페이스는 미리 생성되어야함 - DEFAULT TABLESPACE 부분을 생략하면, 자동으로 시스템 테이블 스페이스를 사용하게 됨
PROFILE	사용자의 접속 및 패스워드 정책에 대한 프로파일을 지정
PASSWORD EXPIRE	사용자의 패스워드를 사용기간 만료 상태로 생성하여, 사용자가 처음 접속할 때 원하는 패스워드를 입력할 수 있도록 한다.
ACCOUNT LOCK	- 사용자를 잠금 상태로 생성 - 생략하면 사용자를 잠금해제 상태로 생성
ACCOUNT UNLOCK	사용자를 잠금해제 상태로 생성한다. (기본값)

- 예제
- create_user
  - 다음은 사용자를 생성하고, **DEFAULT TABLESPACE**문장을 사용해 디폴트 테이블 스페이스를 설정하는 예

```
SQL> create tablespace t1 datafile 't1.dbf' size 10m
   2 extent management local uniform size 256k;

Tablespace 'T1' created.

SQL> create user u1 identified by 'p1'
   2 default tablespace t1;

User 'U1' created.                ^

SQL> select username, default_tablespace
   2 from dba_users where username='U1';

USERNAME
--------------------------------------------------------------------------------
DEFAULT_TABLESPACE
--------------------------------------------------------------------------------
U1
T1


1 row selected.
```

- create_user_clause
  - 다음은 사용자를 생성하고, **PASSWORD EXPIRE**를 사용해 패스워드를 만료된 상태로 변경하는 예

```
SQL> create user u2 identified by 'p2'
   2 password expire;

User 'U2' created.

SQL> grant create session to u2;

Granted.

SQL> conn u2/p2
TBR-17002: Password has expired.
```

```
Enter new password:
Confirm new password:
Password changed successfully.
Connected to Tibero.
```

- 다음은 사용자를 생성하고, **ACCOUNT LOCK**을 사용해 사용자를 잠금 상태로 변경하는 예

```
SQL> conn sys/tibero
Connected.

SQL> create user u3 identified by 'p3'
   2 account lock;

User 'U3' created.

SQL> select username, account_status
   2 from dba_users where username='U3';

USERNAME
--------------------------------------------------------------------------------
ACCOUNT_STATUS
--------------------------------
U3
LOCKED


1 row selected.

SQL> grant create session to u3;

Granted.

SQL> conn u3/p3
TBR-17006: Account is locked.

No longer connected to server.
```

GRANT

- 시스템 특권과 역할, 스키마 객체 특권을 일반 사용자나 역할, 공유 사용자에게 부여
- 특권
  - GRANT를 이용하여 특권 또는 권한을 부여하는 데에는 해당 특권이나 권한에 따라 다른 특권이 필요

구성요소	설명
시스템 특권	- 사용자 자신이 사용할 수 있게 부여받은 시스템 특권에 한해 가능 - 단, 자신이 사용할 수 있는 시스템 특권을 모두 부여할 수 있는 것은 아님 - 사용자 자신이 사용 가능하게 부여받을 때 WITH ADMIN OPTION을 사용하여 부여 받았어야함 - GRANT ANY PRIVILEGE 시스템 특권을 가지고 있는 경우라면 자신이 부여받지 않은 시스템 특권도 모두 부여 가능
역할	- 시스템 특권과 마찬가지로 WITH ADMIN OPTION을 사용하여 부여받은 역할이거나 자신이 만든 역할이어야 부여가 가능

	- GRANT ANY ROLE 시스템 특권을 가지고 있는 경우라면 자신이 부여받지 않은 역할이라도 모두 부여 가능
스키마 객체 특권	- 스키마 객체 특권을 부여하는 것은 자신이 소유한 객체이거나 WITH GRANT OPTION을 사용하여 부여받은 스키마 객체 특권에 한해 가능 - GRANT ANY OBJECT PRIVILEGE 시스템 특권을 부여 받았을 경우라면 자신이 부여받지 않은 스키마 객체 특권도 모두 부여 가능

- 구성요소
  - grant

구성요소	설명
grant_sysprivs	시스템 특권이나 역할을 부여한다.
grant_objprivs	스키마 객체 특권을 부여한다.

  - grant_sysprivs

구성요소	설명
system_privilege	시스템 특권을 정의
role_name	역할을 정의
ALL PRIVILEGES	- 현재 사용가능한 모든 시스템 특권을 부여 - GRANT ANY PRIVILEGE 시스템 특권을 가지고 있는 사용자만이 ALL PRIVILEGE 절을 사용 가능
TO grantee_clause	- 특권이나 역할을 부여받을 대상이 됨 - 대상이 될 수 있는 것은 일반 사용자나 역할, 공유 사용자 3종류 - 공유 사용자에게 특권 또는 역할을 부여하면, 모든 사용자에게 부여한 것과 동일한 효과를 갖게 되므로 주의해야함
WITH ADMIN OPTION	- 시스템 특권과 역할은 해당 특권 또는 역할을 이용할 수 있는 사용 특권과 해당 특권 또는 역할을 남에게 부여할 수 있는 관리 특권으로 나뉨 - WITH ADMIN OPTION을 사용하여 시스템 특권 또는 역할을 부여할 경우에는, 해당 특권 또는 역할을 사용할 수 있는 사용 특권과 함께 남에게 부여할 수 있는 관리 특권까지 같이 부여하게됨 - WITH ADMIN OPTION을 사용하지 않고 시스템 특권 또는 역할을 부여했을 경우에는, 부여받은 사용자는 부여받은 특권 또는 역할을 사용할 수만 있고, 다른 사용자 또는 역할에 부여할 수는 없음 - 반대로 GRANT ANY PRIVILEGE 시스템 특권을 부여받은 사용자의 경우 자신이 부여받지 않은 시스템 특권도 마음대로 부여 가능 - 역할의 경우엔 GRANT ANY ROLE 시스템 특권이 같은 역할을 하게 됨 - 주의해야 할 점은 WITH ADMIN OPTION을 사용하여 해당 특권 또는 역할을 남에게 부여할 수 있는 관리 특권을 주었을 경우 사용 특권은 그대로 둔 상태로 관리 특권만을 회수할 수 없음 - 따라서 그 경우에는 부여한 시스템 특권 또는 역할 전체를 회수한 뒤 WITH ADMIN OPTION 없이 다시 그 시스템 특권 또는 역할을 할당해야함

  - object_objprivs

구성요소	설명
object_privilege_clause	스키마 객체 특권을 명시
(username.) object	- 스키마 객체 특권의 대상이 되는 객체를 명시 - username 부분이 생략되었을 경우 자신의 스키마 객체에서 해당 이름을 가진 객체를 찾게됨

	- 만약 대상으로 동의어가 포함된 경우에는 해당 동의어의 기반 객체로 인식하고 처리하게됨 - 즉 동의어에 스키마 객체 특권을 부여하거나 회수하는 것은 동의어의 기반 객체에 특권을 부여하거나 회수하는 것과 동일한 효과를 갖게됨
TO grantee_clause	- 스키마 객체 특권을 부여받을 대상이됨 - 대상이 될 수 있는 것은 일반 사용자나 역할, 공유 사용자 3 종류
WITH GRANT OPTION	- 시스템 특권의 WITH ADMIN OPTION과 마찬가지 역할을 함 - 즉, 부여한 스키마 객체 특권을 다른 사용자에게 부여할 수 있는 특권까지 같이 부여하고자 할 때 사용됨 - 하지만, 시스템 특권의 WITH ADMIN OPTION과는 커다란 차이가 있음 - 자신의 스키마 객체가 아닌 다른 사용자의 스키마 객체를 WITH GRANT OPTION을 이용하여 스키마 객체 특권을 부여받은 사용자 A가, 또 다른 사용자 B에게 스키마 객체 특권을 부여한 경우 사용자 A의 스키마 객체 특권을 회수하면, 사용자 A가 부여한 사용자 B의 특권도 같이 회수됨 - 또한, 시스템 특권에서의 역할과 마찬가지로, 자신의 스키마 객체에 대해서는 별도의 스키마 객체 특권을 부여받지 않아도, 모든 스키마 객체 특권과 해당 스키마 객체 특권을 다른 사용자에게 부여할 수 있는 특권까지 갖고 있음 - GRANT ANY OBJECT PRIVILEGE 시스템 특권을 부여받은 사용자의 경우 자신이 부여받지 않은 객체에 스키마 객체 특권도 모두 부여 가능 - 이는 시스템 특권에서의 GRANT ANY PRIVILEGE와 동일한 역할 가능 - 또한, 부여받는 객체가 일반 사용자나 공유 사용자가 아닌 역할일 경우에는 WITH GRANT OPTION은 사용할 수 없음

- object_privilege_clause

구성요소	설명
object_privilege	스키마 객체 특권을 명시
colname	- 스키마 객체 특권은 시스템 특권과는 달리 좀 더 세부적인 설정이 가능 - 즉, 특정 객체 전체에 대한 특권이 아닌 객체의 일부 컬럼에 대해서만 제약적으로 특권을 부여 가능 - 객체의 특정 컬럼에 대해서만 제약적으로 스키마 객체 특권을 부여하고자 할 경우에는 특정 컬럼을 나열하는 방식으로 지정 가능 - 하지만, 모든 스키마 객체 특권이 컬럼별 특권을 지정할 수 있는 것은 아님 - 컬럼별 특권을 지정할 수 있는 스키마 객체 특권은 INSERT, REFERENCES, UPDATE뿐
ALL PRIVILEGES	- 현재 사용가능한 모든 스키마 객체 특권을 부여 - GRANT ANY PRIVILEGE 시스템 특권을 가지고 있는 사용자만이 ALL PRIVILEGE 절을 사용 가능

- grantee_clause

구성요소	설명
user_name	스키마 객체 특권을 부여받을 일반 사용자를 명시
role_name	스키마 객체 특권을 부여받을 역할을 명시
PUBLIC	- 스키마 객체 특권을 부여받을 공유 사용자를 명시 - 공유 사용자에게 특권 또는 역할을 부여하면 모든 사용자에게 부여한 것과 동일한 효과를 갖게 되므로 주의

- 예제

- grant_sysprivs
  - 다음은 사용자를 생성하고, **system_privilege**에 특권을 명시해 그 사용자에게 CREATE SESSION 시스템 특권을 부여하는 예

```
SQL> conn sys/tibero
Connected to Tibero.

SQL> drop user u1;

User 'U1' dropped.

SQL> create user u1 identified by 'a';

User 'U1' created.

SQL> select grantee,privilege from dba_sys_privs
   2 where grantee='U1';

0 row selected.

SQL> grant create session to u1;

Granted.

SQL> select grantee,privilege from dba_sys_privs
   2 where grantee='U1';

GRANTEE
--------------------------------------------------------------------------------
PRIVILEGE
----------------------------------------
U1
CREATE SESSION


1 row selected.
```

  - 다음은 생성된 사용자 U1에게 **role_name**에 역할을 명시해 RESOURCE 역할을 부여하는 예

```
SQL> select grantee, granted_role from dba_role_privs
   2 where grantee='U1';

0 row selected.

SQL> select * from dba_roles;

ROLE
--------------------------------------------------------------------------------
PASSWORD_REQUIRED
-----------------
ACCT_PAY
YES

ACCT_REC
NO

CONNECT
NO
```

```
DBA
NO

HS_ADMIN_ROLE
NO

MANAGER
NO

RESOURCE
NO


ROLE
------------------------------------------------------------------------------
PASSWORD_REQUIRED
-----------------
SELECT_CATALOG_ROLE
NO


8 rows selected.

SQL> grant resource to u1;

Granted.

SQL> select grantee, granted_role from dba_role_privs
   2 where grantee='U1';

GRANTEE
------------------------------------------------------------------------------
GRANTED_ROLE
------------------------------------------------------------------------------
U1
RESOURCE


1 row selected.
```

- 다음은 **ALL PRIVILEGE**를 사용하여 모든 시스템 특권을 부여하는 예

```
SQL> grant all privileges to u1;

Granted.

SQL> select grantee, privilege from dba_sys_privs
   2 where grantee='U1';

GRANTEE
------------------------------------------------------------------------------
PRIVILEGE
----------------------------------------
U1
ALTER SYSTEM

U1
```

```
CREATE SESSION

U1
ALTER SESSION

U1
CREATE TABLESPACE

U1
ALTER TABLESPACE

U1
DROP TABLESPACE

U1
CREATE USER
. . .
```

- 여러 역할 서로가 서로를 부여받는 일은 허용하지 않는다.

```
SQL> create role aaa;

Role 'AAA' created.

SQL> create role bbb;

Role 'BBB' created.

SQL> grant aaa to bbb;

Granted.

SQL> grant bbb to aaa;
TBR-7173: Circular role grant detected.
```

- 다음은 **WITH ADMIN OPTION**을 사용했을 때와 사용하지 않았을 때의 차이에 관한 예

```
SQL> drop user u2;

User 'U2' dropped.

SQL> create user u2 identified by xxx;

User 'U2' created.

SQL> drop user u3;

User 'U3' dropped.

SQL> create user u3 identified by zzz;

User 'U3' created.

SQL> grant create session to u2;

Granted.
```

```
SQL> grant create table to u2 with admin option;

Granted.

SQL> desc dba_sys_privs

COLUMN_NAME                              TYPE               CONSTRAINT
---------------------------------------- ------------------ --------------------
GRANTEE                                  VARCHAR(128)
PRIVILEGE                                VARCHAR(40)
ADMIN_OPTION                             VARCHAR(3)

SQL> select grantee, privilege,admin_option
   2 from dba_sys_privs where grantee='U2';

GRANTEE
---------------------------------------------------------------------------------
PRIVILEGE                                ADMIN_OPTION
---------------------------------------- ------------
U2
CREATE SESSION                           NO

U2
CREATE TABLE                             YES


2 rows selected.

SQL> conn u2/xxx
Connected to Tibero.

SQL> grant create session to u3;
TBR-17004: Permission denied.

SQL> grant create table to u3;

Granted.
```

- grant_objprivs
  - 다음은 컬럼별로 특권을 지정하는 예

```
SQL> create user u5 identified by xxx;

User 'U5' created.

SQL> CREATE USER u6 IDENTIFIED BY zzz;

User 'U6' created.

SQL> GRANT CONNECT TO u6;

Granted.

SQL> GRANT CONNECT,RESOURCE TO u5;

Granted.
```

```
SQL> conn u5/xxx;
Connected to Tibero.

SQL> CREATE TABLE t1 (a NUMBER, b NUMBER, c NUMBER);

Table 'T1' created.

SQL> INSERT INTO t1 VALUES (1, 2, 3);

1 row inserted.

SQL> SELECT * FROM t1;

         A          B          C
---------- ---------- ----------
         1          2          3

1 row selected.

SQL> GRANT SELECT ON t1 TO u6;

Granted.

SQL> conn u6/zzz;
Connected to Tibero.

SQL> select * from u5.t1;

         A          B          C
---------- ---------- ----------
         1          2          3

1 row selected.

SQL> conn u5/xxx;
Connected to Tibero.

SQL> grant insert(a,b),update(a,b) on t1 to u6;

Granted.

SQL> conn u6/zzz
Connected to Tibero.

SQL> insert into u5.t1 values(100,200,300);
TBR-8053: Not authorized.

SQL> insert into u5.t1(a,b) values(100,200);

1 row inserted.

SQL> select * from u5.t1;

         A          B          C
---------- ---------- ----------
       100        200
         1          2          3
```

```
2 rows selected.

SQL> update u5.t1 set c=100 where a=100;
TBR-8053: Not authorized.

SQL> update u5.t1 set a=11 where a=100;

1 row updated.

SQL> select * from u5.t1;

         A          B          C
---------- ---------- ----------
        11        200
         1          2          3

2 rows selected.
```

- 다음은 **GRANT ALL**을 사용하여 사용자 자신이 부여할 수 있는 모든 스키마 객체 특권을 부여하는 예

```
SQL> conn u5/xxx
Connected to Tibero.

SQL> create table t2(c1 number, c2 number);

Table 'T2' created.

SQL> grant all on t2 to u6;

Granted.

SQL> conn u6/zzz
Connected to Tibero.

SQL> select owner, table_name privilege
   2 from user_tbl_privs where table_name='T2';

OWNER
--------------------------------------------------------------------------------
PRIVILEGE
--------------------------------------------------------------------------------
U5
T2

U5
T2

U5
T2

U5
T2

U5
T2

U5
T2

U5
T2
```

```
U5
T2


7 rows selected.
```

- 다음은 동의어에 특권을 할당하는 예

```
SQL> conn sys/tibero
Connected to Tibero.

SQL> grant create synonym to u5;

Granted.

SQL> conn u5/xxx
Connected to Tibero.

SQL> create table t3(c1 number,  c2 number);

Table 'T3' created.

SQL> create synonym s3 for t3;

Synonym 'S3' created.

SQL> grant select, insert on s3 to u6;

Granted.

SQL> conn u6/zzz
Connected to Tibero.

SQL> insert into u5.t3 values(1,2);

1 row inserted.

SQL> select * from u5.s3;

        C1          C2
---------- ----------
         1           2

1 row selected.

SQL> select owner, table_name, privilege
   2 from user_tbl_privs where table_name='T3';

OWNER
--------------------------------------------------------------------------------
TABLE_NAME
--------------------------------------------------------------------------------
PRIVILEGE
----------------------------------------
U5
T3
INSERT
```

```
U5
T3
SELECT


2 rows selected.

SQL> select owner, table_name, privilege
   2 from user_tbl_privs where table_name='S3';

0 row selected.
```

- 사용자 u6에게는 동의어가 가리키는 테이블 t3에 대한 특권이 부여되어, 직접 t3이라는 이름으로 질의 또는 로우를 삽입 가능

- grantee_clause
  - 다음은 공유 사용자에게 스키마 객체 특권을 부여하는 예

```
SQL> conn sys/tibero
Connected to Tibero.

SQL> create public synonym s1 for u5.t1;

Synonym 'S1' created.

SQL> grant select on s1 to public;

Granted.

SQL> select owner, table_name, privilege
   2 from dba_tbl_privs
   3 where grantee='PUBLIC' and owner='U5';

OWNER
--------------------------------------------------------------------------------
TABLE_NAME
--------------------------------------------------------------------------------
PRIVILEGE
----------------------------------------
U5
T1
SELECT


1 row selected.

SQL> create user u7 identified by aaa;

User 'U7' created.

SQL> grant connect to u7;

Granted.

SQL> conn u7/aaa
Connected to Tibero.

SQL> select * from s1;
```

```
         A          B          C
---------- ---------- ----------
        11        200
         1          2          3

2 rows selected.
```

- 다음은 **WITH GRANT OPTION**을 사용하는 예

```
SQL> conn u5/xxx
Connected to Tibero.

SQL> create talble t4(c1 number, c2 number);
TBR-7001: General syntax error.
at line 1, column 8 of null:
create talble t4(c1 number, c2 number)
       ^^^^^^

SQL> create table t4(c1 number, c2 number);

Table 'T4' created.

SQL> grant select on t4 to u6 with grant option;

Granted.

SQL> grant insert on t4 to u6;

Granted.

SQL> conn u6/zzz
Connected to Tibero.

SQL> select owner, table_name, privilege, grantable
   2 from user_tbl_privs where table_name='T4';

OWNER                          TABLE_NAME
------------------------------ ------------------------------
PRIVILEGE                      GRANTABLE
------------------------------ ---------
U5                             T4
INSERT                         NO

U5                             T4
SELECT                         YES


2 rows selected.

SQL> insert into u5.t4 values(1,2);

1 row inserted.

SQL> select * from u5.t4;

        C1         C2
---------- ----------
```

```
         1          2

1 row selected.

SQL> grant insert on u5.t4 to u7;
TBR-17004: Permission denied.

SQL> grant select on u5.t4 to u7;

Granted.

SQL> conn u7/aaa;
Connected to Tibero.

SQL> select * from u5.t4;

        C1         C2
---------- ----------
         1          2

1 row selected.
```

- 다음은 **WITH ADMIN OPTION**과 **WITH GRANT OPTION**의 차이점에 관한 예

```
SQL> conn u5/xxx
Connected to Tibero.

SQL> revoke all on t4 from u6;

Revoked.

SQL> conn u7/aaa
Connected to Tibero.

SQL> select * from u5.t4;
TBR-8033: Specified schema object was not found.
at line 1, column 16 of null:
select * from u5.t4
              ^
```

→ WITH GRANT OPTION은 특권을 회수하면 특권을 받은 사용자가 부여한 또 다른 특권도 모두 같이 회수된다. 즉, 위의 예에서 사용자 u6에게 부여한 테이블 t4에 대한 스키마 객체 특권을 모두 회수하면 ?

→ 사용자 u6이 사용자 u7에게 부여한 스키마 객체 특권도 같이 회수가 되어 사용자 u7도 더 이상 사용자 u5의 테이블 t4에 대해 질의할 수 없게 됨

1.1.2 수행

수행 순서	시나리오
1	tbsql 접속
2	사용자 생성, 패스워드 설정
3	권한 설정
4	권한 정보 확인

1.1.3 결과

시나리오	설명

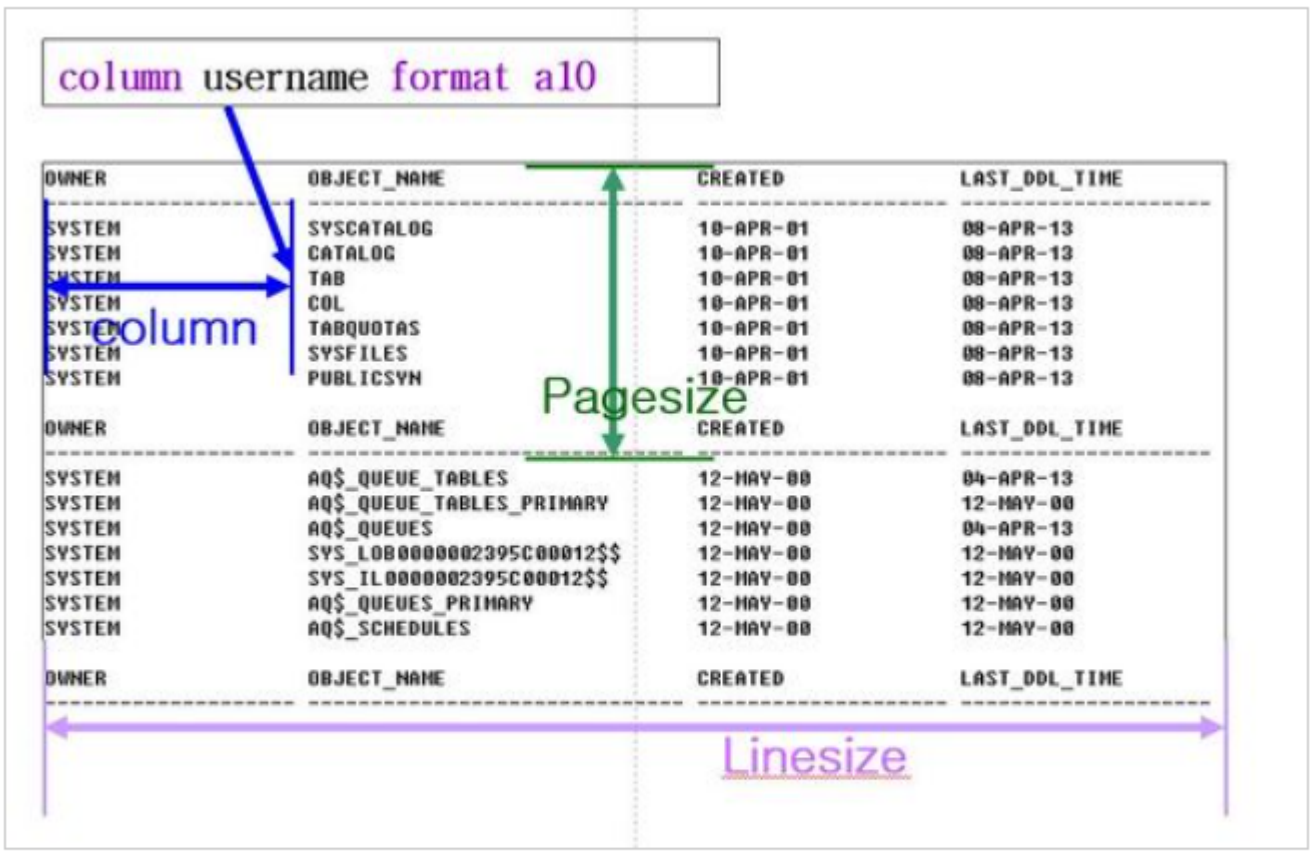
SQL> create user test identified by 'TEST'; User 'TEST' created.	- 사용자(test) 생성, 패스워드(TEST) 설정 - 사용자의 이름은 전체 데이터베이스의 다른 사용자 또는 역할의 이름과 중복되면 안됨
SQL> desc dba_users; COLUMN_NAME TYPE CONSTRAINT ---------- USERNAME VARCHAR(128) USER_ID NUMBER PASSWORD VARCHAR(24) ACCOUNT_STATUS VARCHAR(30) LOCK_DATE DATE EXPIRY_DATE DATE DEFAULT_TABLESPACE VARCHAR(128) CREATED DATE PROFILE VARCHAR(255) DEFAULT_TEMP_TABLESPACE VARCHAR(128) AUTHENTICATION_TYPE VARCHAR(8)	사용자의 특권을 나타내는 table인 dba_users의 colname을 보기 위해 describe
SQL> select username from dba_users where username='TEST'; USERNAME ---------- TEST 1 row selected.	TEST user가 user로 만들어졌는지 확인
SQL> select * from dba_roles; ROLE ---------- PASSWORD_REQUIRED ---------- ACCT_PAY YES ACCT_REC NO CONNECT NO DBA NO HS_ADMIN_ROLE NO MANAGER NO RESOURCE NO ROLE ---------- PASSWORD_REQUIRED	role의 종류 dba_role table에서 확인

---------- SELECT_CATALOG_ROLE NO 8 rows selected.	
SQL> grant resource to test; Granted.	RESOURCE 역할은 기본 역할로, 데이터베이스가 생성될 때 자동으로 생성
SQL> SELECT grantee, privilege FROM dba_sys_privs WHERE grantee='RESOURCE'; GRANTEE ---------- PRIVILEGE ---------- RESOURCE CREATE TABLE RESOURCE CREATE SEQUENCE RESOURCE CREATE PROCEDURE RESOURCE CREATE TRIGGER 4 rows selected.	자신의 스키마 객체를 생성하기 위한 기본적인 시스템 특권 확인
SQL> desc dba_sys_privs COLUMN_NAME TYPE CONSTRAINT ---------- ---------- ---------- GRANTEE VARCHAR(128) PRIVILEGE VARCHAR(40) ADMIN_OPTION VARCHAR(3)	- dba_sys_privs에서 시스템 권한 확인 - grantee : 권한 가지고 있는 사용자 - privilege : 권한 종류 - admin_option : 해당 특권 또는 역할을 사용할 수 있는 사용 특권,남에게 부여할 수 있는 관리 특권
SQL> desc dba_role_privs; COLUMN_NAME TYPE CONSTRAINT ---------- ---------- ---------- GRANTEE VARCHAR(128) GRANTED_ROLE VARCHAR(128) ADMIN_OPTION VARCHAR(3) DEFAULT_ROLE VARCHAR(3)	- dba_role_privs에서 role 권한 확인 - grantee : 권한있는 사용자 이름 - granted_role : role 이름 - admin_option : yes/no - default_role : yes/no
SQL> col grantee for a30 SQL> col grantee_role for a30 SQL> col admin_option for a30 SQL> col default_role for a30 SQL> set linesize 1000 SQL> select * from dba_role_privs where grantee='TEST'; GRANTEE GRANTED_ROLE ADMIN_OPTION DEFAULT_ROLE ---------- ---------- ---------- ----------	table column, linesize 변경

```
--------
TEST RESOURCE NO YES

1 row selected.
```

*column을 col로 바꿔도 무방



추가로 헤더명을 바꿀수도 있습니다.

col 원래컬럼명 for a10 heading "바꿀컬럼명"

1.2 타 사용자의 테이블에 대한 권한이 없는 사용자의 접근 차단 기능

1.2.1 설명

- 타 사용자의 테이블에 대한 권한이 없는 사용자의 접근 차단 기능

1.2.2 수행

수행 순서	시나리오
1	사용자(test1) 생성
2	사용자(test1)에 resource,connect권한 부여
3	사용자(test1)에 table(tbl) 생성
4	사용자(test1)에 table(tbl)에 데이터 입력
5	사용자(test1)에 접속
6	사용자(test1)에 접속해서 table(tbl) 조회
7	다른 사용자(test2) 생성
8	다른 사용자(test2)에 resource,connect권한 부여
9	다른 사용자(test2)에 접속
10	다른 사용자(test2)에 접속해서 test1 사용자의 table(tbl) 조회

1.2.3 결과

시나리오	설명
SQL> create user test1 identified by 'TEST1';	사용자 test1, 패스워트 TEST1 생성

User 'TEST1' created.	
SQL> grant connect,resource to test1; Granted.	connect, resource 권한 test1에게 부여
SQL> create table test1.tbl (id number); Table 'TEST1.TBL' created.	test1의 tbl table 생성
SQL> insert into test1.tbl values(1); 1 row inserted.	tbl table의 값 삽입
SQL> commit; Commit completed.	table 저장
SQL> CONN TEST1/TEST1 Connected to Tibero.	test1 사용자에 접속
SQL> SELECT * FROM TEST1.TBL; ID ---------- 1 1 row selected.	tbl table 조회
SQL> CONN SYS/TIBERO Connected to Tibero.	다시 sys 사용자로 접속
SQL> CREATE USER TEST2 IDENTIFIED BY 'TEST2'; User 'TEST2' created.	사용자 test2, 패스워드 TEST2 생성
SQL> GRANT CONNECT,RESOURCE TO TEST2; Granted.	connect, resource 권한 test2 사용자에게 부여
SQL> CONN TEST2/TEST2 Connected to Tibero.	test2 사용자에 접속
SQL> SELECT * FROM TEST1.TBL; TBR-8033: Specified schema object was not found. at line 1, column 16 of null: SELECT * FROM TEST1.TBL ^	test1의 tbl table 조회 → 사용자 test1이 사용자 test2에게 특권을 부여하지 않은 경우 오류 발생

1.3 타 사용자의 테이블 접근 권한을 부여 받은 사용자에 대한 접근 가능 여부 확인

1.3.1 설명

- 타 사용자의 테이블 접근 권한을 부여 받은 사용자에 대한 접근 가능 여부 확인

1.3.2 수행

수행 순서	시나리오

1	Column format 및 line size 조정
2	테이블(TEST1.TBL)에 대한 권한확인
3	타사용자(TEST2)에게 테이블(TEST1.TBL) 조회권한(select) 부여
4	타사용자(TEST2)에게 부여된 테이블(TEST1.TBL) 조회권한(select) 확인

1.3.3 결과

시나리오	설명
SQL> CONN SYS/TIBERO Connected to Tibero.	test2로 접속되어있으므로 sys사용자로 접속
SQL> desc dba_tab_privs COLUMN_NAME TYPE CONSTRAINT ---------- ---------- ---------- GRANTEE VARCHAR(128) OWNER VARCHAR(128) TABLE_NAME VARCHAR(128) GRANTOR VARCHAR(128) PRIVILEGE VARCHAR(40) GRANTABLE VARCHAR(3)	table privilege를 나타내는 dba_tab_privs table의 colname 조회 - grantee : 권한 받은 사용자 - owner : table 가지고 있는 사용자 - table_name : table 이름 - grantor : 어떤 권한인지(select, update,,,) - privilege : yes/no
SQL> SET LINES 200 SQL> COL GRANTEE FOR A20 SQL> COL OWNER FOR A20 SQL> COL TABLE_NAME FOR A20 SQL> COL GRANTOR FOR A20 SQL> COL PRIVILEGE FOR A20 SQL> COL GRANTABLE FOR A20 SQL> SELECT * FROM DBA_TAB_PRIVS WHERE OWNER='TEST1' AND TABLE_NAME='TBL'; 0 row selected.	test1.tbl이 다른 사용자에게 준 권한이 있나 확인 → 0 row(없음)
SQL> GRANT SELECT ON TEST1.TBL TO TEST2; Granted.	test2에게 test1.tbl 조회 권한 부여
SQL> SELECT * FROM DBA_TAB_PRIVS WHERE OWNER='TEST1' AND TABLE_NAME='TBL'; GRANTEE OWNER TABLE_NAME GRANTOR PRIVILEGE GRANTABLE ---------- ---------- ---------- -------- ---------- ---------- TEST2 TEST1 TBL TEST1 SELECT NO 1 row selected.	test1.tbl이 다른 사용자에게 준 권한이 있나 다시 확인 → 1 row → test2가 test1.tbl을 select할 수 있는 권한 확인

2. 동시성 제어

2.1 Row-level locking

2.1.1 설명

- 동시에 실행되는 여러 개의 트랜잭션이 작업을 성공적으로 마칠 수 있도록 트랜잭션의 실행 순서를 제어하는 기법
- Row-level locking 및 MVCC 기능

2.1.2 수행

수행 순서	시나리오	시나리오
	SESSION 1	SESSION 2
1	테이블(TEST1.RXTEST) 생성	
2	테이블(TEST1.RXTEST)에 데이터 입력	
3	테이블(TEST1.RXTEST)의 데이터 변경 - COMMIT 안함	
4		테이블(TEST1.RXTEST)의 데이터 변경
5	테이블(TEST1.RXTEST)의 데이터 변경 - COMMIT	
6		HANG이 풀리는 것을 확인

2.1.3 결과

시나리오	시나리오	설명
SESSION 1	SESSION 2	
[tibero@T1:/home/tibero]$ tbsql TEST1/TEST1 tbSQL 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. Connected to Tibero.		test1 사용자로 접속
SQL> CREATE TABLE TEST1.RXTEST(ID NUMBER, NAME VARCHAR(20)); Table 'TEST1.RXTEST' created.		test1.rxtest table 생성
SQL> INSERT INTO TEST1.RXTEST VALUES(1,'TEST'); 1 row inserted.		test1.rxtest table에 데이터 입력
SQL> COMMIT; Commit completed.		test1.rxtest table값 저장
SQL> SELECT * FROM TEST1.RXTEST; ID NAME ---------- -------------------- 1 TEST 1 row selected.		test1.rxtest table 조회
SQL> GRANT UPDATE ON RXTEST TO TEST2; Granted.		test2 사용자에게 test1.rxtest table update 권한 부여
SQL> ALTER SESSION SET NLS_DATE_FORMAT='YYYY/MM/DD		날짜 형식 session 변경

HH24:MI:SS'; Session altered.		
SQL> SELECT SYSDATE FROM DUAL; SYSDATE ---------- 2022/11/01 17:43:54 1 row selected.		현재 시간 확인
SQL> UPDATE TEST1.RXTEST SET NAME ='TESTTEST' WHERE ID =1; 1 row updated.		test1 사용자 권한으로 test1.rxtest table의 데이터 변경
SQL> SELECT * FROM TEST1.RXTEST; ID NAME ---------- -------------------- 1 TESTTEST 1 row selected.		test1.rxtest 조회 → name 변경 확인 → COMMIT 안함
	[tibero@T1:/home/tibero]$ tbsql TEST2/TEST2 tbSQL 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. Connected to Tibero.	다른 session으로 test2 사용자로 접속
	SQL> ALTER SESSION SET NLS_DATE_FORMAT='YYYY/MM/DD HH24:MI:SS'; Session altered.	날짜 형식 session 변경
	SQL> SELECT SYSDATE FROM DUAL; SYSDATE ---------- 2022/11/01 17:44:23 1 row selected.	현재 시각 확인
	SQL> UPDATE TEST1.RXTEST SET NAME ='TESTTESTTEST' WHERE ID = 1; <HANG 걸림>	test2 사용자 권한으로 test1.rxtest table update → hang 걸림

SQL> COMMIT; Commit completed.		session1에서 commit(저장)함 → COMMIT함
	1 row updated.	hang 풀림

2.2 MVCC 기능

2.2.1 설명

- 트랜잭션이 한 데이터 아이템을 접근하려 할 때, 그 트랜잭션의 타임스탬프와 접근하려는 데이터 아이템의 여러 버전의 타임스탬프를 비교하여, 현재 실행하고 있는 스케줄의 직렬가능성이 보장되는 적절한 버전을 선택하여 접근하도록 하는 기법

2.2.2 수행

수행 순서	시나리오	시나리오
	SESSION 1	SESSION 2
1	테이블(TEST1.MVCCTEST) 생성	
2	테이블(TEST1.MVCCTEST)에 데이터 입력	
3	테이블(TEST1.MVCCTEST)의 데이터 변경	
4	테이블(TEST1.MVCCTEST)의 변경된 데이터	
5	조회	
6		테이블(TEST1.MVCCTEST)의 데이터 조회
	변경된 데이터 commit	
		테이블(TEST1.MVCCTEST)의 변경된 데이터 조회

2.2.3 결과

시나리오	시나리오	설명
SESSION 1	SESSION 2	
[tibero@T1:/home/tibero]$ tbsql TEST1/TEST1 tbSQL 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. Connected to Tibero.		test1 사용자로 접속
SQL> CREATE TABLE TEST1.MVCTEST(ID NUMBER, NAME VARCHAR(20)); Table 'TEST1.MVCTEST' created.		test1 사용자의 mvctest table 생성
SQL> INSERT INTO TEST1.MVCTEST VALUES(1,'TEST'); 1 row inserted.		test1의 mvctest table의 데이터 입력
SQL> COMMIT; Commit completed.		commit (변경사항 저장)
SQL> SELECT * FROM TEST1.MVCTEST;		test1의 mvctest table 조회

ID NAME ---------- -------------------- 1 TEST 1 row selected.		
SQL> GRANT SELECT ON MVCTEST TO TEST2; Granted.		test2사용자에게 test1의 mvctest table 조회 권한 부여
SQL> UPDATE TEST1.MVCTEST SET NAME='TEST123' WHERE ID=1; 1 row updated.		test1의 mvctest table의 name 칼럼을 'TEST123'로 변경 → COMMIT 하지 않음
SQL> ALTER SESSION SET NLS_DATE_FORMAT='YYYY/MM/DD HH24:MI:SS'; Session altered. SQL> SELECT SYSDATE FROM DUAL; SYSDATE ---------- 2022/11/02 10:41:35 1 row selected.		현재 시각 확인 → 10:41:35
SQL> SELECT * FROM TEST1.MVCTEST WHERE ID=1; ID NAME ---------- -------------------- 1 TEST123		update된 test1의 mvctest table 조회
	[tibero@T1:/home/tibero]$ tbsql TEST2/TEST2 tbSQL 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. Connected to Tibero.	다른 session으로 test2 사용자에게 접속
	SQL> ALTER SESSION SET NLS_DATE_FORMAT='YYYY/MM/DD HH24:MI:SS'; Session altered. SQL> SELECT SYSDATE FROM DUAL; SYSDATE ---------- 2022/11/02 10:46:56 1 row selected.	현재 시각 확인 → 10:46:56

	SQL> SELECT * FROM TEST1.MVCTEST WHERE ID=1; ID NAME ---------- -------------------- 1 TEST 1 row selected.	test2 사용자가 test1의 mvctest table 조회 → 데이터 변경 전 table
SQL> SELECT SYSDATE FROM DUAL; SYSDATE ---------- 2022/11/02 10:47:57 1 row selected.		test1 사용자에서의 현재 시각 확인 → 10:47:57
SQL> COMMIT; Commit completed		commit (update된 변경 사항 저장)
	SQL> SELECT * FROM TEST1.MVCTEST WHERE ID=1; ID NAME ---------- -------------------- 1 TEST123 1 row selected.	test2 사용자의 session에서 변경 사항 조회 → update됨

3. 테이블 관리

3.1 Drop(DDL)된 테이블에 대해 원복 쿼리를 통한 복구 여부 확인

3.1.1 설명

- Drop(DDL)된 테이블에 대해 원복 쿼리를 통한 복구 여부 확인
- FLASHBACK TABLE은 특정 시점의 데이터만을 복원해주며, 테이블과 관련된 인덱스, 트리거, 제약조건은 복원되지 않는다.
- 초기화 파라미터 UNDO_RETENTION 시점까지 FLASHBACK TABLE이 가능하지만 _TSN_TIME_MAP_SIZE의 크기가 UNDO_RETENTION보다 같거나 커야 한다. 이는 $TB_SID.tip 파일에서 옵션으로 값을 설정가능
- 특권
    - FLASHBACK_ANY_TABLE 특권이 있어야 한다.
- 구성요소

구성요소	설명
schema	복원할 테이블을 포함하고 있는 스키마의 이름이다. 생략하면 현재 사용자의 스키마로 인식한다.
table_name	복원할 테이블의 이름이다. 테이블을 제거하기 전의 이름 또는 RECYCLEBIN 뷰를 참조하여 제거한 후 시스템이 변경한 이름으로 지정할 수 있다. 단, 같은 이름의 테이블이 RECYCLEBIN 뷰에 여러 개가 존재하는 경우 원래 이름을 지정하면 가장 최근의 이름으로 사용된다.
TSN	특정 TSN으로 복원할 때 사용한다.
TIMESTAMP	특정 TIMESTAMP로 복원할 때 사용한다.
expression	TSN과 TIMESTAMP 구문으로 복원할 때 필요한 구문이다. TSN의 경우 특정 시점의 TSN이며, TIMESTAMP의 경우 특정 시간을 의미하는 구문이다.

BEFORE DROP	제거된 테이블을 복원한다. RENAME TO table_name 문을 사용하지 않을 때는 생략할 수 있다.
RENAME TO table_name	복원할 테이블의 이름을 원래 이름과 다르게 변경할 때 사용한다. 특히 이전에 사용하던 이름을 다른 테이블이 사용하고 있는 경우에 필요하다.

3.1.2 수행

수행 순서	시나리오
1	환경파일(TIP)에 Drop(DDL)된 테이블 복구기능을 제공하는 파라미터 활성화(USE_RECYCLEBIN)
2	테스트 테이블(TIBERO.FLASHBACK_TEST) 생성
3	테스트 테이블(TIBERO.FLASHBACK_TEST)에 데이터 입력
4	테스트 테이블(TIBERO.FLASHBACK_TEST) 건수 조회
5	테스트 테이블(TIBERO.FLASHBACK_TEST) Drop
6	Drop 된 테스트 테이블(TIBERO.FLASHBACK_TEST)을 Recycle bin 에서 조회
7	Drop 된 테스트 테이블(TIBERO.FLASHBACK_TEST)을 복구
8	테스트 테이블(TIBERO.FLASHBACK_TEST) 건수 조회
9	환경파일(TIP)에 Drop(DDL)된 테이블 복구기능을 제공하는 파라미터 비활성화(USE_RECYCLEBIN)

3.1.3 결과

시나리오	설명
SQL> drop user test; User 'TEST' dropped. SQL> create user test identified by test; User 'TEST' created. SQL> grant connect, resource to test; Granted.	user 충돌 막기 위해 drop후 재생성 test에 권한 부여
SQL> conn sys/tibero Connected to Tibero. SQL> alter system set use_recyclebin=y; system altered.	환경파일(TIP)에 drop(DDL)된 table 복구기능을 제공하는 파라미터 활성화 → use_recyclebin
SQL> conn test/test Connected to Tibero.	test 사용자에 접속
SQL> create table flashback_test(c1 number, c2 number, c3 number); Table 'FLASHBACK_TEST' created.	test의 flashback_test table 생성
SQL> declare 2 begin 3 for i in 1..10000 loop 4 insert into test.flashback_test values(i,i,i);	test 사용자의 flashback_test table에 for loop로 데이터 채우기

5 end loop; 6 end; 7 / PSM completed.	
SQL> commit; Commit completed.	commit(저장)
SQL> select count(*) from flashback_test; COUNT(*) ---------- 10000 1 row selected.	test 사용자의 flashback_test table의 건수 조회 → 조회 O
SQL> drop table test.flashback_test; Table 'TEST.FLASHBACK_TEST' dropped.	test 사용자의 flashback_test table 지우기
SQL> select count(*) from flashback_test; TBR-8033: Specified schema object was not found. at line 1, column 23 of null: select count(*) from flashback_test ^	test 사용자의 flashback_test table의 건수 조회 → 조회 X
SQL> desc user_recyclebin; COLUMN_NAME TYPE CONSTRAINT ---------- ---------- ---------- OBJECT_NAME VARCHAR(128) ORIGINAL_NAME VARCHAR(128) TYPE VARCHAR(7) TS_NAME VARCHAR(128) CREATETIME VARCHAR(19) DROPTIME VARCHAR(19) DROPTSN NUMBER BASE_OBJECT VARCHAR(256) SPACE VARCHAR(0)	table 복구기능을 제공하는 파라미터 있는 table인 user_recyclebin의 칼럼명 조회
SQL> col object_name for a20 SQL> col original_name for a20 SQL> col ts_name for a20 SQL> col space for a30 SQL> select * from user_recyclebin; OBJECT_NAME ORIGINAL_NAME TYPE TS_NAME ---------- ---------- ------- ---------- CREATETIME DROPTIME DROPTSN ---------- ---------- ---------- BASE_OBJECT ---------- SPACE ---------- _TEST_TBL281400 FLASHBACK_TEST TABLE USR 2022-11-02:11:11:06 2022-11-02:11:13:56 78587	table 사이즈 변경 user_recyclebin table 조회

1 row selected.	
SQL> flashback table flashback_test to before drop; Flashbacked.	flashback table을 통해 drop 전으로 table 복구
SQL> select count(*) from flashback_test; COUNT(*) ---------- 10000 1 row selected.	flashback_test table의 건수 조회
SQL> conn sys/tibero Connected to Tibero.	sys 사용자로 접속
SQL> alter system set use_recyclebin=n; System altered.	환경파일(TIP)에 Drop(DDL)된 테이블 복구기능을 제공하는 파라미터 비활성화(use_recyclebin)

3.2 Range partition table 확인

3.2.0 Partition

- 테이블의 크기가 점점 커지고 많은 트랜잭션이 동시에 액세스하는 경우 운영체제는 빈번한 입출력과 잠금(Lock) 현상이 발생하게 된다. 이러한 현상은 데이터베이스 성능이 저하되는 원인
- 이를 해결하기 위해 하나의 논리적 테이블을 여러 개의 물리적인 공간으로 나누는 파티션을 설정할 수 있다. **파티션(Partition)**은 대용량 서비스를 하는 데이터베이스에서 효율적으로 관리하고 동작하기 위해 지원하는 옵션이다. 파티션은 서로 다른 테이블 스페이스에 생성할 수 있으며 입출력과 같은 물리적인 제약을 감소시킬 수 있다.
- 하나의 테이블로만 모든 데이터가 유지된다면 모든 트랜잭션이 한 곳에 집중하게 된다. 이로 인해 각 트랜잭션이 다른 트랜잭션을 대기하는 일이 많아져서 데이터베이스 성능이 저하된다. 하지만 파티션으로 나눈 경우에는 각 트랜잭션은 자신이 접근해야 할 파티션에만 접근하면 되므로 대기 확률이 줄어든다.

파티션	설명
RANGE	각 파티션에 포함될 RANGE를 지정하여 파티션을 정의한다.
HASH	HASH 함수를 이용하여 파티션을 정의한다.
LIST	각 파티션에 포함될 값을 직접 지정하여 파티션을 정의한다.

3.2.1 설명

- Column Value의 범위를 기준으로 하여 행을 분할하는 형태이다.
- Range Partition에서 Table은 단지 논리적인 구조이며 실제 데이터가 물리적으로 저장되는 곳은 Partition으로 나누어진 Tablespace에 저장이 된다.
- PARTITION BY RANGE (column_list) : 기본 Table에서 어느 Column을 기준으로 분할할지를 정함
- VALUES LESS THAN (value_list) : 각 Partition이 어떤 값의 범위를 포함할지 Upper Bound를 정함. ('이전 PARTITION에 들어가지 않고 ~ 보다 작은 값을 갖는 데이터를 포함하는 파티션'**)**
- ALTER TABLE 문에 의해 새로 만들어진 파티션은 기존의 마지막 파티션의 범위보다 높은 범위여야함

3.2.2 수행

수행 순서	시나리오
1	파티션테이블 생성
2	데이터 입력
3	파티션 추가
4	파티션 삭제

5	파티션 이름 변경
6	파티션 병합(MERGE)
7	파티션 분할(SPLIT)
8	파티션 변경(EXCHANGE)
9	파티션 테이블스페이스 변경
10	파티션 데이터 TRUNCATE

3.2.3 결과

시나리오	설명
SQL> create tablespace test_part1 datafile 'TEST_PART1.DBF' size 100m; Tablespace 'TEST_PART1' created. SQL> create tablespace test_part2 datafile 'TEST_PART2.DBF' size 100m; Tablespace 'TEST_PART2' created. SQL> create tablespace test_part3 datafile 'TEST_PART3.DBF' size 100m; Tablespace 'TEST_PART3' created. SQL> create tablespace test_part4 datafile 'TEST_PART4.DBF' size 100m; Tablespace 'TEST_PART4' created	tablespace 4개 생성
SQL> select tablespace_name from dba_tablespaces; TABLESPACE_NAME -- SYSTEM UNDO TEMP USR SYSSUB T1 TEST_PART1 TEST_PART2 TEST_PART3 TEST_PART4 10 rows selected.	만들어진 tablespace 확인
SQL> conn test/test Connected to Tibero. SQL> CREATE TABLE TEST.RANGE_PART (RANGE_NO NUMBER, RANGE_YEAR INT NOT NULL, RANGE_MONTH INT NOT NULL,	test 사용자의 range_part table 생성 (파티션 테이블도 생성)

RANGE_DAY INT NOT NULL, RANGE_NAME VARCHAR2(30), RANGE NUMBER) PARTITION BY RANGE (RANGE_YEAR, RANGE_MONTH, RANGE_DAY) (PARTITION RANGE_Q1 VALUES LESS THAN (2005, 01, 01) TABLESPACE TEST_PART1, PARTITION RANGE_Q2 VALUES LESS THAN (2005, 07, 01) TABLESPACE TEST_PART2, PARTITION RANGE_Q3 VALUES LESS THAN (2006, 01, 01) TABLESPACE TEST_PART3, PARTITION RANGE_Q4 VALUES LESS THAN (2006, 07, 01) TABLESPACE TEST_PART4); Table 'TEST.RANGE_PART' created. SQL> conn test/test Connected to Tibero. SQL> INSERT INTO RANGE_PART VALUES(1, 2004, 06, 12, 'SCOTT', 2500); 1 row inserted. SQL> INSERT INTO RANGE_PART VALUES(2, 2005, 06, 17, 'JONES', 4300); 1 row inserted. SQL> INSERT INTO RANGE_PART VALUES(3, 2005, 12, 12, 'MILLER', 1200); 1 row inserted. SQL> INSERT INTO RANGE_PART VALUES(4, 2006, 06, 22, 'FORD', 5200); 1 row inserted. SQL> INSERT INTO RANGE_PART VALUES(5, 2005, 01, 01, 'LION', 2200); 1 row inserted. SQL> COMMIT; Commit completed.	test 사용자에 들어가 range_part table에 데이터 생성 → COMMIT
SQL> SELECT * FROM TEST.RANGE_PART; RANGE_NO RANGE_YEAR RANGE_MONTH RANGE_DAY RANGE_NAME ---------- ---------- ---------- ---------- ---------- ---	test 사용자의 range_part table 조회

RANGE ---------- 1 2004 6 12 SCOTT 2500 2 2005 6 17 JONES 4300 5 2005 1 1 LION 2200 3 2005 12 12 MILLER 1200 4 2006 6 22 FORD 5200 5 rows selected.	
SQL> col table_name for a20 SQL> col partition_name for a20 SQL> select table_name, partition_name from user_tab_partitions where table_name='RANGE_PART'; TABLE_NAME PARTITION_NAME -------------------- -------------------- RANGE_PART RANGE_Q1 RANGE_PART RANGE_Q2 RANGE_PART RANGE_Q3 RANGE_PART RANGE_Q4 4 rows selected.	table 사이즈 변경 후 partition table 조회
SQL> create tablespace test_part_max datafile 'PART_MAX.DBF' size 100m; Tablespace 'TEST_PART_MAX' created. SQL> alter table range_part 2 add partition range_max values less than (maxvalue, maxvalue, maxvalue) 3 tablespace test_part_max; Table 'RANGE_PART' altered. SQL> select table_name, partition_name from user_tab_partitions where table_name='RANGE_PART'; TABLE_NAME PARTITION_NAME -------------------- -------------------- RANGE_PART RANGE_Q1 RANGE_PART RANGE_Q2 RANGE_PART RANGE_Q3 RANGE_PART RANGE_Q4 RANGE_PART RANGE_MAX 5 rows selected.	upper bound가 max value인 range_max partition table 추가 → test_part_max tablespace 먼저 생성
SQL> alter table range_part drop partition range_max; Table 'RANGE_PART' altered.	앞서 만든 range_max partition table 삭제 후 조회

SQL> select table_name, partition_name from user_tab_partitions where table_name='RANGE_PART'; TABLE_NAME PARTITION_NAME ---------- ---------- RANGE_PART RANGE_Q1 RANGE_PART RANGE_Q2 RANGE_PART RANGE_Q3 RANGE_PART RANGE_Q4 4 rows selected.	
SQL> alter table range_part rename partition range_q4 to range_four; Table 'RANGE_PART' altered. SQL> select table_name, partition_name from user_tab_partitions where table_name='RANGE_PART'; TABLE_NAME PARTITION_NAME ---------- ---------- RANGE_PART RANGE_Q1 RANGE_PART RANGE_Q2 RANGE_PART RANGE_Q3 RANGE_PART RANGE_FOUR 4 rows selected.	range_q4 partition table의 이름 range_four로 rename 후 조회
SQL> alter table range_part 2 merge partitions range_q1, range_q2 into partition range_q2 3 update indexes; Table 'RANGE_PART' altered. SQL> select table_name, partition_name from user_tab_partitions where table_name='RANGE_PART'; TABLE_NAME PARTITION_NAME ---------- ---------- RANGE_PART RANGE_Q2 RANGE_PART RANGE_Q3 RANGE_PART RANGE_FOUR 3 rows selected.	range_q1 partition과 range_q2 partition 합친(merge) 후 조회
SQL> alter table range_part 2 split partition range_q2 at (2005,01,01) 3 into (partition range_q1, 4 partition range_q2); Table 'RANGE_PART' altered. SQL> select table_name, partition_name from user_tab_partitions where table_name='RANGE_PART'; TABLE_NAME PARTITION_NAME ---------- ---------- RANGE_PART RANGE_Q1 RANGE_PART RANGE_Q2 RANGE_PART RANGE_Q3 RANGE_PART RANGE_FOUR	range_q2 partition을 2005,01,01을 기준으로 쪼갠(split)후 조회

4 rows selected.	
SQL> create table range_part_ex 2 (range_no number, 3 range_year int not null, 4 range_month int not null, 5 range_day int not null, 6 range_name varchar2(30), 7 range number) 8 tablespace test_part1; Table 'RANGE_PART_EX' created. SQL> alter table range_part 2 exchange partition range_q1 3 with table range_part_ex; Table 'RANGE_PART' altered.	partition data를 일반 table로 이동하기 위한 table 생성 range_part table의 partition table인 range_q1을 range_part_ex table로 변경 (partition table을 일반 table로 변경)
SQL> select range_no 2 from range_part partition (range_q1); 0 row selected.	partition table의 data조회 → 0row(없음)
SQL> select range_no from range_part_ex; RANGE_NO ---------- 1 1 row selected.	partition data를 이동한 일반 table의 data를 조회 → 1row(존재함)
SQL> alter table range_part move partition range_q3 tablespace test_part_max; Table 'RANGE_PART' altered.	partition tablespace의 변경
SQL> alter table range_part truncate partition range_q3; Table 'RANGE_PART' altered.	table 내용 삭제
SQL> select * from range_part partition(range_q3); 0 row selected.	0row (삭제됨)

3.3 Hash partition table 확인

3.3.1 설명

- Partitioning column의 Partitioning Key 값에 Hash 함수를 적용하여 Data를 분할하는 방식
- 데이터 이력관리의 목적보다 성능 향상의 목적으로 나온 개념이다.

  Hash Partition은 Range Partition에서 범위를 기반으로 나누었을 경우 특정범위의 분포도가 몰려서 각기 Size가 다르게 되는 것을 보완하여, 일정한 분포를 가진 파티션으로 나누고 균등한 데이터 분포도를 이용한 병렬처리로 퍼포먼스를 보다 향상시킬 수 있다.
- Hash Partition에서 Table은 단지 논리적인 구조이며 실제 데이터가 물리적으로 저장되는 곳은 Partition으로 나누어진 Tablespace에 저장이 된다.

3.3.2 수행

수행 순서	시나리오
1	파티션테이블 생성
2	데이터 입력
3	파티션 추가
4	파티션 삭제 (지원하지 않음)
5	파티션 이름 변경
6	파티션 변경 (EXCHANGE)
7	파티션 테이블스페이스 변경
8	파티션 데이터 TRUNCATE

3.3.3 결과

시나리오	설명
[tibero@T1:/home/tibero/project]$ tbsql test/test tbSQL 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. Connected to Tibero.	test 사용자에 접속
SQL> CREATE TABLE TEST.HASH_PART (HASH_NO NUMBER, HASH_YEAR CHAR(4) NOT NULL, HASH_MONTH CHAR(2) NOT NULL, HASH_DAY CHAR(2) NOT NULL, HASH_NAME VARCHAR2(30), HASH NUMBER) PARTITION BY HASH (HASH_NO) (PARTITION HASH_PART1 TABLESPACE TEST_PART1, PARTITION HASH_PART2 TABLESPACE TEST_PART2, PARTITION HASH_PART3 TABLESPACE TEST_PART3, PARTITION HASH_PART4 TABLESPACE TEST_PART4); Table 'TEST.HASH_PART' created.	hash 함수 이용하여 test 사용자의 hash_part table과 partition table 생성
SQL> INSERT INTO TEST.HASH_PART VALUES(1, 2004, 06, 12, 'SCOTT', 2500); 1 row inserted. SQL> INSERT INTO TEST.HASH_PART VALUES(2, 2005, 06, 17, 'JONES', 4300); 1 row inserted. SQL> INSERT INTO TEST.HASH_PART VALUES(3, 2005, 12, 12, 'MILLER', 1200); 1 row inserted. SQL> INSERT INTO TEST.HASH_PART VALUES(4, 2006, 06, 22, 'FORD', 5200); 1 row inserted. SQL> INSERT INTO TEST.HASH_PART VALUES(5, 2005, 01, 01,	hash_part table에 데이터 입력

'LION', 2200); 1 row inserted. SQL> INSERT INTO TEST.HASH_PART VALUES(6, 2006, 12, 22, 'TIGER', 3300); 1 row inserted. SQL> commit; Commit completed.	
SQL> select * from test.hash_part; HASH_NO HASH_YEAR HASH_MONTH HASH_DAY HASH_NAME ---------- --------- ---------- -------- ---------- HASH ---------- 2 2005 6 17 JONES 4300 3 2005 12 12 MILLER 1200 5 2005 1 1 LION 2200 1 2004 6 12 SCOTT 2500 6 2006 12 22 TIGER 3300 4 2006 6 22 FORD 5200 6 rows selected.	test 사용자의 hash_part table 조회
SQL> alter table test.hash_part drop partition hash_part4; TBR-7141: Specified partition 'HASH_PART4' does not exist.	pratition table 삭제 지원X
SQL> alter table test.hash_part rename partition hash_part4 to hash_part_four; Table 'TEST.HASH_PART' altered.	partition table hash_part4의 이름을 hash_part_four로 변경
SQL> COL TABLE_NAME FOR A20 SQL> COL PARTITION_NAME FOR A20 SQL> select table_name, partition_name from user_tab_partitions where table_name='HASH_PART'; TABLE_NAME PARTITION_NAME -------------------- -------------------- HASH_PART HASH_PART1 HASH_PART HASH_PART2 HASH_PART HASH_PART3 HASH_PART HASH_PART_FOUR	바뀐 이름 조회

4 rows selected.	
SQL> CREATE TABLE TEST.HASH_PART_EX (HASH_NO NUMBER, HASH_YEAR CHAR(4) NOT NULL, HASH_MONTH CHAR(2) NOT NULL, HASH_DAY CHAR(2) NOT NULL, HASH_NAME VARCHAR2(30), HASH NUMBER) TABLESPACE TEST_PART1; Table 'TEST.HASH_PART_EX' created.	partition data를 일반 table로 이동하기 위한 hash_part_ex 생성
SQL> ALTER TABLE TEST.HASH_PART EXCHANGE PARTITION HASH_PART2 WITH TABLE TEST.HASH_PART_EX; Table 'TEST.HASH_PART' altered.	partition data를 일반 table(HASH_PART_EX)로 변경
SQL> SELECT COUNT(*) FROM TEST.HASH_PART PARTITION(HASH_PART2); COUNT(*) ---------- 0 1 row selected.	partition table의 data를 조회
SQL> SELECT COUNT(*) FROM TEST.HASH_PART_EX; COUNT(*) ---------- 2 1 row selected.	partition data를 이동한 일반 table의 data 조회
SQL> ALTER TABLE TEST.HASH_PART MOVE PARTITION HASH_PART3 TABLESPACE TEST_PART4; Table 'TEST.HASH_PART' altered. SQL> SELECT COUNT(*) FROM TEST.HASH_PART PARTITION(HASH_PART_FOUR); COUNT(*) ---------- 1 1 row selected.	partition tablespace 변경 hash_part_four partition table 건수 조회
SQL> ALTER TABLE TEST.HASH_PART TRUNCATE PARTITION HASH_PART_FOUR; Table 'TEST.HASH_PART' altered. SQL> SELECT COUNT(*) FROM TEST.HASH_PART PARTITION(HASH_PART_FOUR); COUNT(*) ----------	partition table내에 데이터 삭제 hash_part_four partition table 건수 조회

0 1 row selected.	

3.4 List partition table 확인

3.4.1 설명

- Partitioning Column의 특정 값으로 분할하는 방식
- 데이터 분포도가 낮지 않고, 균등하게 분포되어 있을때 유용하다.
- Composite Partition에서 'Range-List'일 경우 그 효율이 더욱 높아진다.
- 다른 파티션 방식처럼 다중 컬럼을 지원하지 않고 단일 컬럼만 가능하다.

3.4.2 수행

수행 순서	시나리오
1	파티션테이블 생성
2	데이터 입력
3	파티션 추가 (add)
4	파티션 제거 (drop)
5	파티션 이름 변경
6	파티션 변경 (EXCHANGE)
7	파티션 테이블스페이스 변경
8	파티션 데이터 TRUNCATE

3.4.3 결과

시나리오	설명
[tibero@T1:/home/tibero/project]$ tbsql test/test tbSQL 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. Connected to Tibero.	test 사용자에 접속
SQL> CREATE TABLE TEST.LIST_PART (LIST_NO NUMBER NOT NULL, LIST_NAME VARCHAR2(10), LIST_JOB VARCHAR2(9), LIST_MGR NUMBER(4), LIST_HIREDATE DATE, LIST_SAL NUMBER(7, 2), LIST_COMM NUMBER(7, 2), LIST_DEPTNO NUMBER(2)) PARTITION BY LIST (LIST_JOB) (PARTITION LIST_PART1 VALUES ('MANAGER') TABLESPACE TEST_PART1, PARTITION LIST_PART2 VALUES ('SALESMAN') TABLESPACE TEST_PART2, PARTITION LIST_PART3 VALUES ('ANALYST') TABLESPACE TEST_PART3, PARTITION LIST_PART4 VALUES ('PRESIDENT', 'CLERK') TABLESPACE TEST_PART4); Table 'TEST.LIST_PART' created.	test 사용자의 list_part table과 partition table 생성

SQL> INSERT INTO LIST_PART VALUES(1, 'SMITH', 'CLERK', 7902, SYSDATE, 800, NULL, 20); 1 row inserted. SQL> INSERT INTO LIST_PART VALUES(2, 'ALLEN', 'SALESMAN', 7698, SYSDATE, 1600, 300, 30); 1 row inserted. SQL> INSERT INTO LIST_PART VALUES(3, 'WARD', 'SALESMAN', 7698, SYSDATE, 1250, 500, 30); 1 row inserted. SQL> INSERT INTO LIST_PART VALUES(4, 'JONES', 'MANAGER', 7839, SYSDATE, 2975, NULL, 20); 1 row inserted. SQL> INSERT INTO LIST_PART VALUES(5, 'MARTIN', 'SALESMAN', 7698, SYSDATE, 1250, 1400, 30); 1 row inserted. SQL> INSERT INTO LIST_PART VALUES(6, 'BLAKE', 'MANAGER', 7839, SYSDATE, 2850, NULL, 30); 1 row inserted. SQL> INSERT INTO LIST_PART VALUES(7, 'CLARK', 'MANAGER', 7839, SYSDATE, 2450, NULL, 10); 1 row inserted. SQL> INSERT INTO LIST_PART VALUES(8, 'SCOTT', 'ANALYST', 7566, SYSDATE, 3000, NULL, 20); 1 row inserted. SQL> INSERT INTO LIST_PART VALUES(9, 'KING', 'PRESIDENT', NULL, SYSDATE, 5000, NULL, 10); 1 row inserted. SQL> INSERT INTO LIST_PART VALUES(10, 'TURNER', 'SALESMAN', 7698,SYSDATE, 1500, 0, 30); 1 row inserted. SQL> INSERT INTO LIST_PART VALUES(11, 'ADAMS', 'CLERK', 7788,SYSDATE,1100,NULL,20); 1 row inserted.	list_part table에 데이터 입력

SQL> INSERT INTO LIST_PART VALUES(12, 'JAMES', 'CLERK', 7698, SYSDATE, 950, NULL, 30); 1 row inserted. SQL> INSERT INTO LIST_PART VALUES(13, 'FORD', 'ANALYST', 7566, SYSDATE, 3000, NULL, 20); 1 row inserted. SQL> INSERT INTO LIST_PART VALUES(14, 'MILLER', 'CLERK', 7782, SYSDATE, 1300, NULL, 10); 1 row inserted. SQL> COMMIT; Commit completed.	
SQL> SELECT LIST_NO FROM LIST_PART PARTITION(LIST_PART1); LIST_NO ---------- 4 6 7 3 rows selected.	partition table list_part1에있는 list_no 값 조회 (LIST_JOB='MANAGER)
SQL> SELECT LIST_NO FROM LIST_PART PARTITION(LIST_PART2); LIST_NO ---------- 2 3 5 10 4 rows selected.	partition table list_part2에있는 list_no 값 조회 (LIST_JOB='SALESMAN')
SQL> SELECT LIST_NO FROM LIST_PART PARTITION(LIST_PART3); LIST_NO ---------- 8 13 2 rows selected.	partition table list_part3에있는 list_no 값 조회 (LIST_JOB='ANALYST')
SQL> SELECT LIST_NO FROM LIST_PART PARTITION(LIST_PART4); LIST_NO ---------- 1 9 11 12 14	partition table list_part4에있는 list_no 값 조회 (LIST_JOB='PRESIDENT','CLERK')

5 rows selected.	
SQL> ALTER TABLE LIST_PART ADD PARTITION LIST_PART_MAX VALUES('DUMMY') TABLESPACE TEST_PART_MAX; Table 'LIST_PART' altered. SQL> select table_name, partition_name from user_tab_partitions where table_name='LIST_PART'; TABLE_NAME PARTITION_NAME ---------- ---------- LIST_PART LIST_PART1 LIST_PART LIST_PART2 LIST_PART LIST_PART3 LIST_PART LIST_PART4 LIST_PART LIST_PART_MAX 5 rows selected.	LIST_JOB이 DUMMY인 partition table list_part_max 생성후 조회
SQL> ALTER TABLE LIST_PART 2 DROP PARTITION LIST_PART_MAX; Table 'LIST_PART' altered.	list_part table의 list_part_max partition table 삭제(drop)
SQL> ALTER TABLE TEST.LIST_PART 2 RENAME PARTITION LIST_PART4 TO LIST_PART_FOUR; Table 'TEST.LIST_PART' altered.	LIST_PART_4 partition table의 이름 LIST_PART_FOUR로 변경
SQL> CREATE TABLE TEST.LIST_PART_EX (LIST_NO NUMBER NOT NULL, LIST_NAME VARCHAR2(10), LIST_JOB VARCHAR2(9), LIST_MGR NUMBER(4), LIST_HIREDATE DATE, LIST_SAL NUMBER(7, 2), LIST_COMM NUMBER(7, 2), LIST_DEPTNO NUMBER(2)); Table 'TEST.LIST_PART_EX' created.	partition data를 일반 table로 이동하기 위한 table list_part_ex 생성
SQL> SELECT COUNT(*) FROM TEST.LIST_PART PARTITION(LIST_PART3); COUNT(*) ---------- 2 1 row selected.	partition table list_part3의 건수 조회
SQL> ALTER TABLE TEST.LIST_PART EXCHANGE PARTITION LIST_PART3 WITH TABLE TEST.LIST_PART_EX; Table 'TEST.LIST_PART' altered. SQL> SELECT COUNT(*) FROM TEST.LIST_PART PARTITION(LIST_PART3);	partition table list_part3에 있는 partition data를 일반 table인 list_part_ex로 변경 partition table list_part3의 건수 조회 → 0 (일반 테이블로 데이터 이동)

COUNT(*) ---------- 0 1 row selected.	
SQL> SELECT COUNT(*) FROM TEST.LIST_PART_EX; COUNT(*) ---------- 2 1 row selected.	일반 테이블 list_part_ex의 건수 조회 → 2(partition data가 옮겨짐)
SQL> ALTER TABLE TEST.LIST_PART MOVE PARTITION LIST_PART1 TABLESPACE TEST_PART2; Table 'TEST.LIST_PART' altered.	TEST_PART1 tablespace에 있는 partition table list_part1을 TEST_PART2 tablespace로 이동
SQL> SELECT COUNT(*) FROM TEST.LIST_PART PARTITION(LIST_PART2); COUNT(*) ---------- 4 1 row selected. SQL> ALTER TABLE TEST.LIST_PART TRUNCATE PARTITION LIST_PART2; Table 'TEST.LIST_PART' altered. SQL> SELECT COUNT(*) FROM TEST.LIST_PART PARTITION(LIST_PART2); COUNT(*) ---------- 0 1 row selected.	partition table LIST_PART2의 건수 조회 → 4 partition data truncate(삭제) partition table LIST_PART2의 건수 조회 → 0 (삭제됨)

4. 모니터링

4.1 쿼리 실행 계획 및 쿼리 또는 세션에 대한 통계자료(실행시간, IO) 제공 여부 확인

4.1.1 설명

- CLI 방식의 클라이언트 프로그램 화면 또는 별도 trace 파일에서 확인

4.1.2 수행

수행 순서	시나리오
1	테스트 테이블(TIBERO.PLAN_TEST) 생성
2	테스트 테이블(TIBERO.PLAN_TEST)에 데이터 입력
3	테스트 테이블(TIBERO.PLAN_TEST)에 인덱스(TIBERO.PLAN_TEST_IDX) 생성
4	쿼리 실행 계획(PLAN) 보기 활성화

5	쿼리 실행 후 쿼리 실행 계획(PLAN) 확인
6	세션 통계자료(실행시간,IO)확인을 위한 별도 trace 활성화
7	쿼리 실행 후 별도 trace 확인

4.1.3 결과

시나리오	설명
[tibero@T1:/home/tibero/project]$ tbsql tibero/tmax tbSQL 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. Connected to Tibero.	TIBERO 사용자에 접속

SQL> CREATE TABLE PLAN_TEST(C1 NUMBER,C2 NUMBER, C3 NUMBER);
 	
Table 'PLAN_TEST' created.
 	
SQL> declare
begin
for i in 1..10000 loop
insert into plan_test values(i,i,i);
end loop;
end;
/
 	
PSM completed.
 	
SQL> commit;
 	
Commit completed.
 	
SQL> CREATE INDEX PLAN_TEST_IDX ON PLAN_TEST(C1);
 	
Index 'PLAN_TEST_IDX' created.
	TABLE, DATA, INDEX 생성

SQL> set lines 200 SQL> set autot traceonly exp planstat SQL> select * from plan_test where c1 between 10 and 100; SQL ID: 63qt2v8qbnum4 Child number: 150 Plan hash value: 525970105 Execution Plan ---------- ---------- ---------- 1 TABLE ACCESS (ROWID): PLAN_TEST (Cost:3, %%CPU:0, Rows:145) 2 INDEX (RANGE SCAN): PLAN_TEST_IDX (Cost:2, %%CPU:0, Rows:145) Predicate Information ---------- ---------- ---------- 2 - access: ("PLAN_TEST"."C1" >= 10) AND ("PLAN_TEST"."C1" <= 100) (0.999 * 0.015)	쿼리 실행 계획(PLAN) 보기 활성화 쿼리 실행 계획(PLAN) 확인

Note --- --- --- 2 - dynamic sampling used for this table (32 blocks) Execution Stat --- --- --- 1 TABLE ACCESS (ROWID): PLAN_TEST (Time:0. ms, Rows:0, Starts:0) 2 INDEX (RANGE SCAN): PLAN_TEST_IDX (Time:0. ms, Rows:0, Starts:0)	
SQL> set autot off SQL> alter session set sql_trace=y; Session altered. SQL> select * from plan_test where c1 between 10 and 100; C1 C2 C3 ---------- ---------- ---------- 10 10 10 11 11 11 12 12 12 13 13 13 14 14 14 15 15 15 16 16 16 17 17 17 18 18 18 19 19 19 20 20 20 21 21 21 22 22 22 23 23 23 24 24 24 25 25 25 26 26 26 27 27 27 28 28 28 29 29 29 30 30 30 31 31 31 . . . 91 rows selected. SQL> exit Disconnected.	SESSION 통계자료(실행시간,IO) 확인을 위한 별도 TRACE 활성화 쿼리 실행 후 별도 TRACE 확인
[tibero@T1:/tibero/tibero6/instance/tibero/log/sqltrace]$ ls -al total 4 drwxr-xr-x 2 tibero dba 39 Nov 4 13:10 . drwxr-xr-x 8 tibero dba 123 Nov 4 08:27 .. -rw-r--r-- 1 tibero dba 1421 Nov 4 13:10 tb_sqltrc_1814_63_827.trc	경로 (/tibero/tibero6/instance/tibero/log/sqltrace)

[tibero@T1:/tibero/tibero6/instance/tibero/log/sqltrace]$ tbprof tb_sqltrc_1814_63_827.trc tb_sqltrc_1814_63_827.log TBPROF 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. [tibero@T1:/tibero/tibero6/instance/tibero/log/sqltrace]$ vi tb_sqltrc_1814_63_827.log TBPROF 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. input file name : tb_sqltrc_1814_63_827.trc output file name : tb_sqltrc_1814_63_827.log sort option : default aggregate : yes sys : yes print : all ========== count: number of times the procedure was executed cpu: cpu time(seconds) this is not quite accurate due to threaded architecture elapsed: elapsed time(seconds) disk: number of physical reads from disk query: number of blocks for consistent read current: number of blocks in current mode rows: number of rows processed "tb_sqltrc_1814_63_827.log" 114L, 5238C	쿼리 실행 후 별도 TRACE 확인

4.2.1 설명

- CLI 방식의 클라이언트 프로그램 화면 또는 별도 trace 파일에서 확인
- 감사(Auditing)는 데이터베이스 내에서 지정된 사용자의 동작을 기록하는 보안 기술이다. 관리자는 감사 기능을 통해 특정 동작 또는 특정 사용자에 대해 별도의 로그를 남김으로써 데이터베이스를 보다 효과적으로 보호할 수 있다.

4.2.2 수행

수행 순서	시나리오
1	감사(audit) 기능 활성화를 위한 환경파일(TIP) 설정
2	테스트 테이블(TIBERO.AUTDIT_TEST) 생성
3	테스트 테이블(TIBERO.AUTDIT_TEST)에 DML 감사(AUDIT) 설정
4	테스트 테이블(TIBERO.AUTDIT_TEST)에 DDL 감사(AUDIT) 설정
5	테스트 테이블(TIBERO.AUTDIT_TEST)에 DML, DDL 쿼리 수행
6	감사(AUDIT) 로그 확인

4.2.3 결과

시나리오	설명

[tibero@T1:/home/tibero/audit]$ vi $TB_HOME/config/$TB_SID.tip
 	
#AUDIT Setting
AUDIT_TRAIL=OS
AUDIT_SYS_OPERATIONS=Y
AUDIT_FILE_DEST=/home/tibero/audit
	감사(audit) 기능 활성화를 위한 환경 파일 설정

[tibero@T1:/home/tibero/audit]$ tbdown	tibero 재기동

Tibero instance terminated (NORMAL mode). [tibero@T1:/home/tibero/audit]$ tbboot Listener port = 8629 Tibero 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. Tibero instance started up (NORMAL mode). [tibero@T1:/home/tibero/audit]$ tbsql sys/tibero tbSQL 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. Connected to Tibero.	
SQL> create table tibero.audit_test(id number); Table 'TIBERO.AUDIT_TEST' created. SQL> AUDIT insert on tibero.audit_test BY SESSION WHENEVER SUCCESSFUL; Audited. SQL> AUDIT update on tibero.audit_test BY SESSION WHENEVER SUCCESSFUL; Audited. SQL> AUDIT delete on tibero.audit_test BY SESSION WHENEVER SUCCESSFUL; Audited. SQL> AUDIT create table by tibero; Audited. SQL> insert into tibero.audit_test values(1); 1 row inserted. SQL> commit; Commit completed.	tibero.audit_test table생성 DML(insert, update, delete) AUDIT 설정 DDL(create) AUDIT 설정
SQL> update tibero.audit_test set id = 2; 1 row updated. SQL> commit;	TIBERO.AUDIT_TEST에 DML,DDL 쿼리 수행

Commit completed. SQL> delete from tibero.audit_test ; 1 row deleted. SQL> commit; Commit completed. SQL> q Disconnected.	
[tibero@T1:/tibero/tibero6/instance/tibero/log/sqltrace]$ ls -al /home/tibero/audit total 8 drwxr-xr-x 2 tibero dba 23 Nov 4 13:47 . drwx------. 13 tibero dba 4096 Nov 4 13:46 .. -rw-r--r-- 1 tibero dba 2542 Nov 4 13:49 audit.log	경로(/home/tibero/audit)
[tibero@T1:/home/tibero/audit]$ vi audit.log 2022/11/04 13:47:21.889 63 SESS_ID:[63] SERIAL_NO:[21] STMT_ID:[0] USER_NAME:[SYS] USER_HOST:[127.0.0.1] OS_USER:[tibero] CLIENT_ID:[tbsql] PID:[5239] SQLTEXT:[CONNECT] 2022/11/04 13:47:31.720 63 SESS_ID:[63] SERIAL_NO:[21] STMT_ID:[0] USER_NAME:[SYS] USER_HOST:[127.0.0.1] OS_USER:[tibero] CLIENT_ID:[tbsql] PID:[5239] SQLTEXT:[drop table tibero.audit_test] 2022/11/04 13:48:04.335 63 SESS_ID:[63] SERIAL_NO:[21] STMT_ID:[0] USER_NAME:[SYS] USER_HOST:[127.0.0.1] OS_USER:[tibero] CLIENT_ID:[tbsql] PID:[5239] SQLTEXT:[create table tibero.audit_test(id number)] 2022/11/04 13:48:11.721 63 SESS_ID:[63] SERIAL_NO:[21] STMT_ID:[0] USER_NAME:[SYS] USER_HOST:[127.0.0.1] OS_USER:[tibero] CLIENT_ID:[tbsql] PID:[5239] SQLTEXT:[AUDIT insert on tibero.audit_test BY SESSION WHENEVER SUCCESSFUL] 2022/11/04 13:48:20.909 63 SESS_ID:[63] SERIAL_NO:[21] STMT_ID:[0] USER_NAME:[SYS] USER_HOST:[127.0.0.1] OS_USER:[tibero] CLIENT_ID:[tbsql] PID:[5239] SQLTEXT:[AUDIT update on tibero.audit_test BY SESSION WHENEVER SUCCESSFUL] 2022/11/04 13:48:25.847 63 SESS_ID:[63] SERIAL_NO:[21] STMT_ID:[0] USER_NAME:[SYS] USER_HOST:[127.0.0.1] OS_USER:[tibero] CLIENT_ID:[tbsql] PID:[5239] SQLTEXT:[AUDIT delete on tibero.audit_test BY SESSION WHENEVER SUCCESSFUL]	audit.log 확인

5. 백업/복구

5.1 온라인 백업 후 완전 복구 기능

5.1.1 설명

- archive log 정보

→ v$archive_dest_files : log_archive_dest 내의 archive log files의 정보 표시 (현재)

→ v$archive_log : archive log 정보 표시(log switch할 때 archive log file 만들어질 때 입력됨)

- 온라인 백업 후 완전 복구

구분	설명

ARCHIVELOG 모드	온라인 백업(Online Backup) 또는 Hot Backup이라 한다. 데이터베이스가 운영 중일 때 백업할 수 있다. 백업이 가능한 파일은 컨트롤 파일의 생성문과 데이터 파일, 아카이브 로그 파일 등이 있다. 복구는 백업 된 아카이브 로그 파일의 시점에 따라 데이터 파일의 백업 시점 전으로 복구할 수 있다.
NOARCHIVELOG 모드	오프라인 백업(Offline Backup) 또는 Cold Backup이라 한다. 기본적으로 데이터베이스는 NOARCHIVELOG 모드이다. 데이터베이스를 구성하는 전체 파일은 반드시 Tibero가 정상적으로 종료된 상태에서 백업한다. 백업 때문에 서비스가 중지되면 안 된다. 복구는 데이터베이스를 백업받은 시점으로부터 복구할 수 있다.

구분	설명
완전 복구	Archive Log 파일과 Online Log 파일을 모두 사용해서 가장 최근 Log까지 모두 반영한다.
불완전 복구	Log 파일 일부만 적용 하거나 특정 시점으로 복구가 가능하다. – Point-in-Time 복구 – Redo record의 일부만 적용 불완전 복구를 하게 되면 반드시 resetlogs로 데이터베이스를 기동해야 한다. resetlogs에 대한 자세한 내용은 "RESETLOGS"를 참고한다.

5.1.2 수행

수행 순서	시나리오
1	테스트용 데이터 생성 - 테이블 스페이스 생성 - 테스트 유저 및 테이블 생성
2	사전 확인 - 테이블 건수 조회 - 테이블 스페이스 확인 - 데이터 파일 확인
3	Begin Backup 수행 및 확인
4	핫 백업 진행
5	End Backup 수행 및 로그 스위치 수행
6	데이터 입력
7	데이터 조회
8	티베로 종료 및 데이터 파일 전체 삭제
9	티베로 기동하여 마운트 모드 및 장애 상황 확인
10	티베로 종료 및 핫 백업 원복
11	티베로 마운트 모트 기동 및 복구 수행
12	티베로 종료 및 티베로 기동
13	테이블 건수 조회

5.1.3 결과

시나리오	설명
[tibero@T1:/home/tibero/project]$ tbsql sys/tibero tbSQL 6	sql 접속

TmaxData Corporation Copyright (c) 2008-. All rights reserved. Connected to Tibero.	
SQL> select tablespace_name from dba_tablespace; TABLESPACE_NAME -- ---------- SYSTEM UNDO TEMP USR SYSSUB 5 rows selected.	tablespace 조회
SQL> CREATE TABLESPACE TS_TEST DATAFILE 'test001.dtf' SIZE 16M AUTOEXTEND ON NEXT 16M MAXSIZE 1G, 'test002.dtf' SIZE 16M AUTOEXTEND ON NEXT 16M MAXSIZE 1G EXTENT MANAGEMENT LOCAL AUTOALLOCATE; Tablespace 'TS_TEST' created. SQL> CREATE TABLESPACE TS_TEST_IDX DATAFILE 'test_idx_001.dtf' SIZE 8M AUTOEXTEND ON NEXT 8M MAXSIZE 1G EXTENT MANAGEMENT LOCAL AUTOALLOCATE; Tablespace 'TS_TEST_IDX' created. SQL> CREATE USER TEST IDENTIFIED BY TEST DEFAULT TABLESPACE TS_TEST; User 'TEST' created. SQL> GRANT DBA TO TEST; Granted. SQL> CONN TEST/TEST Connected to Tibero. SQL> CREATE TABLE TEST.T1 (ID NUMBER, ANAME VARCHAR2(32), BNAME VARCHAR2(32), ID2 NUMBER) TABLESPACE TS_TEST; Table 'TEST.T1' created. SQL> CREATE INDEX IDX_T1 ON T1(ID, ANAME) TABLESPACE TS_TEST_IDX; Index 'IDX_T1' created.	- tablespace 생성 - test 사용자 생성 및 권한 부여 - test.t1 table 생성 및 index 입력

SQL> SELECT COUNT(*) FROM TEST.T1; COUNT(*) ---------- 0 1 row selected.	
SQL> SELECT TABLESPACE_NAME FROM DBA_TABLESPACES; TABLESPACE_NAME ---------- ---- ---------- SYSTEM UNDO TEMP USR SYSSUB TS_TEST TS_TEST_IDX 7 rows selected. SQL> SELECT FILE_NAME FROM DBA_DATA_FILES; FILE_NAME ---------- ---- ---------- /tibero/tbdata/tibero/system001.dtf /tibero/tbdata/tibero/undo001.dtf /tibero/tbdata/tibero/usr001.dtf /tibero/tbdata/tibero/syssub001.dtf /tibero/tbdata/tibero/test001.dtf /tibero/tbdata/tibero/test002.dtf /tibero/tbdata/tibero/test_idx_001.dtf 7 rows selected.	tablespace 경로 및 이름 조회 datafile 경로 및 이름 조회
SQL> ALTER DATABASE BEGIN BACKUP; Database altered. SQL> SELECT * FROM V$BACKUP; FILE# STATUS CHANGE# ---------- ---------- ---------- TIME ---------- 0 ACTIVE 36797 2022/11/04 1 ACTIVE 36797 2022/11/04 2 ACTIVE 36797 2022/11/04 3 ACTIVE 36797 2022/11/04	backup 시작 (Tibero에 백업의 시작과 종료를 통보) 현재 상태 조회(active)

4 ACTIVE 36797 2022/11/04 5 ACTIVE 36797 2022/11/04 6 ACTIVE 36797 2022/11/04 7 rows selected. SQL> exit Disconnected.	
[tibero@T1:/home/tibero/project]$ ls -al /tibero/tbdata/tibero/*.dtf -rw------- 1 tibero dba 398458880 Nov 4 09:29 /tibero/tbdata/tibero/syssub001.dtf -rw------- 1 tibero dba 398458880 Nov 4 09:30 /tibero/tbdata/tibero/system001.dtf -rw------- 1 tibero dba 398458880 Nov 4 08:25 /tibero/tbdata/tibero/temp001.dtf -rw------- 1 tibero dba 16777216 Nov 4 09:29 /tibero/tbdata/tibero/test001.dtf -rw------- 1 tibero dba 16777216 Nov 4 09:29 /tibero/tbdata/tibero/test002.dtf -rw------- 1 tibero dba 8388608 Nov 4 09:29 /tibero/tbdata/tibero/test_idx_001.dtf -rw------- 1 tibero dba 398458880 Nov 4 09:30 /tibero/tbdata/tibero/undo001.dtf -rw------- 1 tibero dba 104857600 Nov 4 09:29 /tibero/tbdata/tibero/usr001.dtf [tibero@T1:/home/tibero/project]$ cd /tibero/tbdata/tibero [tibero@T1:/tibero/tbdata/tibero]$ mkdir -p /tibero/s/tibero_hot [tibero@T1:/tibero/tbdata/tibero]$ cp /tibero/tbdata/tibero/*.dtf /tibero/s/${TB_SID}_hot/. [tibero@T1:/tibero/tbdata/tibero]$ ls -al /tibero/s/tibero_hot total 1699832 drwxr-xr-x 2 tibero dba 166 Nov 4 09:31 . drwxr-xr-x 7 tibero dba 277 Nov 4 09:30 .. -rw------- 1 tibero dba 398458880 Nov 4 09:31 syssub001.dtf -rw------- 1 tibero dba 398458880 Nov 4 09:31 system001.dtf -rw------- 1 tibero dba 398458880 Nov 4 09:31 temp001.dtf -rw------- 1 tibero dba 16777216 Nov 4 09:31 test001.dtf -rw------- 1 tibero dba 16777216 Nov 4 09:31 test002.dtf -rw------- 1 tibero dba 8388608 Nov 4 09:31 test_idx_001.dtf -rw------- 1 tibero dba 398458880 Nov 4 09:31 undo001.dtf -rw------- 1 tibero dba 104857600 Nov 4 09:31 usr001.dtf	존재하는 datafile 조회 /tibero/s/ 경로에 datafile들 복사
[tibero@T1:/tibero/tbdata/tibero]$ tbsql sys/tibero tbSQL 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. Connected to Tibero. SQL> alter database end backup;	online backup 끝내고 상태 조회 (not active)

Database altered. SQL> select * from v$backup; FILE# STATUS CHANGE# ---------- ---------- ---------- TIME ---------- 0 NOT ACTIVE 36797 2022/11/04 1 NOT ACTIVE 36797 2022/11/04 2 NOT ACTIVE 36797 2022/11/04 3 NOT ACTIVE 36797 2022/11/04 4 NOT ACTIVE 36797 2022/11/04 5 NOT ACTIVE 36797 2022/11/04 6 NOT ACTIVE 36797 2022/11/04 7 rows selected.	
SQL> alter system switch logfile; System altered. SQL> INSERT INTO TEST.T1 SELECT ROWNUM, 'A'||TO_CHAR(ROWNUM), 'B'||TO_CHAR(ROWNUM), ROUND(ROWNUM/50) FROM DUAL CONNECT BY ROWNUM<=50000; 50000 rows inserted. SQL> commit; Commit completed. SQL> select count(*) from test.t1; COUNT(*) ---------- 50000 1 row selected. SQL> exit Disconnected.	log switch test.t1 table에 데이터 입력후 조회

[tibero@T1:/tibero/tbdata/tibero]$tbdown Tibero instance terminated (NORMAL mode). [tibero@T1:/tibero/tbdata/tibero]$cd /tibero/tbdata/tibero [tibero@T1:/tibero/tbdata/tibero]$ rm *.dtf [tibero@T1:/tibero/tbdata/tibero]$ ls -al total 44580 drwxr-xr-x 7 tibero dba 117 Nov 4 09:35 . drwxr-xr-x 3 tibero dba 20 Oct 11 13:34 .. drwxr-xr-x 2 tibero dba 78 Nov 4 09:32 arch -rw------- 1 tibero dba 22822912 Nov 4 09:32 c1.ctl -rw------- 1 tibero dba 22822912 Nov 4 09:32 c2.ctl drwx------ 2 tibero dba 70 Nov 4 08:27 java -r-------- 1 tibero dba 24 Nov 4 08:25 .passwd drwxr-xr-x 2 tibero dba 91 Nov 4 08:25 redo1 drwxr-xr-x 2 tibero dba 91 Nov 4 08:25 redo2 drwxr-xr-x 2 tibero dba 166 Nov 4 09:31 tibero_hot [tibero@T1:/tibero/tbdata/tibero]$ ls -al /tibero/s/tibero_hot total 1699832 drwxr-xr-x 2 tibero dba 166 Nov 4 09:31 . drwxr-xr-x 7 tibero dba 117 Nov 4 09:35 .. -rw------- 1 tibero dba 398458880 Nov 4 09:31 syssub001.dtf -rw------- 1 tibero dba 398458880 Nov 4 09:31 system001.dtf -rw------- 1 tibero dba 398458880 Nov 4 09:31 temp001.dtf -rw------- 1 tibero dba 16777216 Nov 4 09:31 test001.dtf -rw------- 1 tibero dba 16777216 Nov 4 09:31 test002.dtf -rw------- 1 tibero dba 8388608 Nov 4 09:31 test_idx_001.dtf -rw------- 1 tibero dba 398458880 Nov 4 09:31 undo001.dtf -rw------- 1 tibero dba 104857600 Nov 4 09:31 usr001.dtf	datafile 삭제
[tibero@T1:/tibero/tbdata/tibero]$ tbboot Listener port = 8629 ********** * Critical Warning : Raise svmode failed. The reason is * TBR-1024 : Database needs media recovery: open failed(/tibero/tbdata/tibero/system001.dtf). * Current server mode is MOUNT. ********** Tibero 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. Tibero instance started suspended at MOUNT mode. [tibero@T1:/tibero/tbdata/tibero]$tbdown Tibero instance terminated (NORMAL mode).	tibero 기동하여 mount mode 및 장애 확인후 종료
[tibero@T1:/tibero/tbdata/tibero]$ cp -r /tibero/s/tibero_hot/*.dtf /tibero/tbdata/tibero [tibero@T1:/tibero/tbdata/tibero]$ ls -al /tibero/tbdata/tibero total 1744412 drwxr-xr-x 7 tibero dba 277 Nov 4 09:36 . drwxr-xr-x 3 tibero dba 20 Oct 11 13:34 .. drwxr-xr-x 2 tibero dba 78 Nov 4 09:32 arch -rw------- 1 tibero dba 22822912 Nov 4 09:32 c1.ctl -rw------- 1 tibero dba 22822912 Nov 4 09:32 c2.ctl drwx------ 2 tibero dba 70 Nov 4 08:27 java -r-------- 1 tibero dba 24 Nov 4 08:25 .passwd drwxr-xr-x 2 tibero dba 91 Nov 4 08:25 redo1 drwxr-xr-x 2 tibero dba 91 Nov 4 08:25 redo2	backup 원복

-rw------- 1 tibero dba 398458880 Nov 4 09:36 syssub001.dtf -rw------- 1 tibero dba 398458880 Nov 4 09:36 system001.dtf -rw------- 1 tibero dba 398458880 Nov 4 09:36 temp001.dtf -rw------- 1 tibero dba 16777216 Nov 4 09:36 test001.dtf -rw------- 1 tibero dba 16777216 Nov 4 09:36 test002.dtf -rw------- 1 tibero dba 8388608 Nov 4 09:36 test_idx_001.dtf drwxr-xr-x 2 tibero dba 166 Nov 4 09:31 tibero_hot -rw------- 1 tibero dba 398458880 Nov 4 09:36 undo001.dtf -rw------- 1 tibero dba 104857600 Nov 4 09:36 usr001.dtf	
[tibero@T1:/tibero/tbdata/tibero]$ tbboot mount Listener port = 8629 Tibero 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. Tibero instance started up (MOUNT mode).	tibero 마운드 모드 기동
[tibero@T1:/tibero/tbdata/tibero]$ tbsql sys/tibero tbSQL 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. Connected to Tibero. SQL> alter database recover automatic; Database altered. SQL> q	복구 수행
[tibero@T1:/tibero/tbdata/tibero]$ tbdown tbdown failed. proc info file is deleted. Hint: Please check if the tbsvr instance was already stopped. [tibero@T1:/tibero/tbdata/tibero]$ tbboot Listener port = 8629 Tibero 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. Tibero instance started up (NORMAL mode).	tibero 종료 및 tibero 기동
[tibero@T1:/tibero/tbdata/tibero]$ tbsql sys/tibero tbSQL 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. Connected to Tibero. SQL> select count(*) from test.t1; COUNT(*) ---------- 50000 1 row selected.	테이블 건수 조회

SQL> exit Disconnected.	

5.2 오프라인 백업(콜드 백업) 후 완전 복구 기능

5.2.1 설명

- offline 백업 후 완전 복구

5.2.2 수행

수행 순서	시나리오
1	백업 대상 확인 - 컨트롤 파일, 리두 로그, 데이터 파일, 템프 파일
2	테이블 건수 조회
3	티베로 종료 및 콜드 백업 실행
4	티베로 기동
5	데이터 입력
6	데이터 조회
7	티베로 종료 및 데이터 파일 전체 삭제
8	티베로 기동하여 마운트 모드 및 장애 상황 확인
9	티베로 종료 및 콜드 백업 파일 원복
10	티베로 마운트 모트 기동 및 복구 수행
11	티베로 종료 및 티베로 기동
12	테이블 건수 조회

5.2.3 결과

시나리오	설명
[tibero@T1:/home/tibero/project]$ tbsql sys/tibero tbSQL 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. Connected to Tibero.	sys 사용자로 접속
SQL> select name from v$controlfile; NAME ---------- /tibero/tbdata/tibero/c1.ctl /tibero/tbdata/tibero/c2.ctl 2 rows selected. SQL> select member from v$logfile; MEMBER ---------- /tibero/tbdata/tibero/redo1/log01.log /tibero/tbdata/tibero/redo2/log02.log	controlfile 조회 logfile 조회 datafile 조회 temp file 조회

/tibero/tbdata/tibero/redo1/log11.log /tibero/tbdata/tibero/redo2/log12.log /tibero/tbdata/tibero/redo1/log21.log /tibero/tbdata/tibero/redo2/log22.log /tibero/tbdata/tibero/redo1/log31.log /tibero/tbdata/tibero/redo2/log32.log /tibero/tbdata/tibero/redo1/log41.log /tibero/tbdata/tibero/redo2/log42.log 10 rows selected. SQL> select file_name from dba_datafiles; FILE_NAME ---------- /tibero/tbdata/tibero/system001.dtf /tibero/tbdata/tibero/undo001.dtf /tibero/tbdata/tibero/usr001.dtf /tibero/tbdata/tibero/syssub001.dtf /tibero/tbdata/tibero/test001.dtf /tibero/tbdata/tibero/test002.dtf /tibero/tbdata/tibero/test_idx_001.dtf 7 rows selected. SQL> select file_name from dba_temp_files; FILE_NAME ---------- /tibero/tbdata/tibero/temp001.dtf 1 row selected.	
SQL> select count(*) from test.t1; COUNT(*) ---------- 50000 1 row selected. SQL> exit Disconnected.	table 건수 조회
[tibero@T1:/home/tibero/project]$ tbdown Tibero instance terminated (NORMAL mode). [tibero@T1:/home/tibero/project]$ cp -r /tibero/tbdata/tibero /tibero/s/tibero_bak [tibero@T1:/home/tibero/project]$ ls -al /tibero/s/tibero_bak/tibero total 1744425 drwxrwx--- 1 root vboxsf 4096 Nov 4 10:22 . drwxrwx--- 1 root vboxsf 0 Nov 4 10:22 .. drwxrwx--- 1 root vboxsf 4096 Nov 4 10:22 arch	tibero 종료 및 cold backup 실행

-rwxrwx--- 1 root vboxsf 22822912 Nov 4 14:06 c1.ctl -rwxrwx--- 1 root vboxsf 22822912 Nov 4 14:06 c2.ctl drwxrwx--- 1 root vboxsf 0 Nov 4 10:22 java -rwxrwx--- 1 root vboxsf 24 Nov 4 14:06 .passwd drwxrwx--- 1 root vboxsf 0 Nov 4 10:22 redo1 drwxrwx--- 1 root vboxsf 0 Nov 4 10:22 redo2 -rwxrwx--- 1 root vboxsf 398458880 Nov 4 14:06 syssub001.dtf -rwxrwx--- 1 root vboxsf 398458880 Nov 4 14:06 system001.dtf -rwxrwx--- 1 root vboxsf 398458880 Nov 4 14:06 temp001.dtf -rwxrwx--- 1 root vboxsf 16777216 Nov 4 14:06 test001.dtf -rwxrwx--- 1 root vboxsf 16777216 Nov 4 14:06 test002.dtf -rwxrwx--- 1 root vboxsf 8388608 Nov 4 14:06 test_idx_001.dtf -rwxrwx--- 1 root vboxsf 398458880 Nov 4 14:06 undo001.dtf -rwxrwx--- 1 root vboxsf 104857600 Nov 4 14:06 usr001.dtf	
[tibero@T1:/home/tibero/project]$ tbboot Listener port = 8629 Tibero 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. Tibero instance started up (NORMAL mode). [tibero@T1:/home/tibero/project]$ tbsql sys/tibero tbSQL 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. Connected to Tibero.	tibero 기동
SQL> INSERT INTO TEST.T1 SELECT ROWNUM , 'A'||TO_CHAR(ROWNUM), 'B'||TO_CHAR(ROWNUM), ROUND(ROWNUM/50) FROM DUAL CONNECT BY ROWNUM<=50000; 50000 rows inserted. SQL> commit; Commit completed. SQL> SELECT COUNT(*) FROM TEST.T1; COUNT(*) ---------- 100000 1 row selected. SQL> exit Disconnected.	data 입력 data 조회
[tibero@T1:/home/tibero/project]$ tbdown Tibero instance terminated (NORMAL mode). [tibero@T1:/home/tibero/project]$ rm /tibero/tbdata/tibero/*.dtf	tibero 종료 및 data file 전체 삭제 tibero 기동하여 마운트 모드 및 장애 상황 확인

[tibero@T1:/home/tibero/project]$ ls -al /tibero/tbdata/tibero total 44580 drwxr-xr-x 6 tibero dba 99 Nov 4 14:09 . drwxr-xr-x 3 tibero dba 20 Oct 11 13:34 .. drwxr-xr-x 2 tibero dba 78 Nov 4 09:32 arch -rw------- 1 tibero dba 22822912 Nov 4 14:09 c1.ctl -rw------- 1 tibero dba 22822912 Nov 4 14:09 c2.ctl drwx------ 2 tibero dba 70 Nov 4 08:27 java -r-------- 1 tibero dba 24 Nov 4 08:25 .passwd drwxr-xr-x 2 tibero dba 91 Nov 4 08:25 redo1 drwxr-xr-x 2 tibero dba 91 Nov 4 08:25 redo2 [tibero@T1:/home/tibero/project]$ tbboot ********** * BOOT FAILED. * tbsvr process (7021) is alive. * Check if there are any tbsvr instances running. ********** [tibero@T1:/home/tibero/project]$ tbdown Tibero instance terminated (NORMAL mode). [tibero@T1:/home/tibero/project]$ cp /tibero/s/tibero_bak/tibero/*.dtf /tibero/tbdata/tibero/. [tibero@T1:/home/tibero/project]$ ls -al /tibero/tbdata/tibero total 1744420 drwxr-xr-x 6 tibero dba 259 Nov 4 14:10 . drwxr-xr-x 3 tibero dba 20 Oct 11 13:34 .. drwxr-xr-x 2 tibero dba 78 Nov 4 09:32 arch -rw------- 1 tibero dba 22822912 Nov 4 14:10 c1.ctl -rw------- 1 tibero dba 22822912 Nov 4 14:10 c2.ctl drwx------ 2 tibero dba 70 Nov 4 08:27 java -r-------- 1 tibero dba 24 Nov 4 08:25 .passwd drwxr-xr-x 2 tibero dba 91 Nov 4 08:25 redo1 drwxr-xr-x 2 tibero dba 91 Nov 4 08:25 redo2 -rwxr-x--- 1 tibero dba 398458880 Nov 4 14:10 syssub001.dtf -rwxr-x--- 1 tibero dba 398458880 Nov 4 14:10 system001.dtf -rwxr-x--- 1 tibero dba 398458880 Nov 4 14:10 temp001.dtf -rwxr-x--- 1 tibero dba 16777216 Nov 4 14:10 test001.dtf -rwxr-x--- 1 tibero dba 16777216 Nov 4 14:10 test002.dtf -rwxr-x--- 1 tibero dba 8388608 Nov 4 14:10 test_idx_001.dtf -rwxr-x--- 1 tibero dba 398458880 Nov 4 14:10 undo001.dtf -rwxr-x--- 1 tibero dba 104857600 Nov 4 14:10 usr001.dtf	tibero 종료 및 cold backup 파일 원복
[tibero@T1:/home/tibero/project]$ tbboot mount Listener port = 8629 Tibero 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. Tibero instance started up (MOUNT mode). [tibero@T1:/home/tibero/project]$ tbsql sys/tibero tbSQL 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. Connected to Tibero.	tibero mount 모드 기동 및 복구 수행

SQL> alter database recover automatic; Database altered. SQL> exit Disconnected.	
[tibero@T1:/home/tibero/project]$ tbdown Tibero instance terminated (NORMAL mode). [tibero@T1:/home/tibero/project]$ tbboot Listener port = 8629 Tibero 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. Tibero instance started up (NORMAL mode).	tibero 종료 및 tibero 기동
[tibero@T1:/home/tibero/project]$ tbsql sys/tibero tbSQL 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. Connected to Tibero. SQL> select count(*)from test.t1; COUNT(*) ---------- 100000 1 row selected. SQL> exit Disconnected.	table 건수 조회

5.3 매체 복구(Media Recovery) 기능

5.3.1 설명

- 매체 복구 기능

5.3.2 수행

수행 순서	시나리오
1	테이블 건수 조회
2	Begin Backup 수행
3	핫 백업 진행
4	End Backup 수행 및 로그 스위치 수행
5	Sysdate 조회
6	데이터 입력

7	테이블 건수 조회
8	컨트롤 파일 백업 및 로그 스위치 수행
9	티베로 종료 및 데이터 파일 전체 삭제(Redo Log 까지)
10	티베로 기동하여 마운트 모드 및 장애 상황 확인
11	티베로 종료 및 핫 백업 원복
12	노마운트 기동 및 컨트롤 파일 복구
13	티베로 마운트 모드 기동 및 복구 수행
14	티베로 종료
15	티베로 기동 및 복구 마무리
16	건수 확인

5.3.3 결과

시나리오	설명
[tibero@T1:/home/tibero/project]$ tbsql sys/tibero tbSQL 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. Connected to Tibero. SELECT COUNT*() FROM TEST.T1;* *COUNT()* ---------- 100000 1 row selected.	sys 사용자로 접속 test.t1 table 건수 조회
SQL> alter database begin backup; Database altered. SQL> exit Disconnected.	begin backup 수행
[tibero@T1:/tibero/s]$ cp /tibero/tbdata/tibero/*.dtf /tibero/s/tibero_hot2 [tibero@T1:/tibero/s]$ ls -al /tibero/s/tibero_hot2 total 1699848 drwxrwx--- 1 root vboxsf 4096 Nov 4 15:03 . drwxrwx--- 1 root vboxsf 4096 Nov 4 15:03 .. -rwxrwx--- 1 root vboxsf 398458880 Nov 4 15:03 syssub001.dtf -rwxrwx--- 1 root vboxsf 398458880 Nov 4 15:03 system001.dtf -rwxrwx--- 1 root vboxsf 398458880 Nov 4 15:03 temp001.dtf -rwxrwx--- 1 root vboxsf 16777216 Nov 4 15:03 test001.dtf -rwxrwx--- 1 root vboxsf 16777216 Nov 4 15:03 test002.dtf -rwxrwx--- 1 root vboxsf 8388608 Nov 4 15:03 test_idx_001.dtf -rwxrwx--- 1 root vboxsf 398458880 Nov 4 15:03 undo001.dtf -rwxrwx--- 1 root vboxsf 104857600 Nov 4 15:03 usr001.dtf	hot backup 진행 (datafile 복제)
[tibero@T1:/tibero/s]$ tbsql sys/tibero tbSQL 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved.	end backup 수행 및 log switch 수행

Connected to Tibero. SQL> alter database end backup; Database altered. SQL> alter system switch logfile; System altered.	
SQL> ALTER SESSION SET NLS_DATE_FORMAT='YYYY/MM/DD HH24:MI:SS'; Session altered. SQL> SELECT SYSDATE FROM DUAL; SYSDATE ---------- 2022/11/04 15:04:54 1 row selected.	sysdate 조회
SQL> INSERT INTO TEST.T1 (ID) VALUES ('444444'); 1 row inserted. SQL> commit; Commit completed. SQL> SELECT COUNT(*) FROM TEST.T1; COUNT(*) ---------- 100001 1 row selected.	데이터 입력 t1 table 건수 조회
SQL> alter database backup controlfile to trace as '/home/tibero/ctl.sql' reuse resetlogs; Database altered. SQL> alter system switch logfile; System altered. SQL> alter system switch logfile; System altered. SQL> alter system switch logfile; System altered. SQL> alter system switch logfile; System altered. SQL> alter system switch logfile; System altered.	control file backup 및 log switch 수행

SQL> exit Disconnected.	
[tibero@T1:/tibero/s]$ tbdown Tibero instance terminated (NORMAL mode). [tibero@T1:/tibero/s]$ rm /tibero/tbdata/tibero/*.dtf` [tibero@T1:/tibero/tbdata/tibero]$ rm -rf /tibero/tbdata/tibero/redo* [tibero@T1:/tibero/tbdata/tibero]$ rm /tibero/tbdata/tibero/*.ctl [tibero@T1:/tibero/tbdata/tibero]$ ls -al /tibero/tbdata/tibero total 8 drwxr-xr-x 4 tibero dba 45 Nov 4 15:20 . drwxr-xr-x 3 tibero dba 20 Oct 11 13:34 .. drwxr-xr-x 2 tibero dba 4096 Nov 4 15:08 arch drwx------ 2 tibero dba 70 Nov 4 08:27 java -r-------- 1 tibero dba 24 Nov 4 08:25 .passwd	tibero 종료 및 datafile 전체 삭제 (redo log 까지)
[tibero@T1:/tibero/tbdata/tibero]$ tbboot Listener port = 8629 ********** * Warning: Control file open failed * /tibero/tbdata/tibero/c1.ctl ********** ********** * Warning: Control file open failed * /tibero/tbdata/tibero/c2.ctl ********** ********** * Critical Warning : Raise svmode failed. The reason is * TBR-24003 : Unable to read control file. * Current server mode is NOMOUNT. ********** Tibero 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. Tibero instance started suspended at NOMOUNT mode. [tibero@T1:/tibero/tbdata/tibero]$ tbdown Tibero instance terminated (NORMAL mode).	tibero 기동하여 mount 모드 및 장애 상황 확인
[tibero@T1:/tibero/tbdata/tibero]$ cp /tibero/s/tibero_hot2/*.dtf /tibero/tbdata/tibero/. [tibero@T1:/tibero/tbdata/tibero]$ ls -al /tibero/tbdata/tibero total 1699848 drwxr-xr-x 4 tibero dba 205 Nov 4 15:21 . drwxr-xr-x 3 tibero dba 20 Oct 11 13:34 .. drwxr-xr-x 2 tibero dba 4096 Nov 4 15:08 arch drwx------ 2 tibero dba 70 Nov 4 08:27 java -r-------- 1 tibero dba 24 Nov 4 08:25 .passwd -rwxr-x--- 1 tibero dba 398458880 Nov 4 15:21 syssub001.dtf -rwxr-x--- 1 tibero dba 398458880 Nov 4 15:21 system001.dtf -rwxr-x--- 1 tibero dba 398458880 Nov 4 15:21 temp001.dtf -rwxr-x--- 1 tibero dba 16777216 Nov 4 15:21 test001.dtf	tibero 종료 및 핫 백업 원복 (tbdown 정상 종료한 후 백업)

-rwxr-x--- 1 tibero dba 16777216 Nov 4 15:21 test002.dtf -rwxr-x--- 1 tibero dba 8388608 Nov 4 15:21 test_idx_001.dtf -rwxr-x--- 1 tibero dba 398458880 Nov 4 15:21 undo001.dtf -rwxr-x--- 1 tibero dba 104857600 Nov 4 15:21 usr001.dtf	
[tibero@T1:/tibero/tbdata/tibero]$ tbboot nomount Listener port = 8629 Tibero 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. Tibero instance started up (NOMOUNT mode). [tibero@T1:/tibero/tbdata/tibero]$ tbsql sys/tibero tbSQL 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. Connected to Tibero. SQL> @/home/tibero/ctl.sql Control File created. SQL> exit Disconnected.	nomont 기동 및 controlfile 복구
[tibero@T1:/tibero/tbdata/tibero]$ tbdown Tibero instance terminated (NORMAL mode). [tibero@T1:/tibero/tbdata/tibero]$ tbboot mount Listener port = 8629 Tibero 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. Tibero instance started up (MOUNT mode). [tibero@T1:/tibero/tbdata/tibero]$ tbsql sys/tibero tbSQL 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. Connected to Tibero. SQL> ALTER SYSTEM SET NLS_DATE_FORMAT='YYYY/MM/DD HH24:MI:SS'; System altered. SQL> ALTER DATABASE RECOVER AUTOMATIC DATABASE UNTIL TIME '2022/11/04 15:00:00'; TBR-1057: Unable to start a Media Recovery - 2022/11/04 15:00:00 invalid, time should be later than . SQL> exit	tibero mount 모드 기동 및 복구 수행

Disconnected.	
[tibero@T1:/tibero/tbdata/tibero]$ tbdown Tibero instance terminated (NORMAL mode). [tibero@T1:/tibero/tbdata/tibero]$ tbboot resetlogs Listener port = 8629 Tibero 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. Tibero instance started up (NORMAL RESETLOGS mode).	tibero 종료 tibero 기동 및 복구 마무리 (tempfile 생성)
[tibero@T1:/tibero/tbdata/tibero]$ tbsql sys/tibero tbSQL 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. Connected to Tibero. SQL> alter tablespace temp add tempfile 'temp001.dtf' size 512m reuse autoextend on next 16m maxsize 1024m; Tablespace 'TEMP' altered. SQL> select count(*) from test.t1; COUNT(*) ---------- 100000 1 row selected. SQL> exit Disconnected.	t1 table 건수 확인

5.4 부분 백업 기능

5.4.1 설명

- 증분 백업, 차등 백업

백업 형식	백업 기준	백업 속도	사용 공간	유사성	복구에 필요한 미디어
전체 백업	전체 백업	느림	큼	/	가장 최근 백업 만 가능
차등 백업	전체 백업	중간	큼	변경된 파일 만 백업	백업 가장 최근 전체 백업 + 가장 최근 차등 백업
증분 백업	모든 유형의 마지막 백업	빠름	작음	변경된 파일 만 백업	가장 최근 전체 백업 + 전체 이후 모든 증분 백업

- RMGR을 이용하여 임의의 백업 경로에서 Online Full Backup을 수행할 수 있으며(-o 옵션), 백업 경로를 명시하지 않은 경우에는 RMGR_BACKUP_DEST가 기본 dest로 설정

- Archive Log File도 함께 Backup (--with-archivelog 옵션) 해두는 것이 바람직
- 앞서 Online Full Backup을 통해 생성된 Full Backup Set이 최소 1개 이상 존재하는 경우에는 가장 최신의 Backup Set과 현재 상태를 비교하여 변경사항만을 Backup하는 Incremental Backup을 수행

[복구 관리자 옵션]

- -v : RMGR의 진행상황을 자세하게 출력
- -o : 백업받을/백업받은 디렉터리를 지정한다. 백업할 때 옵션을 주지 않을 경우 RMGR_BACKUP_DEST가 기본 dest로 설정된다. 복구할 경우 옵션을 주지 않으면 백업된 directory를 자동으로 찾아간다. 옵션을 줄 경우 모든 full/incremental backup이 해당 directory에 있어야 한다.
- -i : 가장 최신 백업에 대한 Incremental backup을 수행
- cp -r : 디렉터리와 그 안의 내용까지 복사
- -C : 마지막 full backup에 대한 incremental 백업을 수행 (cumulative)

5.4.2 수행 (증분 백업)

수행 순서	시나리오
1	복구 관리자(tbrmgr)를 통한 Online Full Backup
2	테이블 건수 조회
3	데이터 입력
4	테이블 건수 조회
5	Incremental Backup 1
6	데이터 입력
7	테이블 건수 조회
8	Incremental Backup 2
9	데이터 입력
10	테이블 건수 조회
11	Incremental Backup 3
12	티베로 종료 및 데이터 파일 삭제
13	티베로 기동하여 마운트 모드 및 장애 상황 확인 후 다시 종료
14	tbrmgr Recovery
15	건수 조회(Incremental Backup 3 시점과 동일해야함)

5.4.3 결과

시나리오	설명
[tibero@T1:/tibero/tbdata/tibero]$ export TB_BACKUP=/home/tibero/backup [tibero@T1:/tibero/tbdata/tibero]$ mkdir -p $TB_BACKUP/full [tibero@T1:/tibero/tbdata/tibero]$ tbrmgr backup -o $TB_BACKUP/full -v ========== = Recovery Manager(RMGR) starts = = = = TmaxData Corporation Copyright (c) 2008-. All rights reserved. = ========== ========== RMGR - ONLINE backup ========== DB connected DBNAME: tibero START_OFFSET: 0 archive log check succeeded BACKUP (set_id:1, ts_id:0, df_id:0) 100.00% |==========⇒| 48640/48640 blks 0.30s Synchronizing... BACKUP (set_id:1, ts_id:1, df_id:1) 100.00% |==========⇒| 48640/48640 blks 0.27s	복구 관리자(tbrmgr)를 통해 online full backup

Synchronizing... BACKUP (set_id:1, ts_id:3, df_id:2) 100.00% |=====⇒| 12800/12800 blks 0.07s Synchronizing... BACKUP (set_id:1, ts_id:4, df_id:3) 100.00% |=====⇒| 48640/48640 blks 0.46s Synchronizing... BACKUP (set_id:1, ts_id:5, df_id:4) 100.00% |=====⇒| 2048/2048 blks 0.00s Synchronizing... BACKUP (set_id:1, ts_id:5, df_id:5) 100.00% |=====⇒| 2048/2048 blks 0.00s Synchronizing... BACKUP (set_id:1, ts_id:6, df_id:6) 100.00% |=====⇒| 1024/1024 blks 0.00s Synchronizing... Switching an online log file... Database full backup succeeded DB disconnected RMGR backup ends	
[tibero@T1:/tibero/tbdata/tibero]$ tbsql sys/tibero tbSQL 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. Connected to Tibero. SQL> select count(*) from test.t1; COUNT(*) ---------- 100000 1 row selected.	test.t1 table 건수 조회
SQL> INSERT INTO TEST.T1 SELECT ROWNUM , 'A'||TO_CHAR(ROWNUM), 'B'||TO_CHAR(ROWNUM), ROUND(ROWNUM/50) FROM DUAL CONNECT BY ROWNUM<=50000; 50000 rows inserted. SQL> commit; Commit completed. SQL> SELECT COUNT(*) FROM TEST.T1; COUNT(*) ---------- 150000 1 row selected. SQL> exit	data 입력 table 건수 조회

Disconnected.	
[tibero@T1:/home/tibero/backup/full]$ cp -r $TB_BACKUP/full $TB_BACKUP/incremental [tibero@T1:/home/tibero/backup/full]$ tbrmgr backup -o $TB_BACKUP/incremental -i -v ========== = Recovery Manager(RMGR) starts = = = = TmaxData Corporation Copyright (c) 2008-. All rights reserved. = ========== ========== RMGR - INCREMENTAL backup ========== DB connected DBNAME: tibero START_OFFSET: 0 archive log check succeeded BACKUP (set_id:2, ts_id:0, df_id:0) 100.00% |==========⇒| 48640/48640 blks 0.08s Synchronizing... BACKUP (set_id:2, ts_id:1, df_id:1) 100.00% |==========⇒| 48640/48640 blks 0.09s Synchronizing... BACKUP (set_id:2, ts_id:3, df_id:2) 100.00% |==========⇒| 12800/12800 blks 0.00s Synchronizing... BACKUP (set_id:2, ts_id:4, df_id:3) 100.00% |==========⇒| 48640/48640 blks 0.08s Synchronizing... BACKUP (set_id:2, ts_id:5, df_id:4) 100.00% |==========⇒| 2048/2048 blks 0.00s Synchronizing... BACKUP (set_id:2, ts_id:5, df_id:5) 100.00% |==========⇒| 2048/2048 blks 0.00s Synchronizing... BACKUP (set_id:2, ts_id:6, df_id:6) 100.00% |==========⇒| 1024/1024 blks 0.00s Synchronizing... Switching an online log file... Database incremental backup succeeded DB disconnected RMGR backup ends	Incremental Backup 1
[tibero@T1:/home/tibero/backup/full]$ tbsql sys/tibero tbSQL 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. Connected to Tibero. SQL> INSERT INTO TEST.T1 SELECT ROWNUM , 'A'||TO_CHAR(ROWNUM), 'B'||TO_CHAR(ROWNUM), ROUND(ROWNUM/50) FROM DUAL CONNECT BY ROWNUM<=50000; 50000 rows inserted.	data 입력 건수 조회

SQL> commit; Commit completed. SQL> SELECT COUNT(*) FROM TEST.T1; COUNT(*) ---------- 200000 1 row selected. SQL> exit Disconnected.	
[tibero@T1:/home/tibero/backup/full]$ tbrmgr backup -o $TB_BACKUP/incremental -i -v ========== = Recovery Manager(RMGR) starts = = = = TmaxData Corporation Copyright (c) 2008-. All rights reserved. = ========== ========== RMGR - INCREMENTAL backup ========== DB connected DBNAME: tibero START_OFFSET: 0 archive log check succeeded BACKUP (set_id:3, ts_id:0, df_id:0) 100.00% |==========⇒| 48640/48640 blks 0.06s Synchronizing... BACKUP (set_id:3, ts_id:1, df_id:1) 100.00% |==========⇒| 48640/48640 blks 0.06s Synchronizing... BACKUP (set_id:3, ts_id:3, df_id:2) 100.00% |==========⇒| 12800/12800 blks 0.00s Synchronizing... BACKUP (set_id:3, ts_id:4, df_id:3) 100.00% |==========⇒| 48640/48640 blks 0.06s Synchronizing... BACKUP (set_id:3, ts_id:5, df_id:4) 100.00% |==========⇒| 2048/2048 blks 0.00s Synchronizing... BACKUP (set_id:3, ts_id:5, df_id:5) 100.00% |==========⇒| 2048/2048 blks 0.00s Synchronizing... BACKUP (set_id:3, ts_id:6, df_id:6) 100.00% |==========⇒| 1024/1024 blks 0.00s Synchronizing... Switching an online log file... Database incremental backup succeeded DB disconnected RMGR backup ends	Incremental Backup 2
[tibero@T1:/home/tibero/backup/full]$ tbsql sys/tibero tbSQL 6	data 입력 건수 조회

TmaxData Corporation Copyright (c) 2008-. All rights reserved.

Connected to Tibero.

SQL> INSERT INTO TEST.T1
SELECT ROWNUM ,
'A'||TO_CHAR(ROWNUM),
'B'||TO_CHAR(ROWNUM),
ROUND(ROWNUM/50)
FROM DUAL CONNECT BY ROWNUM<=50000;

50000 rows inserted.

SQL> commit;

Commit completed.

SQL> SELECT COUNT(*) FROM TEST.T1;

COUNT(*)

250000

1 row selected.

SQL> exit

Disconnected.

[tibero@T1:/home/tibero/backup/full]$ tbrmgr backup -o $TB_BACKUP/incremental -i -v ========== = Recovery Manager(RMGR) starts = = = = TmaxData Corporation Copyright (c) 2008-. All rights reserved. = ========== ========== RMGR - INCREMENTAL backup ========== DB connected DBNAME: tibero START_OFFSET: 0 archive log check succeeded BACKUP (set_id:4, ts_id:0, df_id:0) 100.00% |==========⇒| 48640/48640 blks 0.13s Synchronizing... BACKUP (set_id:4, ts_id:1, df_id:1) 100.00% |==========⇒| 48640/48640 blks 0.11s Synchronizing... BACKUP (set_id:4, ts_id:3, df_id:2) 100.00% |==========⇒| 12800/12800 blks 0.02s Synchronizing... BACKUP (set_id:4, ts_id:4, df_id:3) 100.00% |==========⇒| 48640/48640 blks 0.17s Synchronizing... BACKUP (set_id:4, ts_id:5, df_id:4) 100.00% |==========⇒| 2048/2048 blks 0.00s Synchronizing... BACKUP (set_id:4, ts_id:5, df_id:5) 100.00% |==========⇒| 2048/2048 blks 0.00s Synchronizing...	Incremental Backup 3

BACKUP (set_id:4, ts_id:6, df_id:6) 100.00% |=====================================⇒| 2048/2048 blks 0.00s Synchronizing... Switching an online log file... Database incremental backup succeeded DB disconnected RMGR backup ends	
[tibero@T1:/home/tibero/backup/full]$ ls -al $TB_BACKUP/incremental total 1439916 drwxr-xr-x 2 tibero dba 4096 Nov 4 17:40 . drwxr-xr-x 4 tibero dba 37 Nov 4 17:36 .. -rw------- 1 tibero dba 398458880 Nov 4 17:36 bkp_20221104_1_0_0_42584 -rw------- 1 tibero dba 398458880 Nov 4 17:36 bkp_20221104_1_1_1_42584 -rw------- 1 tibero dba 104857600 Nov 4 17:37 bkp_20221104_1_3_2_42584 -rw------- 1 tibero dba 398458880 Nov 4 17:37 bkp_20221104_1_4_3_42584 -rw------- 1 tibero dba 16777216 Nov 4 17:37 bkp_20221104_1_5_4_42584 -rw------- 1 tibero dba 16777216 Nov 4 17:37 bkp_20221104_1_5_5_42584 -rw------- 1 tibero dba 8388608 Nov 4 17:37 bkp_20221104_1_6_6_42584 -rw------- 1 tibero dba 23150592 Nov 4 17:37 bkp_20221104_1_cf_42584 -rw------- 1 tibero dba 65536 Nov 4 17:37 bkp_20221104_2_0_0_42584 -rw------- 1 tibero dba 4726784 Nov 4 17:37 bkp_20221104_2_1_1_42584 -rw------- 1 tibero dba 8192 Nov 4 17:37 bkp_20221104_2_3_2_42584 -rw------- 1 tibero dba 1449984 Nov 4 17:37 bkp_20221104_2_4_3_42584 -rw------- 1 tibero dba 1859584 Nov 4 17:37 bkp_20221104_2_5_4_42584 -rw------- 1 tibero dba 81920 Nov 4 17:37 bkp_20221104_2_5_5_42584 -rw------- 1 tibero dba 5726208 Nov 4 17:37 bkp_20221104_2_6_6_42584 -rw------- 1 tibero dba 23150592 Nov 4 17:37 bkp_20221104_2_cf_42584 -rw------- 1 tibero dba 32768 Nov 4 17:40 bkp_20221104_3_0_0_42584 -rw------- 1 tibero dba 1400832 Nov 4 17:40 bkp_20221104_3_1_1_42584 -rw------- 1 tibero dba 8192 Nov 4 17:40 bkp_20221104_3_3_2_42584 -rw------- 1 tibero dba 8192 Nov 4 17:40 bkp_20221104_3_4_3_42584 -rw------- 1 tibero dba 516096 Nov 4 17:40 bkp_20221104_3_5_4_42584 -rw------- 1 tibero dba 1482752 Nov 4 17:40 bkp_20221104_3_5_5_42584 -rw------- 1 tibero dba 5431296 Nov 4 17:40 bkp_20221104_3_6_6_42584 -rw------- 1 tibero dba 23150592 Nov 4 17:40 bkp_20221104_3_cf_42584 -rw------- 1 tibero dba 16384 Nov 4 17:40 bkp_20221104_4_0_0_42584 -rw------- 1 tibero dba 5439488 Nov 4 17:40 bkp_20221104_4_1_1_42584 -rw------- 1 tibero dba 8192 Nov 4 17:40 bkp_20221104_4_3_2_42584 -rw------- 1 tibero dba 8192 Nov 4 17:40 bkp_20221104_4_4_3_42584 -rw------- 1 tibero dba 827392 Nov 4 17:40 bkp_20221104_4_5_4_42584 -rw------- 1 tibero dba 983040 Nov 4 17:40 bkp_20221104_4_5_5_42584 -rw------- 1 tibero dba 9609216 Nov 4 17:40 bkp_20221104_4_6_6_42584 -rw------- 1 tibero dba 23150592 Nov 4 17:40 bkp_20221104_4_cf_42584	Incremental backup list 조회
[tibero@T1:/home/tibero/backup/full]$ tbdown Tibero instance terminated (NORMAL mode). [tibero@T1:/home/tibero/backup/full]$ rm /tibero/tbdata/tibero/*.dtf [tibero@T1:/home/tibero/backup/full]$ tbboot Listener port = 8629 ********** * Critical Warning : Raise svmode failed. The reason is * TBR-1024 : Database needs media recovery: open failed(/tibero/tbdata/tibero/system001.dtf). * Current server mode is MOUNT. ********** Tibero 6	tibero 종료 및 datafile 삭제 tibero 기동하여 mount모드 및 장애 상황 확인 후 다시 종료

TmaxData Corporation Copyright (c) 2008-. All rights reserved. Tibero instance started suspended at MOUNT mode. [tibero@T1:/home/tibero/backup/full]$ tbdown immediate Tibero instance terminated (IMMEDIATE mode).	
[tibero@T1:/home/tibero/backup/full]$ tbrmgr recover -o $TB_BACKUP/incremental -v ========== = Recovery Manager(RMGR) starts = = = = TmaxData Corporation Copyright (c) 2008-. All rights reserved. = ========== ========== RMGR - recovery ========== Tibero instance terminated (ABNORMAL mode). info file is deleted. unlink failed.: No such file or directory Control file #0 (/tibero/tbdata/tibero/c1.ctl) is accessible Control file #1 (/tibero/tbdata/tibero/c2.ctl) is accessible All control files are accessible. No need to restore the backup control file . Listener port = 8629 Tibero 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. Tibero instance started up (MOUNT mode). DB connected RMGR BEGIN RESTORE full backup set_id: 1 last incremental backup set_id: 4 Applying FULL BACKUP (set_id:1, ts_id:0, df_id:0) 100.00% |==========⇒| 48640/48640 blks 0.61s Synchronizing... Applying FULL BACKUP (set_id:1, ts_id:1, df_id:1) 100.00% |==========⇒| 48640/48640 blks 0.84s Synchronizing... Applying FULL BACKUP (set_id:1, ts_id:3, df_id:2) 100.00% |==========⇒| 12800/12800 blks 1.71s Synchronizing... Applying FULL BACKUP (set_id:1, ts_id:4, df_id:3) 100.00% |==========⇒| 48640/48640 blks 1.44s Synchronizing... Applying FULL BACKUP (set_id:1, ts_id:5, df_id:4) 100.00% |==========⇒| 2048/2048 blks 0.00s Synchronizing... Applying FULL BACKUP (set_id:1, ts_id:5, df_id:5) 100.00% |==========⇒| 2048/2048 blks 0.00s Synchronizing... Applying FULL BACKUP (set_id:1, ts_id:6, df_id:6) 100.00% |==========⇒| 1024/1024 blks 0.00s Synchronizing... Applying INCREMENTAL BACKUP (set_id:2, ts_id:0, df_id:0) 100.00% |==========⇒| 48640/48640 blks 0.00s Synchronizing... Applying INCREMENTAL BACKUP (set_id:2, ts_id:1, df_id:1)	tbrmgr recovery

```
100.00% |=====================================⇒| 48640/48640 blks 0.20s
Synchronizing...
Applying INCREMENTAL BACKUP (set_id:2, ts_id:3, df_id:2)
100.00% |=====================================⇒| 12800/12800 blks 0.20s
Synchronizing...
Applying INCREMENTAL BACKUP (set_id:2, ts_id:4, df_id:3)
100.00% |=====================================⇒| 48640/48640 blks 0.60s
Synchronizing...
Applying INCREMENTAL BACKUP (set_id:2, ts_id:5, df_id:4)
100.00% |=====================================⇒| 2048/2048 blks 0.00s
Synchronizing...
Applying INCREMENTAL BACKUP (set_id:2, ts_id:5, df_id:5)
100.00% |=====================================⇒| 2048/2048 blks 0.00s
Synchronizing...
Applying INCREMENTAL BACKUP (set_id:2, ts_id:6, df_id:6)
100.00% |=====================================⇒| 1024/1024 blks 0.00s
Synchronizing...
Applying INCREMENTAL BACKUP (set_id:3, ts_id:0, df_id:0)
100.00% |=====================================⇒| 48640/48640 blks 0.00s
Synchronizing...
Applying INCREMENTAL BACKUP (set_id:3, ts_id:1, df_id:1)
100.00% |=====================================⇒| 48640/48640 blks 0.00s
Synchronizing...
Applying INCREMENTAL BACKUP (set_id:3, ts_id:3, df_id:2)
100.00% |=====================================⇒| 12800/12800 blks 0.00s
Synchronizing...
Applying INCREMENTAL BACKUP (set_id:3, ts_id:4, df_id:3)
100.00% |=====================================⇒| 48640/48640 blks 0.00s
Synchronizing...
Applying INCREMENTAL BACKUP (set_id:3, ts_id:5, df_id:4)
100.00% |=====================================⇒| 2048/2048 blks 0.00s
Synchronizing...
Applying INCREMENTAL BACKUP (set_id:3, ts_id:5, df_id:5)
100.00% |=====================================⇒| 2048/2048 blks 0.01s
Synchronizing...
Applying INCREMENTAL BACKUP (set_id:3, ts_id:6, df_id:6)
100.00% |=====================================⇒| 1024/1024 blks 0.01s
Synchronizing...
Applying INCREMENTAL BACKUP (set_id:4, ts_id:0, df_id:0)
100.00% |=====================================⇒| 48640/48640 blks 0.01s
Synchronizing...
Applying INCREMENTAL BACKUP (set_id:4, ts_id:1, df_id:1)
100.00% |=====================================⇒| 48640/48640 blks 0.00s
Synchronizing...
Applying INCREMENTAL BACKUP (set_id:4, ts_id:3, df_id:2)
100.00% |=====================================⇒| 12800/12800 blks 0.00s
Synchronizing...
Applying INCREMENTAL BACKUP (set_id:4, ts_id:4, df_id:3)
100.00% |=====================================⇒| 48640/48640 blks 0.00s
Synchronizing...
Applying INCREMENTAL BACKUP (set_id:4, ts_id:5, df_id:4)
100.00% |=====================================⇒| 2048/2048 blks 0.00s
Synchronizing...
Applying INCREMENTAL BACKUP (set_id:4, ts_id:5, df_id:5)
100.00% |=====================================⇒| 2048/2048 blks 0.00s
Synchronizing...
Applying INCREMENTAL BACKUP (set_id:4, ts_id:6, df_id:6)
100.00% |=====================================⇒| 2048/2048 blks 0.00s
Synchronizing...

Database restore succeeded

recoverSQL: ALTER DATABASE RECOVER AUTOMATIC
Database automatic recovery succeeded
DB disconnected
```

Tibero instance terminated (NORMAL mode). Listener port = 8629 Tibero 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. Tibero instance started up (NORMAL mode). RMGR recovery ends	
[tibero@T1:/home/tibero/backup/full]$ tbsql sys/tibero tbSQL 6 TmaxData Corporation Copyright (c) 2008-. All rights reserved. Connected to Tibero. SQL> select count(*) from test.t1; COUNT(*) ---------- 250000 1 row selected.	건수 조회 (Incremental Backup3 시점과 동일)

5.4.4 수행 (차등 백업)

수행 순서	
1	tbrmgr 풀 백업 파일 복사
2	테이블 건수 조회
3	데이터 입력
4	테이블 건수 조회
5	Cumulative Backup 1
6	데이터 입력
7	테이블 건수 조회
8	Cumulative Backup 2
9	데이터 입력
10	테이블 건수 조회
11	Cumulative Backup 3
12	티베로 종료 및 데이터 파일 전체 삭제
13	티베로 기동하여 마운트 모드 및 장애 상황 확인 후 다시 종료
14	tbrmgr Recovery
15	건수 조회(Cumulative Backup 3 시점과 동일해야함)

5.4.5 결과