✅

멘토링 2주차 과제 TTA BMT 기능 검증 시나리오_SINGLE

태그

1.1 사용자 계정 생성, 패스워드 설정, 권한 설정 가능

```
1 tbsql 접속
2 사용자 생성, 패스워드 설정
3 권한 설정
4 권한 정보 확인
```

```
tbsql sys/tibero
```

```
CREATE USER TEST IDENTIFIED BY 'TEST';
SELECT USERNAME FROM DBA_USERS WHERE USERNAME='TEST';
GRANT RESOURCE TO TEST;
COL GRANTEE FOR A30
COL GRANTED_ROLE FOR A30
COL ADMIN_OPTION FOR A30
COL DEFAULT_ROLE FOR A30
SET LINESIZE 1000
SELECT * FROM DBA_ROLE_PRIVS WHERE GRANTEE ='TEST';
```

1.2 타 사용자의 테이블에 대한 권한이 없는 사용자의 접근 차단 기능

```
 1 사용자(test1) 생성
2 사용자(test1)에 resource,connect 권한 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,connect 권한 부여
9 다른 사용자(test2)에 접속
10 다른 사용자(test2)에 접속해서 test1 사용자의 table(tbl) 조회
```

```
 CREATE USER TEST1 IDENTIFIED BY 'TEST1';
GRANT CONNECT , RESOURCE TO TEST1;
CREATE TABLE TEST1.TBL (ID NUMBER);
INSERT INTO TEST1.TBL VALUES (1);
```

```
COMMIT;

conn TEST1/TEST1
SELECT * FROM TEST1.TBL;
conn SYS/TIBERO
CREATE USER TEST2 IDENTIFIED BY 'TEST2';
GRANT CONNECT , RESOURCE TO TEST2;
conn TEST2/TEST2
SELECT * FROM TEST1.TBL;
```

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

```
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
```

```
conn SYS/TIBERO
SET LINES 200
COL GRANTEE FOR A20
COL OWNER FOR A20
COL TABLE_NAME FOR A20
COL GRANTOR FOR A20
COL PRIVILEGE FOR A20
COL GRANTABLE FOR A20
SELECT * FROM DBA_TAB_PRIVS
WHERE OWNER ='TEST1' AND TABLE_NAME = 'TBL';
GRANT SELECT ON TEST1.TBL TO TEST2;
SELECT * FROM DBA_TAB_PRIVS WHERE OWNER = 'TEST1' AND TABLE_NAME = 'TBL';
```

```
conn SYS/TIBERO
SET LINES 200
COL GRANTEE FOR A20
COL OWNER FOR A20
COL TABLE_NAME FOR A20
COL GRANTOR FOR A20
COL PRIVILEGE FOR A20
COL GRANTABLE FOR A20
SELECT * FROM DBA_TAB_PRIVS
WHERE OWNER ='TEST1' AND TABLE_NAME = 'TBL';
GRANT SELECT ON TEST1.TBL TO TEST2;
SELECT * FROM DBA_TAB_PRIVS WHERE OWNER = 'TEST1' AND TABLE_NAME = 'TBL';
```

2.1 Row-level locking

```
1 사용자(test1) 생성
2 사용자(test1)에 resource,connect 권한 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,connect 권한 부여
9 다른 사용자(test2)에 접속
10 다른 사용자(test2)에 접속해서 test1 사용자의 table(tbl) 조회
```

```
CREATE USER TEST1 IDENTIFIED BY 'TEST1';
GRANT CONNECT , RESOURCE TO TEST1;
CREATE TABLE TEST1.TBL (ID NUMBER);
INSERT INTO TEST1.TBL VALUES (1);
COMMIT;
conn TEST1/TEST1
SELECT * FROM TEST1.TBL;
conn SYS/TIBERO
CREATE USER TEST2 IDENTIFIED BY 'TEST2';
GRANT CONNECT , RESOURCE TO TEST2;
conn TEST2/TEST2
SELECT * FROM TEST1.TBL;
```

2.2 MVCC 기능

```
세션 1
1 테이블(TEST1.RXTEST) 생성
2 테이블(TEST1.RXTEST)에 데이터 입력
3 테이블(TEST1.RXTEST)의 데이터 변경
- commit 하지 않음
세션 2
4 테이블(TEST1.RXTEST)의 데이터 변경
세션 1
5 테이블(TEST1.RXTEST)의 데이터 변경
commit
세션 2
6 HANG 이 풀리는 것을 확인
```

```
세션 1
tbsql TEST1/TEST1
CREATE TABLE TEST1.RXTEST(ID NUMBER,
NAME VARCHAR(20));
INSERT INTO TEST1.RXTEST VALUES ( 1,
'TEST');
COMMIT;
SELECT * FROM TEST1.RXTEST;
ALTER SESSION SET
NLS_DATE_FORMAT='YYYY/MM/DD
HH24:MI:SS';
SELECT SYSDATE FROM DUAL;
UPDATE TEST1.RXTEST SET NAME =
'TESTTEST' WHERE ID =1;
SELECT * FROM TEST1.RXTEST;
```

```
세션 2 테이블(TEST1.RXTEST)의 데이터 변경
```

```
tbsql TEST2/TEST2
ALTER SESSION SET
NLS_DATE_FORMAT='YYYY/MM/DD
HH24:MI:SS';
SELECT SYSDATE FROM DUAL;
UPDATE TEST1.RXTEST SET NAME =
'TESTTESTTEST' WHERE ID = 1;
```

```
세션 1 테이블(TEST1.RXTEST)의 데이터 변경 commit
COMMIT;
```

```
세션 1
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 조회
```

```
세션 2
5 테이블(TEST1.MVCCTEST)의 데이터 조회
세션 1
6 변경된 데이터 commit
세션 2
7 테이블(TEST1.MVCCTEST)의 변경된 데이터
조회
```

```
세션 1
tbsql TEST1/TEST1
CREATE TABLE TEST1.MVCCTEST(ID NUMBER, NAME VARCHAR(20));
INSERT INTO TEST1.MVCCTEST VALUES(1, 'TEST');
COMMIT;
UPDATE TEST1.MVCCTEST SET NAME = 'TEST123' WHERE ID = 1;
ALTER SESSION SET NLS_DATE_FORMAT='YYYY/MM/DD HH24:MI:SS';
SELECT SYSDATE FROM DUAL;
SELECT * FROM TEST1.MVCCTEST WHERE ID = 1 ;
```

```
세션 2 테이블(TEST1.MVCCTEST)의 데이터 조회
ALTER SESSION SET NLS_DATE_FORMAT='YYYY/MM/DD HH24:MI:SS';
SELECT SYSDATE FROM DUAL;
SELECT * FROM TEST1.MVCCTEST
```

```
세션 1
SELECT SYSDATE FROM DUAL;
COMMIT;
```

```
세션 2 테이블(TEST1.MVCCTEST)의 변경된 데이터 조회
SELECT SYSDATE FROM DUAL;
SELECT * FROM TEST1.MVCCTEST WHERE ID = 1 ;
```

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

```
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)
```

```
conn SYS/TIBERO
ALTER SYSTEM SET USE_RECYCLEBIN=Y;
conn TEST/TEST
CREATE TABLE FLASHBACK_TEST (C1 NUMBER, C2 NUMBER, C3 NUMBER);
declare
begin
for i in 1..10000 loop
insert into test.flashback_test values (i,i,i);
end loop;
end;
/
COMMIT;
DROP TABLE TEST.FLASHBACK_TEST;
SELECT COUNT(*) FROM FLASHBACK_TEST;
COL OBJECT_NAME FOR A20
COL ORIGINAL_NAME FOR A20
COL TS_NAME FOR A20
COL SPACE FOR A30
SELECT * FROM USER_RECYCLEBIN;
FLASHBACK TABLE FLASHBACK_TEST TO BEFORE DROP;
SELECT COUNT(*) FROM FLASHBACK_TEST;
conn SYS/TIBERO
ALTER SYSTEM SET USE_RECYCLEBIN=N;
```

3.2 Range partition table 확인

```
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
```

```
tbsql SYS/TIBERO
GRANT CREATE TABLESPACE TO TEST;
conn TEST/TEST
```

```
CREATE TABLESPACE TEST_PART1 DATAFILE 'TEST_PART1.DBF' SIZE 100M;
CREATE TABLESPACE TEST_PART2 DATAFILE ' TEST_PART2.DBF' SIZE 100M;
CREATE TABLESPACE TEST_PART3 DATAFILE 'TEST_PART3.DBF' SIZE 100M;
CREATE TABLESPACE TEST_PART4 DATAFILE ' TEST_PART4.DBF' SIZE 100M;
```

```
CREATE TABLESPACE TEST_PART1 DATAFILE '/DEV/RAW/RAW60' SIZE 100M;
CREATE TABLESPACE TEST_PART2 DATAFILE '/DEV/RAW/RAW61' SIZE 100M;
CREATE TABLESPACE TEST_PART3 DATAFILE '/DEV/RAW/RAW62' SIZE 100M;
CREATE TABLESPACE TEST_PART4 DATAFILE '/DEV/RAW/RAW63' SIZE 100M;

CREATE TABLE TEST.RANGE_PART
 (RANGE_NO NUMBER,
 RANGE_YEAR INT NOT NULL,
RANGE_MONTH INT NOT NULL,
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 );

INSERT INTO RANGE_PART VALUES(1, 2004, 06, 12, 'SCOTT', 2500);
INSERT INTO RANGE_PART VALUES(2, 2005, 06, 17, 'JONES', 4300);
INSERT INTO RANGE_PART VALUES(3, 2005, 12, 12, 'MILLER', 1200);
INSERT INTO RANGE_PART VALUES(4, 2006, 06, 22, 'FORD', 5200);
INSERT INTO RANGE_PART VALUES(5, 2005, 01, 01, 'LION', 2200);
COMMIT;

SELECT * FROM TEST.RANGE_PART;
```

```
CREATE TABLESPACE TEST_PART_MAX DATAFILE 'PART_MAX.DBF' SIZE 100M;
ALTER TABLE RANGE_PART  ADD PARTITION RANGE_MAX VALUES LESS THAN (MAXVALUE, MAXVALUE, MAXVALUE )
 TABLESPACE TEST_PART_MAX;
```

```
CREATE TABLESPACE TEST_PART_MAX DATAFILE '/DEV/RAW/RAW64' SIZE 100M;
ALTER TABLE RANGE_PART DROP PARTITION RANGE_MAX;
ALTER TABLE RANGE_PART RENAME PARTITION RANGE_Q4 TO RANGE_FOUR;
ALTER TABLE RANGE_PART  MERGE PARTITIONS RANGE_Q1, RANGE_Q2 INTO PARTITION RANGE_Q2 UPDATE INDEXES;
ALTER TABLE RANGE_PART
 SPLIT PARTITION RANGE_Q2 AT (2005, 01, 01)
 INTO (PARTITION RANGE_Q1,
 PARTITION RANGE_Q2);

CREATE TABLE RANGE_PART_EX
 (RANGE_NO NUMBER,
 RANGE_YEAR INT NOT NULL,
 RANGE_MONTH INT NOT NULL,
 RANGE_DAY INT NOT NULL,
 RANGE_NAME VARCHAR2(30),
 RANGE NUMBER)
 TABLESPACE TEST_PART1;

ALTER TABLE RANGE_PART
 EXCHANGE PARTITION RANGE_Q1
 WITH TABLE RANGE_PART_EX;
```

```
SELECT RANGE_NO
 FROM RANGE_PART PARTITION (RANGE_Q1);
SELECT RANGE_NO FROM RANGE_PART_EX;

ALTER TABLE RANGE_PART  MOVE PARTITION RANGE_Q3 TABLESPACE TEST_PART_MAX;
ALTER TABLE RANGE_PART TRUNCATE PARTITION RANGE_Q3;
SELECT * FROM RANGE_PART PARTITION (RANGE_Q3);
```

3.3 Hash partition table 확인

```
• Hash partition table 확인
1 파티션테이블 생성
2 데이터 입력
3 파티션 삭제 ( 지원하지 않음 )
4 파티션 이름 변경
5 파티션 변경 (EXCHANGE)
6 파티션 테이블스페이스 변경
7 파티션 데이터 TRUNCATE
```

```
tbsql TEST/TEST
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);
INSERT INTO TEST.HASH_PART VALUES(1, 2004, 06, 12, 'SCOTT', 2500);
INSERT INTO TEST.HASH_PART VALUES(2, 2005, 06, 17, 'JONES', 4300);
INSERT INTO TEST.HASH_PART VALUES(3, 2005, 12, 12, 'MILLER', 1200);
INSERT INTO TEST.HASH_PART VALUES(4, 2006, 06, 22, 'FORD', 5200);
INSERT INTO TEST.HASH_PART VALUES(5, 2005, 01, 01, 'LION', 2200);
INSERT INTO TEST.HASH_PART VALUES(6, 2006, 12, 22, 'TIGER', 3300);
COMMIT;
SELECT * FROM TEST.HASH_PART;
ALTER TABLE TEST.HASH_PART DROP PARTITION HASH_PART4;
ALTER TABLE TEST.HASH_PART RENAME PARTITION HASH_PART4 TO HASH_PART_FOUR;
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;

ALTER TABLE TEST.HASH_PART
 EXCHANGE PARTITION HASH_PART2
 WITH TABLE TEST.HASH_PART_EX;

SELECT COUNT(*) FROM TEST.HASH_PART PARTITION(HASH_PART2);
SELECT COUNT(*) FROM TEST.HASH_PART_EX;
ALTER TABLE TEST.HASH_PART MOVE PARTITION HASH_PART3 TABLESPACE TEST_PART4;
```

```
SELECT COUNT(*) FROM TEST.HASH_PART PARTITION(HASH_PART_FOUR);
ALTER TABLE TEST.HASH_PART TRUNCATE PARTITION HASH_PART_FOUR;
SELECT COUNT(*) FROM TEST.HASH_PART PARTITION(HASH_PART_FOUR);
```

3.4 List partition table 확인

```
• List partition table 확인
 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
```

```
tbsql TEST/TEST
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);
INSERT INTO LIST_PART VALUES(1, 'SMITH', 'CLERK', 7902, SYSDATE, 800, NULL, 20);
INSERT INTO LIST_PART VALUES(2, 'ALLEN', 'SALESMAN', 7698, SYSDATE, 1600, 300, 30);
INSERT INTO LIST_PART VALUES(3, 'WARD', 'SALESMAN', 7698, SYSDATE, 1250, 500, 30);
INSERT INTO LIST_PART VALUES(4, 'JONES', 'MANAGER', 7839, SYSDATE, 2975, NULL, 20);
INSERT INTO LIST_PART VALUES(5, 'MARTIN', 'SALESMAN', 7698, SYSDATE, 1250, 1400, 30);
INSERT INTO LIST_PART VALUES(6, 'BLAKE', 'MANAGER', 7839, SYSDATE, 2850, NULL, 30);
INSERT INTO LIST_PART VALUES(7, 'CLARK', 'MANAGER', 7839, SYSDATE, 2450, NULL, 10);
INSERT INTO LIST_PART VALUES(8, 'SCOTT', 'ANALYST', 7566, SYSDATE, 3000, NULL, 20);
INSERT INTO LIST_PART VALUES(9, 'KING', 'PRESIDENT', NULL, SYSDATE, 5000, NULL, 10);
INSERT INTO LIST_PART VALUES(10, 'TURNER', 'SALESMAN', 7698,SYSDATE, 1500, 0, 30);
INSERT INTO LIST_PART VALUES(11, 'ADAMS', 'CLERK', 7788,SYSDATE,1100,NULL,20);
INSERT INTO LIST_PART VALUES(12, 'JAMES', 'CLERK', 7698, SYSDATE, 950, NULL, 30);
INSERT INTO LIST_PART VALUES(13, 'FORD', 'ANALYST', 7566, SYSDATE, 3000, NULL, 20);
INSERT INTO LIST_PART VALUES(14, 'MILLER', 'CLERK', 7782, SYSDATE, 1300, NULL, 10);
COMMIT;

SELECT LIST_NO FROM LIST_PART PARTITION (LIST_PART1);
SELECT LIST_NO FROM LIST_PART PARTITION (LIST_PART2);
SELECT LIST_NO FROM LIST_PART PARTITION (LIST_PART3);
SELECT LIST_NO FROM LIST_PART PARTITION (LIST_PART4);

ALTER TABLE LIST_PART
 ADD PARTITION LIST_PART_MAX VALUES ('DUMMY')
TABLESPACE TEST_PART_MAX;
```

```
ALTER TABLE LIST_PART
 DROP PARTITION LIST_PART_MAX ;
ALTER TABLE TEST.LIST_PART
RENAME PARTITION LIST_PART4 TO LIST_PART_FOUR;
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));
SELECT COUNT(*) FROM TEST.LIST_PART PARTITION(LIST_PART3);

ALTER TABLE TEST.LIST_PART
EXCHANGE PARTITION LIST_PART3 WITH TABLE TEST.LIST_PART_EX;
SELECT COUNT(*) FROM TEST.LIST_PART PARTITION(LIST_PART3);
SELECT COUNT(*) FROM TEST.LIST_PART_EX;
ALTER TABLE TEST.LIST_PART MOVE PARTITION LIST_PART1 TABLESPACE TEST_PART2;
SELECT COUNT(*) FROM TEST.LIST_PART PARTITION(LIST_PART2);
ALTER TABLE TEST.LIST_PART TRUNCATE PARTITION LIST_PART2;
SELECT COUNT(*) FROM TEST.LIST_PART ;
```

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

```
모니터링 : 쿼리 실행 계획 및 쿼리 또는 세션에 대한 통계자료(실행시간,IO) 제공 여부 확인
• CLI 방식의 클라이언트 프로그램 화면 또는 별도 trace 파일에서 확인
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
```

```
tbsql TIBERO/TMAX
CREATE TABLE PLAN_TEST(C1 NUMBER,C2 NUMBER, C3 NUMBER);
declare
 begin
 for i in 1..10000 loop
 insert into plan_test values(i,i,i);
 end loop;
 end;
 /
CREATE INDEX PLAN_TEST_IDX ON PLAN_TEST(C1);
SET LINES 200
SET AUTOT TRACEONLY EXP PLANSTAT
select * from plan_test where c1 between 10 and 100;
set autot off
alter session set sql_trace=y;
select * from plan_test where c1 between 10 and 100;
exit
```

```
tblog
cd sqltrace
tbprof tb_sqltrc_13423_61_341750.trc tb_sqltrc_13423_61_341750.log
cat tb_sqltrc_13423_61_341750.log
```

4.2 DML, DDL 쿼리에 대한 감사(audit) 기능

```
DML, DDL 쿼리에 대한 감사(audit) 기능
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
```

```
vi $TB_HOME/config/$TB_SID.tip
---- audit 다른 유저가능
#AUDIT Setting
AUDIT_TRAIL=OS
AUDIT_SYS_OPERATIONS=Y
AUDIT_FILE_DEST=/home/tibero6/audit
```

```
tbsql SYS/TIBERO
```

```
CREATE TIBERO.TABLE AUDIT_TEST(ID NUMBER);
AUDIT insert on tibero.audit_test BY SESSION WHENEVER SUCCESSFUL;
AUDIT update on tibero.audit_test BY SESSION WHENEVER SUCCESSFUL;
AUDIT delete on tibero.audit_test BY SESSION WHENEVER SUCCESSFUL;
AUDIT create table by tibero;
insert into tibero.audit_test values(1);
commit;
update tibero.audit_test set id = 2;
commit;
delete from tibero.audit_test ;
commit;
```

```
감사(AUDIT) 로그 확인
2017/05/19 17:27:12.236
SESS_ID:[59] SERIAL_NO:[38] STMT_ID:[86] USER_NAME:[SYS] USER_HOST:[127.0.0.1]
OS_USER:[tibero6] CLIENT_ID:[tbsql] PID:[14184] SQLTEXT:[insert into tibero.audit_test
values(1)]
2017/05/19 17:27:16.398
SESS_ID:[59] SERIAL_NO:[38] STMT_ID:[87] USER_NAME:[SYS] USER_HOST:[127.0.0.1]
OS_USER:[tibero6] CLIENT_ID:[tbsql] PID:[14184] SQLTEXT:[update tibero.audit_test set
id = 2]
2017/05/19 17:27:18.826
SESS_ID:[59] SERIAL_NO:[38] STMT_ID:[88] USER_NAME:[SYS] USER_HOST:[127.0.0.1]
OS_USER:[tibero6] CLIENT_ID:[tbsql] PID:[14184] SQLTEXT:[delete from
tibero.audit_test]
```

```
e
ㅇ
TTA BMT 기능 검증
-30-
2017/05/19 17:28:01.368
SESS_ID:[59] SERIAL_NO:[198] AUD_NO:[1] STMT_ID:[0] USER_NAME:[TIBERO]
USER_HOST:[127.0.0.1] OS_USER:[tibero6] CLIENT_ID:[tbsql] PRIV_NO:[-489] ACTION:[S]
OBJ_OWNER:[TIBERO] OBJ_NAME:[TEST]USGMT_ID:[5] SLOTNO:[2] WRAPNO:[32]
TSN:[1519821] PID:[14184] SQLTEXT:[create table test(id number)]
```

5.1 온라인 백업 후 완전 복구

```
백업/복구
온라인 백업 후 완전 복구 기능
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
```

```
tbsql sys/tibero
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;
CREATE TABLESPACE TS_TEST_IDX
DATAFILE 'test_idx_001.dtf' SIZE 8M AUTOEXTEND ON NEXT 8M MAXSIZE 1G
EXTENT MANAGEMENT LOCAL AUTOALLOCATE;
CREATE USER TEST IDENTIFIED BY TEST DEFAULT TABLESPACE TS_TEST;
GRANT DBA TO TEST;
CONN TEST/TEST

CREATE TABLE TEST.T1 (ID NUMBER,
ANAME VARCHAR2(32),
BNAME VARCHAR2(32),
ID2 NUMBER)
TABLESPACE TS_TEST;
CREATE INDEX IDX_T1 ON T1(ID, ANAME) TABLESPACE TS_TEST_IDX;
```

```
SELECT COUNT(*) FROM TEST.T1;
SELECT TABLESPACE_NAME FROM DBA_TABLESPACES;
SELECT FILE_NAME FROM DBA_DATA_FILES;

ALTER DATABASE BEGIN BACKUP;
```

```
SELECT * FROM V$BACKUP;
exit
```

```
ls -al $TB_HOME/database/$TB_SID/*.dtf
cd $TB_HOME/database
mkdir -p ${TB_SID}_hot
cp $TB_SID/*.dtf ${TB_SID}_hot/.
ls -al ${TB_SID}_hot
```

```
tbsql sys/tibero
ALTER DATABASE END BACKUP;
SELECT * FROM V$BACKUP;
ALTER SYSTEM SWITCH LOGFILE;
```

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

```
tbdown
cd $TB_HOME/database/$TB_SID
rm *.dtf
ls -al
```

```
tbboot
tbdown
cp -r $TB_HOME/database/${TB_SID}_hot/*.dtf $TB_HOME/database/$TB_SID/.
ls –al $TB_HOME/database/$TB_SID/
```

```
tbboot mount
tbsql sys/tibero

ALTER DATABASE RECOVER AUTOMATIC;

tbdown
tbboot

tbsql sys/tibero
SELECT COUNT(*) FROM TEST.T1;
exit
```

5.2 오프라인 백업 후 완전 복구

```
5.2 오프라인 백업(콜드 백업) 후 완전 복구 기능

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
```

```
tbsql sys/tibero
SELECT NAME FROM V$CONTROLFILE;
SELECT MEMBER FROM V$LOGFILE;
SELECT FILE_NAME FROM DBA_DATAFILES;
SELECT FILE_NAME FROM DBA_TEMP_FILES;
SELECT COUNT(*) FROM TEST.T1;
exit
```

```
tbdown
cp -r $TB_HOME/database/$TB_SID $TB_HOME/database/${TB_SID}_bak
ls -al $TB_HOME/database/${TB_SID}_bak
tbboot
tbsql sys/tibero
INSERT INTO TEST.T1
SELECT ROWNUM ,
 'A'||TO_CHAR(ROWNUM),
 'B'||TO_CHAR(ROWNUM),
 ROUND(ROWNUM/50)
FROM DUAL CONNECT BY ROWNUM<=50000;
COMMIT;
SELECT COUNT(*) FROM TEST.T1;
exit
rm $TB_HOME/database/$TB_SID/*.dtf
ls -al $TB_HOME/database/$TB_SID
tbboot
tbdown
cp $TB_HOME/database/${TB_SID}_bak/*.dtf $TB_HOME/database/$TB_SID/.
ls -al $TB_HOME/database/$TB_SID

tbboot mount
tbsql sys/tibero
ALTER DATABASE RECOVER AUTOMATIC;
exit

tbdown
tbboot
tbsql sys/tibero
SELECT COUNT(*) FROM TEST.T1;
exit
```

5.3 매체 복구 기능

```
매체 복구(Media Recovery) 기능

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
```

```
tbsql sys/tibero
ALTER DATABASE BEGIN BACKUP;
exit
cd $TB_HOME/database
cp $TB_SID/*.dtf ${TB_SID}_hot/.
ls -al ${TB_SID}_hot/
```

```
tbsql sys/tibero
ALTER DATABASE END BACKUP;
ALTER SYSTEM SWITCH LOGFILE;
ALTER SESSION SET NLS_DATE_FORMAT='YYYY/MM/DD HH24:MI:SS';
INSERT INTO TEST.T1 (ID) VALUES ('444444');
SELECT COUNT(*) FROM TEST.T1;
```

```
ALTER DATABASE BACKUP CONTROLFILE TO TRACE AS '/home/tibero6/ctl.sql' REUSE
RESETLOGS;
ALTER SYSTEM SWITCH LOGFILE;
ALTER SYSTEM SWITCH LOGFILE;
ALTER SYSTEM SWITCH LOGFILE;
ALTER SYSTEM SWITCH LOGFILE;
ALTER SYSTEM SWITCH LOGFILE;
exit
```

```
tbdown
rm $TB_HOME/database/$TB_SID/*.dtf
rm $TB_HOME/database/$TB_SID/*.redo
rm $TB_HOME/database/$TB_SID/*.ctl
```

```
ls -al $TB_HOME/database/$TB_SID/

tbboot
tbdown
cp $TB_HOME/database/${TB_SID}_hot/*.dtf $TB_HOME/database/$TB_SID/.
ls -al $TB_HOME/database/$TB_SID/
tbboot nomount

tbsql sys/tibero
@/home/tibero6/ctl.sql
Exit

tbdown
tbboot mount
tbsql sys/tibero
ALTER SYSTEM SET NLS_DATE_FORMAT='YYYY/MM/DD HH24:MI:SS';
ALTER DATABASE RECOVER AUTOMATIC DATABASE UNTIL TIME '2017/05/23 13:26:58';
exit
```

```
tbdown
tbboot resetlogs
tbsql sys/tibero
ALTER TABLESPACE TEMP ADD TEMPFILE 'temp001.dtf' SIZE 512M REUSE
AUTOEXTEND ON NEXT 16M MAXSIZE 1024M;
SELECT COUNT(*) FROM TEST.T1;
exit
```

5.4.(1) 증분백업

```
• 증분 백업, 차등 백업
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)
```

```
export TB_BACKUP=/home/tibero6/backup
mkdir -p $TB_BACKUP/full
tbrmgr backup -o $TB_BACKUP/full -v

tbsql sys/tibero
SELECT COUNT(*) FROM TEST.T1;
INSERT INTO TEST.T1
 SELECT ROWNUM ,
 'A'||TO_CHAR(ROWNUM),
 'B'||TO_CHAR(ROWNUM),
 ROUND(ROWNUM/50)
```

```
 FROM DUAL CONNECT BY ROWNUM<=50000;
COMMIT;
SELECT COUNT(*) FROM TEST.T1;
exit
```

```
cp -r $TB_BACKUP/full $TB_BACKUP/incremental
tbrmgr backup -o $TB_BACKUP/incremental -i -v

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

SELECT COUNT(*) FROM TEST.T1;
exit
tbrmgr backup -o $TB_BACKUP/incremental -i -v
tbsql sys/tibero
INSERT INTO TEST.T1
 SELECT ROWNUM ,
 'A'||TO_CHAR(ROWNUM),
 'B'||TO_CHAR(ROWNUM),
 ROUND(ROWNUM/50)
 FROM DUAL CONNECT BY ROWNUM<=50000;
COMMIT;
SELECT COUNT(*) FROM TEST.T1;
exit
```

```
tbrmgr backup -o $TB_BACKUP/incremental -i -v
ls -al $TB_BACKUP/incremental
tbdown
rm $TB_HOME/database/$TB_SID/*.dtf
tbboot
tbdown immediate
tbrmgr recover -o $TB_BACKUP/incremental -v
```

```
tbsql sys/tibero
SELECT COUNT(*) FROM TEST.T1;
```

5.4 (2)차등백업

```
5.4.4 수행(차등 백업)

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 거수 조회
8 Cumulative Backup 2
9 데이터 입력
10 테이블 거수 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)
```

```
cp -r $TB_BACKUP/full $TB_BACKUP/cumulative
tbsql sys/tibero
SELECT COUNT(*) FROM TEST.T1;
INSERT INTO TEST.T1
 SELECT ROWNUM ,
 'A'||TO_CHAR(ROWNUM),
 'B'||TO_CHAR(ROWNUM),
 ROUND(ROWNUM/50)
 FROM DUAL CONNECT BY ROWNUM<=50000;
COMMIT;
SELECT COUNT(*) FROM TEST.T1;
exit
```

```
tbrmgr backup -o $TB_BACKUP/cumulative -C -v

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

COMMIT;
SELECT COUNT(*) FROM TEST.T1;
exit
```

```
tbrmgr backup -o $TB_BACKUP/cumulative -C -v

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

SELECT COUNT(*) FROM TEST.T1;
exit;
```

```
tbrmgr backup -o $TB_BACKUP/cumulative -C -v
ls -al $TB_BACKUP/cumulative
tbdown
rm $TB_HOME/database/$TB_SID/*.dtf
tbboot
tbdown
tbrmgr recover -o $TB_BACKUP/cumulative -v
```

```
tbsql sys/tibero
SELECT COUNT(*) FROM TEST.T1;
```