Oracle Database 11g Administraion Workshop Ⅱ

Practice 4 Configuring Backup Settings

1. $HOME/backup 에 경로를 생성합니다.

$ mkdir $HOME/backup/→ 이미 있으면 2번으로 진행합니다.

2. RMAN을 시작하고 target database에 접속하여 현재까지의 Backup을 모두 삭제합니다. 그리고나서 여러 가지 RMAN명령어를 실행해 봅니다.

RMAN> connect target /

RMAN> delete backupset;

➔ "Do you really want to delete the above objects(enter Yes or No)?"메시지가 나오면 y를 입력하여 모든 Backup 삭제한 후 다음 실습을 진행합니다.

RMAN> report schema; ➔ DB구조가 보임

RMAN> show all; ➔ 미리 구성된 백업과 복구관련 설정

3. Database 전체 backup 수행 후 Retention Policy를 변경하면서 backup이 필요한 file을 확인한 후 그 설정을 Default값으로 복원하세요.

a) 백업파일이 /home/oracle/backup에 수행하도록 채널을 구성합니다.

RMAN> Configure channel device type disk

      format='/home/oracle/backup/orcl_%s_%p';

b) 현재의 Retention Policy 값 및 백업이 필요한 파일을 확인합니다.

RMAN> show all; ➔ redundancy 확인

RMAN> report need backup;

RMAN> backup database;

RMAN> report need backup; ➔ 백업이 필요한 파일목록

b) Retention Policy를 2로 변경한 후 Users 테이블스페이스만 부분 백업을 실시하고 다시 Backup이 필요한 파일을 분석해 봅니다.

RMAN> configure retention policy to redundancy 2;

RMAN> report need backup;

RMAN> backup tablespace users;

RMAN> report need backup; ➔ USERS만 제외됨

c) Retention Policy 값을 기본값으로 재구성하고 모든 Backupset을 삭제합니다.

RMAN> configure retention policy clear;

RMAN> list backupset

RMAN> delete backupset;

4. Control file Autobackup을 알아봅니다.

a) Control file 자동 백업이 비활성화 상태일 때 백업작업을 모니터 합니다

RMAN> show controlfile autobackup; ➔ OFF임을 확인!

RMAN> backup database; ➔ control file이 백업에 포함됨

RMAN> backup tablespace users; ➔ control file이 백업에 포함되지 않음

b) control file 자동 백업을 활성화한 후 백업작업을 모니터합니다.

RMAN> configure controlfile autobackup on;

RMAN> backup tablespace users; ➔ 모든 백업작업에 controlfile이 포함됨

RMAN> exit;

c) 모든 백업을 삭제하고 controlfile 자동백업을 비활성화 합니다

$ rman target /

RMAN> delete backupset;

RMAN> configure controlfile autobackup off;

5. backup 최적화 기능을 실습해 봅니다.

a) 백업 최적화 기능을 활성화 하고 RETENTION POLICY가 1인지 확인합니다.

RMAN> show retention policy; ➔ 만약 1이 아니면 실습 3-C를 실행합니다.

RMAN> CONFIGURE BACKUP OPTIMIZATION ON;

b) RMAN 에서 다음과 같이 users 테이블스페이스를 READ ONLY로 설정합니다.

RMAN> sql 'alter tablespace users read only';

c) RMAN에서 데이터베이스를 연속 세 번 Backup을 수행합니다.

RMAN> backup database;

RMAN> backup database;

RMAN> backup database; ➔ 세 번(Retention Policy+1번)째 백업부터 USERS제외됨

RMAN> list backupset;

d) backup 최적화 기능을 비활성화하고 모든 backup을 삭제합니다.

RMAN> configure backup optimization clear;

RMAN> show backup optimization;

RMAN> delete backupset;

e) USERS Tablespace를 READ WRITE로 변경합니다.

RMAN> sql 'alter tablespace users read write';

6. 일반 백업과 compressed backup 비교해 봅니다.

a) RMAN에서 두 가지 유형의 백업셋을 만들어 봅니다.

$ rman target /

RMAN> backup database;

RMAN> backup as compressed backupset database;

RMAN> exit

b) OS에서 압축을 하지 않은 백업, 압축을 한 백업의 두 backupset 크기를 비교해 봅니다.

$ ls -al /home/oracle/backup

c) 모든 backupset을 삭제하여 실습을 정리합니다.

$ rman target /

RMAN> delete backupset;

RMAN> exit;

7. 데이터파일의 백업이 여러 개로 만들어지는 경우를 실습해 봅니다.

a) 채널을 병렬로 구성하고 백업셋을 만듭니다. 이 작업은 채널당 백업셋이 하나씩 만들어집니다.

RMAN> configure device type disk parallelism 2;

```
RMAN> backup database;

RMAN> configure device type disk clear;

RMAN> exit
```

$ ls /home/oracle/backup ➔ 백업 셋번호가 서로 다름 ex) 45_1, 46_1

b) 이번에는 채널이 하나의 백업 셋 작업동안 생성될 Backup Piece의 사이즈를 제한하고 백업을 수행해 봅니다.

```
$ rman target /

RMAN> CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT   '/home/oracle/backup/%s_%p' maxpiecesize 300M;

RMAN> backup database;

RMAN> exit
```

$ ls /home/oracle/backup ➔ 동일한 백업셋 번호에 Backup Piece 번호가 다른 백업이 존재 ex) 47_1, 47_2

c) 채널 설정을 초기화하고 모든 백업을 삭제하여 실습내용을 clear 합니다.

```
RMAN> CONFIGURE CHANNEL DEVICE TYPE DISK clear;

RMAN> delete backupset;
```

Practice 5 Creating Backups with RMAN

1. RMAN backup은 두 가지 유형이 있습니다. 다음을 실행하여 두 가지 유형의 백업의 특징을 알아봅니다.

a) RMAN을 실행하고 두 가지 유형의 백업을 실행합니다.

```
$ rman target /

RMAN> backup as backupset tablespace users format '/home/oracle/backup/%s_%p.bk';

RMAN> backup as copy tablespace users format '/home/oracle/backup/users01.bk';

RMAN> exit
```

b) 생성된 두 백업파일의 크기를 비교해 봅니다.

```
$ ls /home/oracle/backup

$ ls
```

c) 백업파일을 RMAN에서 확인하고 삭제합니다.

```
$ rman target /

RMAN> list backup of tablespace users;

RMAN> list copy of tablespace users;

RMAN> delete backupset;

RMAN> delete copy of database;

RMAN> exit
```

$ ls /home/oracle/backup → 백업파일이 없습니다.

2. 두 개의 터미널에서 SQLPlus와 RMAN을 각각실행하여 Incremental Backup을 수행해 봅니다.

a) RMAN을 실행하고 USERS Tablespace를 Incremental Level 0 백업을 합니다.

```
$ rman target /

RMAN> backup incremental level 0 database format '/home/oracle/backup/%s_%p.bk';
```

b) SQL*Plus 에서 hr.employees 테이블에 UPDATE를 실행합니다.

```
$ sqlplus / as sysdba

SQL> UPDATE hr.employees

     SET salary = salary*1.1
```

```
     WHERE department_id  = 80;
SQL> COMMIT;
```

c) RMAN에서 Incremental Level 1 Backup을 수행합니다.

```
RMAN> backup incremental level 1 database format '/home/oracle/backup/%s_%p.bk';
```

d) SQLPlus에서 V$BACKUP_DATAFILE을 query하여 레벨 1 incremental 백업을 생성하기 위해 몇 개의 블록을 읽었는지 확인합니다.

```
SQL> select file#, avg(datafile_blocks), avg(blocks_read),
            avg(blocks_read/datafile_blocks) * 100 as PCT_READ_FOR_BACKUP
    from v$backup_datafile
    where used_change_tracking = 'NO'
    and incremental_level > 0
    group by file#; → PCT_READ_FOR_BACKUP값 기록
```

3. fast incremental backup을 위해 추적 메커니즘(tracking mechanism)을 활성화시킨 후 Incremental Backup을 해 봅니다.

a) SQLPlus에서 Block Change Tracking을 다음과 같이 활성화하고 확인합니다.

```
SQL> alter database enable block change tracking using file
    '/home/oracle/backup/change.log';
SQL> select filename, status from v$block_change_tracking;
```

b) RMAN에서 Incremental level 0 백업을 수행합니다.

```
$ rman target /
RMAN> backup incremental level 0 database format '/home/oracle/backup/%s_%p.bk';
```

c) hr.employees 테이블을 변경 합니다.

```
SQL> UPDATE hr.employees
     SET salary = salary*1.1
     WHERE department_id  = 60;
SQL> COMMIT;
```

d) RMAN에서 incremental level 백업을 실행합니다.

```
RMAN> backup incremental level 1 database format '/home/oracle/backup/%s_%p.bk';
RMAN> exit;
```

e) block change tracking 을 확인합니다.

```
SQL> select * from v$block_change_tracking;
SQL> select file#, avg(datafile_blocks), avg(blocks_read),
            avg(blocks_read/datafile_blocks) * 100 as PCT_READ_FOR_BACKUP
    from v$backup_datafile
    where used_change_tracking = 'YES' and incremental_level > 0
    group by file#; → PCT_READ_FOR_BACKUP 수치 비교
```

4. 추적 메커니즘(tracking mechanism)을 비활성화합니다.

```
SQL> alter database disable block change tracking;
SQL> select filename, status from v$block_change_tracking;
SQL> exit
```

5. Backup Crosscheck 수행 후 'EXPIRED'된 백업파일 정보를 삭제합니다.

a) 현재 RMAN backup을 확인합니다.

RMAN> list backupset;

b) rman으로 백업받은 파일이 디렉토리에 존재하는 것을 확인한 후 모두 삭제합니다.

$ ls /home/oracle/backup → RMAN 백업파일이 존재 합니다.

$ rm /home/oracle/backup/*

c) RMAN 에서 백업정보를 확인합니다.

RMAN> list backupset; → 백업파일이 존재하는 것으로 인식됨(Status: AVAILABLE)

d) CROSSCHECK 명령 수행 후 다시 한번 LIST 명령을 수행하여 봅니다.

RMAN> crosscheck backupset;

RMAN> list backupset; → EXPIRED 로 상태가 변경됨

e) CROSSCHECK 결과 'EXPIRED' 된 백업파일 정보를 RMAN으로 부터 제거합니다.

RMAN> delete expired backupset; → 'y'를 입력하여 삭제

RMAN> list backupset;

RMAN> exit;

6. 여러 방법으로 RMAN Backup을 수행해 봅니다.

a) 데이터베이스 전체 백업을 수행한 후 USERS 및 SYSTEM Tablespace 를 각각 Backupset과 Copy를 하고 현재의 Controlfile을 백업합니다.

$ rman target /

RMAN> backup database;

RMAN> backup tablespace users;

RMAN> report schema; → system01.dbf file이 1번

RMAN> copy datafile 1 to '/home/oracle/backup/system01.bk';

RMAN> backup current controlfile format '/home/oracle/backup/control.bk';

b) 별도의 터미널에서 SQL*Plus를 실행하여 Log Switch를 수행하여 오늘 날짜의 Archived Log를 생성합니다.

$ sqlplus / as sysdba

SQL> alter system switch logfile; → 여러 번 수행

c) RMAN을 사용하여 /home/oracle/backup/ directory에 ar_%s_%p라는 이름형식으로 오늘 생성된 Archived Log file들을 backup 합니다.

RMAN> backup format '/home/oracle/backup/ar_%s_%p' archivelog from time 'sysdate-1';

c) Closed Database Whole Backup을 수행해 봅니다.

RMAN> shutdown immediate;

RMAN> startup mount;

RMAN> backup database;

RMAN> alter database open;

7. Backup File을 나열해 봅니다.

RMAN> list backup of database;

RMAN> list copy of tablespace system;

RMAN> list backup of archivelog from time 'sysdate-1';

RMAN> list backup of controlfile;

8. RMAN에서 백업을 삭제하고 Backup 경로의 파일들이 OS에서 삭제된 것을 확인합니다.

```
RMAN> delete backup of controlfile;
RMAN> delete backup of archivelog all;
RMAN> delete backup of database;
RMAN> delete copy of database;
RMAN> exit
```

Practice 6 Restore and Recovery Tasks

[Scenario 1 : Non-system Datafile Complete Recovery]

1. 데이터베이스를 백업하고 정상적인 운영을 시뮬레이션 합니다.터미널을 두 개 실행 후 각각 rman 과 sqlplus를 실행합니다.

```
$ rman target /
RMAN> backup incremental level 0 database;
$sqlplus / as sysdba
SQL> archive log list ➔ current log 번호 메모
SQL> update hr.bigemp
     Set salary = salary*1.1;
SQL> commit;
SQL> select avg(salary) from hr.bigemp; ➔ 메모
SQL> alter system swtich logfile; ➔ 여러 번 실행
SQL> archive log list
```


2. users01.dbf 파일에 장애가 발생합니다.

```
SQL> !rm /u01/app/oracle/oradata/orcl/users01.dbf
SQL> alter system flush buffer_cache;
SQL> SELECT * FROM hr.employees; ➔ error!
```

3. RMAN에서 테이블스페이스를 OFFLINE하고 복구를 수행합니다.

```
RMAN> sql 'alter tablespace users offline immediate';
RMAN> restore tablespace users;
RMAN> recover tablespace users;
RMAN> sql 'alter tablespace users online';
```

4. 복구가 완료된 것을 확인합니다.

```
SQL> select avg(salary) from hr.bigemp;
```

[Scenario 2 : system datafile complete recovery]

1. 데이터베이스 정상 운영을 시뮬레이트 합니다.

```
SQL> alter system switch logfile; ➔ 여러 번 수행
SQL> UPDATE hr.employees
     SET salary = salary*1.1;
```

SQL> commit;

SQL> SELECT avg(salary) FROM hr.employees;➔ Memo

SQL> alter system switch logfile; ➔ 여러 번 수행

2. system datafile에 장애가 발생합니다.

SQL> !rm /u01/app/oracle/oradata/orcl/system01.dbf

SQL> startup force ➔ MOUNT 단계에서 ERROR 발생

3. RMAN으로 복구합니다.

RMAN> restore tablespace system

RMAN> recover tablespace system;

RMAN> alter database open;

4. 복구완료를 확인합니다.

SQL> SELECT avg(salary) FROM hr.employees;➔

[Scenario 3 : Incremental Backup을 사용한 Recovery]

1. Incremental level 1 단계로 데이터베이스를 백업한 후 데이터베이스 운영을 시뮬레이트 합니다.

SQL> archive log list ➔ current log 번호 메모

RMAN> backup incremental level 1 database;

SQL> alter system switch logfile; ➔ 10회실행

SQL> update hr.employees

Set salary = salary*1.1;

SQL> commit;

2. 다시한번 Incremental level 1 단계로 데이터베이스를 백업한 후 데이터베이스 운영을 시뮬레이트 합니다.

SQL> archive log list ➔ current log 번호 메모

RMAN> backup incremental level 1 database;

SQL> alter system switch logfile; ➔ 10회실행

SQL> update hr.employees

Set salary = salary*1.1;

SQL> commit;

3. users01.dbf 파일에 장애가 발생합니다.

SQL> !rm /u01/app/oracle/oradata/orcl/users01.율

SQL> alter system flush buffer_cache;

SQL> SELECT * FROM hr.employees; ➔ error!

3. 테이블스페이스를 OFFLINE하고 복구를 수행합니다.

RMAN> sql 'alter tablespace users offline immediate';

RMAN> restore tablespace users;

RMAN> recover tablespace users;

Note) 복구과정에서 사용된 archive log sequence 번호를 살펴보면 scenario 1번의 level 0 백업 시점의 archived log file이 아니라 두 번째 incremental 백업 이후의 archived log file이 복구에 사용됩니다.

RMAN> sql 'alter tablespace users online';

4. 복구가 완료된 것을 확인합니다.

SQL> select avg(salary) from hr.bigemp;

[Scenario 4 : Trace file로 Control file 재생성 하기]

1. Trace 경로의 모든 파일을 삭제한 후 controlfile trace를 실행합니다.

$ cd /u01/app/oracle/diag/rdbms/orcl/orcl/trace/*

$rm *

$ sqlplus / as sysdba

SQL> alter database backup controlfile to trace;

SQL> exit

$ ls ➔ orcl_ora_XXXXX.trc 파일 생성 (예) orcl_ora_12345.trc

2. 해당 파일을 /home/oracle/backup 경로로 백업한 후 편집합니다.

$ cp orcl_ora_12345.trc /home/oracle/backup/con.sql

$ cd /home/oracle/backup

$ vi con.sql

상단의 주석부분은 지우기
Set #1. NORESETLOGS case
STARTUP NOMOUNT 부분부터
CREATE CONTROLFILE ~~
…
-- End of tempfile additions. 까지 남김
--   Set #2. RESETLOGS case ➔ 이하의 내용 모두 삭제
:wq

3. 운영 중 모든 controlfile이 삭제되는 장애가 발생합니다.

$ sqlplus / as sysdba

SQL> select name from v$controlfile; ➔ 결과에 조회되는 모든 controlfile을 rm 명령으로 삭제

SQL> startup force

➔ NOMOUNT 단계에서 ERROR !!

4. trace file을 편집해 둔 con.sql 파일로 controlfile을 다시 생성합니다.

SQL> shutdown abort

SQL> @/home/oracle/backup/con.sql

5. 데이터베이스가 정상운영되는 것을 확인한 후 전체 백업을 합니다.

```
SQL> startup force
SQL> conn hr/hr
SQL> exit
rman target /
RMAN> backup database;
```

[Scenario 5 : Temporary Tablespace 자동 복구]

1. Tempfile의 경로와 이름을 조회해 봅니다.

```
$ sqlplus / as sysdba
SQL> select file_name from dba_temp_files;
```

2. Tempfile 장애를 발생시키고 확인합니다.

```
SQL> !rm /u01/app/oracle/oradata/orcl/temp01.dbf
SQL> SELECT * FROM hr.bigemp
     ORDER BY salary;
```

➔ ERROR !!!

3. Tempfile은 DB 재시작 시 자동 생성됩니다.

```
SQL> startup force
SQL> !ls /u01/app/oracle/oradata/orcl
```

4. 만약 데이터베이스를 재시작 할 수 없는 경우에는 새 파일을 생성하고 기존 파일 정보를 삭제합니다.

```
SQL> ALTER TABLESPACE temp ADD TEMPFILE
     '/u01/app/oracle/oradata/orcl/temp02.dbf' SIZE 20M;
SQL> ALTER TABLESPACE temp DROP TEMPFILE
     '/u01/app/oracle/oradata/orcl/temp01.dbf';
```

Practice 7 Using RMAN to Perform Recovery

[Scenario 6 : Read Only Tablespace Recovery]

1. Read only Tablespace는 한 번만 백업합니다.

```
SQL> ALTER TABLESPACE users read only;
RMAN> backup tablespace users;
```

2. 데이터베이스 운영을 시뮬레이터하고 users01.dbf 파일에 장애가 발생합니다.

```
SQL> ALTER SYSTEM SWITCH LOGFILE; ➔ 여러 번 실행
SQL> SELECT * FROM hr.locations; ➔ ERROR!
```

3. 백업한 데이터파일을 복원하여 읽기 전용 테이블스페이스 복구를 완료합니다.

RMAN> sql 'alter tablespace users offline immediate';

RMAN> restore tablespace users;

RMAN> sql 'alter tablespace users online';

4. 복구를 확인합니다.

SQL> SELECT * FROM hr.locations;

5. USERS Tablespace를 다시 READ WRITE로 설정합니다.

SQL> alter tablespace users read write;

6. 모든 백업을 삭제합니다.

RMAN> delete backupset;

[Scenario 7 : Password file 손상 시 재생성]

1. password file을 삭제합니다.

$ cd $ORACLE_HOME/dbs

$ $ mv orapworcl orapworclbk ➔ 다른 이름으로 변경

2. EM 접속을 시도하며 실패를 확인합니다.

3. password file을 다시 생성합니다.

$ orapwd file=$ORACLE_HOME/dbs/orapworcl password=oracle entries=5

4. EM 접속을 시도하며 패스워드 파일 인증을 확인합니다.

[Scenario 8 : spfile의 백업과 복구]

1. spfile로 pfile을 생성하여 백업합니다.

SQL> create pfile from spfile;

SQL> exit

$ cd $ORACLE_HOME/dbs ➔ initorcl.ora 파일 생성 확인

2. spfile이 삭제되는 장애가 발생합니다.

$ rm $ORACLE_HOME/dbs/spfileorcl.ora

3. pfile로 다시 spfile을 생성합니다.

SQL> create spfile from pfile;

Practice 7 Using RMAN to Perform Recovery

[Scenario 1 : Cancel Based Incomplete Recovery]

1. 실습환경을 위해 다음과 같이 수행하여 Archive 경로를 한 개만 사용하도록 설정합니다. 그리고 Database에 대한 Backup 을 수행합니다.

a) EM 실행하고 Availability Tab을 누른 다음 Recovery Settings를 선택합니다.

b) Media Recovery 영역에서 Archived Redo log Destination에서 USE_DB_RECOVERY_FILE_DEST 만 남기고 다른 경로는 삭제한 후 오른쪽 상단의 Apply를 눌러 Archived Redo Log file이 1개의 경로에만 생성되도록 제어합니다.

c) RMAN을 실행하고 Database Backup을 수행합니다.

```
$ rman target /

RMAN> backup database;

RMAN> exit
```

2. 데이터베이스가 정상 운영되는 것을 시뮬레이트 합니다.

```
$ sqlplus  /as sysdba

SQL> alter system switch logfile;

SQL> /  ➔ 여러 번  수행

SQL> create table hr.emp11 as select * from hr.employees;

SQL> alter system switch logfile;

SQL> / ➔ 여러 번  수행

SQL> update hr.emp11
     set salary = salary*1.1;

SQL> commit;

SQL>  select avg(salary) from hr.emp11;

SQL> exit
```


3. users01.dbf file 삭제 및 Archived Log file 중 하나도 누락되는 사고가 발생합니다.

```
$ rm /u01/app/oracle/oradata/orcl/users01.dbf

$ cd /u01/app/oracle/flash_recovery_area/ORCL/archivelog/오늘날짜
```

➔ rm명령으로 최근 생성된 archivelog중 하나를 임의로 삭제(예시 25번)

```
$ sqlplus / as sysdba

SQL> startup force  ➔ ERROR!!!
```

4. 삭제된 archived log 번호를 지정하며 RMAN을 이용하여 불완전복구를 시도합니다.

```
RMAN> run{
```

```
      set until sequence 25 ; sequence 번호는 실습 3 참조
      restore database;
      recover database;
     }
```

5. Resetlogs로 Open하고 복구상태를 확인한 후 다시 RMAN을 이용한 전체 Backup을 수행합니다.

a) RESETLOGS 옵션과 함께 데이터베이스를 OPEN합니다.

```
RMAN> alter database open resetlogs;
RMAN> exit;
```

b) EMP11 테이블이 복구된 것을 확인합니다.

```
SQL> conn /as sysdba
SQL> select avg(salary) from hr.emp11;
```

→ 손상된 Archived Log에 따라 테이블은 복구취소 되었을 수도 있습니다.

```
SQL> archive log list;
```

c) RMAN Backup을 실행하고 불필요한 파일들은 삭제합니다.

```
$ rman target /
RMAN> backup database;
RMAN>  report obsolete;
RMAN>  delete obsolete;
```


[Scenario 2 : Time Based Incomplete Recovery]

1. RMAN을 사용하여 Whole DB Backup을 수행합니다.

```
$ rman target /
RMAN> backup database;
RMAN> exit
```

2. Database가 정상적으로 운영되는 것을 시뮬레이트 합니다.

a) HR.EMP11 테이블에 급여 UDATE가 발생합니다.

```
$ sqlplus  / as sysdba
SQL> alter system switch logfile; → 여러 번  수행
SQL> create table hr.emp11 as select * from hr.employees;
```

→ 테이블이 이미 있으면 UPDATE만 수행~!

```
SQL> update hr.emp11
     Set salary = salary*1.1;
SQL> commit;
SQL> select sum(salary) from hr.emp11; → 기록 : (ex) '2020-08-04:09:30:00'
SQL> alter system switch logfile; → 여러 번 수행
```

b) 시간이 조금 흐른 후 HR.EMP11 테이블에 두 번째 UPDATE가 발생합니다. 두 번째 UPDATE는 유저의 실수로 취소되어야 합니다.

```
SQL> update hr.emp11
     Set salary = salary*1.1;
SQL> commit;
SQL> select sum(salary) from hr.emp11; → 기록 :
SQL> !date → 기록 (ex) '2020-08-04:09:35:00'
```

3. time-based recovery를 수행하기 위해 운영체제 환경변수를 설정해야 합니다. 별도의 터미널을 시작하고 다음을 실행합니다.

```
$ export NLS_LANG=american_america.WE8ISO8859P15
$ export NLS_DATE_FORMAT='YYYY-MM-DD:HH24:MI:SS'
```

4. RMAN 실행하고 Time-Based Recovery를 수행합니다.

```
$ rman target /
RMAN> shutdown abort
RMAN> startup mount
RMAN> restore database;
RMAN> run{
      set until time ='2020-08-04:09:32:00'; → 장애시간보다 약간 빠른 시간
      recover database;}
```

5. Resetlogs로 Open하고 복구상태를 확인한 후 다시 RMAN을 이용한 전체 Backup을 수행합니다.

```
RMAN> alter database open resetlogs;
RMAN> exit
$ sqlplus  / as sysdba
SQL> select sum(salary) from hr.emp11;  → 첫 번째 UPDATE 결과가 나와야 함!
SQL> archive log list
SQL> exit
$ rman target /
RMAN> delete backupset;
```

RMAN> backup database;

RMAN> exit

[Scenario 3 : Autoback을 사용하여 controlfile 복구]

1. RMAN에서 Controlfile Auto Backup을 활성화 한 후 전체 DB Backup을 수행합니다.

$ rman target /

RMAN> CONFIGURE CONTROLFILE AUTOBACKUP ON;

RMAN> CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '%F'; → 기본값이므로 수정된 것이 없으면 그대로 둠.

RMAN> backup database;

RMAN> exit

2. 로그스위치를 수행하여 DB가 정상 운영되는 것을 시뮬레이트 한 후 현재 데이터베이스의 Control file목록을 확인한 후 모든 Control file이 삭제되는 장애를 발생 시킵니다.

a) 로그스위치를 여러 번 발생시킵니다.

SQL> alter system switch logfile; → 여러 번 수행

b) DB의 모든 Control file 확인합니다.

SQL> SELECT name FROM v$controlfile;

c) Current Log 번호 확인 후 목록의 모든 Controlfile 을 삭제합니다.

SQL> archive log list

SQL> !rm /u01/app/oracle/oradata/orcl/control01.ctl

SQL> !rm /u01/app/oracle/flash_recovery_area/orcl/control02.ctl

SQL> !rm /u01/app/oracle/oradata/orcl/control03.ctl ➔ 실습 진행 여부에 따라 없을 수도 있음

d) 데이터베이스를 다시 시작하면서 장애를 식별한 후 복구를 위해 DB를 강제종료 합니다

$ sqlplus / as sysdba

SQL> startup force ➔ ERROR!!

SQL> shutdown abort

SQL> exit

4. Autobackup된 Control file을 이용하여 Database 복구를 수행합니다.

a) RMAN에서 Control file Restore를 수행합니다.

$ rman target /

```
RMAN> restore controlfile from autobackup;
```

b) 다른 터미널에서 SQL*Plus를 실행하여 현재 데이터베이스 시작상태를 조회해 봅니다.

```
$ sqlplus / as sysdba

SQL> select status from v$instance;
```

→ "STARTED"는 NOMOUNT 상태입니다.

c) 데이터베이스를 MOUNT 하고 Recovery를 수행합니다.

```
RMAN> alter database mount;

RMAN> recover database ;

RMAN> alter database open resetlogs;

RMAN> exit
```

d)Resetlogs 수행으로 로그번호가 초기화 된 것과 복구가 완료된 것을 확인한 후 전체 데이터베이스 백업을 수행합니다.

```
SQL> archive log list;

SQL> exit

$ rman target /

RMAN> delete backupset;

RMAN> backup database;
```


[Scenario 4 : Performing a Fast Switch to Image Copies]

1. 실습에 사용할 SALES Tablespace 와 SALES_DATA Table을 생성합니다.

```
SQL> create tablespace sales
     datafile '$ORACLE_BASE/oradata/orcl/sales01.dbf' size 10m;
```

➔ SALES Tablespace가 이미 존재하면 생략하고 sales_data 테이블 생성

```
SQL> create table hr.sales_data
     tablespace sales
     as
     select * from hr.employees;

SQL> exit;
```

2. RMAN에서 SALES Tablespace Image Copy Backup을 수행합니다.

a) SALES Tablespace에 속하는 데이터파일을 식별합니다.

```
$ rman target /

RMAN> report schema;
```

Q3. SALES Tablespace의 데이터파일 번호는 몇 번 인지 확인하십시오.

(Ex) 4번

b) SALES Tablespace에 속하는 데이터파일을 Image Copy 유형으로 백업합니다.

```
RMAN> backup as copy datafile 4 format '/home/oracle/backup/sales01.bk';
```

3. 데이터베이스가 정상적으로 운영되었음을 시뮬레이트 합니다.

```
SQL> alter system switch logfile; → 여러 번 수행
SQL> insert into hr.sales_data
  2  select * from hr.sales_data;
SQL> / → 두 번 수행
SQL> commit;
SQL> select count(*) from hr.sales_data; → 행의 수 확인 :
SQL> alter system switch logfile; → 여러 번 수행
```

4. sales01.dbf 데이터파일을 삭제되는 Failure가 발생합니다.

```
SQL> select name from v$datafile;
SQL> !rm /u01/app/oracle/oradata/orcl/sales01.dbf
SQL> startup force → ERROR !
SQL> exit
```


5. 해당 데이터파일을 OFFLINE 한 후 데이터베이스를 엽니다. 그리고 RMAN에서 해당 파일을 복원하는 대신 Image Copy로 데이터파일을 스위치 한 후 복구를 수행합니다.

```
$ rman target /
RMAN> sql ‘alter database datafile 4 offline’;
RMAN> alter database open;
RMAN> list copy of datafile 4;
RMAN> switch datafile 4 to copy;
RMAN> recover datafile 4;
RMAN> sql "alter tablespace sales online";
RMAN> exit
```

6. SALES Tablespace에 속하는 데이터파일의 경로와 이름을 확인합니다.

```
$ sqlplus / as sysdba
SQL> col file_name for a50
```

```
SQL> col tablespace_name for a10

SQL> select tablespace_name, file_name

     from dba_data_files;
```

7. SALES Tablespace를 삭제하여 실습을 정리합니다.

```
SQL> drop tablespace sales including contents and datafiles;
```

8. 데이터베이스를 Backup 합니다. 그리고나서 불필요한 Backup을 확인한 후 삭제합니다.

```
RMAN> backup database;

RMAN> report obsolete;

RMAN> delete obsolete;
```

Practice 9 Diagnosing the Database

1. RMAN Backup을 수행하여 장애를 대비합니다.

```
$ rman target /

RMAN> backup database;

RMAN> exit;
```


2. 데이터베이스 운영 중 users01.dbf 파일이 삭제되는 장애가 발생합니다.

```
$ rm $ORACLE_BASE/oradata/orcl/users01.dbf
```

3. EM의 Data Recovery Advisor를 실행하여 장애 복구를 위한 Advise를 생성합니다.

a) EM Homepage 하단의 Related Links 영역에서 Advisor Central 을 선택한 후 Data Recovery Advisor를 클릭합니다.

b) 화면 하단의 Host Credentials의 username과 password에 각각 oracle을 입력한 후

이미 장애 내용이 식별되어 " One or more non-system datafiles are missing" 항목에 체크되었는지 확인한 후 Advice 버튼을 클릭합니다.

c) continue with Advice 를 선택합니다.

d) Recovery Advice 페이지의 RMAN Script 내용을 확인한 후 Continue 버튼을 누릅니다.

e) Review Page에서 RMAN 작업을 확인한 후 오른쪽 상단의 Submit Recovery Job 버튼을 누릅니다.

f) Confirmation 페이지의 "The job was created successfully" 메시지 확인 후 시간이 조금 지나서 View Results를 눌러 복구 작업의 성공여부를 확인합니다.

4. users01.dbf 파일이 복원 및 복구되어 DB가 정상 운영되는지 확인해 봅니다.

```
$ ls $ORACLE_BASE/oradata/orcl/users*

$ sqlplus / as sysdba

SQL> select count(*) from hr.employees;

SQL> exit
```

5. USERS 테이블스페이스를 백업한 후, 실습에 사용할 DEPT 테이블을 생성하고 DEPT 테이블에 손상을 주기 위해 데이터가 저장된 OS 파일 이름과 해당 파일의 DEPT 테이블 영역이 있는 OS 파일 블록 ID를 찾아야 합니다.

a) RMAN에서 USERS Tablespace를 백업합니다.

```
$ rman target /

RMAN> backup tablespace users;

RMAN> exit
```

b) 시간이 흐른 것을 가정하기 위해 로그스위치를 수행한 후 DEPT 테이블을 생성합니다.

```
SQL> conn /as sysdba

SQL> alter system switch logfile; → 여러 번 실행

SQL> create table hr.dept
     tablespace users
     As select * from hr.departments;
```


c) DEPT 테이블이 저장된 파일과 블록의 위치를 파악합니다.

```
SQL> select file_id, block_id from dba_extents
     Where segment_name='DEPT'; → file_id 확인 (ex) 4번 파일 49번

SQL> Select file_name from dba_data_files
      Where file_id= 4 (users01.dbf file의 file 번호)
```

→위 쿼리의 결과정보를 다음에 기록해 둡니다.

File id : Block id :

6. 다른 터미널에서 lab_09_02.sh 스크립트를 실행하여 위에서 기록한 블록번호의 users.01.dbf 데이터 파일에 손상을 줍니다. 스크립트에 대한 파라미터의 순서는 전체 파일이름, 블록 번호, 블록크기 순이며, 예제에는 users01.dbf 파일의 49번 블록이며 블록크기는 8192 입니다.

```
$ cd labs

$ chmod 777 lab_09_02.sh  → 파일의 실행권한을 부여합니다.

$ dos2unix lab_09_02.sh  → 유닉스에서 실행가능한 파일로 변환합니다.

$ ./lab_09_02.sh /u01/app/oracle/oradata/orcl/users01.dbf 49 8192
```

7. 버퍼 캐시를 비워서 DEPT 테이블에 대한 qeury가 디스크로부터 실행되도록 합니다. 그런 다음 DEPT 테이블의 모든 열을 선택하고 오류를 봅니다.

```
SQL> ALTER SYSTEM FLUSH BUFFER_CACHE;

SQL> SELECT * FROM hr.dept;
```

8. dbv 유틸리티를 실행하여 users01.dbf 파일의 손상을 검사해봅니다.

```
$ dbv file=/u01/app/oracle/oradata/orcl/users01.dbf blocksize=8192
```

9. RMAN을 사용하여 Recovery Advisor 관련 명령을 실행하여 Recovery 옵션을 알아보고, Block Media Recovery를 수행합니다. 8번 단계의 DBVERIFY 결과를 사용하여 recovery가 필요한 Block을 파악할 수 있습니다.

```
$ rman target /

RMAN> advise failure all;

RMAN> BLOCKRECOVER DATAFILE 4 BLOCK 49, 50, 51, 52

RMAN> exit
```

10. 복구가 완료되었는지 DEPT 테이블을 query하여 알아봅니다.

```
$ sqlplus hr/hr

SQL> SELECT * FROM dept;
```


Practice 10 Using Flashback Technology Ⅰ

1. 실습에 사용할 테이블을 생성하고 Flashback Query를 사용해 봅니다.

a) 실습에 사용할 EMP 테이블을 생성합니다.

```
SQL> conn hr/hr

SQL> drop table emp purge; → 없을 수 있음

SQL> create table emp
     as
     select * from employees;
```

b) 178번 사원의 급여를 확인한 후 update 합니다.

```
SQL> select employee_id, salary from emp
        where employee_id = 178;

SQL> update emp
     set salary=salary*1.1
```

```
     where employee_id = 178;
SQL> commit;
SQL> select employee_id, salary from emp
     where employee_id = 178;
```

c) 시간이 조금 흐른 후 Flashback Query기술을 사용하여 emp 테이블 상태를 쿼리해 봅니다.

(예제는 4분전의 상태를 확인합니다)

```
SQL> select employee_id, salary from emp
     as of timestamp(systimestamp-4/1440)
     where employee_id = 178;
```

d) 급여인상은 잘못된 값이므로 Flashback Query를 서브쿼리로 하여 178번 사원의 급여를 이전 값으로 변경합니다.

```
SQL> update emp
  2  set salary = (select salary from emp
  3               as of timestamp(systimestamp-7/1440)
  4               where employee_id = 178)
  5  where employee_id = 178;
SQL> select employee_id, salary from emp
  2  where employee_id = 178;
SQL> commit;
```


2. Flashbak Versions Query를 사용하여 여러 번 갱신된 행의 버전을 확인합니다.

a) 178번 사원의 급여를 확인한 후 다음과 같이 인상합니다.

```
SQL> select salary from emp
  2  where employee_id = 178;
SQL> update emp
     set salary = salary*1.05
     where employee_id = 178;
SQL> commit;
SQL> select employee_id, salary from emp
where employee_id = 178;
```

b) 시간이 조금 경과한 후 178번 사원의 급여를 한번 더 갱신합니다.

```
SQL> update emp
  2  set salary = salary*1.1
```

```
  3  where employee_id = 178;

SQL> commit;

SQL>  select employee_id, salary from emp

2        where employee_id = 178;
```

c) 시간이 조금 경과한 후 178번 사원의 급여를 한번 더 갱신합니다.

```
SQL> update emp

  2  set salary = salary*1.1

  3  where employee_id = 178;

SQL> commit;
```

d) Flashback Versions Query를 이용하여 178번 사원의 급여 변경내역을 확인합니다.

```
SQL> COL versions_starttime FOR a25

SQL> COL versions_endtime FOR a25

SQL> SELECT  versions_starttime, versions_endtime, versions_xid,

             versions_operation, salary

     FROM emp VERSIONS BETWEEN TIMESTAMP minvalue AND maxvalue

     WHERE employee_id = 178;
```


3. Flashback Table을 수행하여 테이블 전체의 잘못된 데이터를 복구합니다.

a) EMP 테이블을 갱신하는 과정에서 WHERE 절을 쓰지 않아 모든 사원의 급여가 갱신되는 실수를 시뮬레이트 합니다.

```
SQL> update emp

     set sal = 9999 ;

SQL> commit ;

SQL> select * from emp ;
```

b) 오류를 수정하고자 Flashback Table을 수행하여 테이블의 상태를 5분전으로 되돌립니다.

```
SQL> alter table emp enable row movement ;

SQL> flashback table emp to timestamp (sysdate - interval '5' minute) ;
```

c) Flashback Table이 성공했는지 확인 후 행 이동을 비활성화 합니다.

```
SQL> select * from emp ;

SQL> alter table emp disable row movement ;
```

Practice 11 Using Flashback Technology Ⅱ

1. Flashback Data Archive 기술을 사용하여 보다 오랫동안 데이터의 변경이력을 볼 수 있습니다.

a) Flashback Data Archive는 자동 UNDO 관리 방식에서 사용가능 하므로 UNDO 설정을 확인합니다.

```
SQL> show parameter undo
```

b) FDA를 저장할 새 Tablespace를 생성합니다.

```
SQL> create tablespace flash_tbs
     datafile '/u01/app/oracle/oradata/orcl/flash_tbs.dbf' size 10m autoextend on ;
```

c) 실습에 사용할 새 사용자 fa 을 생성하고 권한을 부여합니다.

```
SQL> create user fa
     identified by fa
9362
     default tablespace flash_tbs ;
SQL> grant connect, resource to fa ;
SQL> grant flashback archive administer to fa ;
```

d) fa 사용자로 로그인하고 FDA를 테이블스페이스에 할당하고 Retention 기간을 정합니다.

```
SQL> conn fa/fa
SQL> create flashback archive FLA1
     tablespace flash_tbs
     retention 1 year ;
SQL> create flashback archive FLA2
     tablespace flash_tbs
     retention 2 year ;
SQL> grant flashback archive on FLA1 to hr ;
SQL> grant flashback archive on FLA2 to hr ;
```

e) HR 사용자로 접속하여 EMP테이블의 Flashback Data Archive를 활성화합니다.

```
SQL> conn hr/hr
SQL> create table dept as select * from departments; → 없으면 생성
SQL> alter table hr.emp flashback archive FLA1 ;
```

f) 설정내용을 확인합니다.

```
SQL> select owner_name, flashback_archive_name, retention_in_days
     from user_flashback_archive ;
SQL> select * from user_flashback_archive_tables;
```

g) EMP 테이블을 변경해 봅니다.

```
SQL> update emp
```

```
      set salary = salary * 1.2
       where department_id = 80 ;
SQL> commit ;
```

h) 한 시간 이상의 시간이 흐른 후, 현재 테이블의 데이터와 FDA에 저장된 이전 데이터를 검색해 비교해 봅니다.

```
SQL> select * from emp
     where where department_id = 80 ;
SQL> select *
     from emp as of timestamp (sysdate - interval '1' hour )
     where department_id=80 ;
SQL> exit
```

i) 실습 내용을 정리합니다.

```
SQL> conn / as sysdba
SQL> alter table hr.emp no flashback archive;
SQL> drop tablespace flash_tbs including contents and datafiles;
SQL> drop user fa cascade;
```


Practice 12 Performing Flashback Database

1. Flashback Database 구성을 합니다.

a) flashback database 요구사항을 확인합니다.

```
SQL> conn /as sysdba
SQL> archive log list   → 아카이브 모드임을 확인
SQL> ALTER DATABASE FLASHBACK ON;
SQL> SELECT flashback_on  FROM v$database;
```

b) Flash_Recovery_Area 로 지정된 경로로 이동하여 Flashback log 파일 생성을 확인합니다. (XXX.FLB 파일)

```
SQL> show parameter db_recovery_file_dest
SQL> exit
```

c) Flashback Log 기록을 담당하는 RVWR 프로세스를 확인합니다.

```
$ ps –ef : grep ora_
```

2. Flashback Database를 사용해 봅니다.

a) 실습에 사용할 새 사용자 및 테이블을 생성합니다.

```
SQL> conn /as sysdba
```

```
SQL> CREATE USER tflash IDENTIFIED BY tflash;

SQL> GRANT connect, resource TO tflash;

SQL> conn tflash/tflash

SQL> CREATE TABLE ftest

     (a number);

SQL> INSERT INTO ftest

     VALUES(1);

SQL> commit;
```

b) 현재의 SCN과 시간을 확인한 후 사용자를 삭제합니다. Scn은 시간을 대신합니다.

```
SQL> conn /as sysdba

SQL> SELECT current_scn FROM v$database;

CURRENT_SCN
-----------  → 예시
    794323
SQL> SELECT scn_to_timestamp('794323') from dual;

SCN_TO_TIMESTAMP('794323')
---------------------------------------------------------------------------  → 예시

01-SEP-20 11.55.45.000000000 AM

SQL> drop user tflash cascade;
```


c) tflash 사용자가 삭제된 후에도 DB 작업이 계속 됨을 시뮬레이트 합니다.

```
SQL> conn hr/hr

SQL> create table ftest2

  2  as select * from departments;

SQL> conn /as sysdba

SQL> SELECT current_scn FROM v$database; (예) 1159153
```

d) tflash 유저 삭제 취소를 위한 Flashback Database를 실행합니다.

```
SQL> shutdown immediate;

SQL> startup mount exclusive

SQL> Flashback database to scn 794323; → 실습 (b)에서 확인한 정보
```

또는

```
SQL> flashback database to timestamp to_timestamp('2020-09-01:11:55:45','yyyy-mm-dd:hh24:mi:ss');
```

f) 읽기전용으로 데이터베이스를 열고 복구가 되었는지 확인합니다.

```
SQL> alter database open read only;
```

```
SQL> conn tflash/tflash

SQL> select * from ftest;
```

g) Flashback 시점 이후의 데이터베이스 작업은 취소되었음을 확인합니다.

```
SQL> conn hr/hr

SQL> select * from ftest2; →  Flashback 되지 않음
```

h) 데이터베이스를 종료하고 RESETLOGS로 다시 OPEN 합니다.

```
SQL> conn /as sysdba

SQL> shutdown immediate;

SQL> startup mount

SQL> alter database open resetlogs;
```

Practice 13 Managing Memory

1. 현재 메모리 관리 설정이 어떻게 되어 있는지 확인합니다.

```
$ sqlplus  / as sysdba

SQL> SHOW PARAMETER memory_target

SQL> SHOW PARAMETER memory_max_target
```

→ memory_target 파라미터가 설정되어 있으면 Oracle 11g의 Automatic Memory Management (AMM) 기능을 사용하는 것입니다.

```
SQL> col name format a30

SQL> col value format a30

SQL> select name,value from v$parameter
    where name in ('shared_pool_size', 'db_cache_size',
                   'large_pool_size', 'java_pool_size');
```

2. EM에서 다음을 실행하여 AMM을 비활성화 하고 ASMM(Automatic Shared Memory Management)를 사용하도록 설정한 후 확인합니다.

a) EM > Server Tab > Momory Advisor > Automatic Memory Management를 Disable 합니다.

b) SGA와 PGA를 별도로 관리하는 Automatic Shared Memory Management 가 활성화 됨을 확인합니다.

c) SQL*Plus에서 ASMM 관련 파리미터를 확인합니다.

```
$ sqlplus / as sysdba

SQL> show parameter sga_target

SQL> show parameter pga_aggregate_target

SQL> show parameter memory_target
```

d) EM의 Memory Advisors 페이지에서 Automatic Memory Management 를 다시 Enable합니다.

3. Non-Standard Blocksize의 테이블스페이스를 생성해 봅니다.

a) 현재 테이블스페이스의 block size를 알아봅니다.

```
SQL> select tablespace_name, block_size from dba_tablespaces;
```

b) Blocksize 가 4k인 SALES 테이블스페이스를 생성합니다.

```
SQL> create tablespace sales
  2  datafile '/u01/app/oracle/oradata/orcl/sales01.dbf' size 20m
  3  blocksize 4k; → 4K 버퍼 메모리가 없어 실패
```

c) db_4k_cache_size Memory 설정을 알아본 후 해당 Memory를 할당합니다.

```
SQL> show parameter db_4k  → 0으로 조회됨
SQL> alter system set db_4k_cache_size = 4m;
```

d) 다시 SALES Tablespace를 생성하고 blocksize를 확인합니다.

```
SQL> create tablespace sales
  2  datafile '/u01/app/oracle/oradata/orcl/sales01.dbf' size 20m
  3  blocksize 4k;
SQL> select tablespace_name , block_size
  2  from dba_tablespaces;
```

e) 실습내용을 지웁니다.

```
SQL> drop tablespace sales including contents and datafiles;
SQL> alter system set db_4k_cache_size=0;
```

Practice 14 Managing Database Performance

1. Database의 모든 resource 항목과 event 항목을 알아봅니다.

```
SQL> select name, class from v$statname;
SQL> select name from v$event_name;
SQL> select pool, sum(bytes) from v$sgastat
  2  group by pool;
```

2. HR 세션을 두 개 활성화하여 lock contention을 발생시킨 후 현상을 모니터링 합니다.

a) 두 개의 터미널을 열고 HR 사용자로 접속한 후 동일한 행을 동시에 수정합니다.

Session 1	session 2
SQL> conn hr/hr SQL> create table dept2 as select * from departments; SQL> update dept2 set department_name='CP' where department_id=60;	SQL> conn hr/hr SQL> update dept2 set department_name='CP' where department_id=60;

b) HR 세션들을 그대로 둔 채 세 번째 터미널을 실행하고 SYSDBA로 로그인하여 v$system_event를 통해 STARTUP 이후에 등록된 시스템 대기 이벤트를 확인합니다.

```
$ sqlplus  / as sysdba
SQL> select event, total_waits, time_waited from v$system_event;
```

c) v$session_wait를 사용하여 자원을 기다리고 있는 활성세션이 있는지 여부를 확인합니다.

```
SQL> select sid, username from v$session
     where username='HR';  → SID 확인 (ex)9, 10
```

```
SQL> select sid, event, p1text, wait_time, state
    from v$session_wait
     where sid in (9,10);
```

3. HR 세션을 모두 rollback 하고 종료한 후 v$session을 통해 세션정보가 더 이상 없음을 확인합니다.

session 1	session 2
SQL> rollback; SQL> exit;	SQL> rollback; SQL> exit;

SYSDBA

```
SQL> select sid, username, event, wait_time From v$session;
```

4. AWR snapshot 생성 및 ADDM 호출을 수동으로 합니다.

a) 수동으로 Snapshot을 생성한 후 실습에 사용할 bigemp 테이블을 생성합니다.

```
$ sqlplus  / as sysdba
SQL> EXECUTE  dbms_workload_repository.create_snapshot('TYPICAL');
SQL> CONNECT hr/hr
SQL> DROP TABLE bigemp PURGE;
SQL> CREATE TABLE bigemp AS SELECT * FROM employees;
SQL> DECLARE
n NUMBER;
BEGIN
FOR n IN 1..12
LOOP
INSERT INTO bigemp SELECT * FROM bigemp;
END LOOP;
COMMIT;
END;
/
SQL> UPDATE bigemp SET employee_id  = ROWNUM;
SQL> COMMIT;
```

b) 두 번째 performance data snapshot을 생성합니다.

```
SQL> EXECUTE dbms_workload_repository.create_snapshot('TYPICAL');
```

c) 생성된 snapshot을 확인하고 ADDM report를 생성해 봅니다.

```
SQL> SELECT snap_id, begin_interval_type FROM dba_hist_snapshot
     ORDER BY snap_id;
```

→ 생성되어진 snapshot ID 확인

```
SQL> @$ORACLE_HOME/rdbms/admin/addmrpt
```

→ 실행 시 start snapshot ID와 end snapshot ID addm report file 이름을 입력해야 한다.(ex) addm_test.txt

5. 중복 SQL을 식별하고 CURSOR_SHARING 파라미터의 설정으로 문제를 해결해 봅니다.

a) HR이 상수부분만 다른 중복 SQL을 여러 번 작성합니다.

```
$ sqlplus hr/hr

SQL> SELECT * FROM bigemp

     WHERE employee_id = 1; → 번호만 바꾸어가며 9번까지 조회합니다.
```

b) 별도의 터미널에서 V$SQL을 조회하여 중복SQL의 CURSOR 공유여부를 진단합니다.

```
$ sqlplus / as sysdba

SQL> select sql_text, executions from v$sql

     where sql_text like 'select * from bigemp%';

SQL> show parameter cursor_sharing
```

→ EXACT로 조회되면 상수가 다른 동일한 SQL이 CURSOR를 공유하지 않습니다.

c) 상수만 서로 다른 중복 SQL이 커서를 공유하도록 cursor_sharing의 값을 FORCE로 변경한 후 다음 실습을 위해 shared pool을 비웁니다.

```
SQL> alter system set cursor_sharing=force;

SQL> alter system flush shared_pool;
```

d) 다시 HR이 중복SQL을 작성합니다.

```
$ sqlplus hr/hr

SQL> SELECT * FROM bigemp

     WHERE employee_id = 11; → 번호만 바꾸어가며 19번까지 조회합니다.
```

e) SYS 세션에서 V$SQL을 조회하여 중복SQL의 CURSOR 공유여부를 확인합니다.

```
SQL> select sql_text, executions from v$sql

     where sql_text like 'select * from bigemp%';
```


Practice 15 Managing Performance by SQL Tuning

1. sqlplus 에서 dbms_sqltune 패키지를 호출하고 SQL Tuning Advisor 실행하여 다음과 같이 BIGEMP_COUNT라는 이름의 튜닝 태스크를 생성하고 실행해봅니다.

```
SQL> conn /as sysdba

SQL> grant dba  to

SQL> conn hr/hr

SQL> declare

  l_task_id    varchar2(20);

  l_sql        varchar2(2000);

begin

     l_sql := ' SELECT last_name FROM bigemp

            WHERE  employee_id NOT IN

                       (SELECT manager_id FROM   bigemp

                        WHERE  manager_id IS NOT NULL)';

     l_task_id := dbms_sqltune.create_tuning_task (

       sql_text  => l_sql,

       user_name  => 'HR',
```

```
        scope     => 'COMPREHENSIVE',
        time_limit => 120,
        task_name  => 'BIGEMP_COUNT'
      );
      dbms_sqltune.execute_tuning_task ('BIGEMP_COUNT');
end;
/
```

2. 태스크 실행 결과를 확인한 후 삭제합니다.

```
SQL> set serveroutput on size 999999
SQL> set long 999999
SQL> select dbms_sqltune.report_tuning_task ('BIGEMP_COUNT')
     from dual;
SQL> exec dbms_sqltune.drop_tuning_task ('BIGEMP_COUNT');
```

→ 실습 환경에 따라 결과가 나오지 않을 수도 있습니다.

Practice 16 Managing Resources

시스템 리소스 활용 및 제어 성능을 향상시키기 위해 Database Resource Manager를 사용하려고 합니다.

1. EM을 사용하여 APPUSER와 BATCHUSER라는 이름의 resource group을 생성하고 그룹에 유저를 추가합니다.

a) Server > Resource Manager > Consumer Groups를 선택합니다.

b) Create 버튼을 클릭한 후 Consumer Group 필드에 APPUSER를 입력합니다.

c) ADD 버튼을 누르고 HR, SCOTT 유저를 select 하여 그룹에 포함시킵니다.

d) OK 버튼을 눌러 Consumer Group 생성을 완료합니다.

e) 동일한 방법으로 BATCHUSER그룹을 생성하고 SYSTEM, HR사용자를 포함시키십시오.

2. WEEKLY_DAY, WEEKLY_NIGHT 라는 이름의 Resource Plan을 생성합니다.

a) Server > Resource Manager > Plans를 선택합니다.

b) Create 버튼을 누르고 Plan 필드에 WEEKLY_DAY 를 입력합니다.

c) Modify버튼을 클릭, Available Groups/Subplans 영역에서 APPUSER 및 BATCHUSER를 선택하여 Resource Allocations 영역으로 Move 한 후 OK합니다.

d) General 탭에서 Mode를 Percentage로 설정한 후 APPUSER 70, BATCHUSER 25, OTHER_GROUP 5를 입력하여 CPU사용을 분배합니다.

e) Session Pool 탭을 선택하고 Maximum Number of Active Sessions 필드에 APPUSER , BATCHUSER, OTHER_GROUPS 에 각각 2, 1, 1을 할당합니다. Activation Queue Timeout (sec)은 모두 5로 지정합니다.

f) Idle Time영역을 선택한 후 max_idle_time 필드에 모두 60을 입력합니다. Max Idle Time if Blocking Another Session 필드는 모두 0을 입력합니다.

g) OK버튼을 눌러 완료합니다.

h) 동일한 프로세스로 WEEKLY_NIGHT Plan을 생성합니다.

3. WEEKLY_DAY plan을 활성화합니다.

a) Server > Resource Manager > Plans 페이지에서 WEEKLY_PLAN을 선택한 후 Actions 리스트에서 Activate를 선택하고 Go 버튼을 누릅니다.

b) "Are you sure you want to activate Resource Plan WEEKLY_DAY?"에 Yes를 클릭하면 Resource Plan이 활성화 됩니다.

c) SQLPlus에서 resource_manager_plan 파라미터에 WEEKLY_DAY 플랜이 선택되어 있는지 확인합니다.

```
SQL> show parameter resource_manager_plan
```

4. 주중 오전에 활성화되는 윈도우를 생성합니다.

a) Server > Oracle Scheduler > Windows를 선택합니다.

b) Create를 누르고 Name 필드에 WEEKDAY를 입력하고 Resource Plan 필드를 드롭다운하여 WEEKLY_DAY 플랜을 선택합니다.

c) Schedule영역에서 다음 항목을 만족하도록 윈도우를 생성합니다.

- Repeat : 매일 오전 9:00 부터 3시간
- End Date : 2020.09.16

d) Server > Resource Manager > Plans를 선택한 후 Delete를 클릭하여 해당 Plan을 삭제합니다.

Practice 17 Automating Tasks with the Scheduler

1. HR의 COPY_EMP 테이블과 INS_CEMP procedure를 생성합니다.

```
SQL> conn hr/hr
SQL> create table copy_emp
  2  as
  3  select * from employees
SQL> create or replace procedure ins_cemp
     is
     begin
     insert into copy_emp
     select * from employees;
     commit;
     end ins_cemp;
     /
```

2. 작업 내에서 프로그램을 사용한다는 사실을 DB가 인지하도록 프로그램을 생성합니다.

```
SQL> begin
          dbms_scheduler.create_program(
              program_name   => 'INS_COPY_EMP',
              program_type   => 'STORED_PROCEDURE',
              program_action =>  'HR.INS_CEMP',
              enabled        => TRUE,
              comments       => 'Insert into COPY_EMP');
      end;
      /
```

3. 매 10분 간격으로 실행되는 EVERY_10_MIN이라는 이름의 스케줄을 생성합니다.

```
SQL> begin
```

```
          dbms_scheduler.create_schedule
           ( schedule_name   => 'EVERY_10_MINS',
             repeat_interval => 'FREQ=MINUTELY; INTERVAL=10',
             comments        => 'Every 10-mins'
           );
     end;
     /
```

4. 프로그램과 스케줄을 연관시킴으로써 새로운 작업을 생성합니다.

```
SQL> begin
     dbms_scheduler.create_job
    (
         job_name      => 'JOB_INS_COPY_EMP',
         program_name  => 'INS_COPY_EMP',
         schedule_name => 'EVERY_10_MINS',
         comments      => 'Scheduled job for insert into copy_emp ',
         enabled       => TRUE
          );
end;
/
```


5. 작업한 내용을 EM에서 확인할 수 있다.

a) Database Control >> Server tab >> Oracle Scheduler

b) Jobs, Schedules, Programs 에서 각각 확인합니다.

6. 'EVERY_10_MINS' 라고 명명한 schedule의 종료날짜를 EM에서 수정합니다.

a) Database Control >> Server tab >> Oracle Scheduler >> Schedule을 선택합니다.

b) EVERY_10_MINS를 클릭하고 end date를 1시간 후로 설정합니다.

7. 시간이 지난 후 작업의 성공여부를 확인해 봅니다.

a) Database Control >> Server tab >> Oracle Scheduler >> Jobs를 선택

b) History tab을 누르고 해당 작업의 성공실패 여부를 확인할 수 있습니다.

8. 10분 간격으로 HR의 COPY_EMP 테이블의 데이터가 증가되고 있는지 확인합니다.

```
SQL> conn hr/hr
SQL> select count(*) from copy_emp;
```

9. 해당 작업이 끝나면 clean up 합니다.

```
SQL> exec dbms_scheduler.drop_job('JOB_INS_COPY_EMP');
```

```
SQL> exec dbms_scheduler.drop_program('INS_COPY_EMP');
SQL>  exec  dbms_scheduler.drop_schedule('HR.EVERY_5_MINS');
```

Practice 18 Managing Space

1. 테이블스페이스에 설정되어있는 Threshold 값 확인해 봅니다.

```
SQL> conn /as sysdba
SQL> select  METRICS_NAME,  WARNING_VALUE, CRITICAL_VALUE
     from dba_thresholds
     where metrics_name like 'Tablespace%'
```

2. HR의 BIGEMP 테이블에 DELETE가 많이 발생하여 공간을 재구성해야 합니다.

a) 현재의 테이블상태를 알기 위해 분석을 수행합니다.

```
SQL> exec dbms_stats.gather_table_stats('HR','BIGEMP');
SQL> select table_name, num_rows, blocks, empty_blocks
  2  from user_tables
  3  where table_name = 'BIGEMP';
```

→ 조회되는 값을 기록해 둡니다.

b)BIGEMP에 삭제가 많이 일어나도록 한 후 다시 한번 분석을 수행하고 결과를 봅니다.

```
SQL> delete from bigemp
  2  where department_id in (50,60,80);
SQL> commit;
SQL> exec dbms_stats.gather_table_stats('HR','BIGEMP');
SQL> select table_name, num_rows, blocks, empty_blocks
  2  from user_tables
  3  where table_name = 'BIGEMP';
```

c) BIGEMP 테이블을 Shrink Space기능으로 재구성합니다.

```
SQL> alter table bigemp enable row movement;
SQL> alter table bigemp shrink space;
SQL> alter table bigemp disable row movement;
```

d) 테이블 공간이 다시 구성되었는지 확인합니다.

```
SQL> exec dbms_stats.gather_table_stats('HR','BIGEMP');
SQL> select table_name, num_rows, blocks, empty_blocks
  2  from user_tables
  3  where table_name ='BIGEMP';
```

2. BIGEMP 테이블에 UPDATE로 인한 Row Migration이 심각하게 발생하는 상황에 대하여 해결해 봅니다.

a) email 열의 사이즈를 늘린 후 이전보다 긴 행으로 UPDATE합니다.

```
SQL> desc bigemp
SQL> alter table bigemp
```

```
  2  modify email varchar2(100);
SQL> update bigemp
 set email='aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa';
SQL> commit;
SQL> create index bigemp_id_ix on bigemp(employee_id);
```

b) ANALYZE 명령으로 테이블을 분석하여 문제를 감지합니다.

```
SQL> analyze table bigemp compute statistics;
SQL> select table_name, num_rows, blocks, empty_blocks, chain_cnt
  2  from user_tables
  3  where table_name = 'BIGEMP';
SQL> select index_name, status from user_indexes
  2* where table_name = 'BIGEMP';
```

→ DBMS_STATS는 대부분의 테이블 분석을 수행하지만 Migrated Row를 분석하지는 못합니다.테이블의 Migration은 USER_TABLES의 CHAIN_CNT에서 확인 할 수 있습니다.

c) 테이블 이동으로 문제를 해결합니다. 이 작업은 관련 인덱스를 UNUSABLE 상태로 만들기 때문에 인덱스도 재구축 되어야 합니다.

```
SQL> alter table bigemp move;
SQL>  select index_name, status from user_indexes
  2   where table_name = 'BIGEMP';
SQL>  alter index bigemp_id_ix  rebuild;
SQL> select index_name, status from user_indexes
  2   where table_name = 'BIGEMP';
SQL> analyze table bigemp compute statistics;
SQL> select table_name, num_rows, blocks, empty_blocks, chain_cnt
  2  from user_tables
  3  where table_name ='BIGEMP';
```


Practice 19 Managing Space for the Database

1. Transportable Tablespace 를 해 봅니다.

a) SOURCE System에서 터미널 실행 후 다음을 실행합니다.

```
<SOURCE>
$ mkdir $HOME/dir_1
SQL> conn /as sysdba
SQL> CREATE TABLESPACE insa
     DATAFILE '/u01/app/oracle/oradata/orcl/insa01.dbf' SIZE 10M;
SQL> CREATE TABLE hr.xtrans
     TABLESPACE insa
     AS SELECT employee_id, last_name FROM hr.employees;
```

b) Directory Object를 생성합니다.(이미 있을 수 있습니다.)

```
SQL> create directory dir_1 as '/home/oracle/dir_1';
SQL> grant read, write on directory dir_1 to public;
```

c) INSA 테이블스페이스를 Transportable 방식으로 Export 합니다.

SQL> alter tablespace insa read only;

SQL> exit

$ expdp transport_tablespaces=insa directory=dir_1dumpfile=transinsa.dmp

→username : sys/oracle as sysdba 입력

$ cp /u01/app/oracle/oradata/orcl/insa01.dbf $HOME/dir_1/.

$ cd $HOME/dir_1

ls

d) export후 다시 READ WRITE 상태가 된 INSA 테이블스페이스에 객체가 추가되는 변경이 발생합니다.

$ sqlplus / as sysdba

SQL> ALTER TABLESPACE insa read write;

SQL> create table hr.xtrance_dept

     As select * from hr.departments;

e) Target System에 INSA Tablespace가 없다고 가정하기위해 INSA Tablespace를 DROP 하여 Target System을 대신합니다.

SQL> drop tablespace insa including contents and datafiles;

SQL> exit

f) Target System에 데이터파일을 배치한 후 import를 수행합니다.

$ cp /home/oracle/dir/insa01.dbf /u01/app/oracle/oradata/orcl/.

$impdp transport_datafiles='/u01/app/oracle/oradata/orcl/insa01.dbf' directory=dir_1 dumpfile=transinsa.dmp

g) 테이블스페이스의 상태를 확인한 후 READ WRITE상태로 변경합니다.

SQL> SELECT name FROM v$datafile;

SQL> SELECT name, status FROM dba_tablespaces;

SQL> alter tablespace insa read write;

SQL> desc hr.xtrans

SQL> desc hr.xtrans_dept → export후 추가된 객체는 없습니다.

h) 실습을 Clean-Up 합니다.

SQL> drop tablespace insa including contents and datafiles;