Oracle9i Dataguard 기술서

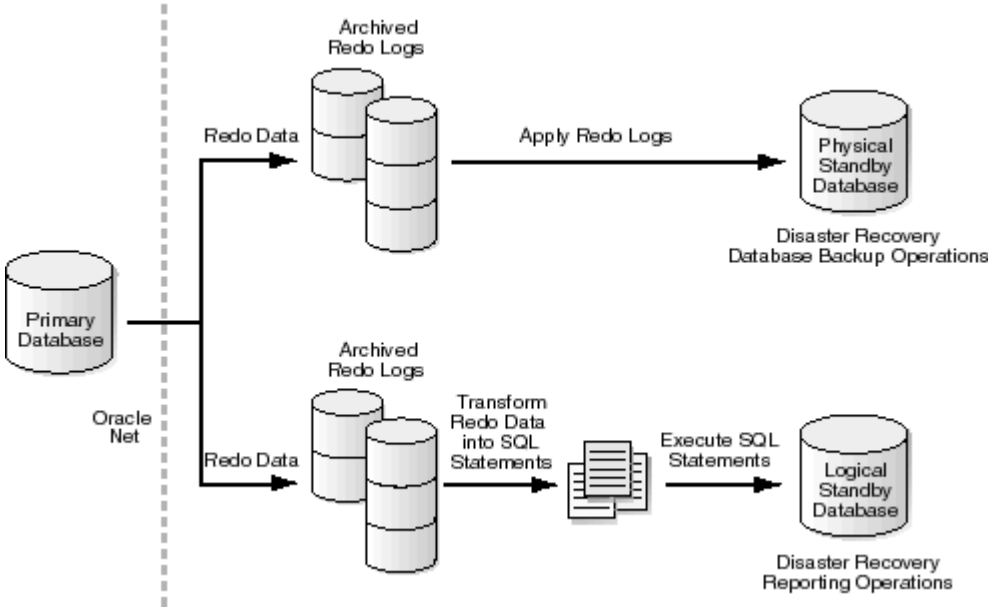
작성일 : 2005년 3월 24일
업데이트 : 2006년 1월 22일 v1.8 Final
작성자 : LG카드 중형서버운영파트 DBA 민연홍
Phone : 016-744-0220
E-Mail : ses0124@hanmail.net

목 차

1. dataguard 개요 및 아키텍처

(1) dataguard 란 무엇인가?

- primary DB와 standby DB를 동기화시켜, primary DB가 하드웨어 장애 등의 문제가 생겼을 경우 standby DB로 failover 또는 switchover 시킬 수 있는 시스템 구성을 말한다.
- Oracle Net을 통해서 primary DB의 변경정보를 standby DB로 적용시켜 운영된다.



(2) switchover and failover

① 자동실행이 아니라 DBA가 action을 취해야 한다.

② switchover

- OS 작업 또는 서버 PM작업 시 사용(primary -> standby , standby -> primary)

③ failover

- 디스크 fail 등 긴급상황에서 사용, dataguard 재구성 필요

(3) standby DB의 종류

① Physical standby database

- block 대 block 기반으로 primary DB의 redo log를 적용시켜 standby DB를 동기화

② Logical standby database

- 같은 schema 정의로 공유
- primary DB의 sql 문장을 standby DB에 적용

(4) dataguard의 서비스 종류

① Log transport Services

- primary DB에서 standby DB로 redo log 정보를 자동으로 전송

② Log Apply Services

- redo log를 standby DB에 적용

③ Role Management Service

- 데이터베이스는 primary/standby로 두 가지의 상대적으로 배타적인 role을 가진다. Role Management Service는 log transport service와 log apply service를 failover 또는 switchover의 상황에 동적으로 변경할 수 있다.

(5) protection mode

① Maximum Protection - primary DB와 standby DB의 redo log를 동기화 시킨다. standby DB가 네트워크 이상 등의 이유로 standby로의 전송이 안될 경우 primary DB를 halt시킨다. 데이터는 서로 동기화되어 primary DB에서 commit을 하게 되면 standby DB에서 commit이 완료될 때까지 primary DB에서 commit 완료를 하지 않는다. 성능에는 문제를 줄 소지가 있으나 failover 상황이 오더라도 데이터 손실은 없다.
physical standby DB에만 가능하다.

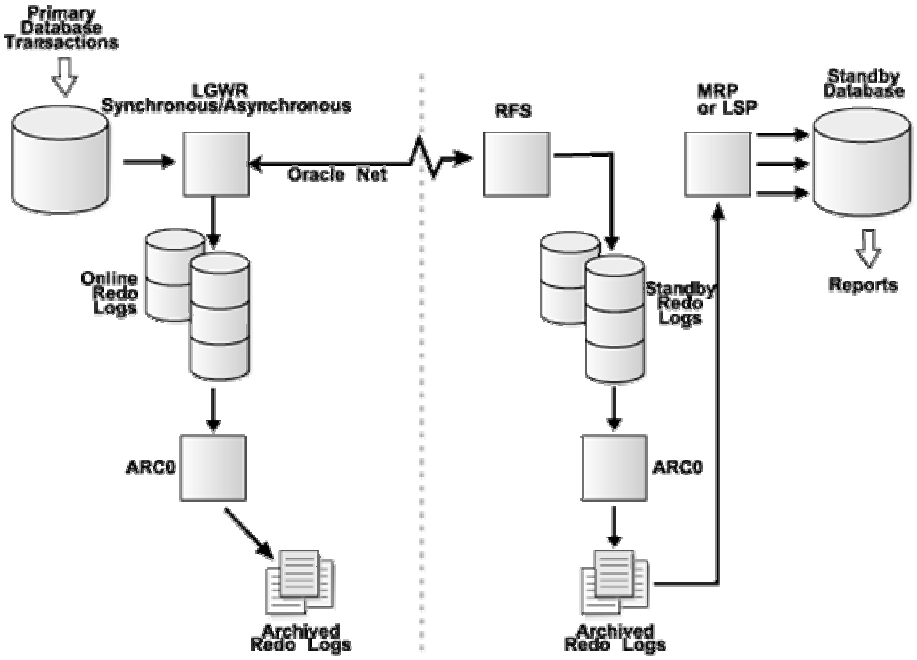
② Maximum availability - Maximum Protection 과 마찬가지로 primary DB와 standby DB를 동기화시킨다. 단 standby DB가 네트워크 문제 등의 이유로 전송이 안될지라도 halt되지는 않는다. 데이터는 maximum protection 과 마찬가지로 primary DB에서 commit을 하게 되면 standby DB에서 commit이 완료될 때까지 primary DB에서 commit 완료를 하지 않는다. 만약 standby DB가 unavailable상태일 경우 임시로 불일치 시킨다. 또 다시 standby DB가 available하면 자동으로 동기화 시킨다. 성능에는 문제를 줄 소지가 있으나 failover 상황이 오더라도 데이터 손실은 거의 없다. physical standby, logical standby DB 모두 가능하다.

③ Maximum Performance - default protection mode이다. primary data에 대한 protection 이 가장 낮다. primary database에 transaction이 수행되면 이것을 asynchronous 하게 standby DB에 적용한다. 즉, maximum protection, maximum availability의 경우에는 standby DB에 적용(commit) 될 때까지 primary db의 transaction 이 적용(commit)되지 않았으나, Maximum Performance 모드에서는 비 동기화 시키므로 primary DB에서 standby DB가 transaction 적용이 끝날 때까지 기다리지 않는다. 즉 standby db의 문제로 인해서 primary DB에 성능영향이 가지 않는다. 단, failover시에는 약간의 데이터 손실을 가져올 수 있다.

(6) dataguard의 시스템 구성(2가지 종류)

① physical standby database 구성 (LGWR processes를 사용한 Physical standby DB)

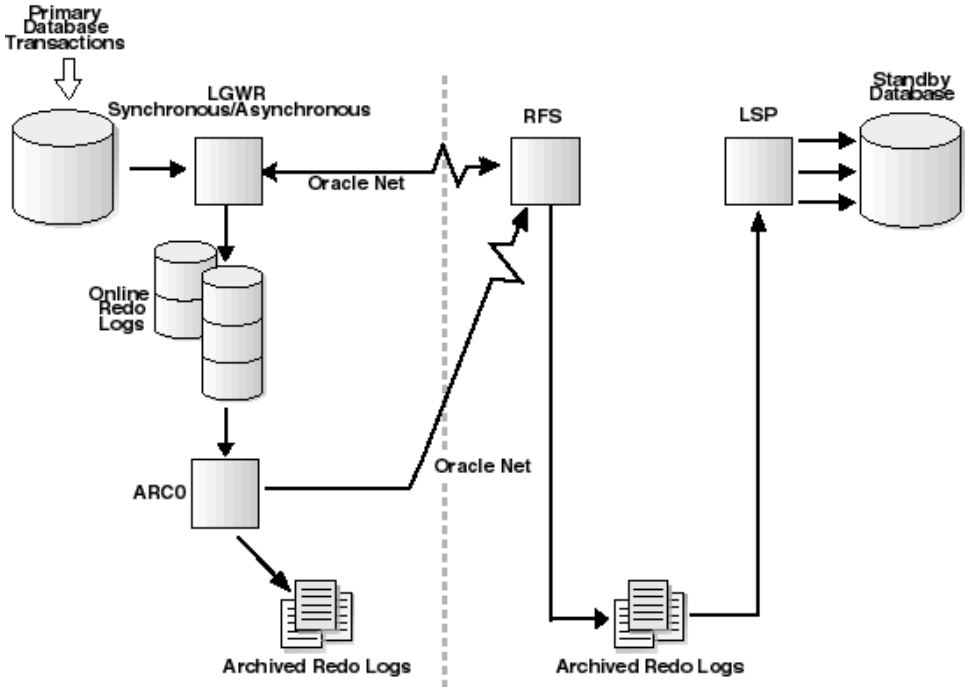
Archiving to a Physical standby Destination Using the Logwriter Process



- primary db의 LGWR 프로세스가 standby DB로 redo log를 보내고, standby db의 RFS 프로세스가 이 redo log를 standby redo log에 적용시킨다. archiving되면 archived redo logs가 되고 이것을 MRP process가 standby DB에 적용시킨다.

② logical standby DB 구성

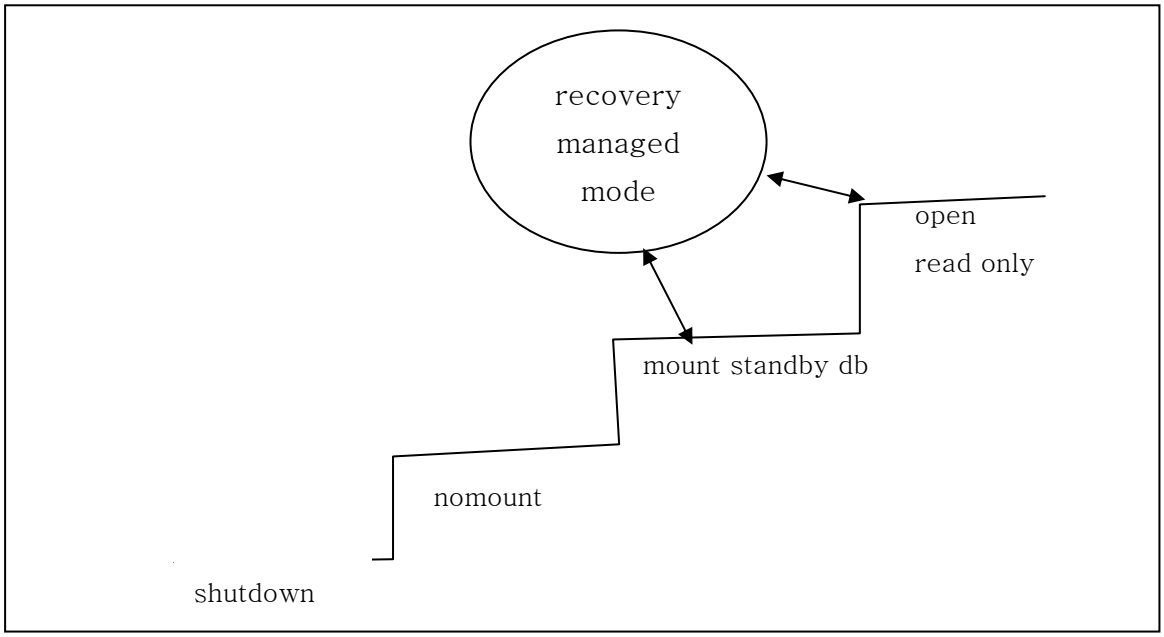
Archiving to a Logical standby Destination Using the Logwriter Process



- logical standby DB는 primary DB에서 수행된 SQL문장을 LGWR프로세스가 standby DB로 보내고 RFS 프로세스가 받아서 Archived redo logs에 쓴다. LSP (Logical standby process)가 standby DB에 적용시킨다.
- primary DB에서 log switch가 일어나게 되면 standby DB의 RFS 프로세스에 이를 알려주어 log switch가 되도록 한다.

2. standby DB 기동방법

- 주의 standby db의 startup 방식을 보면 아래와 같다. 아래 그림을 기억해두면 편하다.



① standby DB를 read only mode에서 managed recovery mode로 변경

- 그대로 명령 또는 shutdown immediate 이후 재기동
- 첫번째 방법

SQL> alter database open read only;
SQL> recover managed standby database disconnect;
SQL>

- 두번째 방법

SQL> shutdown immediate
Database dismounted.
SQL> startup nomount
SQL> alter database mount standby database;
SQL> recover managed standby database disconnect;

② shutdown 에서 managed recovery mode 로 변경

SQL> startup nomount
SQL> alter database mount standby database;
SQL> recover managed standby database disconnect;

③ managed recovery mode 에서 read only mode 로 변경

SQL> recover managed standby database cancel;
SQL> alter database open read only;

④ read only standby DB 에서 managed recovery mode 로 변경

(먼저 standby DB에 연결된 모든 세션을 종료)

SQL> recover managed standby database disconnect;

- 만약 유저의 세션 때문에 실패할 경우

SQL> shutdown immediate

SQL> startup nomount

SQL> alter database mount standby database;

SQL> recover managed standby database disconnect;

3. 시스템 구축 (실습)

- primary db이름은 MIN 이고, standby db이름은 STBY 이다.
- 여기서는 하나의 서버에서 2개 DB를 구성하는 방법으로 수행한다.
- 2개 서버에서도 똑같이 수행할 수 있다.
- primary DB는 /data1/oradata/MIN 에 구성되어 있다.
- standby DB는 /data1/oradata/STBY에 구성되어 있다.

(1) 리스너 설정 및 기동

아래와 같은 네트워크 설정을 해준다. 각 서버마다 설정해준다.

- MIN DB에서 설정(primary DB)

```
vi $ORACLE_HOME/network/admin/listener.ora
smsvr1_MIN =
  (ADDRESS_LIST =
        (ADDRESS= (PROTOCOL= TCP)(Host= smsvr1)(Port=2001))
  )
SID_LIST_smsvr1_MIN =
  (SID_LIST =
    (SID_DESC =
      (ORACLE_HOME= /u/pkg/oracle/ora9i/app/oracle/product/9.2.0)
      (SID_NAME = MIN)
    )
  )
```

- STBY DB 에서 설정(physical standby DB)

```
vi $ORACLE_HOME/network/admin/listener.ora
smsvr1_STBY =
  (ADDRESS_LIST =
        (ADDRESS= (PROTOCOL= TCP)(Host= smsvr1)(Port=2002))
  )
SID_LIST_smsvr1_STBY =
```

```
  (SID_LIST =
    (SID_DESC =
      (ORACLE_HOME= /u/pkg/oracle/ora9i/app/oracle/product/9.2.0)
     (SID_NAME= STBY)
    )
 )
```

(2) tnsnames.ora 설정

tnsnames.ora 파일을 설정한다. 서로 네트워크가 가능하도록 하는데 이름을 재대로 써야 한다.

log_archvie_dest_2='service=STBY LGWR SYNC AFFIRM' 일 경우

STBY 는 tnsnames.ora 에서의 접속이름을 말한다.

- MIN DB 설정 primary DB에서 설정 (standby DB로 가는 네트워크 구성)

```
vi $ORACLE_HOME/network/admin/tnsnames.ora
STBY =
 (DESCRIPTION =
    (ADDRESS = (PROTOCOL= TCP)(Host= smsvr1)(Port= 2002))
    (CONNECT_DATA = (SID = STBY))
  )
```

- STBY DB 설정, standby DB에서 설정 (primary DB로 가는 네트워크 구성)

```
vi $ORACLE_HOME/network/admin/tnsnames.ora
MIN =
 (DESCRIPTION =
    (ADDRESS = (PROTOCOL= TCP)(Host= smsvr1)(Port=2001))
    (CONNECT_DATA = (SID = MIN))
  )
```

(3) 오라클 초기화 파라미터 설정

- MIN DB (primary DB)

```
vi $ORACLE_HOME/dbs/initMIN.ora

db_name=MIN
compatible=9.2.0
control_files=('/data1/oradata/MIN/control01.ctl','/data1/oradata/MIN/control02.ctl')
shared_pool_size=64M
db_cache_size=16M
undo_management=auto
background_dump_dest=/data1/oradata/MIN/bdump
```

```
user_dump_dest=/data1/oradata/MIN/udump
core_dump_dest=/data1/oradata/MIN/cdump
log_archive_start=true
log_archive_dest_1="location=/data1/oradata/MIN/archive1"
log_archive_format=%t_%s.arc

#### 데이터가드를 위해서 변경하지 말 것
remote_archive_enable=true
dg_broker_start=true
log_archive_dest_state_1=enable
log_archive_max_processes=3
standby_file_management=auto
db_file_name_convert='/data1/oradata/STBY','/data1/oradata/MIN'
log_file_name_convert='/data1/oradata/STBY','/data1/oradata/MIN'
standby_archive_dest='/data1/oradata/MIN/archive2'

##### standby DB이면 열고, primary DB이면 닫는다.
#fal_server=STBY
#fal_client=MIN
#lock_name_space=STBY

## primary DB이면 열고, standby DB이면 닫는다.
log_archive_dest_2='SERVICE=STBY LGWR SYNC'
```

- STBY DB (standby DB) -주의할 것은 DB이름은 MIN 이다. instance 이름만 STBY 이다.

```
vi $ORACLE_HOME/dbs/initSTBY.ora

db_name=MIN
compatible=9.2.0
control_files=('/data1/oradata/STBY/control01.ctl','/data1/oradata/STBY/control02.ctl')
shared_pool_size=64M
db_cache_size=16M
undo_management=auto
background_dump_dest=/data1/oradata/STBY/bdump
user_dump_dest=/data1/oradata/STBY/udump
core_dump_dest=/data1/oradata/STBY/cdump
log_archive_start=true
log_archive_dest_1="location=/data1/oradata/STBY/archive1"
log_archive_format=%t_%s.arc
```

```
### 데이터가드를 위해서 변경하지 말 것
remote_archive_enable=true
dg_broker_start=true
log_archive_dest_state_1=enable
log_archive_max_processes=3
standby_file_management=auto
db_file_name_convert='/data1/oradata/MIN','/data1/oradata/STBY'
log_file_name_convert='/data1/oradata/MIN','/data1/oradata/STBY'
standby_archive_dest='/data1/oradata/STBY/archive2'

## standby DB일 경우 아래를 열기, primary DB일 경우 닫기
fal_server=MIN
fal_client=STBY
lock_name_space=STBY
##primary DB이면 열고, standby DB이면 닫는다.
#log_archive_dest_2='SERVICE=MIN LGWR SYNC'
```

- 파라미터 설정에서 주의해야 할 것을 먼저 보면,
  standby_file_management=auto 로 되어 있어야 primary DB에서 수행한 물리적인 테이블스페이스 추가 및 데이터파일 추가 시 standby DB에 자동으로 생성이 된다.
  standby_file_management=auto로 되어 있을 경우
  db_file_name_convert='/data1/oradata/STBY','/data1/oradata/MIN'
  log_file_name_convert='/data1/oradata/STBY','/data1/oradata/MIN'
  의 파라미터에서 ' A 부분','B 부분' 에서 B부분이 자신의 primary DB의 경로를 말하고 A부분이 변환할 standby DB의 경로를 말한다. 파라미터를 확인해보자.
  fal_server, fal_client 는 standby DB에서만 사용한다. fal_server는 primary DB를 설정하고 fal_client는 standby DB를 설정해둔다. 이것을 설정할 경우 primary DB와 standby DB에 redo log의 gap이 발생했을 경우 자동으로 맞추어주는 역할을 한다.
  fal_server=MIN
  fal_client=STBY
  lock_name_space는 한 대의 서버에서 primary, standby DB를 운영할 경우 사용한다. primary , standby DB 모두 db_name은 같다. 단 instance_name만 다를 뿐이며 똑같은 DB이름을 가진 instance가 startup 하기 위해서는 lock_name_space를 지정해주어야 한다. 이것은 standby DB에서만 지정해둔다.
  lock_name_space=STBY

(4) primary DB를 online backup으로 standby DB 위치로 restore

<MIN DB primary DB>

- primary DB를 DB를 online backup으로 이동. online backup을 하는 것이므로 primary DB의 redo log는 standby DB로 전송하지 않는다. standby DB구성 시 자동으로 redo log가 생성된다.

- primary DB는 24시간 서비스 이므로 shutdown 이 불가능한 것을 가정하에 구성한다.

SQL> select tablespace_name, file_name, bytes/1024/1024 mega from dba_data_files;

TABLESPACE_NAME	FILE_NAME	MEGA
SYSTEM	/data1/oradata/MIN/system01.dbf	250
UNDOTBS	/data1/oradata/MIN/undotbs.dbf	100
USERS	/data1/oradata/MIN/users01.dbf	100

SQL> select name , bytes/1024/1024 mega from v$tempfile;

NAME	MEGA
/data1/oradata/MIN/temp01.dbf	100

- 여기서는 cp 명령이 standby DB로 전송하는 것을 뜻한다. ftp로 primary db의 백업을 전송한다.

```
SQL> alter tablespace system begin backup;
SQL> !cp /data1/oradata/MIN/system01.dbf /data1/oradata/STBY/system01.dbf
SQL> alter tablespace system end backup;
SQL> alter tablespace undotbs begin backup;
SQL> !cp /data1/oradata/MIN/undotbs.dbf /data1/oradata/STBY/undotbs.dbf
SQL> alter tablespace undotbs end backup;
SQL> alter tablespace users begin backup;
SQL> !cp /data1/oradata/MIN/users01.dbf /data1/oradata/STBY/users01.dbf
SQL> alter tablespace users end backup;
```

- tempfile은 그대로 복사를 한다. begin backup, end backup 이 필요 없다. 단 tempfile 이어야 한다.

```
SQL> !cp /data1/oradata/MIN/temp01.dbf /data1/oradata/STBY/temp01.dbf
```

(5) primary DB에서 standby control file을 생성해서 standby DB로 전송

<MIN DB primary DB>

primary DB에서 standby control file을 생성해서 standby DB로 전송

```
SQL> alter database create standby controlfile as '/data1/oradata/STBY/stbyctl.ctl';
```

(6) standby DB에서 사용할 control file을 배치

<STBY DB standby DB의 control file에서>

standby DB에서 사용할 control file을 initSTBY.ora 파일에 있는 control file 위치에 배치를 한다.

```
SQL> !cp /data1/oradata/STBY/stbyctl.ctl /data1/oradata/STBY/control01.ctl
SQL> !cp /data1/oradata/STBY/stbyctl.ctl /data1/oradata/STBY/control02.ctl
```

(7) standby DB 기동

<STBY DB standby DB 에서 수행>

standby DB를 기동시킨다. startup mount standby database 라는 명령은 없다.

nomount까지 기동한 후 standby DB로 mount를 하고 recovery managed mode로 MRP 프로세스를 기동시켜야 한다.

```
SQL> startup nomount
SQL> alter database mount standby database;
```

(8) standby DB에 standby redo log file 생성

<STBY DB standby DB 에서 수행>

우리는 처음으로 standby DB를 구성하였으므로 standby redo log를 확인해서 넣어주어야 한다.
이후에는 새로 만들 필요가 없다.

```
SQL> select * from v$logfile;

    GROUP# STATUS  TYPE    MEMBER
---------- ------- ------- ----------------------------------------
         1         ONLINE  /data1/oradata/STBY/log01a.log
         2         ONLINE  /data1/oradata/STBY/log02a.log
         3         ONLINE  /data1/oradata/STBY/log03a.log
SQL> alter database add standby logfile
     '/data1/oradata/STBY/stbylog01a.log' size 10M;
SQL> alter database add standby logfile
    '/data1/oradata/STBY/stbylog02a.log' size 10M;
SQL> alter database add standby logfile
    '/data1/oradata/STBY/stbylog03a.log' size 10M;
SQL> select * from v$logfile;
    GROUP# STATUS  TYPE    MEMBER
---------- ------- ------- ----------------------------------------
         1         ONLINE  /data1/oradata/STBY/log01a.log
         2         ONLINE  /data1/oradata/STBY/log02a.log
         3         ONLINE  /data1/oradata/STBY/log03a.log
         4         STANDBY /data1/oradata/STBY/stbylog01a.log
         5         STANDBY /data1/oradata/STBY/stbylog02a.log
         6         STANDBY /data1/oradata/STBY/stbylog03a.log
```

(9) primary DB에 standby redo log file 생성

<MIN DB primary DB 에서 수행>

서버문제가 발생했을 경우 takeover를 해야 하므로 primary db도 standby DB가 될 수 있기 때문에 미리 standby redo log를 만든다.

```
SQL> select * from v$logfile;
    GROUP# STATUS  TYPE    MEMBER
---------- ------- ------- ----------------------------------------
```

```
         1          ONLINE   /data1/oradata/MIN/log01a.log
         2          ONLINE   /data1/oradata/MIN/log02a.log
         3          ONLINE   /data1/oradata/MIN/log03a.log

SQL> alter database add standby logfile
     '/data1/oradata/MIN/stbylog01a.log' size 10M;
SQL> alter database add standby logfile
     '/data1/oradata/MIN/stbylog02a.log' size 10M;
SQL> alter database add standby logfile
     '/data1/oradata/MIN/stbylog03a.log' size 10M;

SQL> select * from v$logfile;
    GROUP# STATUS   TYPE     MEMBER
---------- ------- ------- ----------------------------------------
         1          ONLINE   /data1/oradata/MIN/log01a.log
         2          ONLINE   /data1/oradata/MIN/log02a.log
         3          ONLINE   /data1/oradata/MIN/log03a.log
         4          STANDBY /data1/oradata/MIN/stbylog01a.log
         5          STANDBY /data1/oradata/MIN/stbylog02a.log
         6          STANDBY /data1/oradata/MIN/stbylog03a.log
```

(10) standby DB를 recovery managed mode로 기동

<STBY DB standby DB 에서 수행>

standby DB를 recovery managed mode로 변경한다. MRP 프로세스가 생긴다.

```
SQL> recover managed standby database disconnect;
```

(11) log switch 적용

<MIN DB primary DB 에서 수행>

standby DB를 구성하는 동안 primary DB와 gap이 생겼을 것이다.

Current redo log를 적용시킨다.

```
SQL> alter system archive log current;
```

(12) primary DB 점검

```
SQL> select i.instance_name, i.status instance_status, d.name dbname, d.database_role db_role,
     d.switchover_status switchover_status , d.protection_mode
          from v$database d, v$instance i;
```

=> 중요한 점검 포인트이다. Switchover_Status가 TO_STANDBY 이어야 한다.

```
INSTANCE_NAME     INSTANCE_STA DBNAME     DB_ROLE          SWITCHOVER_STATUS   PROTECTION_MODE
---------------- ------------ --------- ---------------- ------------------ --------------------
MIN               OPEN         MIN       PRIMARY          TO STANDBY         MAXIMUM PERFORMANCE
```

```
SQL> select dest_id id,database_mode db_mode,recovery_mode,
       protection_mode,standby_logfile_count "SRLs",
       standby_logfile_active ACTIVE,
       archived_seq#
       from v$archive_dest_status;
```

==> 2번째로 설정한 곳에 mounted_standby 이어야 하고 MANAGED MODE 이어야 한다.

```
 ID DB_MODE              RECOVER PROTECTION_MODE        SRLs ACTIVE ARCHIVED_SEQ#
--- -------------------- ------- -------------------- ---- ------ -------------
  1 OPEN                 IDLE    MAXIMUM PERFORMANCE     0      0            99
  2 MOUNTED-STANDBY      MANAGED RESYNCHRONIZATION       3      0            96
  3 OPEN                 IDLE    MAXIMUM PERFORMANCE     0      0             0
```

```
SQL> select process, status from v$managed_standby;
```

==> 우리는 파라미터에서 LGWR 프로세스가 standby DB로 전송하도록 하였다.

```
PROCESS STATUS
------- ------------
ARCH    CLOSING
ARCH    CLOSING
LGWR    WRITING
```

```
SQL> select dest_id "ID",destination,status,target,
      schedule,process,mountid mid
      from v$archive_dest order by dest_id;
```

=> destination 2번에 우리는 service=STBY로 설정하였다. STBY는
tnsnames.ora 에 나오는 접속이름이었다. STATUS=VALID 상태이고 STANDBY 이어야 한다.

```
ID DESTINATION                          STATUS    TARGET    SCHEDULE PROCESS MID
--- ---------------------------------- --------- ------- -------- ------- ---
  1 /data1/oradata/MIN/archive1        VALID     PRIMARY ACTIVE   ARCH        0
  2 STBY                               VALID     STANDBY ACTIVE   LGWR        0
  3                                    INACTIVE  PRIMARY INACTIVE ARCH        0
```

```
SQL> select dest_id,status,error from v$archive_dest;
```

=> archive dest 가 유효해야 한다. 1번 2번 destination 모두 valid상태이어야 한다.

```
 DEST_ID STATUS    ERROR
-------- --------- ----------------------------------------------------------
       1 VALID
       2 VALID
       3 INACTIVE
```

```
SQL> select message, timestamp
     from v$dataguard_status
     where severity in ('Error','Fatal')
     order by timestamp;
```

=> 아무런 에러도 나와서는 안된다. 여기에서 에러가 났다면 primary DB를 먼저 기동하고 standby DB를 기동했을 경우 발생할 수도 있으나, standby로 전송이 안된 것일 수도 있으므로 다른 것도 확인을 해보아야 한다. 만약 standby DB를 먼저 기동하고 recovery managed mode에서 MRP 프로세스를 띄우고 그리고 나서야 primary DB를 기동시켰다면 아래에서는 아무런 메세지도 나와서는 안된다. 예제에서는 primary DB를 먼저 기동했으므로 메세지가 발생했을 것이다.

```
MESSAGE                        TIMESTAMP
-------------------------------
-------------------------------
```

```
SQL> select dest_id id,database_mode db_mode,recovery_mode,
     protection_mode,standby_logfile_count "SRLs",
     standby_logfile_active ACTIVE,
     archived_seq#
     from v$archive_dest_status;
```

==> db_mode가 MOUNTED_STANDBY 이어야 한다. recovery_mode 가 managed가 되어 있어야 primary DB에서 전송된 redo log정보를 standby DB에 적용시킬 수 있다.

```
 ID DB_MODE           RECOVER PROTECTION_MODE        SRLs ACTIVE ARCHIVED_SEQ#
--- ---------------- ------- -------------------- ---- ------ -------------
  1 OPEN             IDLE    MAXIMUM PERFORMANCE     0      0            45
  2 MOUNTED-STANDBY  MANAGED MAXIMUM AVAILABILITY    2      0            45
  3 OPEN             IDLE    MAXIMUM PERFORMANCE     0      0             0
```

(13) standby DB 점검

```
SQL> select i.instance_name, i.status instance_status, d.name dbname, d.database_role db_role,
d.switchover_status switchover_status , d.protection_mode
from v$database d, v$instance i;
```

=> STANDBY 이어야 한다.

```
INSTANCE_NAME    INSTANCE_STA DBNAME    DB_ROLE      SWITCHOVER_STATUS   PROTECTION_MODE
---------------- ------------ --------- ---------------- ----------------- --------------------
STBY     MOUNTED      MIN      PHYSICAL STANDBY    NOT ALLOWED         MAXIMUM PERFORMANCE
```

```
SQL> select process, status from v$managed_standby;
```

=> 꼭 MRP 프로세스가 띄워져 있어야 한다.

```
PROCESS STATUS
------- ------------
ARCH    CONNECTED
```

```
ARCH      CONNECTED
ARCH      CONNECTED
MRP0      WAIT_FOR_LOG
RFS       RECEIVING
RFS       ATTACHED
```

(14) primary DB 테이블스페이스 생성 및 데이터 입력

<MIN DB primary DB>

test 테이블스페이스를 만들고 테이블을 만들고 데이터를 넣어본다.

주의할 것은 db_file_name_convert 에서 나오는 것처럼 /data1/oradata/MIN 안에만 생성을 해야 한다. 그래야 standby DB에 데이터파일이 자동으로 생기게 된다.

또한 파라미터에서 standby_file_management=auto로 되어 있어야 standby DB에 테이블스페이스의 데이터파일이 생긴다.

```
SQL> create tablespace test
     datafile '/data1/oradata/MIN/test01.dbf' size 10M;
SQL> select tablespace_name, file_name, bytes/1024/1024 mega from dba_data_files;
TABLESPACE FILE_NAME                                        MEGA
---------- ---------------------------------------- ----------
SYSTEM       /data1/oradata/MIN/system01.dbf                250
UNDOTBS      /data1/oradata/MIN/undotbs.dbf                 100
USERS        /data1/oradata/MIN/users01.dbf                 100
TEST         /data1/oradata/MIN/test01.dbf                  10

SQL> create table test
      (id number(10),name varchar2(30)) tablespace users;
SQL> insert into test
      values (1,'min1');
SQL> commit;
```

- archive를 적용시킨다.

```
SQL> alter system archive log current;
SQL> alter system archive log current;
```

(15) standby DB에 데이터 입력 여부 확인

<STBY DB standby DB>

- primary DB에서 만든 테이블스페이스가 있는지 데이터는 들어갔는지 확인한다.

recovery managed mode를 해제하고 read only로 open한다.

```
SQL> recover managed standby database cancel;
SQL> alter database open read only;
```

==> 정상적으로 primary DB에서 만든 테이블스페이스가 적용되었으며, test 라는 테이블에 데이터 insert가 정상적으로 된 것을 확인할 수 있다.

```
SQL>  select tablespace_name, file_name, bytes/1024/1024 mega from dba_data_files;
TABLESPACE FILE_NAME                                        MEGA
---------- ---------------------------------------- ----------
SYSTEM      /data1/oradata/STBY/system01.dbf              250
UNDOTBS     /data1/oradata/STBY/undotbs.dbf               100
USERS       /data1/oradata/STBY/users01.dbf               100
TEST        /data1/oradata/STBY/test01.dbf                10
SQL> select * from test;
        ID NAME
---------- ------------------------------
         1 min1
```

- 다시 recovery managed mode로 만들어서 primary 에서 전송된 redo log 정보가 standby DB에 적용되도록 한다. fal_server, fal_client 파라미터가 설정되어 있으므로 자동으로 gap이 생긴 부분을 맞추어 준다. DB를 open 상태에서도 recovery managed mode로 변경이 가능하다.

```
SQL> recover managed standby database disconnect;
```

(16) takeover 하기

시스템 문제가 발생하였다. takeover를 수행한다.

① <MIN DB primary DB>

가장 먼저 primary DB에서 standby로 변경을 한 후 standby DB를 standby DB로 변경 해야한다. 왜냐하면 standby DB를 primary로 변경하면 hang 상태로 primary DB가 standby DB가 될 때까지 기다리게 된다.

primary DB를 shutdown 한다.

```
SQL> alter database commit to switchover to physical standby with session shutdown wait;
SQL> shutdown immediate
```

② <STBY DB standby DB>

standby DB를 primary DB로 바꾸고 shutdown 한다.

유저접속은 없으므로 with session shutdown 절은 안 들어가도 된다.

```
SQL> alter database commit to switchover to primary;
SQL> shutdown immediate
```

③ 파라미터를 변경한다. # 을 붙인 것을 빼거나 추가해서 설정해준다.

실 환경에서는 파라미터를 따로 만들어서 스크립트로 수행하는 것도 좋겠다.

<MIN DB primary DB>

```
vi $ORACLE_HOME/dbs/initMIN.ora
<변경전>
#####standby DB이면 열고, primary DB이면 닫는다.
#fal_server=STBY
#fal_client=MIN
#lock_name_space=STBY
```

```
##primary DB이면 열고, standby DB이면 닫는다.
log_archive_dest_2='SERVICE=STBY LGWR SYNC'

<변경후>
#####standby DB이면 열고, primary DB이면 닫는다.
fal_server=STBY
fal_client=MIN
lock_name_space=STBY

##primary DB이면 열고, standby DB이면 닫는다.
#log_archive_dest_2='SERVICE=STBY LGWR SYNC'
```

<STBY DB standby DB>

```
vi $ORACLE_HOME/dbs/initSTBY.ora
<변경전>
##standby 이면 열고, primary DB이면 닫는다.
fal_server=MIN
fal_client=STBY
lock_name_space=STBY

##primary DB이면 열고, standby DB이면 닫는다.
#log_archive_dest_2='SERVICE=MIN LGWR SYNC'

<변경후>
##standby 이면 열고, primary DB이면 닫는다.
#fal_server=MIN
#fal_client=STBY
#lock_name_space=STBY
##primary DB이면 열고, standby DB이면 닫는다.
log_archive_dest_2='SERVICE=MIN LGWR SYNC'
```

④ < MIN DB , new standby DB >

- 새롭게 standby DB가 된 MIN DB를 recovery managed mode로 변경한다.

```
SQL> startup nomount
SQL> alter database mount standby database;
SQL> recover managed standby database disconnect;
```

```
SQL> select process, status from v$managed_standby;
```

==> MRP 프로세스 기동 확인

```
PROCESS STATUS
------- ------------
ARCH    CONNECTED
ARCH    CONNECTED
MRP0    WAIT_FOR_LOG
RFS     WRITING
RFS     ATTACHED
```

⑤ < STBY DB, new primary DB >

- new primary DB를 기동한다.

```
SQL> startup
```

- 확인하기

```
SQL> select i.instance_name, i.status instance_status, d.name dbname, d.database_role db_role,
          d.switchover_status switchover_status , d.protection_mode
             from v$database d, v$instance i;
```

=> 중요한 점검포인트 이다. TO_STANDBY 인지 확인한다.

```
INSTANCE_NAME    INSTANCE_STA DBNAME    DB_ROLE          SWITCHOVER_STATUS   PROTECTION_MODE
---------------- ------------ --------- ---------------- ------------------- --------------------
STBY             OPEN         MIN       PRIMARY          TO STANDBY          MAXIMUM PERFORMANCE
```

```
SQL>  select dest_id id,database_mode db_mode,recovery_mode,
          protection_mode,standby_logfile_count "SRLs",
          standby_logfile_active ACTIVE,
          archived_seq#
          from v$archive_dest_status;
 ID DB_MODE          RECOVER PROTECTION_MODE        SRLs ACTIVE ARCHIVED_SEQ#
--- --------------- ------- -------------------- ---- ------ -------------
  1 OPEN            IDLE    MAXIMUM PERFORMANCE     0      0           109
  2 MOUNTED-STANDBY MANAGED MAXIMUM AVAILABILITY    3      0           109
  3 OPEN            IDLE    MAXIMUM PERFORMANCE     0      0             0
```

(17) 서비스 원복(takeover)

다시 원복을 시킨다. STBY DB를 standby DB로 변경한 후 MIN DB를 primary DB로 변경한다.

<STBY DB new primary DB>

- primary DB를 standby DB로 만들고 나서 standby DB를 physical standby 만든다.
  순서를 잊지 말자.

```
SQL> alter database commit to switchover to physical standby with session shutdown wait;
SQL> shutdown immediate
```

<MIN DB new standby DB>

- MIN DB를 primary DB로 변경한다.

SQL> alter database commit to switchover to primary ;

SQL> shutdown immediate

<MIN DB primary DB>

- 초기화 파라미터를 변경한다.

```
vi $ORACLE_HOME/dbs/initMIN.ora
<원복시킨다>
#####standby DB이면 열고, primary DB이면 닫는다.
#fal_server=STBY
#fal_client=MIN
#lock_name_space=STBY

##primary DB이면 열고, standby DB이면 닫는다.
log_archive_dest_2='SERVICE=STBY LGWR SYNC'
```

<STBY DB standby DB>

```
vi $ORACLE_HOME/dbs/initSTBY.ora
<원복시킨다>
##standby 이면 열고, primary DB이면 닫는다.
fal_server=MIN
fal_client=STBY
lock_name_space=STBY

##primary DB이면 열고, standby DB이면 닫는다.
#log_archive_dest_2='SERVICE=MIN LGWR SYNC'
```

<STBY DB standby DB>

- standby DB를 먼저 기동시킨다.

SQL> startup nomount

SQL> alter database mount standby database;

SQL> recover managed standby database disconnect;

<MIN DB primary DB>

- primary DB를 기동시킨다.

SQL> startup

(18) failover 하기

primary DB가 디스크 이상으로 인해서 DB 데이터파일이 손상되었다. standby DB를 긴급하게 기동시켜야 한다. MIN DB를 shutdown abort로 Down시키고, standby DB를 primary DB로 기동시킨다.

Failover을 한 후에 시스템을 복구해서 MIN DB를 primary DB, STBY DB를 standby DB로

원래대로 구성하려면 Dataguard를 재구성 해야 한다. 즉, failover을 했다면 failover이후에 New primary DB인 STBY DB를 통해서 MIN DB를 standby DB로 구성하고 takeover 시키면 되겠다.

<MIN DB primary DB>

SQL> shutdown abort

<STBY DB standby DB>

- recovery managed mode를 해제가 아닌 끝내도록 한다. primary DB로 변경한다.

SQL> **recover managed standby database finish;**

SQL> **alter database commit to switchover to primary;**

- 파라미터를 변경 후 startup 시킨다.

<주의> 만약 New standby DB인 STBY DB까 failover가 정상적으로 수행되지 않으면서 recovery 하라고 나온다면? 이럴 경우엔 아래와 같은 명령을 쓰도록 한다.
standby redo log가 gap이 많이 있는데 standby DB에 적용하지 못한 경우에 발생할 수 있다.
이 때에는 skip하는 만큼의 gap을 DB에 적용하지 못할 수 있다.

SQL> **alter database recover managed standby database finish skip wait;**

SQL> **alter database commit to switchover to primary;**

$ORACLE_HOME/dbs/initSTBY.ora 파일에서 primary DB로 파라미터를 설정하고 기동한다.

SQL> startup

---------- 끝 ----------