리눅스에서 도전하는 오라클9i 3

데이터 가드를 이용한 스탠바이 데이터베이스 구성

류명환 ryu2811@netian.com

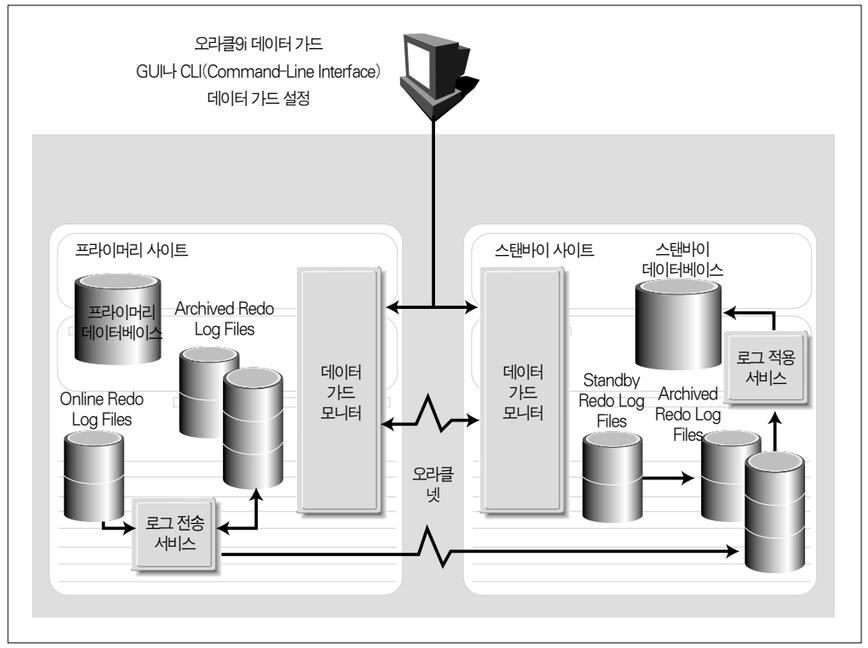
오라클을 처음 배울 때 스탠바이 데이터베이스(Standby Database)를 보면서 느낀 점이 "참, 별걸 다 하는구나" 였다. 그러나 실무를 수행하면서 백업의 중요성을 절실히 느껴왔고 백업을 게을리 하여 데이터를 잃어버리는 경우를 많이 보았다. "복구에 실패한 DBA는 용서 받을 수 있어도 백업에 실패한 DBA는 용서 받을 수 없다"는 말이 있다. 우스개 소리같은 이 말은 백업이 그만큼 중요하다는 뜻이다. 모른다면 지금부터 공부하면 된다. 그래서 독자들이 백업을 게을리 하여 DBA의 책임을 져버리지 않기 바란다.

세계적으로 인터넷이 활성화하면서 e비즈니스란 말을 자주 들어왔을 것이다. 이 e비즈니스에서 가장 강조되는 것이 '24시간! 365일! 무정지!' 시스템이다. 인터넷을 통해 전 세계를 대상으로 서비스가 확대되면서 언제 어디서 접속할지 모르는 상황에서 절대로 다운되지 않고 멈추지 않는 시스템이 필요하게 된 것이다. 이것은 단순히 잘 만들어진 시스템만으로는 실현할 수 없다. 예기치 못한 관리자의 실수나 인간의 힘으로 어쩔 수 없는 자연재해에 대비해 언제든지 어떤 상황에서나 다운 타임을 최소화해 최대한 빨리 시스템을 복구할 수 있어야 한다. 그리하여 백업이 필요하게 된 것이다. 다운 타임을 최소화하고 최신 데이터를 유지, 보호하는 것이 바로 백업의 목적이다.

스탠바이 데이터베이스란 무엇인가

이제 스탠바이 데이터베이스의 개념을 살펴 보자. **〈그림 1〉**을 보면 프라이머리 데이터베이스와 스탠바이 데이터베이스가 있다. 먼저 프라이머리 데이터베이스는 현재 서비스 중인 데이터베이스이고, 스탠바이 데이터베이스는 만약을 대비한 백업 시스템이다. 처음 스탠바이 데이터베이스를 생성할 때 프라이머리 데이터베이스의 모든 데이터 · 제어 · 매개변수 파일을 스탠바이 데이터베이스로 복사한다. 그리고 프라이머리 데이터베이스에서 나오는 로그들을 적용하면 스탠바이 데이터베이스의 내용은 프라이머리 데이터베이스의 내용과 가장 가까운 데이터를 유지한다는 것이 이 시스템의 개요다. 이때 프라이머리 데이터베이스와 스탠바이 데이터베이스는 오라클 넷(Oracle Net)으로 연결된다. 여기서 유념해 볼 것이 있다. 양쪽 시스템 모두 오라클 데이터 가드 브로커(Oracle Data Guard Broker)가 있다는 것이다.

〈그림 1〉 오라클9i 스탠바이 데이터베이스



데이터 가드 브로커

오라클 데이터 가드 브로커는 분산 관리 프레임워크(Distributed Management Framework)다. 이것은 데이터 가드 설정(Data Guard Configuration)을 생성 · 관리 · 모니터링하는 것을 자동화해 간편하게 만든다. 다음은 데이터 가드 브로커가 하는 일 중 몇 가지 예다.

- 로컬과 원격 스탠바이 데이터베이스 생성(Creation of local and remote standby databases)
- 로그 전송 서비스 설정, 제어, 모니터링(Configuration, control, and monitoring of log transport services)
- 로그 적용 서비스 설정, 제어, 모니터링(Configuration, control, and monitoring of log apply services)

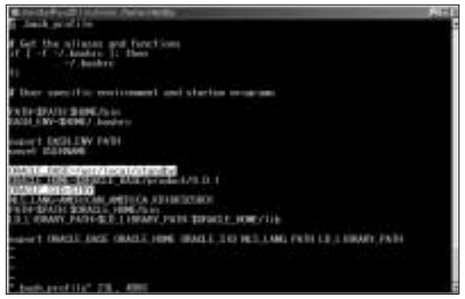
데이터 가드 브로커는 GUI 환경인 오라클9i 데이터 가드 매니저(Oracle9i Data Guard Manager)와 CLI 환경인 DGMGRL(Data Guard Command-Line Interface)을 제공한다. GUI 환경에서 사용자는 마법사(Wizard)를 통해 모든 작업을 할 수 있고, CLI 환경에서는 스탠바이 데이터베이스를 생성하는 것을 제외한 모든 작업을 할 수 있다. 다음은 데이터 가드 브로커의 구성 요소다.

- 오라클9i 데이터 가드 매니저
- 데이터 가드 커맨드 라인 인터페이스(DGMGRL)
- 데이터 가드 모니터

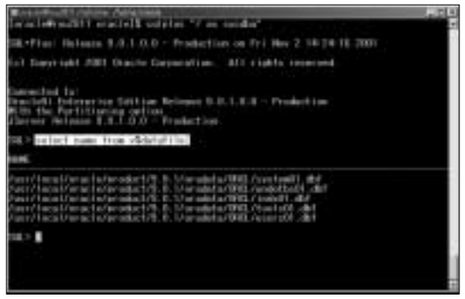
앞에서 언급한 첫 번째와 두 번째 것은 클라이언트 쪽 관리 툴이고 세 번째 것은 모니터링을 위해 서버 쪽에서 실행되는 것이다. 정확히 서버 쪽에는 모니터링을 위한 DMON 프로세스와 설정 파일이 존재한다. DMON 프로세스는 오라클의 백그라운드 프로세스로서 데이터 가드 브로커에 의해 관리되는 모든 사이트에서 실행된다. 사용자가 데이터 가드 매니저나 DGMGRL을 실행하면 DMON 프로세스와 통신하면서 작업하는 것이다.

〈표 1〉에 데이터 가드 브로커를 사용한 경우와 사용하지 않은 경우의 차이를 구분해 놓았다. 거의 대부분의 작업을 쉽고 간편하게 자동화하는 것을 알 수 있다. 하지만 일부 개발자나 DBA는 이러한 자동화 도구를 사용하지 않는다. 필자도 그런 부류의 사람 중 하나다. 필자는 개인적으로 RAD를 매우 싫어하고(비록 그 생산성만큼은 인정하지만), 모든 것이 편집기 하나로 가능해야 한다는 것이 지론이다. 만약 독자들이 취직한 회사에 오라클 엔터프라이즈 매니저(Oracle Enterprise Manager)가 없다면 "저는 엔터프라이즈 매니저로만 관리합니

〈화면 1〉 standby 계정을 위한 환경변수



〈화면 2〉 프라이머리 데이터베이스의 데이터 파일



다. 그것이 없다면 아무것도 할 수 없습니다. 그걸 구입해 주십시오"라고 말할 것인가? 어디까지나 이러한 도구들은 하나의 도구일 뿐 그 이상도 이하도 아니라는 것을 독자들도 명심하길 바란다. 도구란 것은 모든 것을 다 아는 사용자가 편의를 위해 사용하는 것이라 필자는 믿는다.

이제 오라클 스탠바이 데이터베이스의 개념을 살펴봤으니 실제로 한번 구성해 보자. 이번에 스탠바이 데이터베이스를 구성하는 것은 어디까지나 학습(연습)용으로서 로컬 머신에 두 개의 오라클9i를 설치해 구성하겠다. 하지만 실제로 필드에서 뛰는 사람이라면 당연히 리모트 머신에 구성하라. 원격 장비에서 구성하는 것이 스탠바이 데이터베이스의 개념에 맞다. 그러나 오라클에서 스탠바이 데이터베이스만큼은 리모트 머신에 구성하는 것이 더 쉽다(쉽다기보다는 몇 가지 귀찮은 일이 사라진다).

먼저 데이터 가드 브로커를 사용하기 위한 조건을 살펴보자.

- 프라이머리 데이터베이스와 스탠바이 데이터베이스는 반드시 오라클 9i여야 한다.
- 싱글 인스턴스 환경에서 실행되어야 한다. 리얼 애플리케이션 클러스터는 안된다.
- 데이터베이스 서버는 반드시 엔터프라이즈 에디션이어야 한다.
- 초기화 매개변수 DRS_START는 TRUE이어야 한다.
- 프라이머리 데이터베이스와 스탠바이 데이터베이스는 오라클 넷이 구성되어야 한다.
- 프라이머리 데이터베이스는 반드시 Archivelog Mode이어야 한다.

〈표 1〉 데이터 가드 브로커 사용시 특징

	브로커가 있을 경우	브로커가 없을 경우
General	단일화한 설정 적용	분리된 설정 적용
스탠바이 데이터베이스 생성	데이터 가드 매니저 마법사를 통해 쉽고 간편하게 생성할 수 있다.	데이터베이스 · 제어 · 초기화 매개변수 파일을 관리자가 직접 생성하고 복사한다.
설정	설정과 관리를 자동으로 단일화한다.	로그 전송 서비스와 로그 적용 서비스를 관리자가 설정해야 한다. 각각의 데이터베이스를 따로 관리한다.
제어	마우스 조작만으로 모든 것을 할 수 있다. 로그 전송과 로그 적용을 자동으로 수행한다. 데이터베이스의 상태를 간단히 보고 수정할 수 있다. 데이터베이스의 상태와 설정을 단일화할 수 있다.	데이터베이스 상태를 보고 관리하기 위해 SQL*Plus를 사용해야 한다. 각 서버마다 같은 작업을 반복해야 한다.
모니터링	연속적인 데이터베이스의 상태와 매개변수를 볼 수 있다. 경고 로그와 데이터 가드 설정 로그를 통합한 데이터베이스의 상태를 볼 수 있다.	각각의 데이터베이스 상태와 매개변수를 보기 위해 Fixed View들을 사용한다.

대략 이렇다. 하지만 우리는 이런 자동화 도구를 사용하기 보단 앞서 언급했듯이 스탠바이 데이터베이스 구성을 수동으로 해보겠다.

스탠바이 데이터베이스 구성

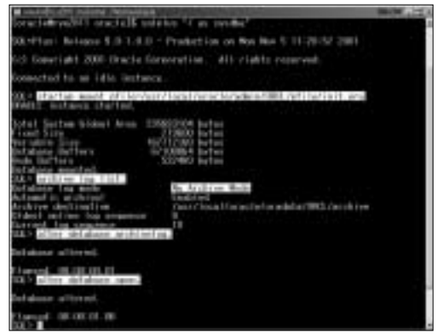
〈표 2〉는 스탠바이 데이터베이스를 구성하기 위한 작업 목록이다. 이제 하나하나 과정을 진행해 보자.

일단 스탠바이 데이터베이스를 위한 새로운 오라클을 설치한다. 새로운 계정을 만들고(필자는 standby라는 계정을 만들었다) 오라클을 소프트웨어만 설치한다. 데이터베이스는 생성하지 않는다. 〈화면 1〉은 standby 계정을 위한 .bash_profile이다. oracle 계정의 환경과 비슷하다.

첫 번째 과정은 프라이머리 데이터베이스의 데이터 파일들을 복사하는 것이다. 앞에서도 말했듯이 최소한 현재 프라이머리 데이터베이스를 그대로 복사하고 그 다음부터 추가적인 아카이브 로그 파일을 받아 적용해 프라이머리 데이터베이스와 최대한 비슷한 내용을 유지하는 것이다. 그러기 위해서는 프라이머리 데이터베이스의 현재 데이터

〈화면 3〉 프라이머리 데이터베이스의 초기화 매개변수 파일

〈화면 4〉 로그 모드 변환



파일들을 복사한다. 반드시 지켜야 하는 것은 아니지만 프라이머리 데이터베이스를 종료(Shutdown)하고 난 후 복사하도록 권유한다. 〈화면 2〉를 보면 현재 프라이머리 데이터베이스에 어떤 파일들이 있는지 알 수 있다. 이 파일들을 확인하고 데이터베이스를 종료한 후 데이터 파일들을 복사한다.

다음 과정은 현재 프라이머리 데이터베이스가 아카이브 로그 모드(Archive Log Mode)인지 검사하는 것이다. 만약 현재 데이터베이스가 아카이브 로그 모드가 아니라면 초기화 매개변수 파일을 수정하고 데이터베이스의 로그 모드를 변환한다. 〈화면 3〉은 초기화 매개변수 파일의 일부분과 로그 모드를 수정한 부분이 있다.

만약 〈화면 3〉과 비슷한 부분이 있다면 별도로 수정할 필요는 없다. 없다면 추가한다. log_archive_start는 로그 파일을 아카이빙하는 백그라운드 프로세스를 데이터베이스 가동시에 시작할 것인지를 결정하는 매개변수다. log_archive_dest_1은 아카이빙한 로그 파일들을 저장할 디렉토리를 말한다. log_archive_dest_state_1은 이름에서 짐작할 수 있듯이 저장될 곳의 상태를 말한다. log_archive_format은 아카이빙된 로그 파일의 이름을 결정하는 매개변수다. 이것은 프라이머리 데이터베이스의 아카이브 로그의 위치와 상태를 지정한 것이다. 스탠바이 데이터베이스를 위해 log_archive_dest_2를 추가한다. 매개변수 값은 'SERVICE=STBY LGWR'이다. SERVICE는 아카이브할 대상이 STBY라는 오라클 네트워크 서비스를 이용해 전송하라는 뜻이고 LGWR은 온라인 리두로그를 스탠바이 데이터베이스에 전송하라는 뜻이다. log_archive_dest_state_2의 값은 당연히 'Enable'로 설정한다. 초기화 매개변수 파일을 수정했으면 데이터베이스의 로그 모드를 변환한다.

〈화면 4〉에서 보듯 로그 모드를 변환하기 위해 스타트업(Startup) 과정 중에서 마운트까지만 수행했다. 참고로 오라클에는 종료, 노마운트(NoMount), 마운트, 열기 네 가지 상태가 있고 오라클이 가동될 때 네 가지 단계를 거치게 된다(노마운트나 마운트까지만 진행해 어떤 작업을 하는 경우가 종종 있다. 이에 대해 자세히 알고 싶은 독자는 OTN에 가보면 매뉴얼이 있다. 그중에서 "Oracle9i Administrator's Guide"를 참고한다). 마운트 후 현재 로그 모드를 확인하고 만약 위와 같이 노 아카이브 모드(No Archive Mode)로 나온다면 아카이브 로그 모드로 변환한다. 성공했다면 데이터베이스를 열기 상태로 변환해 사용할 수 있게 만든다. 만약 이미 아카이브 로그 모드라면 log_archive_ dest를 찾아서 그 안에 있는 Archived Log File도 스탠바이 데이터베이스의 standby_archive_ dest로 복사하는 것이 좋다.

프라이머리 데이터베이스를 열어서 이번에는 스탠바이 데이터베이스에서 사용할 제어 파일을

〈표 2〉 스탠바이 데이터베이스 구성

단계	작업
1	스탠바이 데이터베이스를 위한 오라클을 설치한다. 이때 데이터베이스는 생성하지 않는다. 과정 중 선택사항에서 'Software Only'를 선택하면 된다.
2	프라이머리 데이터베이스에서 데이터 파일들을 백업받는다.
3	프라이머리 데이터베이스의 초기화 매개변수들을 수정한다.
4	프라이머리 데이터베이스를 ARCHIVELOG Mode가 아니라면 ARCHIVELOG Mode로 바꾼다.
5	프라이머리 데이터베이스에 접속해 스탠바이 데이터베이스에서 사용할 제어 파일을 생성한다.
6	백업받은 데이터 파일들과 새로 생성한 제어 파일들을 스탠바이 데이터베이스로 옮긴다.
7	스탠바이 데이터베이스에서 사용할 초기화 매개변수 파일을 생성하고 초기화 매개변수들을 수정한다.
8	스탠바이 데이터베이스를 시작한다. 이때 스탠바이 데이터베이스는 반드시 마운트까지만 실행한다.
9	필요한 경우 스탠바이 데이터베이스의 Redo Log File을 생성한다(선택사항).
10	필요하다면 데이터베이스 파일들과 Redo Log File들의 이름을 직접 변환한다(선택사항).
11	스탠바이 데이터베이스의 리스너(Listener)를 설정하고 수행한다. 필요하면 Oracle Net Configuration Assistant를 실행해 구성한다. 데이터 가드 브로커를 사용하길 원한다면 반드시 TCP/IP를 사용해야만 한다.
12	스탠바이 데이터베이스가 프라이머리 데이터베이스에 접근하기 위해 넷 서비스 이름(Net Service Name)을 생성한다. 필요하다면 Oracle Net Configuration Assistant를 실행해 구성한다.
13	프라이머리 데이터베이스가 스탠바이 데이터베이스에 접근하기 위해 넷 서비스 이름을 생성한다. 필요하다면 Oracle Net Configuration Assistant를 실행해 구성한다.

〈화면 5〉 제어 파일 생성

〈화면 6〉 스탠바이 데이터베이스 초기화 매개변수 파일

만들 차례다. 아주 간단히 수행할 수 있다. **〈화면 5〉**에서 보이는 대로 alter database 문장으로 제어 파일을 만든 후 따로 이전에 복사한 데이터 파일들과 같이 잘 보관한다.

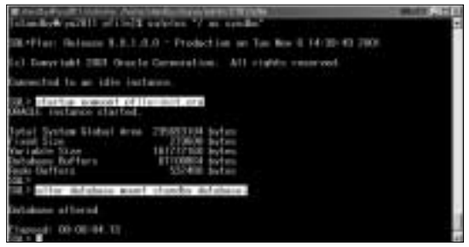
이제 프라이머리 데이터베이스를 정상으로 가동한다. 프라이머리 데이터베이스에서 할 수 있는 일은 끝났다. 지금까지 한 일은 스탠바이 데이터베이스를 위한 준비 작업이라 할 수 있다. 이제부턴 실제로 스탠바이 데이터베이스를 구성한다.

지금까지 준비한 파일들을 스탠바이 데이터베이스로 옮긴다. 매뉴얼을 살펴보면 기본적으로 온라인 리두(redo) 로그 파일은 필요 없지만 같이 복사하도록 권유한다. 이때 디렉토리 구조는 독자들이 알아서하면 되지만 웬만하면 프라이머리 데이터베이스와 비슷하게 하는 것이 좋다. 지금은 로컬 머신에 프라이머리 데이터베이스와 스탠바이 데이터베이스를 같이 실행하는 경우니 어쩔 수 없이 다를 것이다. 그러나 리모트 머신에 스탠바이 데이터베이스를 구성한다면 완전히 같은 환경에서 구성하는 것이 몇 가지 귀찮은 작업을 줄여 줄 수 있다. 필자는 〈SID〉 이름으로 생성되는 디렉토리와 $ORACLE_BASE 디렉토리를 제외하면 모두 같은 구조로 사용하려 노력했다.

스탠바이 데이터베이스에서 사용할 초기화 매개변수 파일을 생성한다. 특별히 생성하는 것이 아니라 프라이머리 데이터베이스에서 복사해서 수정하면 된다. **〈화면 6〉**을 보면 반드시 수정해야 할 항목과 추가해야 할 항목이 나와 있다.

〈화면 6〉에 보이듯 반드시 db_name은 프라이머리 데이터베이스와 같이 ORCL을 그대로 사용한다. 다른 dump_dest와 control_files의 위치, log_archive_dest 등은 약간만 변경

〈화면 7〉 스탠바이 데이터베이스 시작



해 자신의 디렉토리 구조에 맞게 설정하면 된다. 이 부분에서 중요한 것은 db_file_name_convert와 log_file_name_convert다. db_file_name_convert는 프라이머리 데이터베이스에서 스탠바이 데이터베이스로 데이터 파일을 이전해 오면서 어쩔 수 없이 변하는 디렉토리 위치를 바꾸어 주는 것이고, log_file_name_convert는 로그 파일에 대한 디렉토리 위치 변경을 지정하는 것이다. 특히 remote_archive_enable 값은 프라이머리 데이터베이스와 스탠바이 데이터베이스에서 반드시 TRUE이어야만 한다. 이 값은 스탠바이 데이터베이스가 프라이머리 데이터베이스의 리두 로그를 전송받는 것을 결정하는 값이다.

필요한 파일이 모두 준비됐다면 이제 스탠바이 데이터베이스를 시작한다. 여기서 주의할 것은 스탠바이 데이터베이스를 마운트까지만 올리는 것이다. 이것도 한번에 마운트하는 것이 아니라 종료에서 노마운트로, 노마운트에서 마운트로 진행한다(**〈화면 7〉**).

네트워크 설정과 스탠바이 데이터베이스 수행

이제 스탠바이 데이터베이스 준비도 끝났다. 남은 것은 양쪽의 네트워크를 맞추고 서로 통신이 가능하도록 하면 된다. 이 예제에서는 로컬 머신을 사용하므로 IPC를 사용할 수도 있으나 TCP/IP를 사용해 구성한다. IPC를 사용해 구성하는 것은 독자들의 과제로 남긴다. 매뉴얼을 조금만 찾아보면 아주 간단히 해결할 수 있다. 사실 IPC는 그렇게 많이 쓰이지 않는다.

두 개의 데이터베이스 서버는 양쪽 모두 서로를 바라볼 수 있어야 한다. 그러므로 둘 다 리스너를 띄우고 tnsnames.ora에 해당 데이터베이스를 위한 엔트리를 추가한다. 오라클 넷에 관해 약간의 지식이 있다면 파일을 직접 수정하고 잘 모르는 사용자는 netca나 netmgr(8.xx 대의 netasst)을 사용해 구성하면 된다. 오라클 넷에서는 기본적으로 리스너에서 1521 포트를 사용한다. 그러나 이 예제에는 두개의 데이터

〈화면 8〉 스탠바이 데이터베이스의 리스너 구성

〈화면 9〉 스탠바이 데이터베이스의 리스너 실행

베이스가 로컬 머신에서 수행되므로 반드시 스탠바이 데이터베이스에는 1526 포트를 사용한다. **〈화면 8〉**에 스탠바이 데이터베이스의 listener.ora의 내용이 있다.

각각의 계정에서 리스너를 실행한다. Oracle 계정에서 리스너 ORCL을 실행하고 Standby 계정에서 리스너 STBY를 실행한다. **〈화면 9〉**는 STBY를 실행한 것이다.

모두 실행한 후에 'ps ax'를 실행해 보면 두개의 리스너가 동작하고 있음을 확인할 수 있다. 이제 모든 과정이 끝났다. 스탠바이 데이터베이스에서 Log Apply Services를 수행하기 위해 다음을 실행한다.

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE;
```

백그라운드로 실행하고 싶다면 다음과 같이 한다.

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE
DISCONNECT FROM SESSION;
```

이 경우 현재 연결하고 있는 세션이 종료하면 Recovery를 수행하게 된다. Recovery 수행을 중지하고 싶다면 따로 다음 명령을 실행한다.

〈화면 10〉 아카이브 로그 전송 확인

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCAL
[IMMEDIATE | NOWAIT ];
```

두 가지 방법 중에서 선택은 어디까지나 독자들의 몫이다.

이제 프라이머리 데이터베이스에서 약간의 작업을 하고 다음 명령을 수행하면 현재 로그가 다 쓰이지 않았더라도 스위칭이 일어난다.

```
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
```

이 명령은 현재 로그를 강제로 스위칭함으로써 아카이브 로그 파일을 생성한다. 스위칭이 일어나면서 프라이머리 데이터베이스에서 스탠바이 데이터베이스로 아카이브 로그가 전송된 것이 보일 것이다(**〈화면 10〉**).

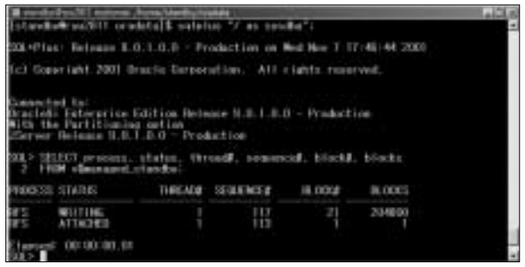
이로써 스탠바이 데이터베이스 구성이 끝났다. 이 시스템은 프라이머리 데이터베이스와 최대한 가까운 값을 유지할 것이다. 만약 프라이머리 데이터베이스에 문제가 발생하면 이 스탠바이 데이터베이스는 훌륭히 프라이머리 데이터베이스를 대체할 수 있을 것이다.

관리자는 항상 프라이머리 데이터베이스와 스탠바이 데이터베이스가 제대로 동작하는지 확인해야만 한다. 이때 활용할 수 있는 간단한 명령 하나가 있다.

```
SQL> SELECT GROUP#, THREAD#, SEQUENCE#, BYTES, ARCHIVED,
STATUS
2> FROM V$LOG;
```

이 명령을 수행하면 현재 로그 파일의 상황을 알 수 있다. 이 값은 프라이머리 데이터베이스의 로그 시퀀스(Log Sequence)

〈화면 11〉 온라인 리두 로그 전송 확인



가 스탠바이 데이터베이스보다 항상 1이 많아야 하므로 한번씩 살펴보라. 프라이머리 데이터베이스에서 강제로 로그 스위칭을 몇 번 일으킨 후 프라이머리 데이터베이스와 스탠바이 데이터베이스에서 각각 조사해보면 두 값이 계속 증가함을 보여준다. 이것은 아카이브 로그가 자동으로 스탠바이 데이터베이스에 전송되고 적용됐다는 것을 말한다.

```
SQL> SELECT PROCESS, STATUS, THREAD#, SEQUENCE#, BLOCK#,
BLOCKS
2> FROM V$MANAGED_STANDBY;
```

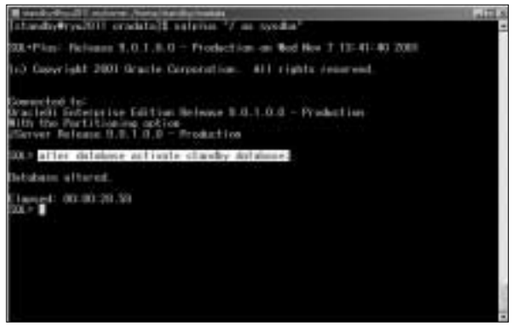
이 명령을 실행하면 현재 스탠바이 데이터베이스의 Current Log Sequence가 스탠바이 데이터베이스에서 쓰기 중임을 알 수 있다(〈**화면 11**〉). 시퀀스 117번은 현재 프라이머리 데이터베이스에서 Current Redo Log의 Sequence Number이다.

스탠바이 데이터베이스의 승격

이제 프라이머리 데이터베이스에 갑자기 문제가 발생했다고 가정하고 스탠바이 데이터베이스를 프라이머리 데이터베이스로 승격시키는 작업만 남았다. 이 작업은 스탠바이 데이터베이스를 한번 구성하고 나면 단 한번만 할 수 있다. 한번 프라이머리 데이터베이스로 승격된 스탠바이 데이터베이스는 다시 되돌릴 수 없다. 이때는 스탠바이 데이터베이스를 다시 구성해야 한다. 절대 실무에 있는 사람은 연습삼아 실행해 보는 일이 없기를 바란다. 〈**화면 12**〉는 스탠바이 데이터베이스를 프라이머리 데이터베이스로 승격한 것이다.

만약의 사태가 발생해 프라이머리 데이터베이스를 복구하는 데 시간이 오래 걸린다면 스탠바이 데이터베이스를 승격시켜 사용하라. 한 가지 당부할 것은 스탠바이 데이터베이스를 프라이머리 데이터베이스로 승격시키고 나면 반드시! 데이터베이스를 종료하고 전체 백업(Full Backup)을 따로 받

〈화면 12〉 스탠바이 데이터베이스의 승격



아두라고 강력히! 권유한다. 항상 말하는 것이지만 컴퓨터는 믿을 만한 것이 절대 아니다.

한 가지 팁을 소개하자면 현재 스탠바이 데이터베이스를 프라이머리 데이터베이스로 승격시키지 않고 읽기 전용 모드로 열 수 있다.

```
SQL> ALTER DATABASE OPEN READ ONLY;
```

역시 자세한 사항을 보려면 매뉴얼을 참고하라.

마치며

지금까지 오라클 스탠바이 데이터베이스를 살펴봤다. 데이터가드 매니저를 사용하는 것이나 좀더 많은 내용을 알고 싶은 독자는 매뉴얼을 참고하라. 필자의 생각으로 오라클 관련 책자는 서점에 가도 그다지 많이 나오지 않았고 몇 권 없는 것들마저도 그렇게 좋은 내용을 담고 있지는 않았다. 언제든지 모든 문제의 해답은 매뉴얼에 있으므로 매뉴얼을 찾는 것을 습관화하라고 충고한다.

정리 : 송우일 wooil@sbmedia.co.kr