아키텍처 개요

오라클은 이식이 매우 편하도록 설계된 데이터베이스다. 윈도우즈와 유닉스 그리고 메인프레임까지 다양한 플랫폼에서 적절하게 이용할 수 있다는 뜻이다. 그러나 오라클의 물리 구조는 서로 다른 운영체제에서는 다르게 보이기도 한다. 예를 들면 유닉스 OS에서 주요 기능을 담당하는 프로세스들은 가상으로 활동하며, 마치 OS의 다른 많은 프로세스들처럼 사용되고 있다. 유닉스에서는 이러한 구조가 멀티프로세스 기반 위에서 정확히 실행되는 반면, 윈도우즈 플랫폼에서 오라클은 마치 단일 프로세스처럼 실행되는데, 이를 '쓰레드(thread)' 프로세스라 한다. OS/390이나 z/OS의 OS 위에서 구동되는 IBM 메인프레임의 경우, 오라클은 OS에 특화된 구조로 되어 있어서 복수의 OS/390의 '주소 공간(address space)'을 사용하고, 단일 인스턴스에서 모두 실행되도록 구성되어 있다. 하나의 데이터베이스 인스턴스에 대해 255개의 주소 공간을 적용할 수 있다. 그뿐만 아니라 오라클은 OS/390의 Workload Manager(WLM)와 함께 OS/390 시스템에서 구동되는 모든 작업들과 다른 작업들에 대비하여 오라클의 특정 작업에 대해 실행우선권을 부여할 수 있다. 이러한 물리 구조가 오라클을 특정 플랫폼에서 다른 플랫폼으로 전환하는 도구로 사용함에 있어 이식이 편하도록 설계되었다 할 수 있으며, 오라클이 모든 플랫폼에서 어떻게 작동되는지 여러분이 손쉽게 이해할 수 있도록 충분히 일반화되었다고 할 수 있다.

이 장에서는 오라클의 이식이 편하도록 설계된 구조에 대해 설명할 것이다. 오라클 서버와 함께 '데이터베이스', '인스턴스'(항상 혼란을 가져오는 용어)와 같은 용어들에 대해서도 정의할 것이다. 오라클에 접속하고 서버가 메모리를 어떻게 관리하고 있는지도 살펴볼 것이다. 이어지는 세 개의 장에서는 오라클 아키텍처의 세 가지 주요 요소에 대해 상세하게 살펴볼 것이다.

- 3장은 파일에 대해 다룬다. 이 장에서는 데이터베이스를 구성하는 다섯 가지의 파일 세트, 즉 파라미터 파일, 데이터 파일, 임시(템프) 파일, 제어(컨트롤) 및 리두 로그 파일에 대해 알아본다. 오라클 10g 이상의 버전에서 플래시백복구(Flashback Recovery)라고 부르는 새로운 파일 영역에 대해 살펴볼 것이며, Automatic Storage Management(ASM)가 위에 열거한 파일 스토리지에 미치는 영향 등에 대해 이야기할 것이다.
- 4장은 System Global Area(SGA)와 Process Global Area(PGA)라고 부르는 오라클 메모리 구조를 다룬다. 두 구조 간의 관계를 예제를 통해 비교할 예정이며, SGA의 다양한 구성 요소인 shared

pool, large pool, Java pool 등에 대해서도 다루도록 하겠다.

- 5장은 오라클의 물리적 프로세스 또는 쓰레드에 대해 다룬다. 데이터베이스에서 구동하는 프로세스의 세 가지 서로 다른 유형(server process, background processes, slave process)에 대해 다루도록 하겠다.

필자는 이런 요소들을 이 책의 맨 앞에서 다루는 것에 대해 많은 고민을 했다. 프로세스는 SGA를 사용하고 있으므로 프로세스를 다루기 전에 SGA에 대해 이야기하는 것은 바른 순서가 아닌 듯하다. 반면, 프로세스가 어떤 역할을 하는지에 대해 이야기할 때 SGA를 거론할 필요가 있다. 이 두 개의 요소는 밀접한 관계가 있고, 파일은 파일과 관련된 각각의 프로세스에 의해 작동되는데, 프로세스가 무슨 일을 하는지에 대한 이해가 없는 상태에서 이야기하는 것 또한 의미가 없다고 할 수 있다.

그러므로 이 장에서는 몇몇 용어들에 대해 미리 정의하고, 오라클이 어떻게 생겼는지(만약 당신이 화이트보드에 이 구조를 그려놓았다면, 훨씬 잘 이해할 수 있으리라)에 대한 일반적인 개론을 설명하려 한다.

데이터베이스와 인스턴스

'인스턴스'와 '데이터베이스'라는 두 용어는 '오라클'이라는 단어를 이해하는 데 많은 혼란을 일으키곤 한다. 용어적인 측면에서 볼 때 두 용어의 정의는 아래와 같다.

- 데이터베이스: 물리적인 운영체제의 파일이나 디스크의 집합. 오라클의 Automatic Storage Management(ASM)나 RAW 파티션을 사용할 때 데이터베이스는 운영체제에 존재하는 개별적인 분리된 파일처럼 보이지는 않지만, 같은 뜻으로 여전히 정의하고 있다.
- 인스턴스: 오라클의 프로세스/쓰레드와 공유 메모리 영역을 합한 집합. 인스턴스는 하나의 컴퓨터에서 쓰레드/프로세스가 운영 중에 서로 공유하는 메모리 영역이다. 인스턴스는 휘발성을 가지고 영속성을 띄지 않는데, 이런 인스턴스의 메모리 영역들 중 몇몇은 디스크를 함께 사용하기도 한다. 데이터베이스 인스턴스는 어떠한 종류의 디스크 스토리지가 없이도 존재할 수 있다. 이런 이론들은 실생활에서는 그리 유용하지는 않지만, 이러한 고민은 인스턴스와 데이터베이스 간에 명확하게 선을 긋는 데 유용하다.

인스턴스와 데이터베이스 두 용어는 가끔은 서로 혼용하여 사용하기도 하지만, 두 용어는 완전히 다른 개념이다. 두 용어 간의 관계를 설명하면 다수의 인스턴스가 데이터베이스를 마운트하고 오픈하며, 하나의 인스턴스는 특정 시점에는 하나의 데이터베이스만 마운트하고 오픈한다. 인스턴스는 자신의 수명 동안 기껏해야 한 개의 데이터베이스를 마운트하고 오픈하는 것이라고 말하는 것이 옳을 것이다.

아직도 혼란스러운가? 다음 예제가 좀 더 이해하는 데 도움을 줄 것이다. 하나의 인스턴스는 운영체제

고유 프로세스의 단순한 집합이거나 다수의 쓰레드를 가지는 단일 프로세스와 몇몇 메모리의 집합을 뜻한다. 이 프로세스들은 데이터베이스에서 작동된다. 데이터베이스는 단순히 파일들(데이터 파일, 임시 파일, 리두 로그 파일, 컨트롤 파일)의 모음이다. 인스턴스는 항상 하나의 데이터베이스가 가지고 있는 하나의 파일 세트만 가진다. 그런데 하나의 데이터베이스는 오로지 하나의 인스턴스만 구동할 수 있다라는 식으로 반대로 알려져 있는 게 사실이다. 그러나 오라클 Real Application Clusters(RAC)와 같은 특별한 경우에는 클러스터 환경에서 다수의 컴퓨터가 작동할 수 있도록 된 오라클의 옵션으로 인해, 공유된 물리적 디스크의 세트로 존재하는 하나의 데이터베이스에서 마운트와 동시에 오픈되는 많은 인스턴스를 가질 수 있다. 이런 환경은 동시에 서로 다른 컴퓨터에서 단일 데이터베이스에 접근이 가능하도록 만들어준다. 오라클 RAC는 고가용성과 확장성을 제시해줄 수 있다.

간단한 예제를 통해 이야기해보도록 하겠다. 오라클 11g 버전 11.2.0.1을 리눅스 OS 환경의 서버에 설치할 때 오라클 소프트웨어만 설치하고 데이터베이스 구동에 필요한 프로그램은 설치하지 않았다.

pwd 명령어는 현재 작업 디렉터리를 보여주는데(이 예제는 리눅스 기반에서 실행하였다), dbs 디렉터리(윈도우즈에서는 데이터베이스 디렉터리)를 ls –l 명령어를 통해 확인해본 결과, 디렉터리가 비어 있음을 알 수 있다. init.ora 파일도 없고, SPFILES(파라미터 파일에 대해서는 3장에서 자세히 살펴보도록 하자)도 없다는 것을 확인할 수 있다.

```
[ora11gr2@dellpe dbs]$ pwd
/home/ora11gr2/dbs
[ora11gr2@dellpe dbs]$ ls -l
total 0
```

ps(process status) 명령어를 실행하면, ora11gr2 사용자(이 예에서는 오라클 소프트웨어 소유자가 되겠다)가 실행한 모든 프로세스를 확인할 수 있다. 이 명령어를 실행한 시점에는 어떠한 오라클 데이터베이스 프로세스도 발견할 수 없다.

```
[ora11gr2@dellpe dbs]$ ps -aef ¦ grep ora11gr2
ora11gr2  4447  4446  0 13:15 pts/1    00:00:00 -bash
ora11gr2  4498  4447  0 13:17 pts/1    00:00:00 ps -aef
ora11gr2  4499  4447  0 13:17 pts/1    00:00:00 grep ora11gr2
```

공유 메모리, 세마포어(semaphores) 등과 같은 내부 프로세스들 간의 커뮤니케이션 장치들을 조회할 수 있는 ipcs 명령어를 사용하면, 현재 이 시스템에서는 아무것도 없음을 확인할 수 있다.

```
[ora11gr2@dellpe dbs]$ ipcs -a
```

```
------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch     status

------ Semaphore Arrays --------
key        semid      owner      perms      nsems

------ Message Queues --------
key        msqid      owner      perms      used-bytes   messages
```

오라클의 '명령어 라인(command-line)' 인터페이스인 SQL*Plus를 통해 데이터베이스 안에서 모든 권한을 소유한 계정인 sysdba로 접속해보도록 하겠다. 초기에 미리 ORACLE_SID 변수와 같은 환경을 설정하지 않았다고 가정하면, 아래와 같은 메시지를 받을 것이다.

```
[ora11gr2@dellpe dbs]$ sqlplus / as sysdba

SQL*Plus: Release 11.2.0.1.0 Production on Fri Dec 11 14:07:14 2009

Copyright (c) 1982, 2009, Oracle. All rights reserved.

ERROR:
ORA-12162: TNS:net service name is incorrectly specified


Enter user-name:
```

이와 같은 오류 메시지는 데이터베이스 소프트웨어가 접속할 대상을 알 수 없을 때 발생한다. 만약 여러분이 오라클에 접속할 때는 '네트워크 커넥션(network connection)'을 위한 TNS '접속 문자열(connect string)'을 찾을 것이고, 오라클 환경변수에 설정된 경로에서 ORACLE_SID를 정의한 변수를 찾을 것이다(윈도우즈에서는 레지스트리(registry)에서 ORACLE_SID 변수를 찾는다). ORACLE_SID는 오라클의 'site identifier'이며, 인스턴스에 접근하기 위한 핵심 키다. ORACLE_SID를 설정하면 어떤 변화가 생길까?

```
[ora11gr2@dellpe dbs]$ export ORACLE_SID=ora11g
```

ORACLE_SID를 정의한 후 접속을 시도하면 접속에 성공하고, SQL*Plus가 idle 인스턴스에 접속하였다는 메시지를 확인할 수 있다.

```
[ora11gr2@dellpe dbs]$ sqlplus / as sysdba

SQL*Plus: Release 11.2.0.1.0 Production on Fri Dec 11 13:48:01 2009
```

```
Copyright (c) 1982, 2009, Oracle. All rights reserved.

Connected to an idle instance.

SQL>
```

'인스턴스'는 볼드체로 표시한 라인 밑에 따라오는 메시지들을 통해 오라클 서버 프로세스만 기동되고 있는 것을 확인할 수 있다. 공유 메모리가 아직 할당되지 않은 상태며, 다른 프로세스가 없음을 확인할 수 있다.

```
SQL> !ps -aef ¦ grep ora11gr2
ora11gr2  4447  4446  0 13:15 pts/1    00:00:00 -bash
ora11gr2  4668  4667  0 13:48 pts/2    00:00:00 sqlplus   as sysdba
ora11gr2  4669  4668  0 13:48 ?        00:00:00 oracleora11g ↩
 (DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=beq)))
ora11gr2  4678  4668  0 13:48 pts/2    00:00:00 /bin/bash -c ps -aef ¦ grep ora11gr2
ora11gr2  4679  4678  0 13:48 pts/2    00:00:00 ps -aef
ora11gr2  4680  4678  0 13:48 pts/2    00:00:00 grep ora11gr2

SQL> !ipcs -a

------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch     status

------ Semaphore Arrays --------
key        semid      owner      perms      nsems

------ Message Queues --------
key        msqid      owner      perms      used-bytes   messages
SQL>
```

| Note | 윈도우즈에서 오라클 프로세스들은 리눅스에서 확인한 것과 같이 각각 독립적으로 분리된 프로세스로 존재하지 않고, 쓰레드들을 포함한 단일 프로세스(역자 주_ Oracle.exe)로 나타난다. 또한 윈도우즈 쓰레드는 각각의 프로세스들의 이름을 바로 확인할 수도 없다. 여러분이 쉽게 개별 프로세스의 이름을 구분할 수 있도록 이 책에서는 리눅스 환경을 사용하도록 하겠다.

흥미로운 점은 ps의 결과물에서 프로세스의 이름들이 모두 oracle ora11g라는 것이다. 실행 가능 바이너리 파일은 $ORACLE_HOME/bin/oracle에 있는 바이너리 파일들인데, 아무리 열심히 시스템을 뒤져본다 하더라도 이름으로 실행할 수 있는 것을 찾아낼 수 없을 것이다.

| Note | ORACLE_HOME이라는 유닉스의 환경변수나 윈도우즈의 레지스트리 설정을 완료하고, 각각의 환경에 오라클 소프트웨어가 설치된 전체 경로를 표현하였다고 가정하겠다.

오라클 개발자들은 메모리에 적재할 때 프로세스의 이름을 간단하게 바꿀 수 있다. 지금 실행 중인 (dedicated server process: 다음에 프로세스를 다루는 장에서 설명하겠다) 단일 오라클 프로세스의 이름은 oracle$ORACLE_SID다. 이런 명명 규칙은 프로세스가 어떤 인스턴스에서 사용되는지 쉽게 알아낼 수 있게 한다. 이제 인스턴스를 구동시켜 보자.

```
SQL> startup
ORA-01078: failure in processing system parameters
LRM-00109: could not open parameter file '/home/ora11gr2/dbs/initora11gr2.ora'
SQL>
```

initora11gr2.ora라는 이름의 파일이 없어 발생한 오류라는 점에 주목하자. 우리가 평상시 init.ora 파일이라고 부르거나 좀 더 보편적으로는 **파라미터 파일**이라 부르는 이 파일은, 인스턴스를 기동하기 위해 반드시 필요한 유일한 파일이다. 또한 파라미터 파일(좀 더 상세하고 짧게 기술한 단순한 텍스트 파일) 혹은 내장 파라미터 파일이 필요하다.

그럼 이제 파라미터 파일을 생성해볼 텐데, 실제 데이터베이스 인스턴스(일반적으로는 데이터베이스 블록 크기, 컨트롤 파일의 위치 등과 같이 더 많은 파라미터를 담고 있다)를 구동하기 위한 최소한의 정보만 기록해 넣을 것이다. 기본적으로 이 파일들은 $ORACLE_HOME/dbs 디렉터리에 위치하고, init${ORACLE_SID}.ora 형식을 갖는다.

```
[ora11gr2@dellpe ~]$ cd $ORACLE_HOME/dbs
[ora11gr2@dellpe dbs]$ echo db_name=ora11g > initora11g.ora
[ora11gr2@dellpe dbs]$ cat initora11g.ora
db_name=ora11g
```

다시 SQL*Plus로 돌아가보자.

```
SQL> startup nomount
ORACLE instance started.

Total System Global Area  150667264 bytes
Fixed Size                  1335080 bytes
Variable Size              92274904 bytes
Database Buffers           50331648 bytes
```

```
Redo Buffers              6725632 bytes
SQL>
```

데이터베이스 구동 명령 옵션 중 nomount 옵션을 사용하였기 때문에 아직까지는 마운트 상태는 아니다(SQL*Plus 문서에 데이터베이스 구동과 정지에 관련된 모든 옵션에 대해 설명하고 있다).

| Note | 윈도우즈 OS에서 데이터베이스 구동 명령어를 실행하기 전이라면, oradim.exe라는 유틸리티를 이용하여 윈도우즈 서비스 생성 구문을 실행할 필요가 있을 것이다(역자 주_ 윈도우즈에서는 SQL*Plus를 이용하지 않아도 oradim.exe를 이용하여 데이터베이스 인스턴스를 명령어 라인에서 바로 구동시킬 수 있다. 예: C:\oradim.exe –startup –sid XXXX).

이제 무엇을 인스턴스라 부르는지 알 수 있게 되었다. 실제로 데이터베이스를 구동하기 위해 필요한 프로세스 모니터(pmon), Log Writer(lgwr)와 다수의(이런 종류의 프로세스들은 5장에서 자세하게 다루도록 하겠다) 백그라운드 프로세스를 아래에서 볼 수 있다

```
SQL> !ps -aef ¦ grep ora11gr2
ora11gr2  4447  4446  0 13:15 pts/1    00:00:00 -bash
ora11gr2  4900  4899  0 14:15 pts/2    00:00:00 /home/ora11gr2/bin/sqlplus
ora11gr2  4904     1  0 14:16 ?        00:00:00 ora_pmon_ora11g
ora11gr2  4906     1  0 14:16 ?        00:00:00 ora_vktm_ora11g
ora11gr2  4910     1  0 14:16 ?        00:00:00 ora_gen0_ora11g
ora11gr2  4912     1  0 14:16 ?        00:00:00 ora_diag_ora11g
ora11gr2  4914     1  0 14:16 ?        00:00:00 ora_dbrm_ora11g
ora11gr2  4916     1  0 14:16 ?        00:00:00 ora_psp0_ora11g
ora11gr2  4918     1  0 14:16 ?        00:00:00 ora_dia0_ora11g
ora11gr2  4920     1  0 14:16 ?        00:00:00 ora_mman_ora11g
ora11gr2  4922     1  0 14:16 ?        00:00:00 ora_dbw0_ora11g
ora11gr2  4924     1  0 14:16 ?        00:00:00 ora_lgwr_ora11g
ora11gr2  4926     1  0 14:16 ?        00:00:00 ora_ckpt_ora11g
ora11gr2  4928     1  0 14:16 ?        00:00:00 ora_smon_ora11g
ora11gr2  4930     1  0 14:16 ?        00:00:00 ora_reco_ora11g
ora11gr2  4932     1  0 14:16 ?        00:00:00 ora_mmon_ora11g
ora11gr2  4934     1  0 14:16 ?        00:00:00 ora_mmnl_ora11g
ora11gr2  4935  4900  0 14:16 ?        00:00:00 oracleora11g ↩
(DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=beq)))
ora11gr2  4953  4900  0 14:18 pts/2    00:00:00 /bin/bash -c ps -aef ¦ grep ora11gr2
ora11gr2  4954  4953  0 14:18 pts/2    00:00:00 ps -aef
ora11gr2  4955  4953  0 14:18 pts/2    00:00:00 grep ora11gr2
```

부가적으로, ipcs는 첫 번째는 공유 메모리와 세마포어들의 사용량에 대한 정보를 보여주며, 두 번째로 유닉스의 커뮤니케이션 장치들 간의 중요한 내부 통신에 대한 정보를 보여준다.

```
SQL> !ipcs -a

------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch     status
0x873d6bdc 753667     ora11gr2  660        153092096  16

------ Semaphore Arrays --------
key        semid      owner      perms      nsems
0x420a82a0 1015808    ora11gr2  660        104

------ Message Queues --------
key        msqid      owner      perms      used-bytes   messages

SQL>
```

아직까지는 '데이터베이스' 라고 할 만한 것은 없는 상태다. 비록 우리가 생성한 파라미터 파일 안에 데이터베이스 이름을 정의하였다 하지만, 데이터베이스는 아직 존재하지 않는다. 이 데이터베이스를 '마운트' 한다면 아직 데이터베이스 자체가 존재하지 않기에 마운트를 실패한다. 이제 데이터베이스를 생성해보도록 하겠다. 오라클 데이터베이스를 생성하는 몇 가지 단계에 대해 이야기할 것이다.

```
SQL> create database;
Database created.
```

실제로 위와 같이 문장 한 줄로도 데이터베이스를 생성할 수 있다. 그러나 실전에서는 로그 파일, 데이터 파일, 컨트롤 파일 등을 어디에 둘 것인지 오라클에 알려주어야만 하기 때문에 CREATE DATABASE 명령어에 좀 더 복잡하게 기술하여 생성해야 한다. 우리가 매일 사용하는 나머지 데이터 딕셔너리를 생성하기 위해 $ORACLE_HOME/rdbms/admin/catalog.sql 스크립트와 다른 catalog 스크립트를 실행해야 하지만(ALL_OBJECTS와 같이 자주 사용되는 뷰들은 아직 데이터베이스에 생성하지 않았다), 데이터베이스는 생성되었다고 할 수 있다. 오라클의 V$와 같은 뷰, 더 자세하게 이야기를 하자면 V$DATAFILE, V$LOGFILE, V$CONTROLFILE과 같이 데이터베이스를 구성하는 파일들에 대한 리스트를 보여주는 뷰들은 간단히 조회해볼 수 있다.

```
SQL> select name from v$datafile;

NAME
--------------------------------------------------------------------------------
/home/ora11gr2/dbs/dbs1ora11g.dbf
/home/ora11gr2/dbs/dbx1ora11g.dbf
/home/ora11gr2/dbs/dbu1ora11g.dbf
```

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

MEMBER
--------------------------------------------------------------------------------
/home/ora11gr2/dbs/log1ora11g.dbf
/home/ora11gr2/dbs/log2ora11g.dbf

SQL> select name from v$controlfile;

NAME
--------------------------------------------------------------------------------
/home/ora11gr2/dbs/cntrlora11g.dbf

SQL>
```

오라클은 오라클과 관련된 파일을 하나의 디렉터리에 생성되도록 하는 기본 모드를 사용하고 있으며, 영구적으로 존재하는 파일들의 집합으로서 데이터베이스를 생성한다. 만약 현재의 데이터베이스를 닫은 다음 다시 열고자 한다면, 아래와 같이 오류가 발생한다.

```
SQL> alter database close;
Database altered.

SQL> alter database open;
alter database open
*
ERROR at line 1:
ORA-1619c6: database has been previously opened and closed
```

인스턴스는 인스턴스가 구동 중에는 기껏해야 하나의 데이터베이스만을 마운트하고 오픈할 수 있다. 새로운 데이터베이스나 다른 데이터베이스를 열기 위해서는 이 인스턴스를 버리고(shutdown) 새로운 인스턴스를 생성(startup)하여야 한다.

요약하면 다음과 같다.

- 하나의 인스턴스는 백그라운드 프로세스와 공유 메모리의 합집합이다.
- 데이터베이스는 디스크에 저장된 데이터의 집합이다.
- 인스턴스는 오직 하나의 데이터베이스만 마운트하고 오픈할 수 있다.
- 하나의 데이터베이스를 하나 혹은 여러 인스턴스(RAC를 이용하여)에서 마운트하고 오픈할 수 있고, 하나의 데이터베이스에 마운트된 인스턴스 수는 시간에 따라 변경될 수 있다.

앞에서 이야기한 것처럼 대부분의 경우, 인스턴스와 데이터베이스는 1:1 관계를 갖는다. 이런 관계는 용어를 이해하는 측면에서 혼란을 초래하는 요인이 된다. 일반적으로 사람들의 경험에서 하나의 데이터베이스는 하나의 인스턴스며, 하나의 인스턴스 또한 하나의 데이터베이스라는 고정관념이 있기 때문이다.

그러나 다양한 테스트 환경에서는 다른 대칭 관계를 보인다. 필자의 장비에는 다섯 개의 서로 독립된 데이터베이스를 보유하고 있다. 물론, 테스트 장비에서 어느 특정 시점에는 오직 하나의 오라클 인스턴스만 구동한다. 그러나 데이터베이스는 필자의 요구에 따라 날마다 시시각각 다르게 변화하고 있다. 간단하게 다수의 서로 다른 환경 파일을 보유하는 것으로 데이터베이스를 서로 다른 모습으로 마운트하고 오픈할 수 있다. 지금 이 시간 필자는 하나의 인스턴스를 가지고 있지만 다수의 데이터베이스를 갖고 있으며, 특정 시점에는 그 중 단 하나(데이터베이스)에만 접근할 수 있다.

이제 인스턴스에 대해 이야기해보겠다. 이미 인스턴스는 오라클의 프로세스들과 메모리의 집합이라는 것을 알고 있을 것이다. 데이터베이스에 대해 묻는다면 데이터를 담고 있는 물리 파일들에 대해 이야기하는 것으로 가름할 것이다. 데이터베이스는 다수의 인스턴스가 접근이 가능해야 하지만, 특정 시점에는 정확하게 말하면 오직 하나의 데이터베이스에만 접근할 수 있다.

SGA와 백그라운드 프로세스

여러분은 이미 오라클 인스턴스와 데이터베이스가 무엇인지 그림 2-1과 같이 추상화한 그림을 준비했을지도 모른다.

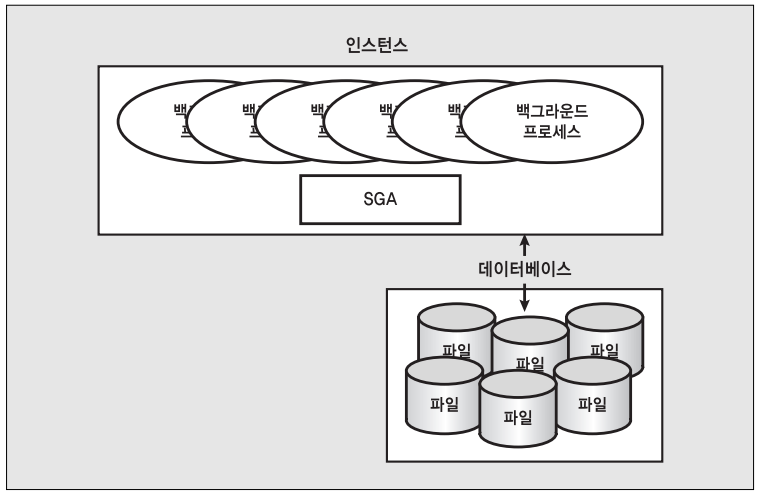

그림 2-1 | 오라클 인스턴스와 데이터베이스

그림 2-1은 오라클의 인스턴스와 데이터베이스에 대해 간단하게 설명한 것이다. 오라클은 SGA라고 부르는 커다란 단위의 메모리 영역을 갖는데, 이 SGA는 아래와 같은 특징이 있다.

- 모든 프로세스가 액세스하기 위해 필요한 다수의 내부 데이터 구조를 갖는다.
- 디스크에서 데이터를 읽어 캐시하기. 디스크에 쓰기 전 리두 데이터(redo data) 버퍼
- 파싱된 SQL의 실행계획 저장
- 기타 등등

오라클은 SGA에 덧붙여진 프로세스들의 집합이며, 운영체제에 붙여진 것과는 다른 메커니즘을 가지고 있다. 유닉스 환경에서 오라클 프로세스(일반적으로 세마포어의 shmget()와 shmat()를 사용하고 있음)들은, 다수의 프로세스들이 액세스하는 운영체제가 할당한 메모리의 큰 덩어리인 대형 공유 메모리 세그먼트에 물리적으로 붙어 있을 것이다.

윈도우즈에서 이러한 프로세스들은 간단하게 malloc()라고 불리는 C 함수를 이용해서 메모리를 할당받는다. 이것들은 실제로 하나의 큰 프로세스 안에 쓰레드로 존재하고, 그런 다음에 같은 가상 메모리 공간을 공유한다.

오라클은 또한 데이터베이스 프로세스 또는 쓰레드가 읽고 쓰는 파일 집합을 가지고 있으며, 오라클 프로세스는 이 파일을 읽고 쓰도록 유일하게 허용된 프로세스다. 테이블, 인덱스, 임시 공간, 리두 로그 및 그 밖의 것들을 포함하고 있다.

여러분이 유닉스 기반 시스템에서 오라클을 시작하고 ps 명령어를 실행했다면, 서로 다른 많은 이름의 프로세스가 실행 중인 것을 볼 것이다. 실제 예로서 pmon, smon 및 기타 여러 프로세스를 이미 보았다. 이런 프로세스는 5장에서 구체적으로 다룰 것이어서, 이후부터는 통칭해서 오라클 백그라운드 프로세스(oracle background process)라고 하겠다. 백그라운드 프로세스는 인스턴스를 구성하는 프로세스들이고, 인스턴스를 시작한 시점부터 셧다운하기 전까지 발견할 수 있다.

흥미로운 점은 그것들이 프로세스지 개별적인 프로그램이 아니라는 점이다. 유닉스에서는 단 하나의 오라클 바이너리 실행 파일만 존재한다. 그것은 실행될 때 어떻게 정의되느냐에 따라 다양한 '개성'을 지닌다. ora_pmon_ora11g를 실행하는 바이너리 실행 파일은 roa_ckpt_ora11g 프로세스를 얻기 위해서도 사용된다. 이진 실행 프로그램은 단 하나며, 그 이름은 바로 oracle이다. 그것은 다른 이름으로 여러 번 실행되었을 뿐이다.

윈도우즈에서 Windows XP Resource Kit(만약 가지고 있지 않다면, 검색엔진에서 'pstat download'라고 검색해보라)에서 제공하는 프로그램 중 하나인 pstat 툴을 사용하면, oracle.exe라는 하나의 프로세스를 찾을 것이다. 즉, 윈도우즈에서는 oracle.exe라는 하나의 실행 파일만 데이터베이스를 구동하기 위해

필요하고, 데이터베이스가 구동된 이후로도 실제로 구동하는 프로세스로 오직 하나만 존재한다는 뜻이다. 이 프로세스 안에서 오라클의 백그라운드 프로세스들을 대표하는 다양한 쓰레드를 발견할 수 있을 것이다.

pstat(혹은 다른 툴들)를 사용하면 이런 프로세스들을 볼 수 있다.

```
C:\WINDOWS> pstat

Pstat version 0.3:  memory: 523760 kb  uptime:  0  1:37:54.375

PageFile: \??\C:\pagefile.sys
        Current Size: 678912 kb  Total Used: 228316 kb   Peak Used 605488 kb

 Memory: 523760K Avail: 224492K  TotalWs: 276932K InRam Kernel:  872K P:20540K
 Commit: 418468K/ 372204K Limit:1169048K Peak:1187396K  Pool N:10620K P:24588K

    User Time   Kernel Time    Ws   Faults  Commit Pri Hnd Thd Pid Name
                           56860  2348193                        File Cache
  0:00:00.000   1:02:23.109    28       0       0  0    0   1   0 Idle Process
  0:00:00.000   0:01:50.812    32    4385      28  8  694  52   4 System
  0:00:00.015   0:00:00.109    60     224     172 11   19   3 332 smss.exe
  0:00:33.234   0:00:32.046  2144   33467    1980 13  396  14 556 csrss.exe
  0:00:00.343   0:00:01.750  3684    6811    7792 13  578  20 580 winlogon.exe
  0:00:00.078   0:00:01.734  1948    3022    1680  9  275  16 624 services.exe
  0:00:00.218   0:00:03.515  1896    5958    3932  9  363  25 636 lsass.exe
  0:00:00.015   0:00:00.078    80     804     592  8   25   1 812 vmacthlp.exe
  0:00:00.093   0:00:00.359  1416    2765    3016  8  195  17 828 svchost.exe
  0:00:00.062   0:00:00.453  1340    3566    1764  8  244  10 896 svchost.exe
  0:00:00.828   0:01:16.593  9632   36387   11708  8 1206  59 1024 svchost.exe
  0:00:00.046   0:00:00.640  1020    2315    1300  8   81   6 1100 svchost.exe
  0:00:00.015   0:00:00.234   736    2330    1492  8  165  11 1272 svchost.exe
  0:00:00.015   0:00:00.218   128    1959    3788  8  117  10 1440 spoolsv.exe
  0:00:01.312   0:00:19.828 13636   35525   14732  8  575  19 1952 explorer.exe
  0:00:00.250   0:00:00.937   956    1705     856  8   29   1 228 VMwareTray.exe
  0:00:00.812   0:00:04.562  1044    4619    3800  8  165   4 240 VMwareUser.exe
  0:00:00.015   0:00:00.156    88    1049    1192  8   88   4 396 svchost.exe
  0:00:00.109   0:00:04.640   744    1229    2432  8   81   3 460 cvpnd.exe
  0:00:02.015   0:00:12.078  1476   17578    1904 13  139   3 600 VMwareService.exe
  0:00:00.031   0:00:00.093   124    1004    1172  8  105   6 192 alg.exe
  0:00:00.062   0:00:00.937  2648   13977   22656  8  101   3 720 TNSLSNR.EXE
  0:04:00.359   0:02:57.734164844 2009785  279168  8  550  29 1928 oracle.exe
  0:00:00.093   0:00:00.437  6736    2316    2720  8  141   6 1224 msiexec.exe
  0:00:00.015   0:00:00.031  2668     701    1992  8   34   1 804 cmd.exe
  0:00:00.015   0:00:00.000   964     235     336  8   11   1 2856 pstat.exe
```

여기서 하나의 오라클 프로세스가 29개의 쓰레드(Thd 열)를 포함하고 있다는 것을 확인할 수 있다. 유

닉스에서 프로세스라고 나타나는 쓰레드는 pmon, arch, lgwr과 오라클의 몇 가지 프로세스들임을 알 수 있었다. pstat 리포트의 아래 페이지로 내려가보면 각각 쓰레드에 대해 보다 더 자세히 살펴볼 수 있다.

```
pid:788 pri: 8 Hnd:  550 Pf:2009785 Ws: 164844K oracle.exe
 tid pri Ctx Swtch StrtAddr    User Time  Kernel Time  State
 498   9       651 7C810705  0:00:00.000  0:00:00.203 Wait:Executive
 164   8        91 7C8106F9  0:00:00.000  0:00:00.000 Wait:UserRequest
 ...
 a68   8        42 7C8106F9  0:00:00.000  0:00:00.031 Wait:UserRequest
```

위 리포트에서는 유닉스에서 볼 수 있던 '이름'(ora_pmon_ora11g 외)을 가진 쓰레드를 볼 수 없으나 ID(Tid) 쓰레드와 priority(Pri)를 볼 수 있고, 각각에 대한 운영체제 계정의 정보를 확인할 수 있다.

오라클에 접속하기

이번 절에서는 오라클에 서비스를 요청하는 두 가지 일반적인 방법(dedicated server와 shared server 커넥션)에 대해 알아보기로 하자. 클라이언트와 서버 간에 커넥션이 이루어지기 위해 어떤 일이 일어나는지 알아보도록 하겠다. 데이터베이스에서 수행되는 모든 일은 로그인한 이후 수행된다. 마지막으로 TCP/IP 커넥션에 대해서도 잠시 알아보도록 하겠다. 그리고 실제로 서버에 물리적인 접속이 이루어지는 데 결정적 역할을 담당하는 서버에 물리적인 접속을 담당하는 리스너 프로세스가 dedicated 커넥션과 shared server 커넥션에서 어떻게 서로 다르게 동작하는지 알아보도록 하겠다.

Dedicated Server

그림 2-1과 pstat 출력물은 오라클이 구동된 바로 직후 어떤 모습을 갖게 되는지 보여주고 있다. dedicated server를 이용하여 데이터베이스에 접속하게 되면, 서비스를 실행하기 위해 새롭게 생성된 프로세스를 확인할 수 있다.

```
C:\Documents and Settings\tkyte>sqlplus tkyte/tkyte

SQL*Plus: Release 11.1.0.7.0 - Production on Fri Dec 11 18:05:32 2009

Copyright (c) 1982, 2008, Oracle. All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.7.0 - Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
```

```
SQL> host pstat
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
Pstat version 0.3:  memory: 523760 kb  uptime:  0  1:40:36.687

PageFile: \??\C:\pagefile.sys
        Current Size: 678912 kb  Total Used: 227744 kb   Peak Used 605488 kb

 Memory: 523760K Avail: 194928K  TotalWs: 315172K InRam Kernel:  876K P:20616K
 Commit: 447888K/ 401420K Limit:1169048K Peak:1187396K  Pool N:10636K P:24628K

    User Time   Kernel Time    Ws   Faults  Commit Pri Hnd Thd Pid Name
...
  0:04:00.515   0:02:58.546166948  2020411  279216  8  549  30 1928 oracle.exe
...
SQL>
```

쓰레드 29개 대신 쓰레드 30개가 있는 것을 확인할 수 있는데, 나머지 쓰레드 한 개는 dedicated server 프로세스다(dedicated server 프로세스에 대한 내용은 바로 다음에 다루도록 하겠다). 로그아웃을 하면 여분의 쓰레드인 dedicated server 프로세스는 사라질 것이다. 유닉스 환경에서 실행 중인 오라클 프로세스에 추가된 또 다른 프로세스들을 볼 수 있는데, 바로 dedicated server 프로세스들이다.

```
[tkyte@dellpe ~]$ ps -aef ¦ grep oracle$ORACLE_SID
tkyte    26935 19699  0 16:05 pts/5    00:00:00 grep oracleora11gr2
[tkyte@dellpe ~]$ sqlplus /

SQL*Plus: Release 11.2.0.1.0 Production on Mon May 10 16:05:22 2010

Copyright (c) 1982, 2009, Oracle. All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

ops$tkyte%ORA11GR2> !ps -aef ¦ grep oracle$ORACLE_SID
ora11gr2 26938 26937  1 16:05 ?        00:00:00 oracleora11gr2 ←
 (DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=beq)))
tkyte    26947 26945  0 16:05 pts/2    00:00:00 grep oracleora11gr2
```

이 dedicated server는 다음 그림에서 반복적으로 나타난다. 가장 일반적인 구성 방식으로 설치된 오라클 서버에 접속을 한다면, 그림 2-2와 같은 모습을 보게 될 것이다.

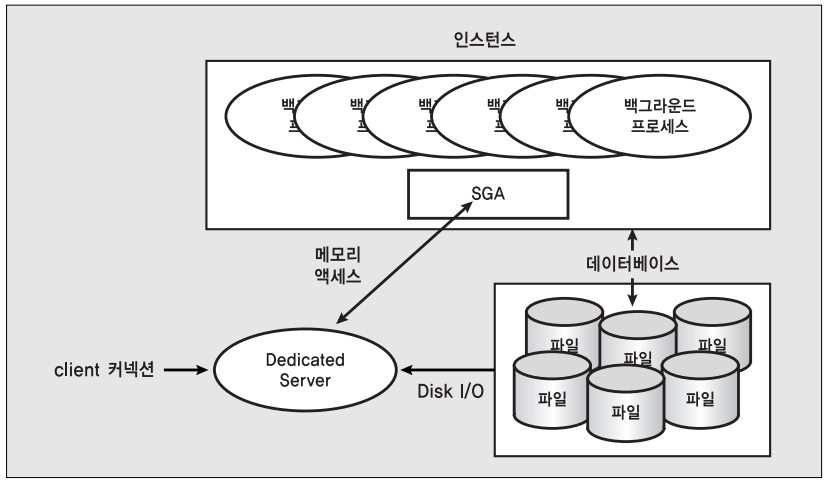

그림 2-2 | 일반적인 dedicated server 구성도

이미 이야기한 것과 같이 필자가 로그인할 경우, 필자의 서버 프로세스를 위해 새로운 프로세스를 생성하는 것이 오라클의 일반적인 모습이다. 이것이 dedicated server 환경 구성에 대한 일반적인 방식이며, 세션의 수명이 다할 때까지 서버 프로세스 하나가 전용으로 연결된다. 각각의 모든 세션에 대해 새로운 dedicated server는 1:1로 매핑한 형태로 나타날 것이다. 이 dedicated server 프로세스는 인스턴스의 일부분은 아니다. 필자의 클라이언트 프로세스(데이터베이스에 접속하려고 하는 어떤 종류의 프로그램에 생성된 것이든)가 TCP/IP 소켓과 같은 네트워크를 따라 dedicated server를 통해 직접 통신한다. 이 서버 프로세스는 필자의 SQL을 접수하고 실행하는데, 필요하면 데이터 파일을 읽고, 읽은 결과의 데이터를 데이터베이스의 캐시에서 볼 수 있을 것이다. dedicated server 프로세스는 필자의 업데이트 문장을 수행할 것이다(역자 주_ 사용자가 데이터베이스에 던진 SQL 문은 사용자의 프로세스를 통해 데이터베이스에 직접 연결된 dedicated server 프로세스를 통해 데이터베이스에 SQL 문이 전달되고 수행된다). 필자의 PL/SQL 코드를 실행하고, 필자가 데이터베이스에 보낸 SQL 구문에 대해 답을 한다.

Shared Server

오라클은 shared server라는 방식의 커넥션을 또한 허용하는데, 부가적인 새로운 쓰레드를 생성한다거나 모든 사용자 커넥션별로 새로운 유닉스 프로세스를 할당하거나 하지 않는다.

| Note | 오라클 버전 7.x와 8.x에서 shared server는 Multi-Threaded Server 혹은 MTS라고 알려졌다. 이런 과거의 이름은 더 이상 사용하지 않는다.

오라클은 shared server에서 다수 사용자를 위한 공유 프로세스들의 풀(pool)을 사용한다. 공유 서버

는 단순한 접속 풀링 구조인데, 10,000개의 데이터베이스 세션을 위해 10,000개의 전용 서버를 갖는 대신 모든 세션이 공유할 수 있는 작은 비율 정도의 프로세스/쓰레드를 가질 수 있다. dedicated server 구조에서 접속이 가능한 수보다 훨씬 더 많은 사용자가 접속할 수 있다. 10,000개의 프로세스 부하로 인해 서버가 멈춰버릴지도 모르는 상황이라면 100이나 1,000개의 프로세스에서는 운영이 가능할지 모른다. shared server 모드에서 shared server는 일반적으로 데이터베이스와 함께 구동되며, ps list에 나타난다.

shared server와 dedicated server의 가장 큰 차이점이라 하면, 클라이언트 프로세스가 dedicated server를 경유하여 직접 데이터베이스와 통신하는 반면, shared server는 다른 매개체를 거친다는 것이다. 사실, shared server가 프로세스를 공유한다고 말할 수는 없다. 이 프로세스(shared process)들을 공유하기 위해서는 데이터베이스와 프로세스 간의 '대화'에 대한 또 다른 메커니즘이 필요하다. 오라클은 이러한 목적(역자 주_ 서버 프로세스와 데이터베이스 서버 프로세스 간의 대화)을 위해 dispatcher라는 프로세스(혹은 프로세스 집합)를 사용한다. 클라이언트 프로세스는 네트워크에서 dispatcher 프로세스와 통신할 것이다. dispatcher 프로세스는 클라이언트의 요청을 SGA에 있는 request queue에 넣는다. busy하지 않은 제일 첫 번째 shared server는 이 요청 메시지를 꺼내서 실행한다(예제를 보자. 요청 메시지는 UPDATE T SET X = X + 5 WHERE Y = 2라고 하겠다). 이 명령이 끝날 때까지 shared server는 dispatcher의 응답 queue에 응답을 위치시킬 것이다. dispatcher 프로세스는 이 queue를 모니터링하며, 결과를 확인하고, 클라이언트에 이 결과를 전달할 것이다. shared server 요청에 대한 흐름은 그림 2-3과 같다.

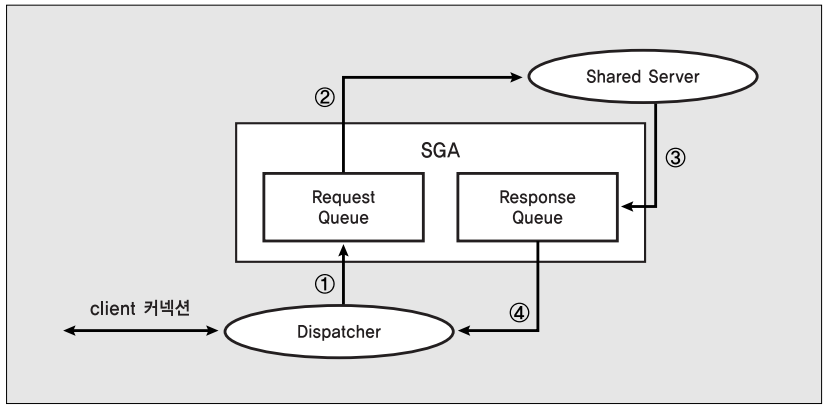

그림 2-3 | shared serve의 request 처리 절차

그림 2-3과 같이 클라이언트 커넥션은 dispatcher에 요청을 한다. dispatcher는 첫 번째로 요청건을 SGA(1)에 있는 요청 queue에 집어넣는다. 첫 번째 가용 shared server 프로세스는 이 요청을 빼내서 (2) 실행할 것이다. 공유 서버의 실행이 완료되면, 응답결과(코드, 결과 데이터 등을 반환)는 응답 queue(3)

에 옮겨지고, 곧바로 dispatcher(4)가 응답 queue에서 결과를 집어내어 클라이언트에 전달한다.

개발자의 입장에서 보면, shared server 커넥션과 dedicated server 커넥션 사이에 개념적인 차이는 없다.

두 가지 방식에 대한 차이를 이해하였으니 아래의 질문에 답할 수 있을 것이다.

- 제일 처음 어떻게 데이터베이스에 접속할 것인가?
- 무엇이 dedicated server를 시작할 것인가?
- 어떻게 dispatcher에 접근할 것인가?

해답은 여러분의 고유한 환경에 달렸으나 여기서는 일상적인 용어 측면에서 개괄적인 개념을 전달하도록 하겠다.

TCP/IP를 이용하여 접속하는 기법

이제부터 일반적인 네트워킹 사례에 대해 살펴볼 것이다. 네트워크 기반 접속은 TCP/IP 접속 기반에서 요청한다. 이 경우에 한쪽 장비에는 클라이언트가, 다른 쪽 장비에는 서버가 있고, 둘 간에는 TCP/IP 네트워크로 연결되어 있다. 모든 접속은 클라이언트에서 시작된다. 클라이언트는 데이터베이스에 접속하기 위해 오라클 클라이언트 소프트웨어(오라클이 제공하는 '애플리케이션 프로그램 인터페이스(API)'의 모음)를 사용하여 요청한다. 예를 들어 클라이언트는 다음과 같이 접속을 요청한다.

```
[tkyte@dellpe ~]$ sqlplus scott/tiger@orcl

SQL*Plus: Release 11.2.0.1.0 Production on Fri Dec 11 16:00:31 2009

Copyright (c) 1982, 2009, Oracle. All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

scott%ORA11GR2>
```

| Note | 위에서 사용한 orcl은 필자가 설정한 환경에서 유일한 것이며, orcl이란 명칭으로 tnsnames.ora(잠시 후에 좀 더 자세히 설명하겠다)에 포함되어 있다. 그것은 TNS 접속 구문으로서, 필자의 네트워크에 설치되어 존재하는 오라클 11g R2 인스턴스를 가리킨다. 여러분은 여러분의 환경에서 식별 가능한 TNS 접속 구문을 사용할 것이다.

테스트에서 사용할 클라이언트 프로그램은 SQL*Plus이며, 사용자이름과 패스워드로 scott/tiger를 사용한다. orcl은 TNS 서비스 이름인데, TNS는 Transparent Network Substrate의 약어로서, peer-to-peer 커뮤니케이션을 가능케 하는 원격 접속을 제어하도록 오라클 클라이언트 프로그램에 내장된 '핵심' 소프트웨어다. TNS 접속 구문은 오라클 소프트웨어가 어떻게 원격 접속할 것인가를 말해준다(역자 주_ 접속할 원격 데이터베이스에 대한 주소 정보). 일반적으로 여러분의 장비에서 돌아가는 클라이언트 소프트웨어(역자 주_ SQL*Plus나 T*, O*와 같은 데이터베이스 관련 프로그램들)는 tnsnames.ora 파일을 열어 접속하고자 하는 장비에 대한 정보를 읽어볼 것이다. tnsnames.ora 파일은 일반적으로 $ORACLE_HOME/network/admin 디렉터리($ORACLE_HOME은 오라클 소프트웨어가 설치된 디렉터리의 전체 경로를 대표한다)에서 발견할 수 있는 텍스트 형태로 기록된 환경 파일이다. tnsnames.ora에는 다음과 같은 항목들이 있다.

```
[tkyte@dellpe ~]$ cat $ORACLE_HOME/network/admin/tnsnames.ora

ORCL =
  (DESCRIPTION =
    (ADDRESS =
         (PROTOCOL = TCP)
         (HOST = somehost.somewhere.com)
         (PORT = 1521)
    )
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = orcl)
    )
  )

[tkyte@dellpe ~]$
```

이런 구성 정보는 오라클 클라이언트 소프트웨어에 우리가 사용하는 TNS 접속 문자열(orcl)과 리스너 프로세스가 접속을 허용할 호스트의 포트(port) 번호, 호스트 이름, 우리가 접속하고자 하는 호스트에 있는 데이터베이스의 서비스 이름 등을 기록하고 있다. 서비스 이름은 공통 속성, 서비스 수준의 한계, 우선순위 등이 포함된 애플리케이션 그룹을 상징한다. 서비스를 제공하는 인스턴스 개수는 애플리케이션에 투명하며(가령 서비스 이름이 orcl이라고 하면, orcl로 서비스를 수행하는 인스턴스 중 누가 서비스하든 애플리케이션은 orcl에만 서비스를 요청하면 된다는 의미), 각각의 데이터베이스 인스턴스는 다수의 서비스를 제공하기 위해 리스너에 등록하여야 한다. 그래서 서비스는 물리 데이터베이스에 각각 매핑되며, DBA가 확실한 임계치와 인스턴스에 대한 우선순위 결정을 허용한다.

orcl 구문은 다른 도구를 이용하여 해석할 수도 있다. 예를 들어 orcl은 호스트 이름을 해석하기 위해 사용되는 DNS와 같이, 분산 Lightweight Directory Access Protocol(LDAP) 서버인 Oracle Internet

Directory(OID)를 이용하여 해석될 수 있다. 그렇지만 tnsnames.ora 파일을 사용하면 환경 구성 파일을 손쉽게 복사할 수 있으므로, 다수의 중소규모 데이터베이스를 설치할 때 훨씬 더 보편적이라 할 수 있다.

클라이언트 소프트웨어가 어디에 접속하고자 하는지 안다면, 클라이언트 소프트웨어는 1521 포트로 somehost.somewhere.com이라는 호스트 이름을 가진 서버로 접속하기 위해 TCP/IP 소켓을 열 것이다. 서버를 담당하는 DBA가 Oracle Net을 설치하고 구성하였다면, 접속에 대한 요청에 대해 1521 포트로 리스너를 구성했을 것이기에 접속이 허용될 것이다. 네트워크 환경에서 우리 서버 안에 TNS Listener라고 부르는 프로세스가 떠 있는 것을 확인할 수 있을 것이다. 리스너 프로세스가 서버로 접속하겠다는 요청을 접수하면, 각 요청건에 대해 리스너의 환경 파일을 이용하여 접속 요청을 허락할 것인지 불허할 것인지(예를 들어 서비스가 명시되지 않았다거나, 설사 서비스가 명시되었더라도 접속하고자 하는 호스트의 IP 주소가 접속 불가하다거나 하는 경우)를 결정할 것이다.

dedicated server 커넥션 환경을 구성하였다면 리스너 프로세스는 우리에게 dedicated server를 하나 생성해줄 것이다. 유닉스에서 fork(), exec() system call(유닉스에 오라클이 설치된 이후 새로운 프로세스를 생성할 수 있는 방법은 fork() 이외에는 없다)을 거쳐 새로운 dedicated server를 획득한다. 새로운 dedicated server 프로세스는 리스너에 의해 커넥션 권한을 상속받아 데이터베이스에 물리적인 커넥션을 하게 된다. 윈도우즈에서 리스너 프로세스는 데이터베이스 프로세스에 접속을 위한 새로운 쓰레드를 생성해달라고 요청한다. 이 쓰레드가 생성되면 클라이언트는 쓰레드로 '리다이렉트(redirect)'하여 물리적으로 커넥션을 맺게 된다. 유닉스에서 도식화한 것이 그림 2-4다.

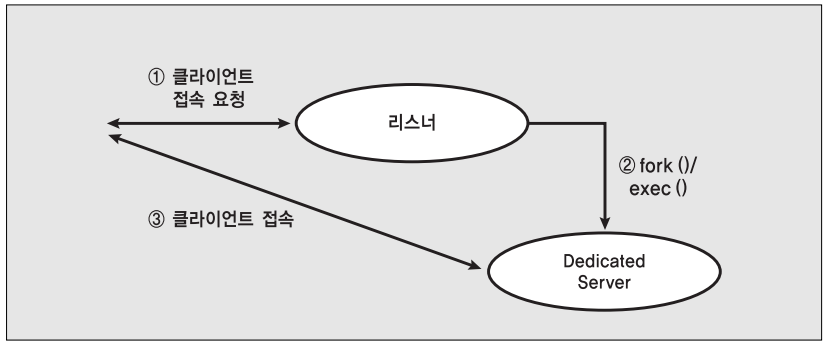

그림 2-4 | 리스너 프로세스와 dedicated server 커넥션

반면, shared server 접속을 요구하면 리스너는 dedicated server 커넥션에서 보여주었던 양식과는 다른 모습을 보인다. 리스너는 인스턴스에서 운영 중인 dispatcher 프로세스에 대한 정보를 가지고 있다. 접속 요구를 받으면 리스너는 현재 사용 가능한 dispatcher들의 pool에서 한 개의 dispatcher 프로세서를 선택한다. 리스너는 dispatcher 프로세스에 접속할 수 있게 하는 커넥션 정보를 클라이언트에 보

내주거나, 할 수만 있다면 dispatcher 프로세스에 연결 요청을 넘긴다(이 기능은 운영체제 및 데이터베이스 버전에 의존적이긴 하나 실제 효과는 동일하다). 리스너가 커넥션 정보를 접속을 요청한 프로세스에 되돌려주는데, 리스너가 외부에 잘 알려진 호스트 이름과 포트를 운영하고 있기 때문이다. 그러나 dispatcher는 서버에 랜덤하게 포트를 할당하여 커넥션을 수락한다. 리스너는 이런 dispatcher가 무작위로 할당한 포트 정보를 알게 되고, 데이터베이스에 접속하려는 사용자에게 dispatcher를 할당하게 된다. 즉, 클라이언트는 리스너 접속을 끝내면 바로 dispatcher에 직접 접속한다. 물리적으로 데이터베이스에 연결된 것이다.

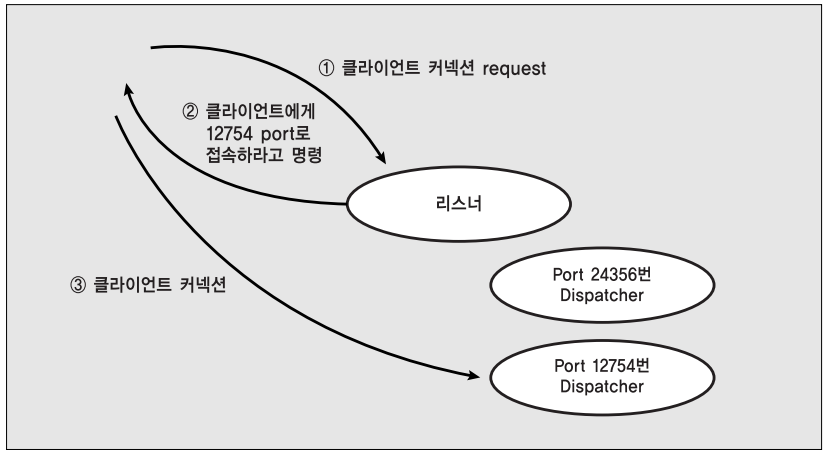

그림 2-5 | 리스너 프로세스와 shared server 커넥션

정리

이제 오라클 아키텍처에 대한 개요를 마치고자 한다. 이번 장에서는 '인스턴스'와 '데이터베이스'란 용어를 정의했고, 어떻게 데이터베이스에 접속하는지를 dedicated server 커넥션과 shared server 커넥션을 통해 알아보았다. 그림 2-6은 이번 장을 요약한 그림이다. shared server 커넥션과 dedicated server 커넥션 사이의 상호작용을 보여주고 있다. 이것은 오라클 인스턴스가 두 커넥션 타입을 동시에 사용할 수도 있음을 보여준다(사실, 오라클 데이터베이스는 항상 dedicated server 커넥션을 제공한다. 심지어 shared server 환경일 때도)(역자 주_ 일반적으로 TNSNAMES.ORA 파일에

```
...
 (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = 서비스 명)
    )
```

라고 정의하면 shared server 환경에서도 dedicated server 커넥션으로 접속할 수 있음).

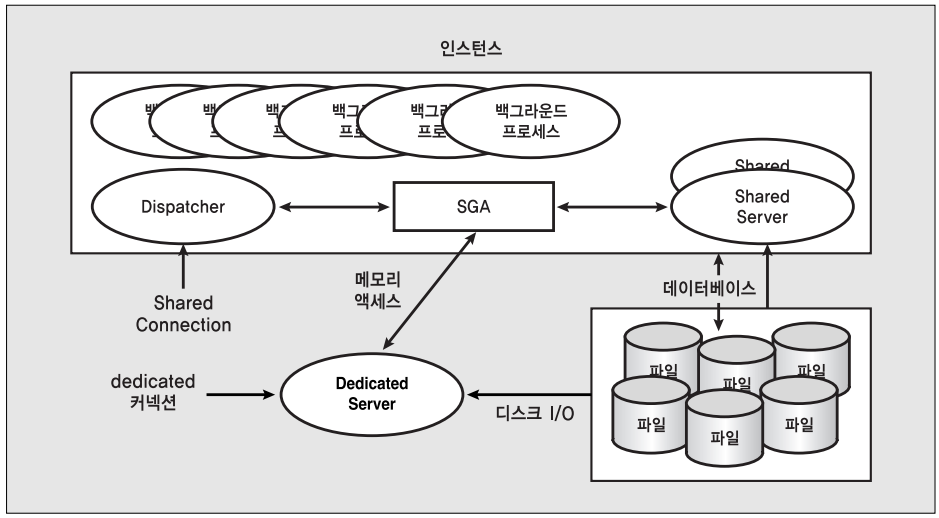

그림 2-6 | 커넥션 개념

이제 서버 뒤에 숨겨진 프로세스들이 구체적으로 무슨 일을 하고 다른 프로세스들과 어떤 상호 관계가 있는지 알아볼 것이다. 또한 SGA의 깊은 내면을 알아보기 위해 프로세스들이 무엇을 포함하고 무엇을 제공하는지 알아볼 준비가 되었다. 다음 장에서 오라클 파일의 타입을 알아보고, 데이터 관리와 각 파일의 역할이 무엇인지 알아보기로 하자.