

# 데이터베이스개론

2023년 2학기 실습자료

ex\_db

# 실습 1

## 제 3 장 오라클 소개

- 오라클 소개
- 오라클 설치 방법
- 오라클 구조

# 오라클(Oracle)의 역사

---

- 1978년
  - 로렌스 J. 엘리슨(현 회장)이 관계형 DBMS인 오라클 첫 번째 버전(Version 1)을 개발
- 1979년
  - 회사명을 RSI(Relational Software Inc.)로 바꾸고 첫 번째 상용 DBMS인 오라클 두 번째 버전(Version 2)을 개발
- 1983년
  - 회사 이름을 지금의 오라클로 바꾸고 C언어로 개발된 오라클 세 번째 버전(Version 3)을 출시
- 1999년
  - 오라클 8i 출시 (i는 인터넷의 약자)
- 2003년
  - 오라클 10g 출시 (g는 그리드 컴퓨팅의 약자)
- 현재
  - 오라클 11g가 최신 버전\*\*

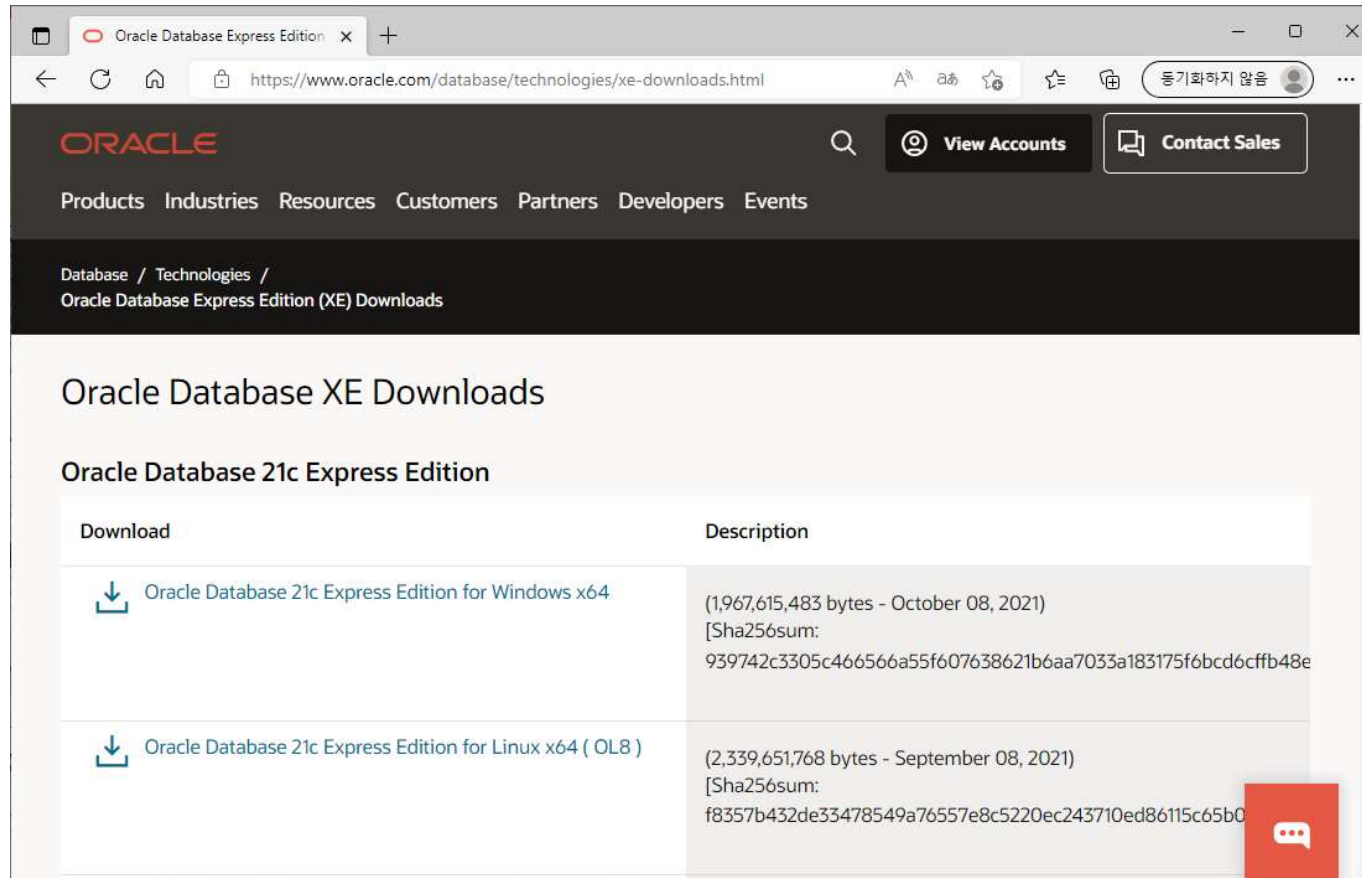
# 오라클 DBMS

---

- 클라이언트/서버 및 분산 처리 환경 지원
- 다양한 운영체제 지원
  - MS Windows, Unix(Solaris, HP-UX 등), Linux
- 대용량 데이터 처리 지원
  - Petabyte 크기의 데이터 저장 관리
- 많은 사용자의 동시 접속 지원
- 신뢰성 높은 보안 기능 제공
  - 인증, 권한 관리, 암호화 등
- 오류 및 장애에 대한 대비책 지원
  - backup, recovery 기능
- 다양한 데이터 및 응용 분야 지원
  - Multimedia objects, XML
  - Data Warehouse/OLAP, Mobile, Grid Computing, Cloud Computing

# Oracle Express Edition

- 다운로드



The screenshot shows the Oracle Database XE Downloads page. The browser address bar displays the URL: <https://www.oracle.com/database/technologies/xe-downloads.html>. The page header includes the Oracle logo, a search icon, and links for 'View Accounts' and 'Contact Sales'. Below the header, a navigation menu lists 'Products', 'Industries', 'Resources', 'Customers', 'Partners', 'Developers', and 'Events'. The main content area is titled 'Oracle Database XE Downloads' and 'Oracle Database 21c Express Edition'. It features a table with two columns: 'Download' and 'Description'.

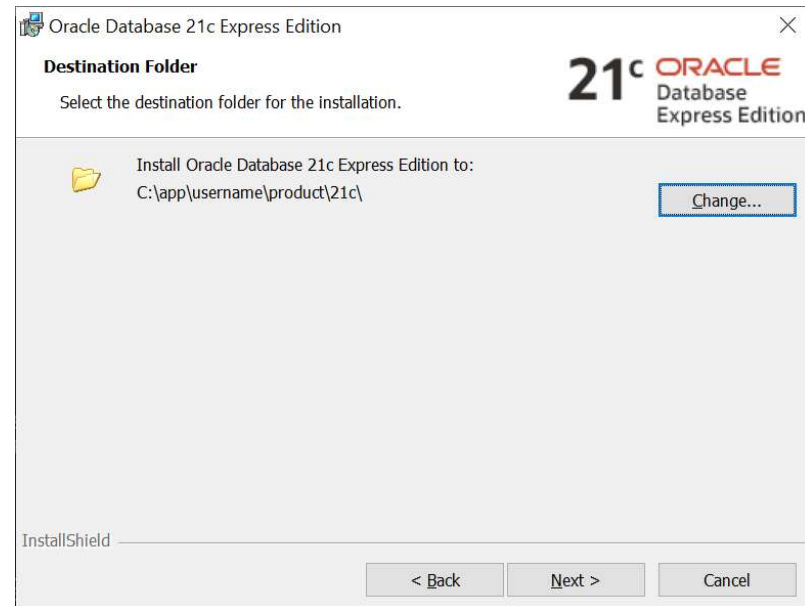
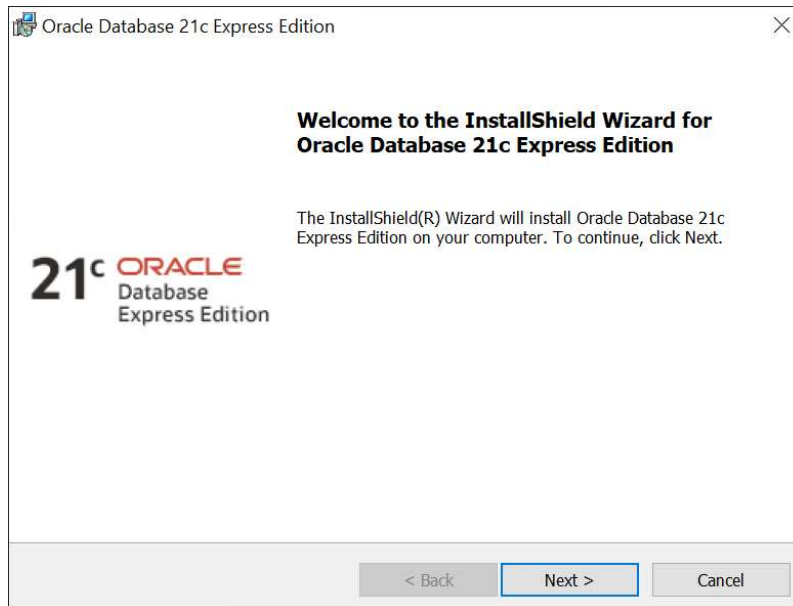
Download	Description
<a href="#">Oracle Database 21c Express Edition for Windows x64</a>	(1,967,615,483 bytes - October 08, 2021) [Sha256sum: 939742c3305c466566a55f607638621b6aa7033a183175f6bcd6cffb48e
<a href="#">Oracle Database 21c Express Edition for Linux x64 ( OL8 )</a>	(2,339,651,768 bytes - September 08, 2021) [Sha256sum: f8357b432de33478549a76557e8c5220ec243710ed86115c65b0

# Oracle Express Edition

---

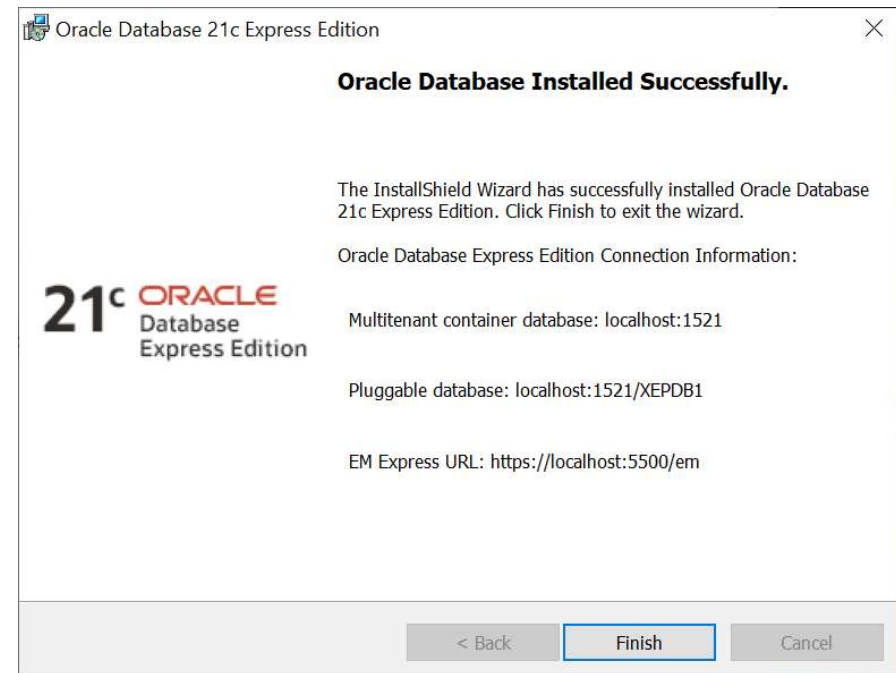
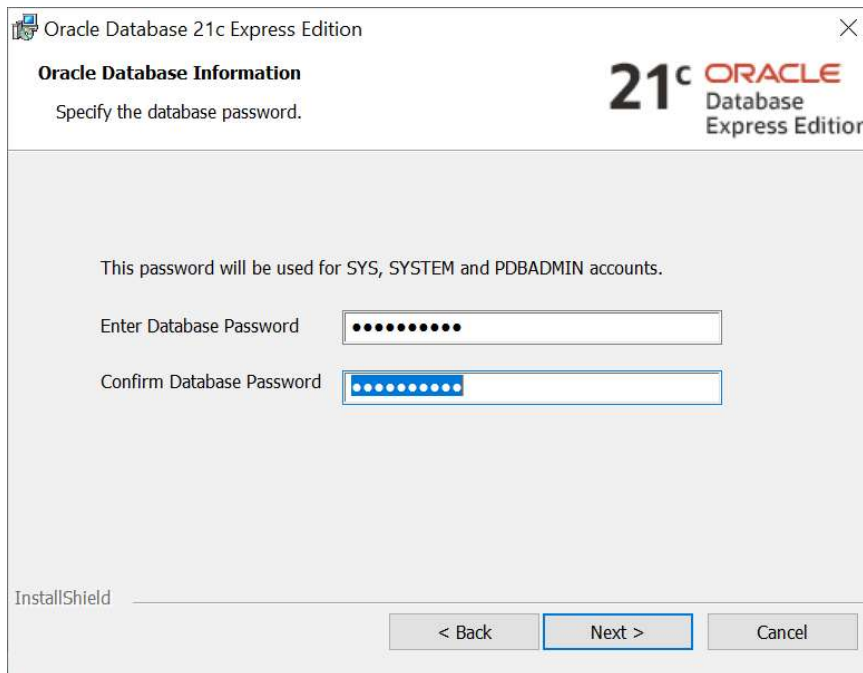
- 설치

- Administrator(윈도우즈 계정) 권한이 있는 계정으로 로그인 후 다운로드 실행
- Zip 파일을 압축 해제하고 database/setup.exe 파일을 실행
- 설치 폴더 지정



# Oracle Express Edition

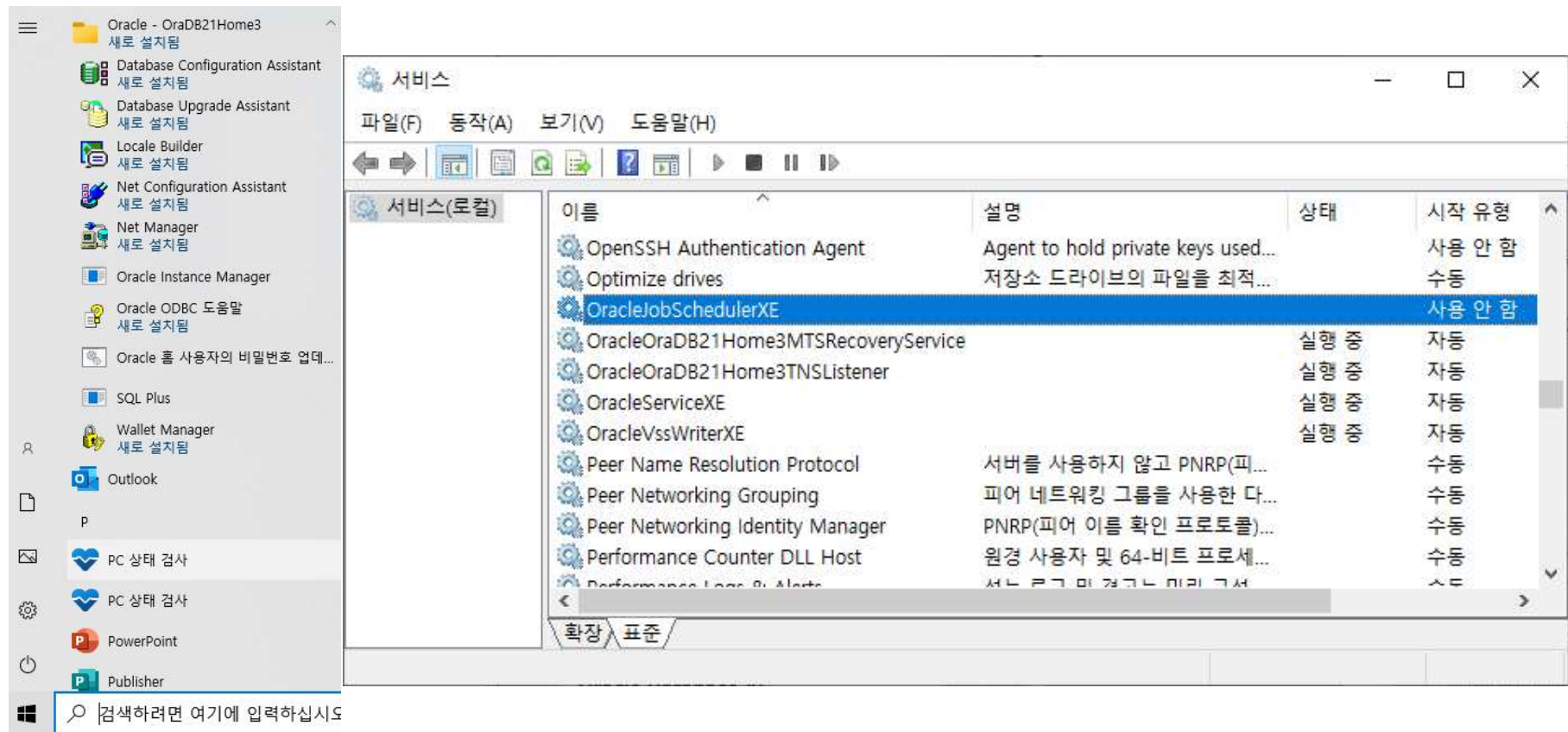
- 설치
  - 시스템 관리자 계정(SYS, **SYSTEM**)을 위한 암호 설정
  - 자세한 사항은 Installation Guide 참조
    - ▶ <https://docs.oracle.com/en/database/oracle/oracle-database/21/xeinw/index.html>





# Oracle Express Edition

- Windows 메뉴 및 Service 등록 확인
  - OracleServiceXE, OracleOraDB21Home1TNSListener 서비스 실행 중



# Oracle Express Edition

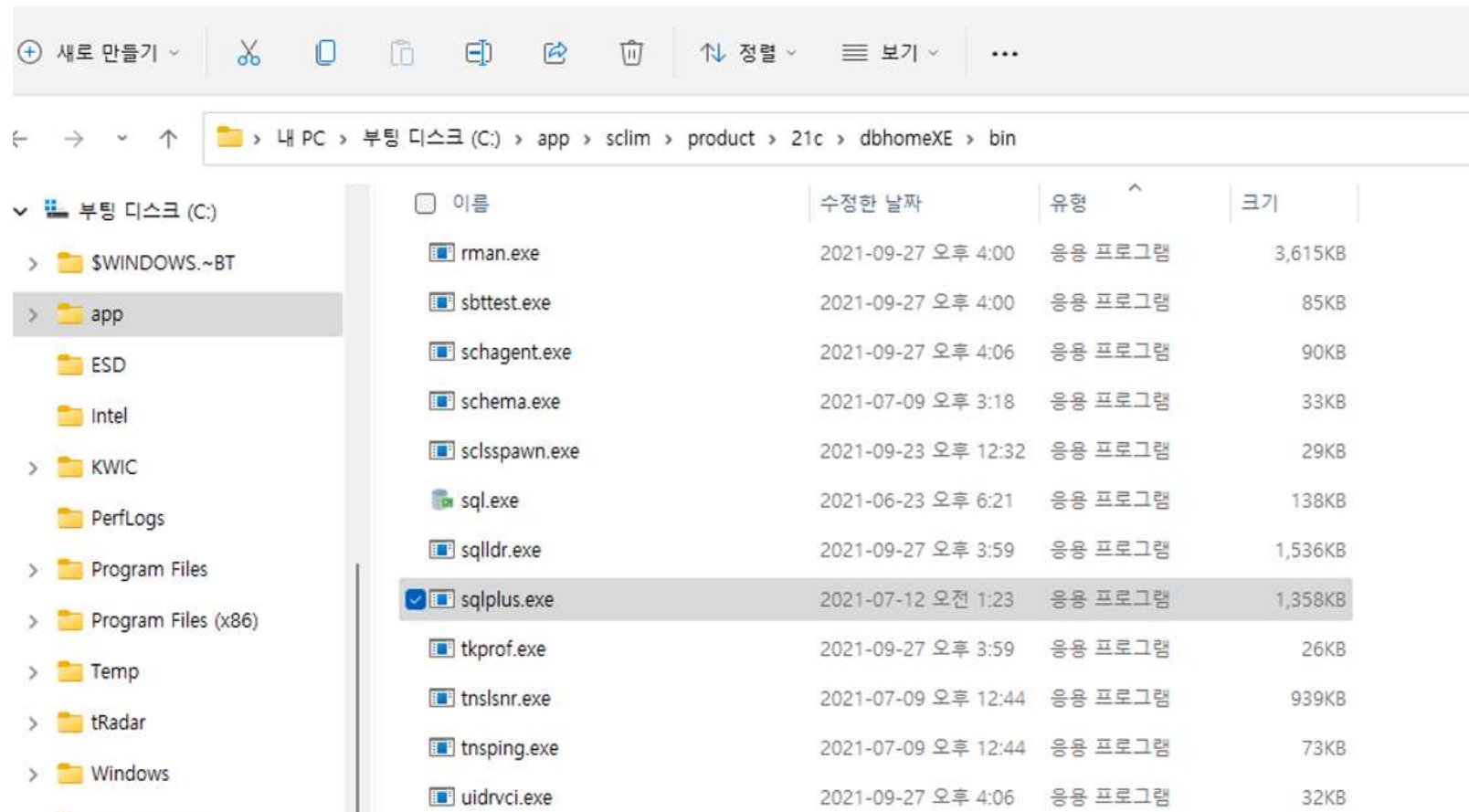
---

- 설치 폴더

File Name and Location	Purpose
<INSTALL_DIR>	Oracle Base This is the root of the Oracle Database XE directory tree.
<INSTALL_DIR>\dbhomeXE	Oracle Home This home is where the Oracle Database XE is installed. It contains the directories of the Oracle Database XE executables and net work files.
<INSTALL_DIR>\oradata\XE	Database files
<INSTALL_DIR>\diag\rdbms\XE\XE\trace	Diagnostic logs The database alert log is <INSTALL_DIR>\diag\rdbms\XE\XE\trace\alert_XE.log
<INSTALL_DIR>\cfgtoollogs\	Database installation, creation, and configuration logs. The <INSTALL_DIR>\cfgtoollogs\dbca\XE\XE.log file contains the results of the database creation script execution.

# Oracle Express Edition

- 설치 폴더



새로 만들기 | ✂ | 📄 | 📁 | 📌 | 🗑 | 정렬 | 보기 | ...

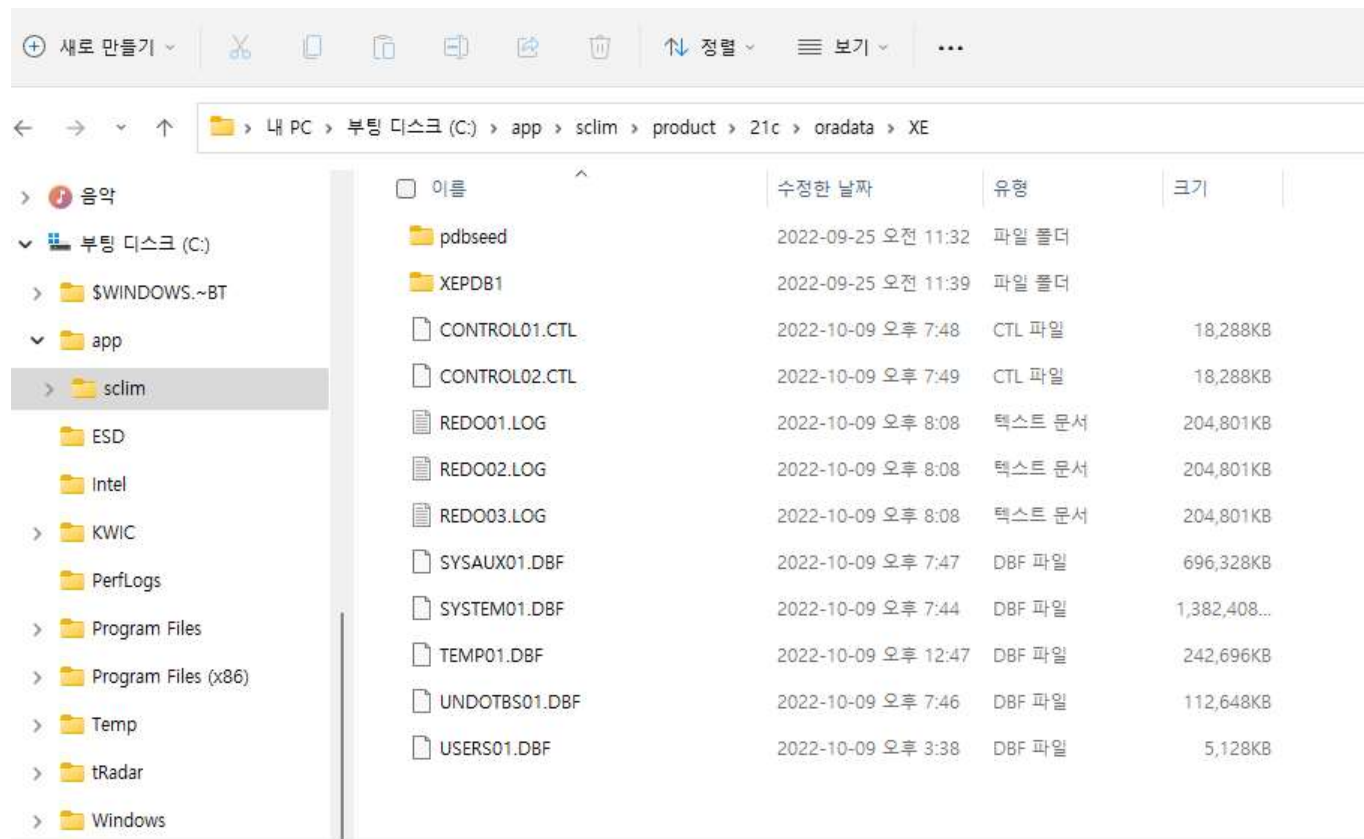
← → ↕ ↑ 📁 > 내 PC > 부팅 디스크 (C:) > app > sclim > product > 21c > dbhomeXE > bin

이름	수정한 날짜	유형	크기
rman.exe	2021-09-27 오후 4:00	응용 프로그램	3,615KB
sbttest.exe	2021-09-27 오후 4:00	응용 프로그램	85KB
schagent.exe	2021-09-27 오후 4:06	응용 프로그램	90KB
schema.exe	2021-07-09 오후 3:18	응용 프로그램	33KB
sclsspawn.exe	2021-09-23 오후 12:32	응용 프로그램	29KB
sql.exe	2021-06-23 오후 6:21	응용 프로그램	138KB
sqlldr.exe	2021-09-27 오후 3:59	응용 프로그램	1,536KB
<input checked="" type="checkbox"/> sqlplus.exe	2021-07-12 오전 1:23	응용 프로그램	1,358KB
tkprof.exe	2021-09-27 오후 3:59	응용 프로그램	26KB
tnlsnr.exe	2021-07-09 오후 12:44	응용 프로그램	939KB
tnsping.exe	2021-07-09 오후 12:44	응용 프로그램	73KB
uidrvci.exe	2021-09-27 오후 4:06	응용 프로그램	32KB

# Oracle Express Edition

- 주요 파일

- CONTROL.CTL, REDO.LOG: 시스템 관리와 복구를 위한 정보 저장
- UNDOTBS.DBF: 데이터 복구를 위한 정보 저장
- USERS.DBF: 사용자가 생성한 테이블(데이터) 저장

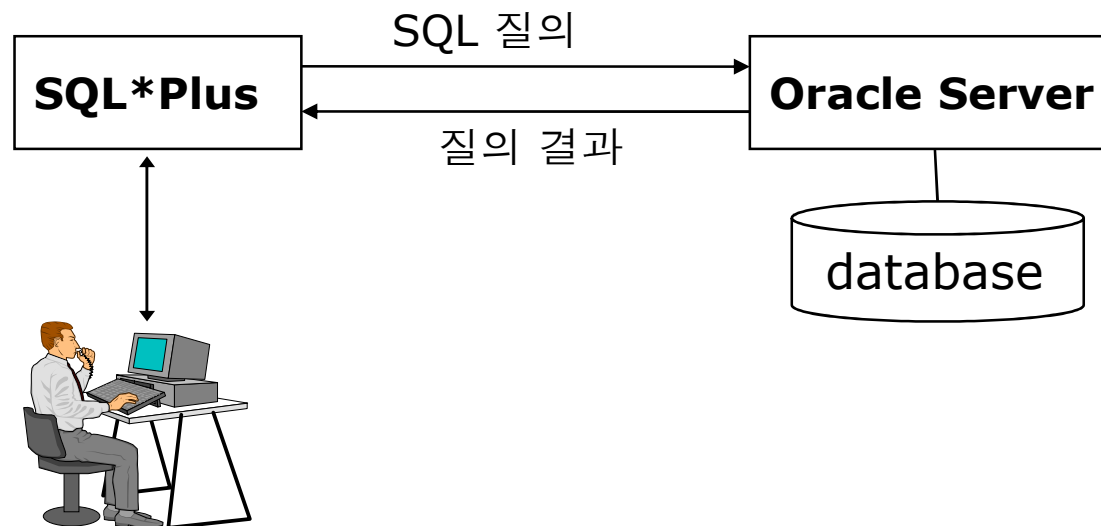


이름	수정된 날짜	유형	크기
pdbseed	2022-09-25 오전 11:32	파일 폴더	
XEPDB1	2022-09-25 오전 11:39	파일 폴더	
CONTROL01.CTL	2022-10-09 오후 7:48	CTL 파일	18,288KB
CONTROL02.CTL	2022-10-09 오후 7:49	CTL 파일	18,288KB
REDO01.LOG	2022-10-09 오후 8:08	텍스트 문서	204,801KB
REDO02.LOG	2022-10-09 오후 8:08	텍스트 문서	204,801KB
REDO03.LOG	2022-10-09 오후 8:08	텍스트 문서	204,801KB
SYSAUX01.DBF	2022-10-09 오후 7:47	DBF 파일	696,328KB
SYSTEM01.DBF	2022-10-09 오후 7:44	DBF 파일	1,382,408...
TEMP01.DBF	2022-10-09 오후 12:47	DBF 파일	242,696KB
UNDOTBS01.DBF	2022-10-09 오후 7:46	DBF 파일	112,648KB
USERS01.DBF	2022-10-09 오후 3:38	DBF 파일	5,128KB

# SQL\*Plus 개요

---

- Oracle DBMS에서 사용자가 SQL 질의를 작성 및 실행할 수 있도록 인터페이스를 제공하는 클라이언트 도구
  - Oracle Server에 접속하여 대화식으로 이용
    - ▶ local 뿐만 아니라 remote에 있는 서버에도 접근 가능
  - SQL문과 Oracle PL/SQL문을 수행하며, 자체 명령어를 제공
  - 주의: SQL은 데이터베이스를 접근하기 위해 사용되는 언어이고, SQL\*Plus는 SQL문과 PL/SQL 코드를 실행시키기 위한 도구임



# SQL\*Plus 개요

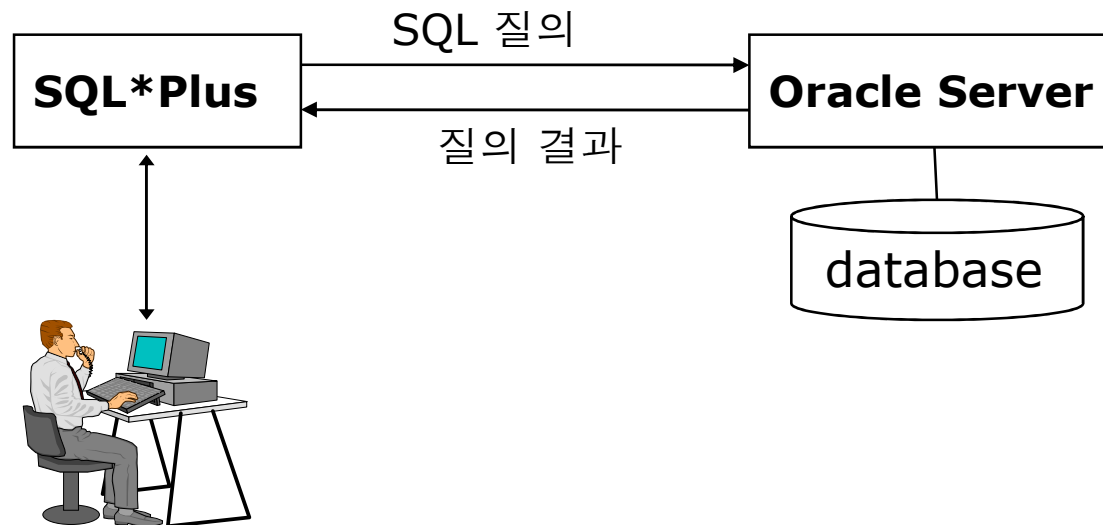
---

- 주요 기능
  - 데이터베이스 접속 및 종료
  - 테이블 생성, 편집, 저장 및 검색
  - SQL 문 또는 PL/SQL 프로그램 코드를 작성 및 실행 가능
  - 질의 결과를 정렬하고 지정된 형식으로 출력 (보고서 작성)
  - 데이터베이스 간의 데이터 복사 등 데이터베이스 관리 작업 수행

# SQL\*Plus 개요

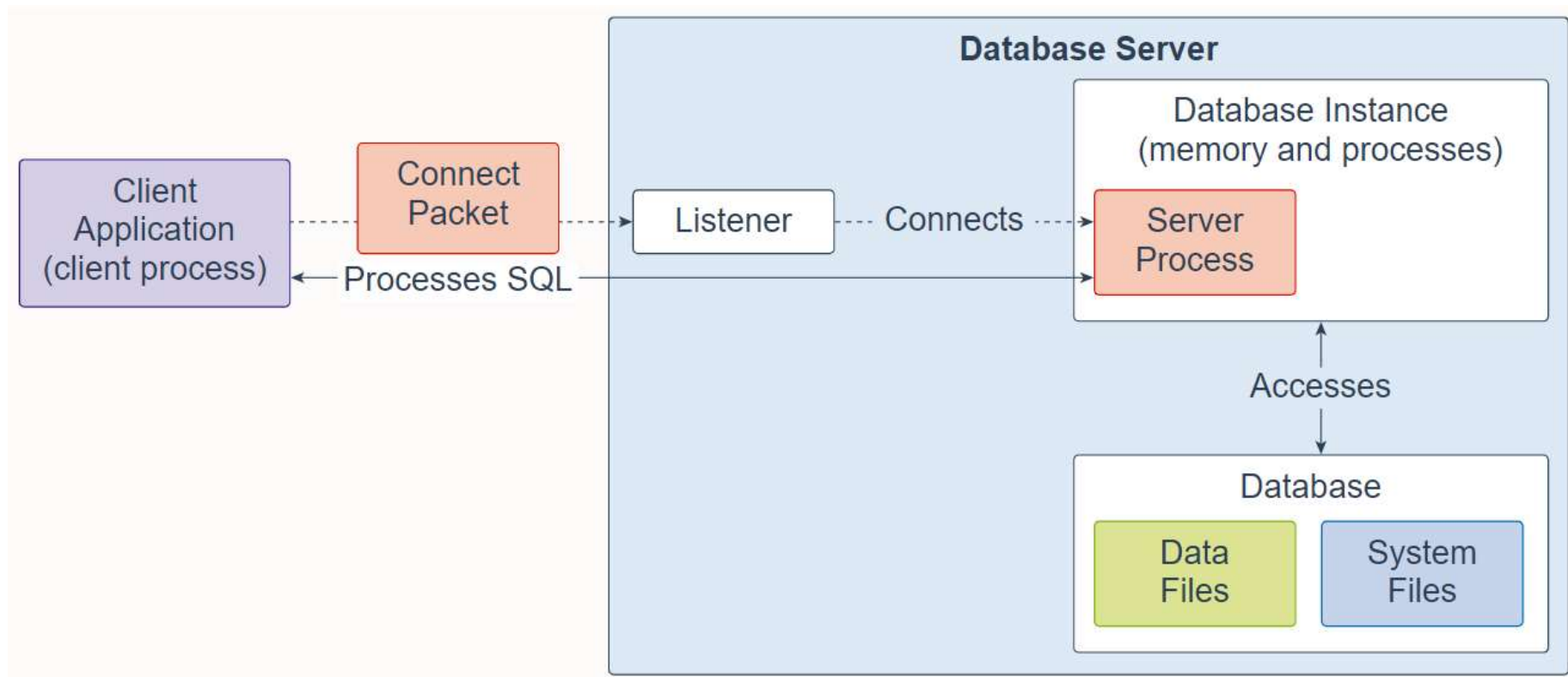
---

- Oracle DBMS에서 사용자가 SQL 질의를 작성하고 실행할 수 있도록 하는 콘솔 프로그램
  - Oracle Server에 접속하여 대화식으로 이용
    - ▶ local 뿐만 아니라 remote에 있는 서버에도 접근 가능
  - SQL문과 Oracle PL/SQL문을 수행하며, 자체 명령어를 제공



# Architecture of Oracle DB

- 우리가 사용하는 실습환경인 sqlplus는 아래 그림의 client app.에 해당





# Oracle SQL Developer

- 다운로드 및 설치
  - 검색: "oracle sql developer download"
  - Windows 64-bit with JDK 11 included 버전을 선택하여 다운로드
  - Zip 파일을 압축 해제 후 sqldeveloper.exe 실행



ORACLE Products Industries Resources Customers Partners Developers Events Company

Database / SQL Developer / SQL Developer Downloads

## SQL Developer 22.2.1 Downloads

Version 22.2.1.234.1810 - September 12, 2022

- [Release Notes](#)
- [Documentation](#)

Platform	Download	Notes
Windows 64-bit with JDK 11 included	 <a href="#">Download</a> (444 MB)	<ul style="list-style-type: none"><li>• MD5: c917657ae52a31af66b3cd16f100cf56</li><li>• SHA1: 9da21417bcd8a8fab4bba985ffd1d9087be16be</li><li>• <a href="#">Installation Notes</a></li></ul>
Windows 32-bit/64-bit	 <a href="#">Download</a> (490 MB)	<ul style="list-style-type: none"><li>• MD5: 92fe45220660c178ad8656c43ea6e9e2</li><li>• SHA1: b0a6e46b314bb5f976b42564c361d589cc2795a9</li><li>• <a href="#">Installation Notes</a></li><li>• <a href="#">JDK 11 required</a></li></ul>

# Oracle SQL Developer

- SQL 질의 작성 및 실행

The screenshot displays the Oracle SQL Developer interface with the following components and annotations:

- Left Panel (Tree View):** Shows the database structure for 'localhost.XEPDB1.scoot'. A red arrow points to the '실행' (Execute) button in the toolbar, labeled '2. 실행'.
- SQL Worksheet:** Contains the query: 

```
SELECT *  
FROM emp  
WHERE deptno = '10';
```

 A red arrow points to this area, labeled '1. 질의 작성(편집)'.
- SQL Results:** Displays the execution results in a table. A red arrow points to this area, labeled '3. 결과 확인'.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	7782 CLARK	MANAGER	7839	81/06/09	2450	(null)	10
2	7839 KING	PRESIDENT	(null)	81/11/17	5000	(null)	10
3	7934 MILLER	CLERK	7782	82/01/23	1300	(null)	10

Bottom status bar: | 3 행 21 열 | 삽입 | 수정됨 | Windows: CF

# Oracle SQL Developer

- 테이블 구조 및 데이터 확인

1. 테이블 선택

2. 열 선택

3. 데이터 선택

The first screenshot shows the Oracle SQL Developer interface with the 'EMP' table selected in the 'Table' tab. The table structure is displayed in the main window, showing columns: EMPNO, ENAME, JOB, and MGR. The second screenshot shows the same interface with the 'EMP' table selected in the 'Data' tab, displaying the table's data.

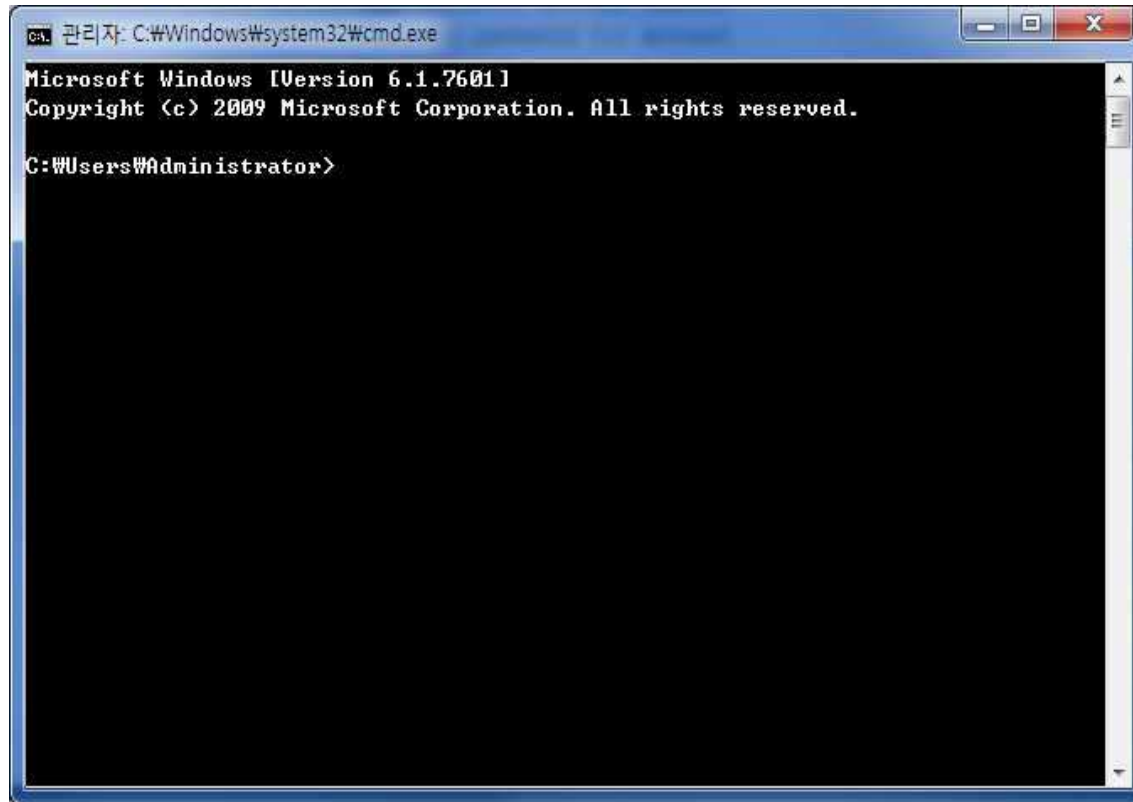
EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	7369 SMITH	CLERK	7902	80/12/17	800	(null)	20
2	7499 ALLEN	SALESMAN	7698	81/02/20	1600	300	30
3	7521 WARD	SALESMAN	7698	81/02/22	1250	500	30
4	7566 JONES	MANAGER	7839	81/04/02	2975	(null)	20
5	7654 MARTIN	SALESMAN	7698	81/09/28	1250	1400	30
6	7698 BLAKE	MANAGER	7839	81/05/01	2850	(null)	30
7	7782 CLARK	MANAGER	7839	81/06/09	2450	(null)	10
8	7839 KING	PRESIDENT	(null)	81/11/17	5000	(null)	10
9	7844 TURNER	SALESMAN	7698	81/09/08	1500	0	30
10	7900 JAMES	CLERK	7698	81/12/03	950	(null)	30
11	7902 FORD	ANALYST	7566	81/12/03	3000	(null)	20
12	7664 MILLER	CLERK	7782	81/01/13	1300	(null)	10

성채

# 사용자 계정의 잠금 해제 방법

---

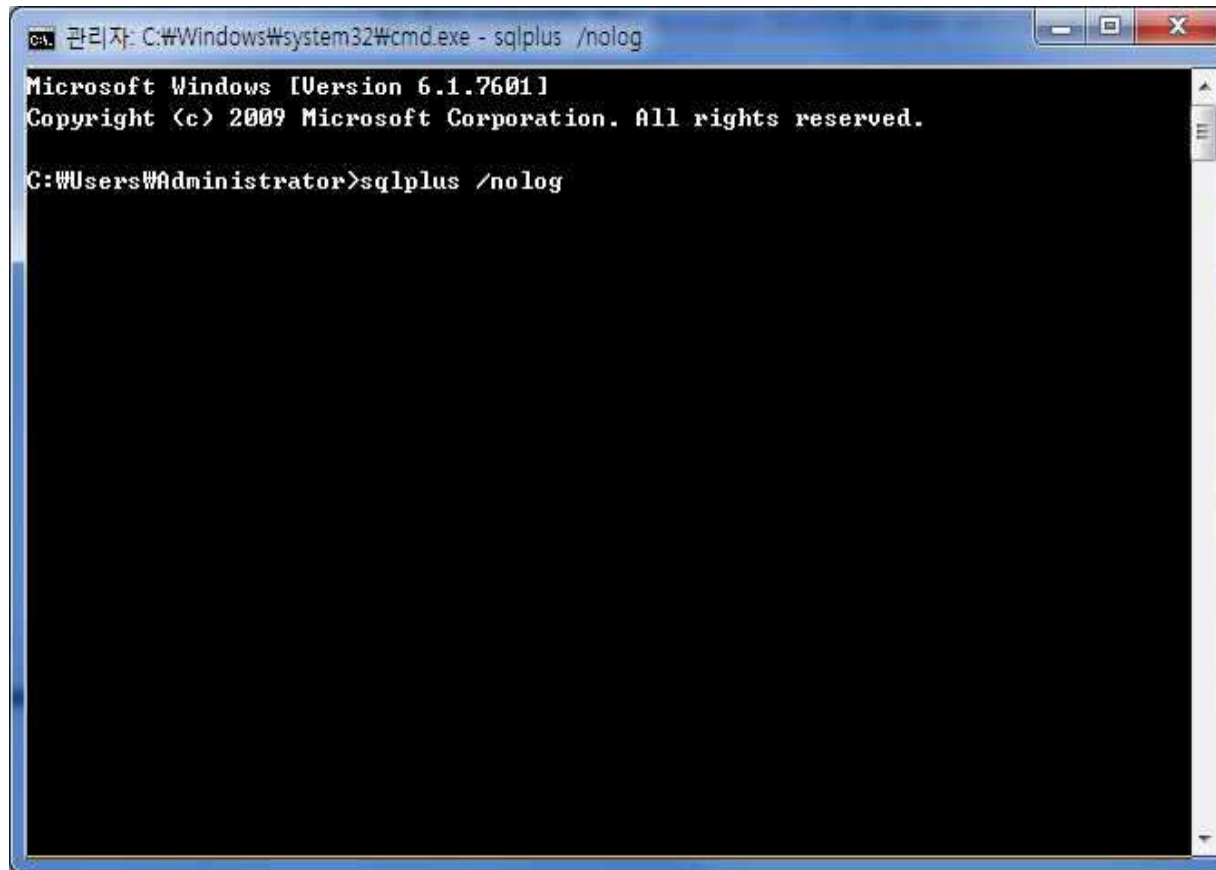
- 실행창에서 **cmd** 명령어를 입력해서 명령어 입력창을 실행



# 사용자 계정의 잠금 해제 방법

---

- `sqlplus /nolog` 명령으로 SQL\*Plus 를 실행



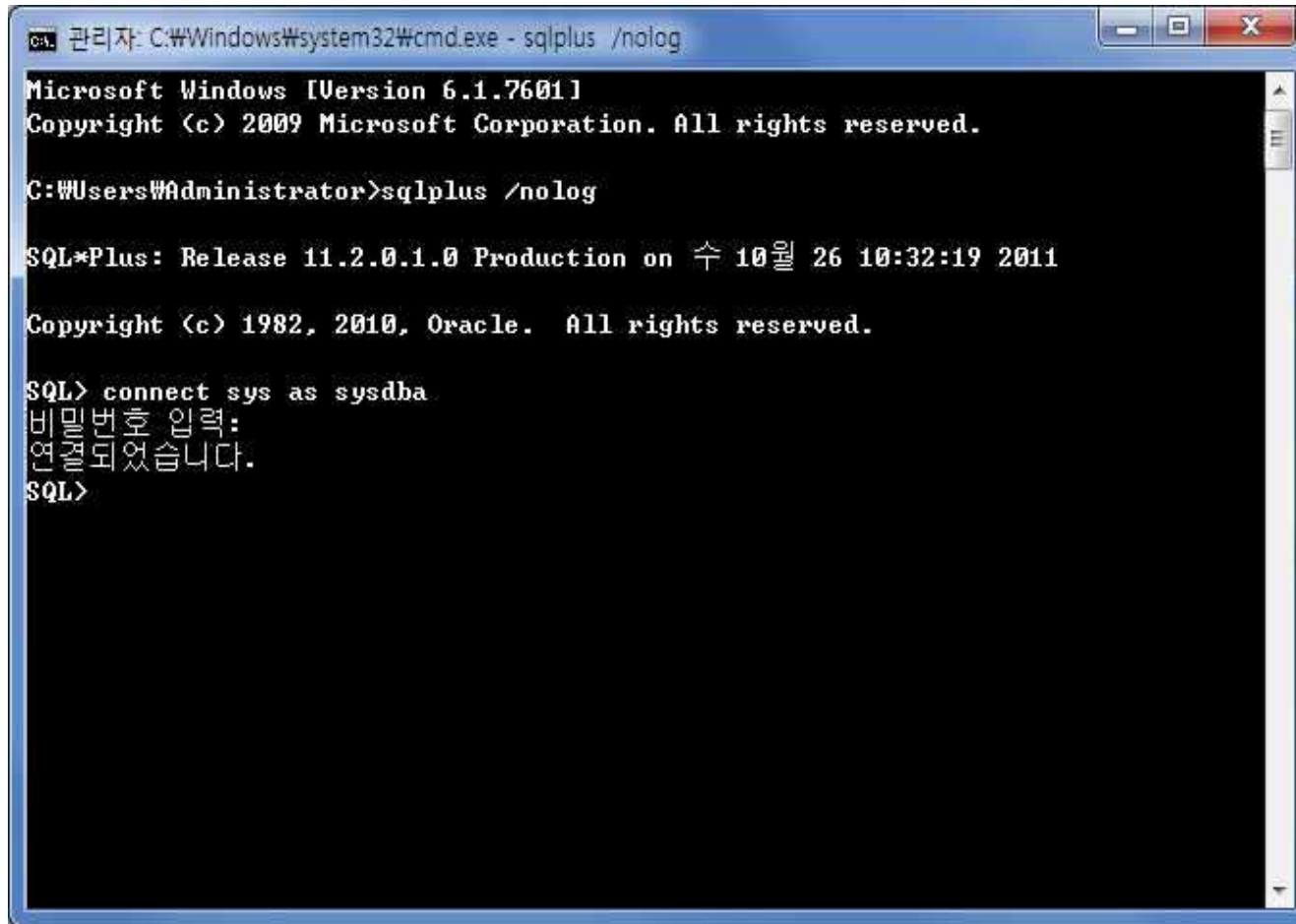
```
C:\> 관리자: C:\Windows\system32\cmd.exe - sqlplus /nolog

Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>sqlplus /nolog
```

# 사용자 계정의 잠금 해제 방법

- **connect sys as sysdba** 명령으로 시스템 관리자인 **sysdba** 계정으로 접속



```
관리자: C:\Windows\system32\cmd.exe - sqlplus /nolog
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>sqlplus /nolog

SQL*Plus: Release 11.2.0.1.0 Production on 수 10월 26 10:32:19 2011

Copyright (c) 1982, 2010, Oracle. All rights reserved.

SQL> connect sys as sysdba
비밀번호 입력:
연결되었습니다.
SQL>
```

## 접속의 시작

---

- **sqlplus /nolog** 명령으로 SQL\*Plus 를 실행
  - “/nolog” 옵션 사용시 계정 입력 화면이 나오지 않고 수행됨
- “**connect sys as sysdba**” 명령으로 시스템 관리자인 **sysdba** 계정으로 접속

```
C:\Users\Administrator>sqlplus /nolog

SQL*Plus: Release 11.2.0.1.0 Production on 수 10월 26 10:32:19 2011

Copyright (c) 1982, 2010, Oracle. All rights reserved.

SQL> connect sys as sysdba
비밀번호 입력:
연결되었습니다.
SQL>
```

# PL/SQL 실행 \*\* 이후 수업에서 다루겠음

---

- PL/SQL

- 오라클에서 DBMS의 표준 질의어인 SQL을 확장하여 개발한 고급 프로그래밍 언어
- SQL이 가진 편리함과 유연함에 '제어문', '반복문' 등과 같은 구조적 프로그래밍 언어의 요소가 결합
- 기본 단위는 블록(block)
  - ▶ 변수를 선언하는 부분인 **선언부**와 실행코드가 나오는 **실행부**, 실행 중 에러가 발생했을 때 실행되는 **예외처리부**로 구성



# PL/SQL 실행하기

---

- PL/SQL의 기본 형식

```
DECLARE
    변수 선언문;
BEGIN
    실행문;
EXCEPTION
    예외처리문;
END;
```

예) 변수 n을 생성하고 10을 저장한 후 출력

1	DECLARE
2	n INTEGER;
3	BEGIN
4	n := 10;
5	dbms_output.put_line(n);
6	END;

# 테이블스페이스 관리

---

- 테이블스페이스 생성
- 테이블스페이스 변경
- 테이블스페이스 조회

# 테이블 스페이스 생성

---

- 오라클에서 테이블을 생성하려면 테이블 스페이스를 사용해야 함
- 테이블 스페이스 사용
  - 오라클을 설치할 때 만들어지는 기본 테이블스페이스를 사용
  - 또는 새로운 테이블스페이스를 생성하여 사용
- 테이블 스페이스는 오라클 관리자만이 생성
- 생성할 때 실제 데이터가 저장될 디스크 상의 파일인 데이터파일을 지정

# 테이블스페이스 생성

---

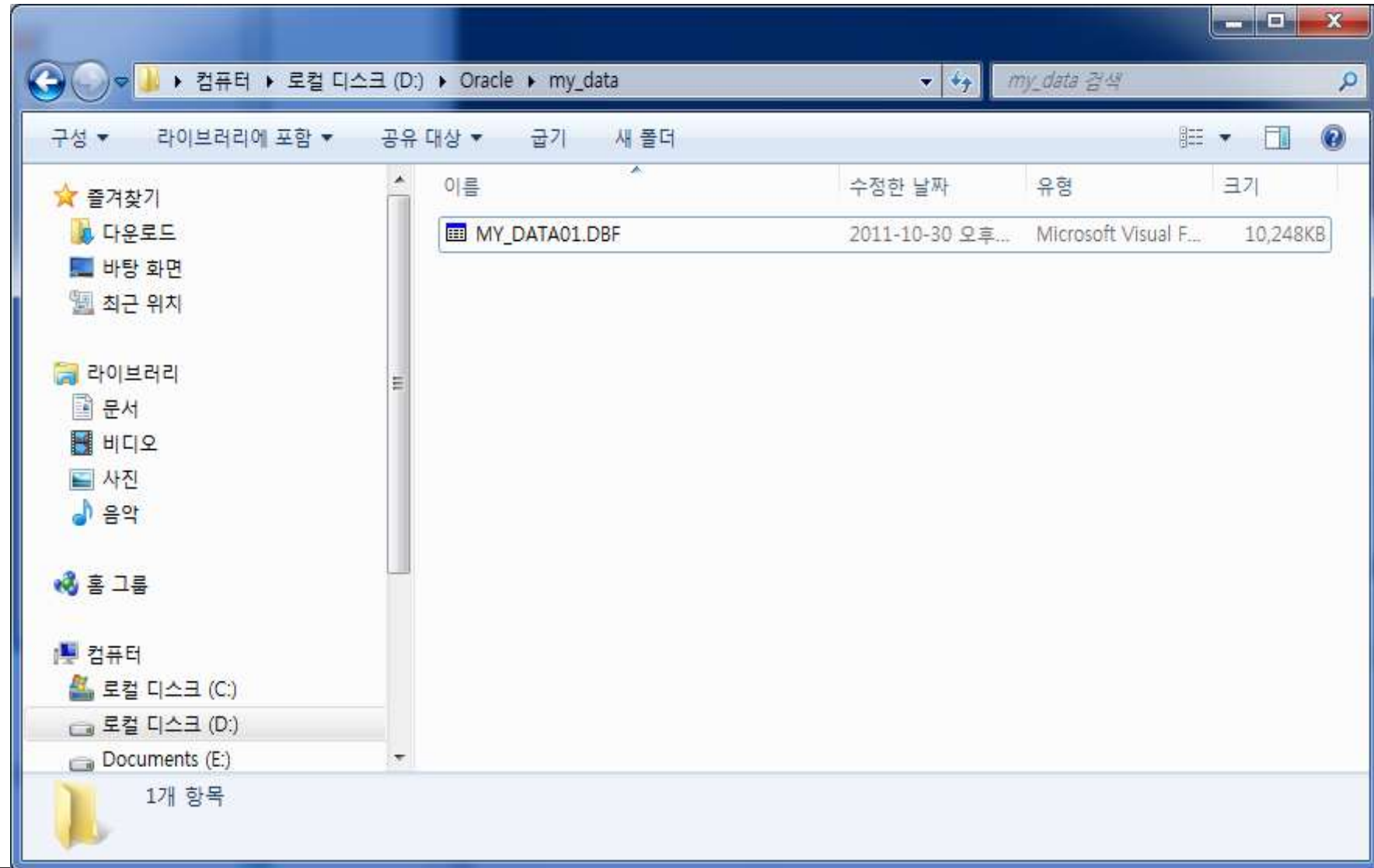
- 형식

**create tablespace** <테이블스페이스 이름>

**datafile** '<데이터파일 경로명>' **size** <데이터파일 크기>

```
SQL> create tablespace my_space  
2 datafile 'd:\#Oracle\my_data\my_data01.dbf' size 10M;  
  
테이블스페이스가 생성되었습니다.
```

## 생성된 데이터파일 확인



# 테이블스페이스 변경

---

- 테이블스페이스에 새로운 데이터파일을 추가
- 형식

**alter tablespace** <테이블스페이스 이름>

**add datafile** '<데이터파일 경로명>' **size** <데이터파일 크기>

```
SQL> alter tablespace my_space  
2 add datafile 'd:\#Oracle\my_data\my_data02.dbf' size 10M;
```

테이블스페이스가 변경되었습니다.

## 테이블스페이스 변경

---

- 테이블스페이스를 삭제할 때에는 **drop tablespace** 명령을 사용
- 형식

**drop tablespace** <삭제할 테이블스페이스 이름>

## 테이블스페이스의 사용

---

- 사용자 계정 생성
- 형식

**create user** <사용자 계정>

**identified by** <비밀번호>

**default tablespace** <사용할 테이블스페이스 이름>

**quota** <용량> **on** <사용할 테이블스페이스 이름>



## 테이블스페이스의 사용

---

무제한(unlimited)  
또는 20K, 5M

```
SQL> create user jimmy
  2  identified by jimmy_pass
  3  default tablespace my_space
  4  quota unlimited on my_space;
```

사용자가 생성되었습니다.

## 사용자 계정 scott 으로 접속 – 현재는 없음

---

SQL명령입력

(SQL을 실행하려면 반드시 ;으로 실행문을 끝내야 함)

```
SQL> select ename, sal from emp;
```

세미콜론 없이 enter를 치면 여러 줄로 입력 가능

```
SQL> select ename, sal  
2 from emp  
3 where sal > 10000;
```

## SQL\*Plus 명령어 실행

- 테이블 구조 보기 – **desc(ribe)**
  - 테이블에 어떤 필드들이 정의되어 있는지 확인
  - 형식
    - ▶ **desc** <테이블 명>

```
SQL> desc emp
이름                                길?   유형
-----
EMPNO                               NOT NULL NUMBER(4)
ENAME                               VARCHAR2(10)
JOB                                 VARCHAR2(9)
MGR                                 NUMBER(4)
HIREDATE                           DATE
SAL                                 NUMBER(7,2)
COMM                               NUMBER(7,2)
DEPTNO                             NUMBER(2)
```

# SQL\*Plus 명령어 실행

- 직전에 실행했던 명령문 보기 - **list**
  - 바로 직전에 실행시켰던 명령을 출력
  - 형식
    - ▶ list

```
SQL> select empno, ename, job, sal
2  from emp
3  where sal > 1500;
```

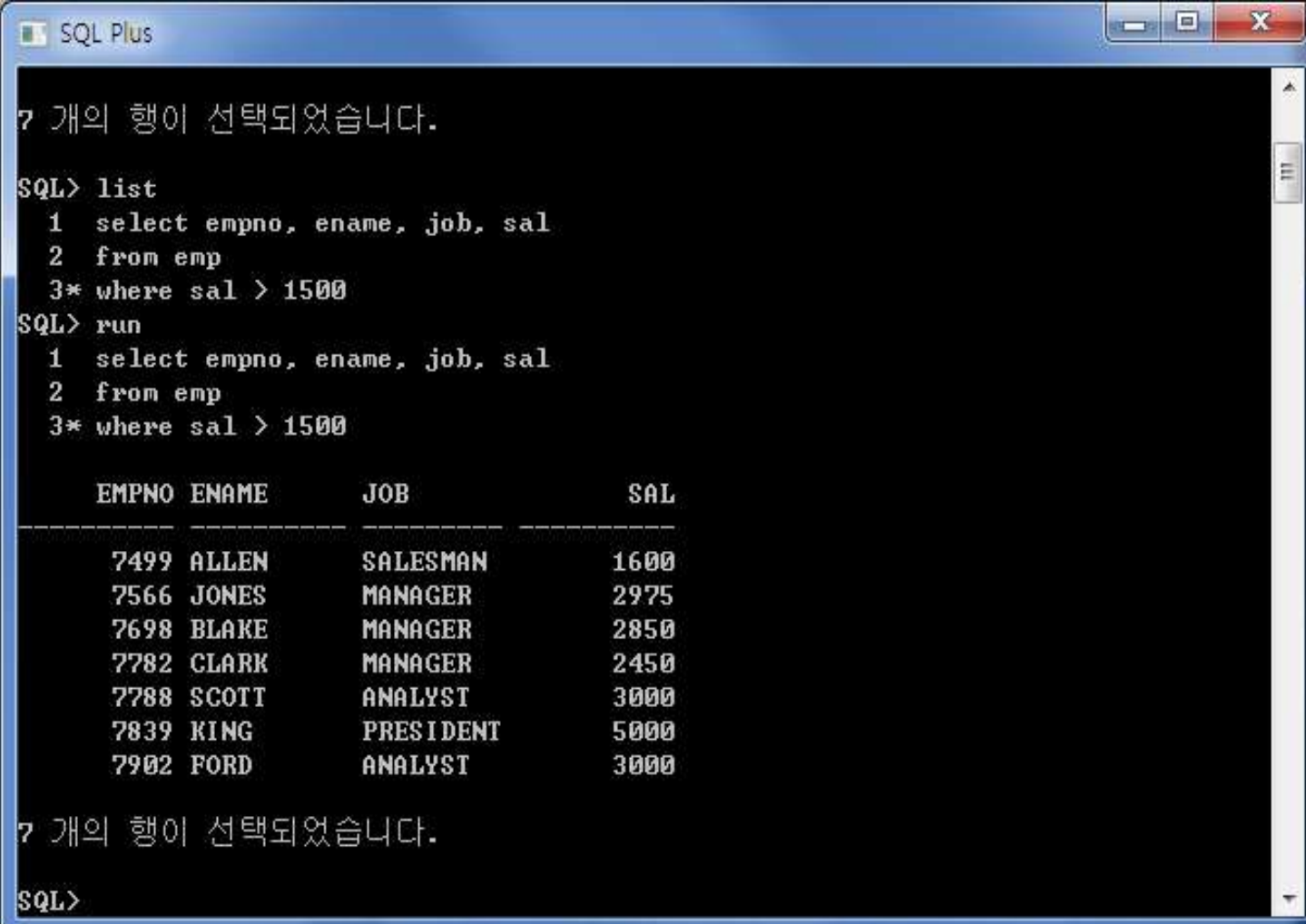
EMPNO	ENAME	JOB	SAL
7499	ALLEN	SALESMAN	1600
7566	JONES	MANAGER	2975
7698	BLAKE	MANAGER	2850
7782	CLARK	MANAGER	2450
7788	SCOTT	ANALYST	3000
7839	KING	PRESIDENT	5000
7902	FORD	ANALYST	3000

7 개의 행이 선택되었습니다.

```
SQL> list
1  select empno, ename, job, sal
2  from emp
3* where sal > 1500
```

# SQL\*Plus 명령어 실행

- 직전에 실행했던 명령문 다시 실행하기 – **run**
  - 직전에 실행했던 명령문을 다시 입력하지 않고 반복해서 실행
  - 형식
    - ▶ run



```
SQL Plus
7 개의 행이 선택되었습니다.

SQL> list
  1  select empno, ename, job, sal
  2  from emp
  3* where sal > 1500
SQL> run
  1  select empno, ename, job, sal
  2  from emp
  3* where sal > 1500

      EMPNO ENAME      JOB          SAL
-----
      7499 ALLEN      SALESMAN     1600
      7566 JONES      MANAGER     2975
      7698 BLAKE      MANAGER     2850
      7782 CLARK      MANAGER     2450
      7788 SCOTT      ANALYST     3000
      7839 KING      PRESIDENT   5000
      7902 FORD      ANALYST     3000

7 개의 행이 선택되었습니다.

SQL>
```

EMPNO	ENAME	JOB	SAL
7499	ALLEN	SALESMAN	1600
7566	JONES	MANAGER	2975
7698	BLAKE	MANAGER	2850
7782	CLARK	MANAGER	2450
7788	SCOTT	ANALYST	3000
7839	KING	PRESIDENT	5000
7902	FORD	ANALYST	3000

# SQL\*Plus 명령어 실행

- 직전에 실행했던 명령문 파일로 저장하기 - **save**
  - SQL\*Plus에서 실행시킨 명령문을 종류에 상관없이 파일로 저장
  - 형식
    - ▶ **save** <파일 이름>

```
SQL> select empno, ename, job, sal
2  from emp
3  where sal > 1500;
```

EMPNO	ENAME	JOB	SAL
7499	ALLEN	SALESMAN	1600
7566	JONES	MANAGER	2975
7698	BLAKE	MANAGER	2850
7782	CLARK	MANAGER	2450
7788	SCOTT	ANALYST	3000
7839	KING	PRESIDENT	5000
7902	FORD	ANALYST	3000

7 개의 행이 선택되었습니다.

```
SQL> save emp
file emp.sql<이>가 생성되었습니다
```

저장된 파일은 오라클  
시스템 폴더에 저장

C:\wappw[윈도우즈 사용자 이름]\product\11.2.0\wdhome\_1\BIN

# SQL\*Plus 명령어 실행

- 저장된 명령문 파일 불러오기 - **get**
  - **save**로 저장된 명령문을 불러올 때에는 **get** 명령어를 사용
  - 형식
    - ▶ **get** <파일 이름>

```
SQL> get emp
1 select empno, ename, job, sal
2 from emp
3* where sal > 1500
SQL> run
1 select empno, ename, job, sal
2 from emp
3* where sal > 1500
```

EMPNO	ENAME	JOB	SAL
7499	ALLEN	SALESMAN	1600
7566	JONES	MANAGER	2975
7698	BLAKE	MANAGER	2850
7782	CLARK	MANAGER	2450
7788	SCOTT	ANALYST	3000
7839	KING	PRESIDENT	5000
7902	FORD	ANALYST	3000

## SQL\*Plus 명령어 실행

---

- 운영체제 명령어 실행시키기 - **host**
  - SQL\*Plus 실행 도중 운영체제 명령을 실행
  - 형식
    - ▶ **host**
  - 복귀는 exit

```
SQL> host
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\wapp\Administrator\product\11.2.0\dbhome_1\BIN>exit

SQL> _
```



# SQL\*Plus 명령어 실행

---

- 기타 명령어
  - **connect** 계정명
    - ▶ 계정명으로 DB 연결(로그인)
    - ▶ conn으로 명령어 축약 가능
    - ▶ Sql> conn tiger
    - ▶ Sql> conn system
    - ▶ Sql> conn sys as sysdba
  - cl(ear) src
    - ▶ 화면 지우기
  - quit, exit
    - ▶ SQL\*Plus를 종료

# System 계정 사용 연습

---

- 앞서 수행한 sqlplus 콘솔에서 아래 명령어 입력
  - > conn(ect) system
  - System이란 Id로 DBMS에 연결(통신 연결을 통해 연결). 패스워드 입력
  - > disc(onnect)
  - 로그인 연결에서 빠져 나옴
- 앞에서 한 실습을 다시 한번 반복(암기해서...)
  - conn/disc
  - cl scr
  - host/exit
- System으로 로그인 한 후 아래 sql 문 수행
  - > **select** username **from** dba\_users;
  - 현재 이미 DBMS에 생성되어 있는 계정의 이름을 리스팅

# 계정 및 테이블의 생성

---

- 계정 생성하는 SQL 문
  - **주의**) system으로 로그인한 상태에서 수행 (로그인 상태 확인? "show user")
  - Sql> **create user sclim identified by 1;**
  - 계정 생성 확인:  
select username from dba\_users;
  - 위의 sql 문 수행의 결과 시스템에 sclim 계정이 생성
  - 계정이 생성되었지만 **로그인할 수 있는 권한 및 테이블을 생성할 수 있는 권한이 없음**
  - 이런 권한을 추가로 부여해야 함 (by system)

## 계정 및 테이블의 생성

---

- 로그인 할 수 있는 권한(네트워크 접속 즉, session 생성 권한) 부여
  - System 계정자가 아래 sql문 수행
  - Sql> grant connect to scim;
- 테이블을 생성할 수 있는 권한 주기
  - Sql> grant resource to scim;
- 보통은 아래처럼 한번에 권한을 부여함(권한 리스트 사용)
  - Sql> grant connect, resource to scim;

## 주의) 18c 버전부터는 아래처럼 계정 생성

---

- SQL> **create user** c##sclim identified by 1;
- SQL> **grant** connect, resource to c##sclim ; // 권한 부여
- SQL> conn c##sclim ; // 로그인
  
- SQL> **drop user** c##sclim ; // 계정 삭제 시
  
- **C##을 붙이고 싶지 않다면**
  - SQL> ALTER SESSION SET "\_ORACLE\_SCRIPT"=true;

# 계정 및 테이블의 생성

---

- 테이블을 만들어 보자 (**주의!!!!** Sclim으로 로그인한 상태에서 수행)
  - **SQL> create table test\_table ( name varchar2(10), age int );**
  - 위의 문에서 붉은색 표시된 이름이 바로 필드 명
  - 위의 sql 문을 일반 사용자가 수행하도록 한다.
  - **절대 system(or sys)로는 수행하지 않는다**
- 실습 중에 내가 어떤 계정 상태인지 헛갈림. 그렇다면 아래와 같이 입력
  - ▶ SQL> show user
  - ▶ 현재 어떤 계정으로 접속 중인지 확인 바람
  - ▶ 질문) **위의 입력에서 ';' 없어도 되나?**

## 실습 종료

---

- 실습이 끝나면 자신이 만든 계정은 삭제 --- **system 계정에서 수행해야 함**
  - SQL> conn system (계정 변경)
  - System> **drop user** 생성한-계정 **cascade**;
  - 생성 계정을 삭제
  - 뒤에 붙이는 cascade 키워드를 통해 생성된 테이블과 색인이 함께 삭제됨
- 삭제가 되었는지 확인
  - System> select username from dba\_users;
  - 읊

## 실습 과제 \*\*

---

- 계정 ex\_db을 생성한다. 그리고 이 계정에 **적절히 권한을 부여한다** (로그인 연결 및 테이블 생성 가능하도록)
- 다시 ex\_db **계정으로 로그인**
- 그리고 ex\_db의 권한으로 아래와 같은 구조의 레코드(구조체)와 유사한 필드를 갖는 테이블 person를 생성해 보자( "show user"를 통해 사용자 확인)

```
struct person {  
    char name[10];  
    int age;  
    char gender;  
    char address[30];  
};
```

- **계정 생성, 권한 부여, 그리고 테이블 생성하는 과정을 화면 캡처하여 제출한다**



# 실습 2

# 오라클 데이터 구조

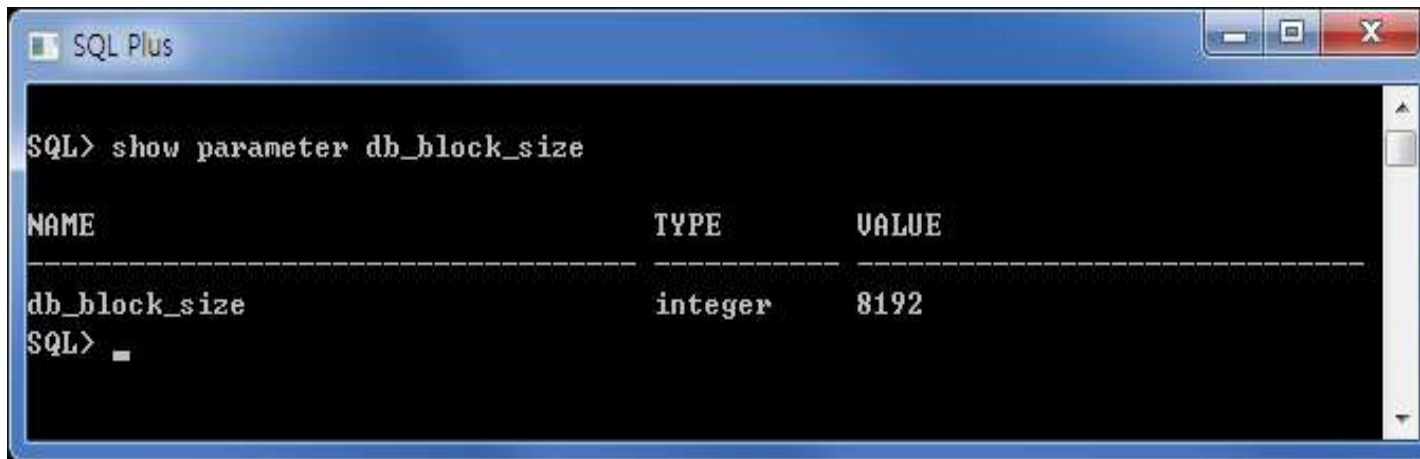
---

- 논리적 구성요소
  - 데이터 블록(data block)
  - 익스텐트(extent)
  - 세그먼트(segment)
  - 테이블스페이스(tablespace)

## 데이터 블록 - 논리적 구성요소

---

- 데이터가 저장되는 가장 작은 단위
- 저장해야 할 데이터가 늘어나면 데이터 블록의 배수로 저장 공간을 확보하여 저장
- 데이터 블록 표준 크기는 **db\_block\_size**라는 설정 값에 저장
- 블록 크기 확인 명령
  - **show parameter db\_block\_size**



```
SQL> show parameter db_block_size
```

NAME	TYPE	VALUE
db_block_size	integer	8192

```
SQL> _
```

## 익스텐트(extent)- 논리적 구성요소

---

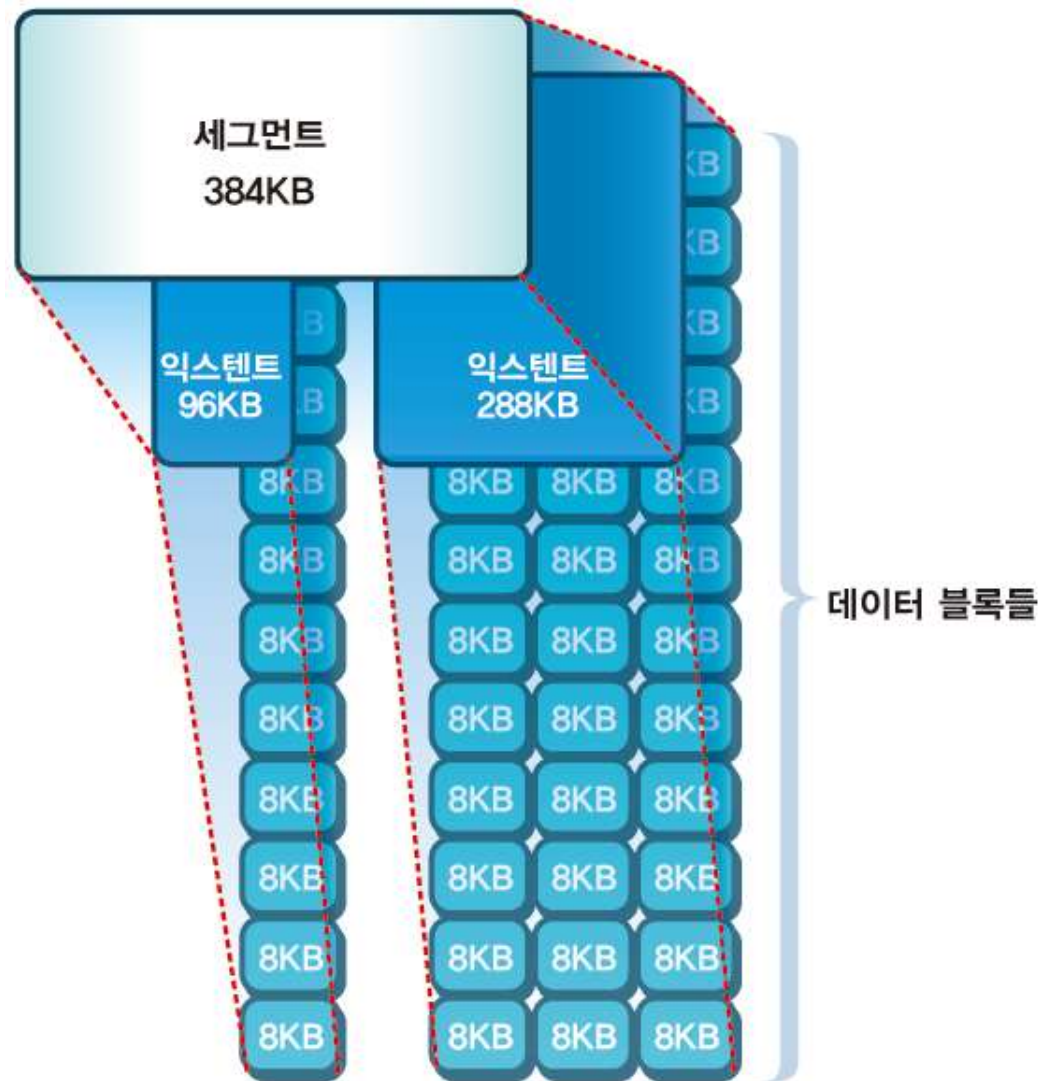
- 데이터 블록 다음 단계의 논리적 데이터 저장 공간
- 연속적인 여러 개의 데이터 블록이 모여서 하나의 익스텐트를 구성
- 익스텐트가 모여 다음에 설명할 세그먼트를 구성
  - 하나의 세그먼트에 할당된 공간이 모두 사용되면 오라클은 새로운 익스텐트를 만들어 그 세그먼트에 할당

## 세그먼트(segment)- 논리적 구성요소

---

- 여러 개의 익스텐트들이 모여 하나의 세그먼트를 구성
- 하나의 세그먼트에는 같은 종류의 데이터가 저장
  - 데이터 세그먼트
    - ▶ 테이블이 저장되는 세그먼트
  - 인덱스 세그먼트
    - ▶ 인덱스(index) 정보가 저장되는 세그먼트
- 하나의 세그먼트는 뒤에 설명할 하나의 테이블스페이스에 저장
- 하나의 세그먼트를 구성하는 익스텐트들은 디스크상에 연속적으로 저장되지 않을 수도 있음

# 데이터 블록, 익스텐트, 세그먼트의 관계



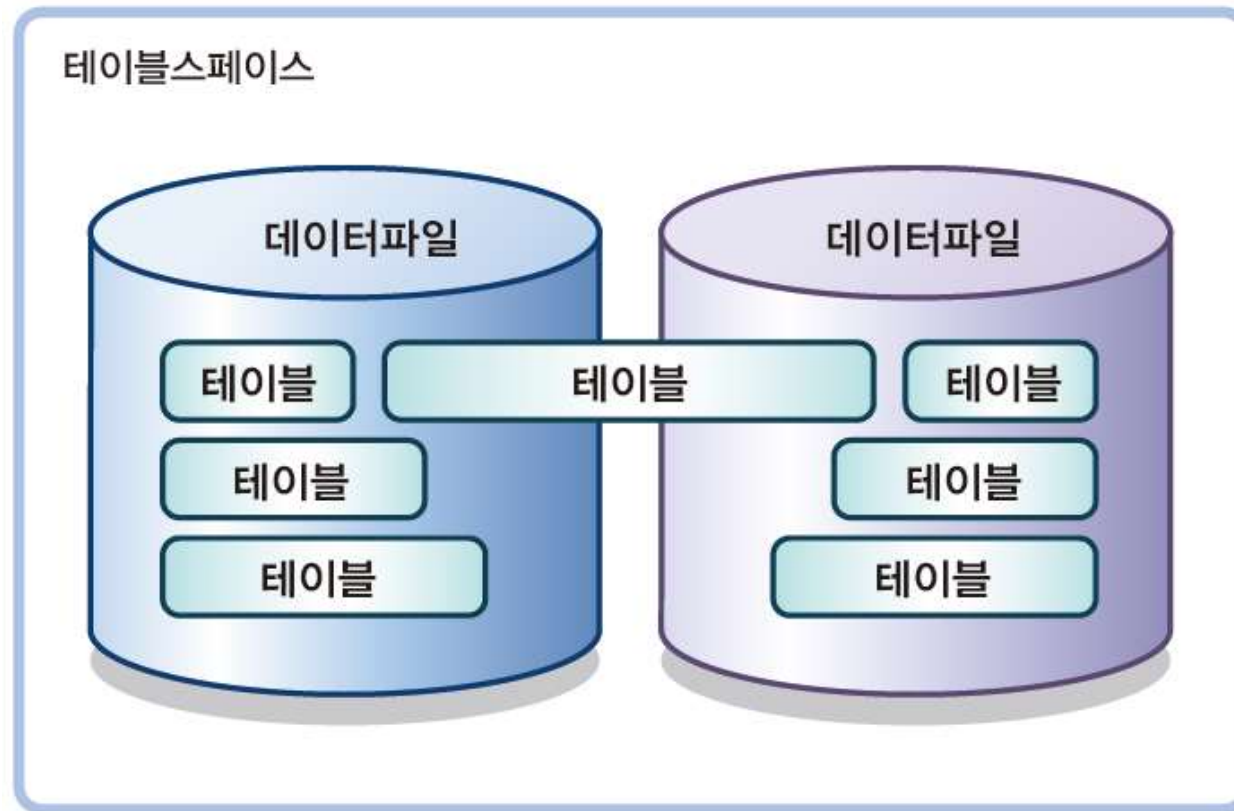
## 테이블 스페이스(table space)- 논리적 구성요소

---

- 하나의 데이터베이스는 오라클의 논리적 저장 단위인 테이블 스페이스들로 구성
- 하나의 테이블스페이스에는 하나 이상의 세그먼트를 포함

## 데이터파일(datafile)- 물리적 구성요소

- 오라클에서 관리하는 데이터가 실제로 저장되는 디스크 상의 파일

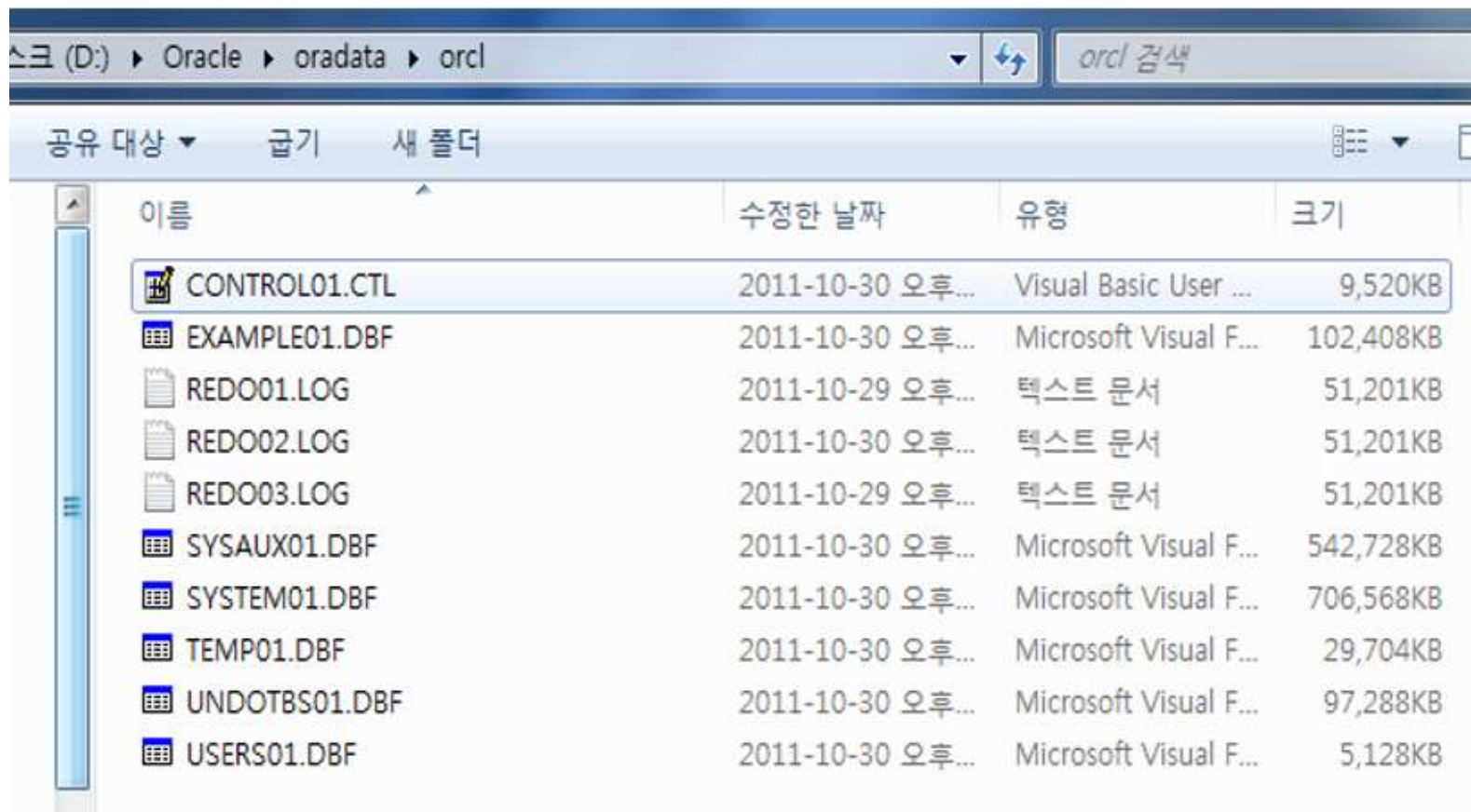


데이터파일과 테이블스페이스의 관계



## 데이터파일(datafile)- 물리적 구성요소

- 오라클에서 관리하는 데이터가 실제로 저장되는 디스크 상의 파일
- **oradata** 폴더의 데이터파일들



이름	수정한 날짜	유형	크기
CONTROL01.CTL	2011-10-30 오후...	Visual Basic User ...	9,520KB
EXAMPLE01.DBF	2011-10-30 오후...	Microsoft Visual F...	102,408KB
REDO01.LOG	2011-10-29 오후...	텍스트 문서	51,201KB
REDO02.LOG	2011-10-30 오후...	텍스트 문서	51,201KB
REDO03.LOG	2011-10-29 오후...	텍스트 문서	51,201KB
SYSAUX01.DBF	2011-10-30 오후...	Microsoft Visual F...	542,728KB
SYSTEM01.DBF	2011-10-30 오후...	Microsoft Visual F...	706,568KB
TEMP01.DBF	2011-10-30 오후...	Microsoft Visual F...	29,704KB
UNDOTBS01.DBF	2011-10-30 오후...	Microsoft Visual F...	97,288KB
USERS01.DBF	2011-10-30 오후...	Microsoft Visual F...	5,128KB

# 데이터파일(datafile)- 물리적 구성요소

---

## **SYSAUX01.DBF, SYSTEM01.DBF**

오라클 시스템 관리를 위해 만들어진 데이터파일

## **TEMP01.DBF**

임시 데이터들을 저장하기 위한 데이터파일

## **USER01.DBF**

사용자 계정을 위해 만들어진 데이터파일

## **EXAMPLE01.DBF**

예제 테이블들을 저장하고 있는 데이터파일

## **UNDOTBS01.DBF**

데이터에 문제가 발생했을 때 복구를 위한 정보

## 기타 물리적 구성요소

---

- 컨트롤 파일(control file)
  - 데이터베이스의 물리적 구조, 데이터베이스 이름, redo 로그 파일들의 위치 정보, 데이터베이스 생성 시간, 현재 로그 번호, 체크포인트 정보 등이 저장
- Redo 로그 파일
  - 데이터베이스의 변경 내역을 저장하는 파일
  - 데이터 변경 과정에서 장애가 발생하여 변경내용이 데이터베이스에 반영되지 못했을 경우 온라인 redo 로그 파일을 이용하여 복구
- 설정 파일(parameter file)
  - 데이터베이스와 데이터베이스 서버와 관련된 설정 정보들이 저장
- alert/trace 로그 파일
  - 오라클 서버 내부에서 오류가 발생할 경우 그 오류에 대한 정보나 메시지를 저장하는 파일

# SQL 문의 연습

---

- 오라클 SQL 문을 공부한다

# 질의어와 SQL

---

- SQL: Structured Query Language
- 1974년 IBM의 System R project에서 개발된 Sequel이란 언어에 기초
- 표준 질의어로 채택되어 널리 쓰이는 관계형 질의언어
  - 1986년 ANSI와 ISO에서 표준 질의어로 채택
  - 1992년 SQL2(SQL-92) 발표
  - 2003년 SQL3발표 (최신)
- 관계 대수나 관계 해석은 이론적 배경을 제공하나 상용으로 쓰이기에는 어렵고 적절치 않음
  - SQL은 자연어와 유사하고 상대적으로 비절차적 언어 특성이 있음
  - 습득과 이용이 용이함

# SQL의 구성: DDL & DML

---

- SQL은 크게 DDL과 DML로 구성됨
- 데이터 정의 언어 (DDL: Data Definition Language)
  - 데이터 저장 구조를 명시하는 언어
  - 테이블 스키마의 정의, 수정, 삭제
- 데이터 조작 언어 (DML: Data Manipulation Language)
  - 사용자가 데이터를 접근하고 조작할 수 있게 하는 언어
  - 레코드의 검색(search), 삽입(insert), 삭제(delete), 수정(update)

# 데이터 정의 언어

---

- 테이블 생성 (create table)
- 기본키, 외래키 설정
- 테이블 삭제 (drop table)
- 테이블 수정 (alter table)

# 데이터 정의 언어

- 종류
  - 테이블 생성
  - 테이블 삭제
  - 테이블 수정
- 필드의 Data type 종류

분류	표준 SQL	오라클	설명
문자	char(n)	char(n)	길이가 n byte인 고정길이 문자열 오라클의 경우 최대 2000byte까지 지정 가능
	varchar(n)	varchar2(n)	최대 길이가 n byte인 가변길이 문자열 오라클의 경우 최대 4000byte까지 지정 가능
숫자	int	int	정수형
	float	float	부동 소수
날짜 시간	date	date	년, 월, 일을 갖는 날짜형 오라클의 경우 날짜의 기본 형식은 'yy/mm/dd'이다.
	time timestamp	timestamp	년, 월, 일, 시, 분, 초를 갖는 날짜시간형



# 테이블 생성

---

- 형식

```
create table <테이블이름> (<필드리스트>)
```

- ▶ <필드리스트>는 '필드명 데이터타입'

- department 테이블을 생성하는 SQL문

(질의 1)

```
create table department
(
    dept_id          varchar2(10)    not null,
    dept_name        varchar2(14)    not null,
    office           varchar2(10)
)
```

- ▶ 키워드 **not null**은 해당 필드에 널을 허용하지 않는다는 것을 의미함

# 기본키, 외래키 설정

---

- 테이블을 생성할 기본키 역할을 하는 필드를 지정

(질의 2)

```
create table department
(
    dept_id          varchar2(10) ,
    dept_name        varchar2(20) not null,
    office           varchar2(20),
    constraint pk_department primary key(dept_id)
)
```

## 테이블 생성(student table)

---

- not null과 기본키를 지정한 student 테이블 생성 예

(질의 3)

```
create table student
(
    stu_id                varchar2(10),
    resident_id           varchar2(14) not null,
    name                  varchar2(10) not null,
    year                  int,
    address               varchar2(10),
    dept_id               varchar2(10),
    constraint pk_student primary key(stu_id)
)
```

## 테이블 생성(student table)

---

- 외래키까지 포함된 student 테이블 생성 예

(질의 3)

```
create table student
(
    stu_id                varchar2(10),
    resident_id           varchar2(14) not null ,
    name                  varchar2(10) not null,
    year                  int,
    address               varchar2(10),
    dept_id               varcahr2(10),
    constraint pk_student primary key(stu_id),
    constraint fk_student foreign key(dept_id) references
                        department(dept_id)
)
```

# 테이블 생성

---

- professor 테이블

(질의 5)

```
create table professor
```

```
(
```

```
    prof_id          varchar2(10) ,  
    resident_id      varchar2(14)   not null,  
    name             varchar2(10)   not null,  
    dept_id          varchar2(10),  
    position         varchar2(10),  
    year_emp         int,  
    constraint       pk_professor    primary key(prof_id),  
    constraint       fk_professor    foreign key(dept_id)  
                                references department(dept_id)
```

```
)
```

# 테이블 생성

---

- course 테이블

(질의 6)

```
create table course
```

```
(
```

```
    course_id
```

```
varchar2(10) ,
```

```
    title
```

```
varchar2(14)
```

```
not null,
```

```
    credit
```

```
int,
```

```
constraint
```

```
    pk_course
```

```
primary key(course_id)
```

```
)
```

# 테이블 생성

- class 테이블

(질의 7)

```
create table class
(
    class_id          varchar2(10) ,
    course_id         varchar2(10),
    year              int,
    semester           int,
    division           char(1),
    prof_id            varchar2(10),
    classroom          varchar2(9),
    enroll             int,
    constraint pk_class primary key(class_id),
    constraint fk_class1 foreign key(course_id)
                        references course(course_id),
    constraint fk_class2 foreign key(prof_id)
                        references professor(prof_id)
)
```

# 테이블 생성

---

- takes 테이블

(질의 8)

**create table takes**

(

stu\_id

**varchar2(10) ,**

class\_id

**varchar2(10),**

grade

**char(5),**

**constraint**

pk\_takes

**primary key**(stu\_id, class\_id),

**constraint**

fk\_takes1

**foreign key**(stu\_id)

**constraint**

**references**

student(stu\_id),

fk\_takes2

**foreign key**(class\_id)

**references**

class(class\_id)

)



# 기본키, 외래키 관련 사항

---

- 기본키
  - 테이블에는 한 개의 기본키 생성 가능(기본키 생성이 필수는 아님)
  - 기본키 : 하나 혹은 2개 이상의 필드에 대해 생성 가능
    - ▶ 보통 하나의 필드로 기본키 생성
- 기본키 생성에 따른 제약(constraint) 사항
  - 기본키 값은 반드시 서로 달라야 함 (즉, unique해야 함)
  - 따라서, 기본키 값으로 각 레코드를 유일하게 식별할 수 있음
  - 예. 주민번호. 주민번호가 같은 사람을 존재할 수 없음. 데이터 무결성과 연관을 가짐 있음

# 기본키, 외래키 관련 사항

---

- 외래키
  - 두 개의 서로 다른 테이블 사이에 존재하는 의미성을 표현하기 위해 사용됨
- (예) 두 개의 Table T1, T2가 있다고 가정하자
  - T1에 존재하는 필드 Fx(즉, T1.Fx)를 T1의 기본키라고 가정하자
  - T2를 생성할 때, T1.Fx를 참조하는 외래키를 생성할 수 있음
  - T2.Fy를 외래키로 정의:  $T2.Fy \rightarrow T1.Fx$  (참조 관계)
  - 위와 같은 외래키가 존재한다면, T2에 새로운 레코드 R을 삽입하려 한다면, R.Fy의 값과 같은 T1.Fx 값에 이미 존재하고 있어야 함
    - ▶ 학생.dept\_id 와 같은 학과.dept\_id 값이 있어야...
  - 또한, T1에서 레코드 R을 삭제할 때, R.Fx 값을 Fy 필드값으로 가진 레코드가 T2에 존재한다면 외래키 제약 위반으로 인해 레코드 삭제 안됨

## 테이블 생성: 외래키/기본키

---

```
create table dept (  
    dept_id int primary key,  
    name varchar2(20)  
);
```

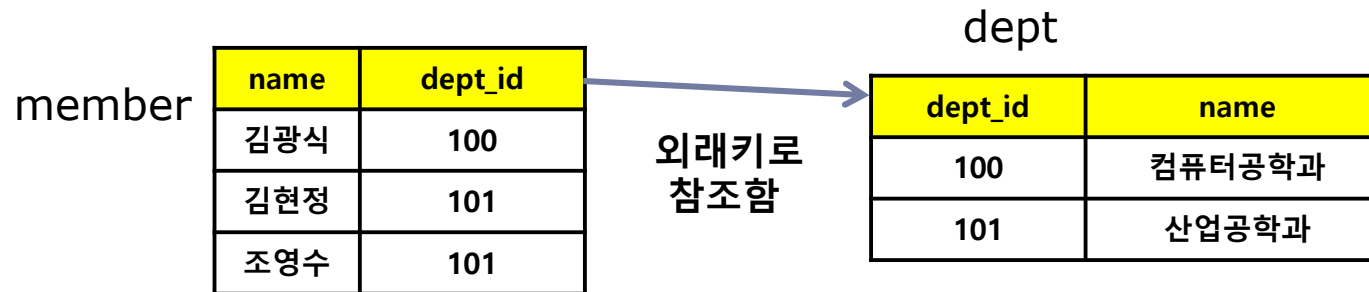
```
create table member (  
    person_name varchar2(20),  
    dept_id int references dept(dept_id)  
);
```

테이블 생성시: **member** 테이블을 생성하기 전에, **dept** 테이블이 이미 존재하고 있어야 함.

**\*\*** 외래키로 인해 참조 관계가 있는 테이블 간에는 생성/삭제 순서에 대한 제약 사항이 발생

## 기본키, 외래키 관련 주의사항

- 예) (주의: 필드명이 반드시 일치할 필요는 없음)



- 만일 department 테이블이 존재하지 않는 상태에서 student 테이블을 생성하려 한다면 오류발생
- member 테이블이 존재하는 동안에는 dept 테이블을 삭제할 수 없음
  - ▶ dept 테이블을 삭제하려면 member 테이블을 먼저 삭제하던지, 외래키 제약조건을 삭제해야 함

# 테이블 삭제

---

- 형식

```
drop table <테이블 이름> ;
```

- 주의

- 다른 테이블에서 외래키로 참조되는 경우에는 삭제할 수 없음
- 예)
  - ▶ class 테이블은 takes 테이블에서 외래키로 참조됨

## 테이블 수정

---

- 기존 테이블에 새로운 필드를 추가하거나 기존 필드를 삭제
- 필드 추가 형식

```
alter table <테이블이름> add <추가할 필드>
```

- 예) student 테이블에 age 필드를 추가

(질의 9)

```
alter table student  
add age int ;
```

# 테이블 수정

---

- 필드 삭제 형식

```
alter table <테이블이름> drop column <삭제할 필드> ;
```

- 예)

(질의 10)

```
alter table student  
drop column age;
```

# 데이터 조작 언어(DML)

---

- 레코드 삽입
- 레코드 수정
- 레코드 삭제
- 레코드 검색



# 레코드 삽입

---

- 형식

**insert into** <테이블이름> (<필드리스트>) **values** (<값 리스트>)

- <필드리스트>
  - ▶ 삽입에 사용될 테이블의 필드들
- <값 리스트>
  - ▶ <필드리스트>의 순서에 맞춰 삽입될 값
- <필드 리스트>에 나열되지 않은 필드에 대해서는 널 값이 입력됨
- <필드 리스트>를 생략할 경우 <값 리스트>에는 테이블에 존재하는 **필드의 순서에 맞춰서 값을** 나열해 줘야 함

## 18c 버전에서의 차이

---

- 테이블 생성 후 레코드 삽입
  - Insert 문 실행 시 다음 같은 오류 발생
  - ORA-01950 : 테이블스페이스 'USERS'에 대한 권한이 없습니다.
  - 이전 버전에서는 이런 오류 발생하지 않았음.
  - 18c 버전에서는 레코드가 삽입될 저장 공간(table space)에 대한 사용권한을 따로 부여해줘야 한다
- System 계정에서 아래처럼 권한을 부여해 줘야 함
  - SQL> alter user c##cs default tablespace users quota unlimited on users;

## 레코드 삽입

---

(질의 11)

```
insert into department (dept_id, dept_name, office)  
values ('920', '컴퓨터공학과', '201호') ;
```

## 학사 데이터베이스의 데이터 삽입 예

---

```
insert into department values('920', '컴퓨터공학과', '201호')
```

```
insert into department values('923', '산업공학과', '207호')
```

```
insert into department values('925', '전자공학과', '308호')
```

```
insert into student
```

```
    values('1292001', '900424-1825409', '김광식', 3, '서울', 920)
```

```
insert into student
```

```
    values('1292002', '900305-1730021', '김정현', 3, '서울', 920)
```

```
insert into student
```

```
    values('1292003', '891021-2308302', '김현정', 4, '대전', 920)
```

```
insert into student
```

```
    values('1292301', '890902-2704012', '김현정', 2, '대구', 923)
```

```
insert into student
```

```
    values('1292303', '910715-1524390', '박광수', 3, '광주', 923)
```

```
insert into student
```

```
    values('1292305', '921011-1809003', '김우주', 4, '부산', 923)
```

```
insert into student
```

```
    values('1292501', '900825-1506390', '박철수', 3, '대전', 925)
```

```
insert into student
```

```
    values('1292502', '911011-1809003', '백태성', 3, '서울', 925)
```

## 레코드 삽입

---

- 필드 이름을 나열한다면, 그 순서는 테이블을 생성할 때 지정한 순서와 일치할 필요 없음
- 예) (질의 11)과 동일한 SQL

### (질의 12)

```
insert into department (office, dept_id, dept_name)
values ('201호', '920', '컴퓨터공학과')
```

- department 테이블의 필드들 중에서 office 필드를 생략하는 경우
  - 생략된 필드에는 널이 입력

### (질의 13)

```
insert into department (dept_id, dept_name)
values ('920', '컴퓨터공학과')
```

- 단, **not null**로 설정된 필드는 널 값이 들어갈 수 없는 필드이기 때문에 **insert**문의 <필드리스트>에서 생략할 수 없음

## 레코드 삽입

---

- <필드리스트>를 사용하지 않고 데이터를 삽입하는 예

(질의 14)

```
insert into  
values
```

```
department  
( '923', '산업공학과', '207호' )
```

Department 테이블의 필드 명이나 순서는 어떻게 확인?

# 레코드 수정

---

- 형식

```
update <테이블이름>  
set <수정내역>  
where <조건>
```

- <수정내역>

- 대상 필드에 들어가는 값을 수정하기 위한 산술식
- ','를 이용해서 여러 필드에 대한 수정 내역을 지정

- <조건>

- 대상이 되는 레코드에 대한 조건을 기술
- 테이블의 **모든 레코드에 대해 수정하려면 where 절을 생략**

## 레코드 수정

---

- 예) student 테이블에서 모든 학생들의 학년을 하나씩 증가

(질의 16)

```
update student
set year = year + 1;
```

- 예) professor 테이블에서 '고희석' 교수의 직위를 '교수'로 수정하고 학과번호를 '923'으로 수정

(질의 17)

```
update professor
set position='교수', dept_id='923'
where name='고희석';
```



# 레코드 삭제

---

- 형식

<b>delete from</b>	<테이블이름>
<b>where</b>	<조건>

- **where**절에 지정된 조건을 만족하는 레코드를 삭제
- **where**절이 생략되면 테이블에서 모든 레코드를 삭제

- 예) professor 테이블에서 이름이 '김태석'인 교수를 삭제

(질의 18)

<b>delete from</b>	professor
<b>where</b>	name='김태석';

- **delete**문을 이용하여 테이블의 모든 레코드를 삭제하더라도 테이블은 삭제되지 않음
  - ▶ 예) delete from professor;

## 레코드 삽입 시 주의사항

- 외래키로 사용되는 필드에 대해 데이터를 삽입할 때
  - 참조하는 테이블의 해당 필드에 그 값을 먼저 삽입해야 함
  - 예) department 테이블이 생성되긴 했지만 아직 레코드가 삽입되지 않은 상태에서 다음질의의 실행 결과

(질의 19)

**insert into**  
**values**

student  
( '1292002', '900305-1730021', '김정현2', 3, '서울',  
'920')

```
SQL> insert into student
  2  values( '1292002', '900305-1730021', '김정현', 3, '서울', '920');
insert into student
*
1행에 오류:
ORA-02291: 무결성 제약조건(SCOTT.FK_STUDENT)이 위반되었습니다- 부모 키가
없습니다
```

## 레코드 수정/삭제 시 주의사항

---

- 외래키 필드의 값을 수정할 때
  - 외래키가 참조하는 테이블에 존재하는 값으로만 수정 가능
- 외래키로 참조되는 필드를 가지고 있는 테이블에서 레코드를 삭제할 경우에도 오류가 발생할 수 있음
  - student 테이블에서 외래키로 참조하는 department 테이블의 레코드에 대한 삭제 시도
  - 해당 dept\_id를 가진 학생이 존재할 경우, 오류 발생

## 연습

---

- 6개의 테이블(학사 DB)을 생성해 보자
  - Department 테이블에 레코드를 삽입해 보자
  - 자신의 이름으로 학생 레코드를 하나 삽입하자. 학번은 적당히. 소속 학과는 컴퓨터공학과로..
    - ▶ 검색되는지 확인!
- 고재 실습내용 수행
  - 에러 발생 가능(제약 사항때문에) 적절히 수행하도록 하자
- 앞의 실습 후 "drop table"을 사용하여 6개의 테이블을 모두 삭제해 보자(
  - 삭제 순서를 미리 잘 생각한 후 수행한다
  - 테이블 간의 참조 관계를 표시해 보자.  
  - (테이블 삭제 가능 순서를 하나 기록: )

# 과제

---

- 사용자 계정에서 다음을 수행해 보자
- 다음 sql 문을 통해 테이블을 생성한다 (기본키 존재)
  - **Sql> create table** stu  
    (   stu\_id int,  
        stu\_name    varchar2(20) **not null**,  
        address     varchar2(20),  
        **constraint** stu\_pk **primary key** (stu\_id)  
    );
- 위의 테이블에 아래와 같은 레코드를 삽입한다 (잘 생성되었는지 확인은 "select \* from stu;"를 수행).
  - [100, '이태규', '부산']     /// 100은 정수 타입
  - [101, '최성희', '대전']
  - [102, '강만희', '부산']
- 위의 테이블에 대해서 sql 문 작성
  - Sql> '강만희' 학생의 주소를 출력

## 과제(continued)

---

- 이어서 아래의 레코드를 삽입해 보자
  - [101, '홍길동', '서울']
  - 어떤 문제가 발생하는가 살펴보자.
    - ▶ 에러 메시지 체크하여, 제출한다.
- Stu 테이블에 "tel\_num"이란 필드를 추가해 보자. 타입은 varchar2(10)
  - 수정된 테이블에 레코드 하나를 추가해 보자(레코드 값은 적당히)
  - 이태규의 주소를 '인천'으로 수정해 본다
  - 이태규 레코드를 삭제한다
- 추가로 friend라는 테이블을 생성한다
  - Sql> create table friend ( stu\_id int not null,  
friend\_id int references stu(stu\_id) );  
(friend 테이블은 stu 테이블을 referencing.
  - friend 테이블에 레코드를 하나 삽입하여, 최성희는 강만희를 친구로 가짐을 표현해 보자