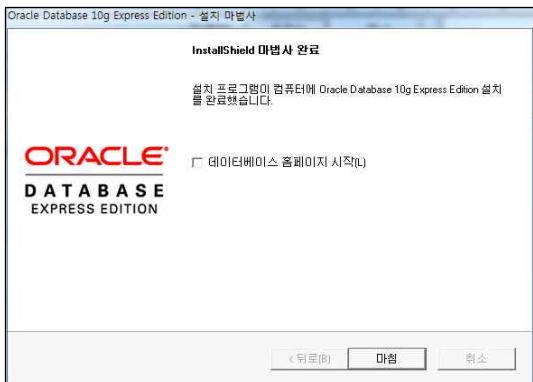
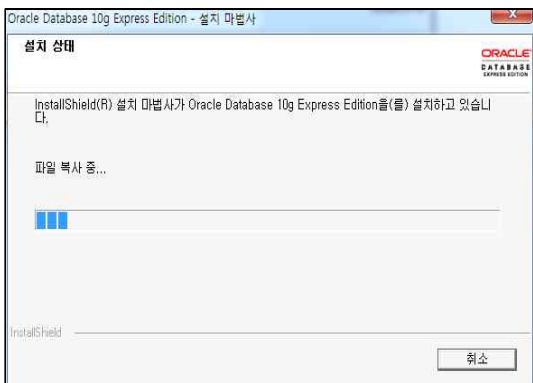
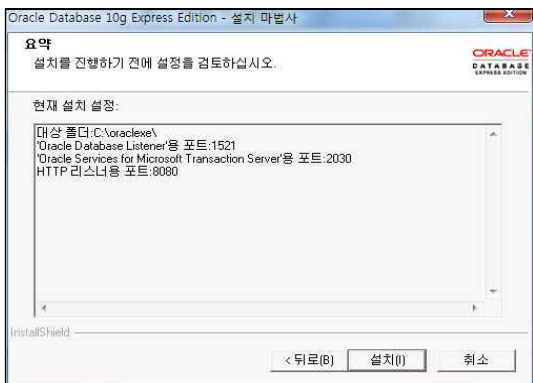
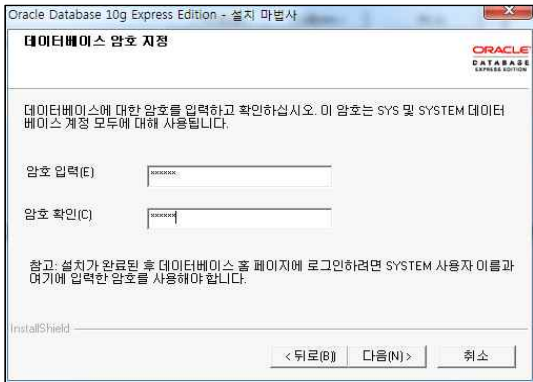
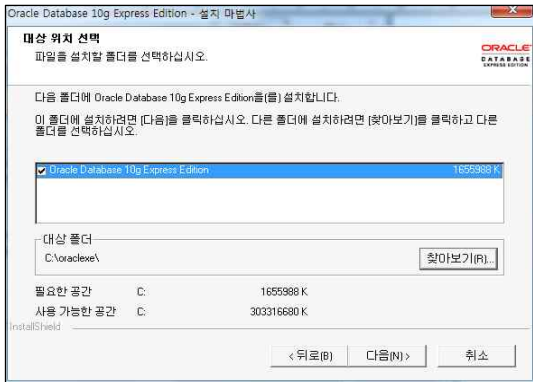
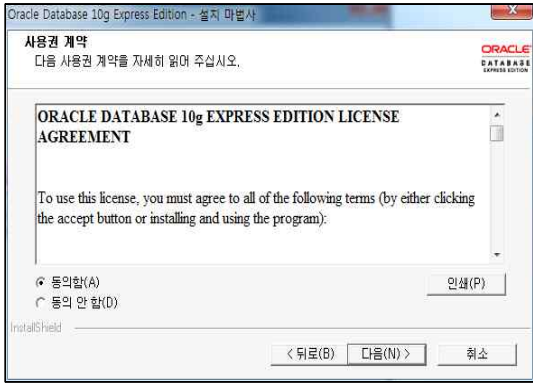
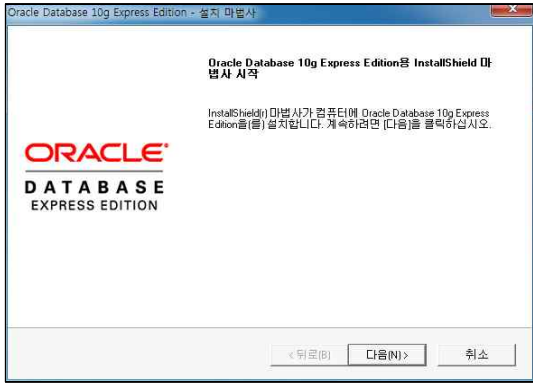


# 1장

## 1교시 : 오라클 설치

|  |   |                  |
|--|---|------------------|
| <a href="http://www.oracle.com/index.html">http://www.oracle.com/index.html</a><br>[Download]-<br>[Database 11gRelease 2 Express Edition]-<br>Oracle Database Express Edition 10g~ | Oracle Database 10gExpress Edition<br>for Microsoft Windows-<br>라이센스 동의 | OracleXEUniv.exe |
|--|---|------------------|



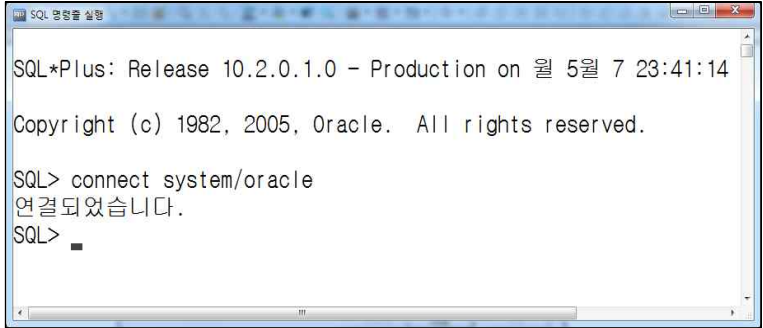
2교시-1 : 접속과 사용자 생성

[오라클 접속 포트 변경]

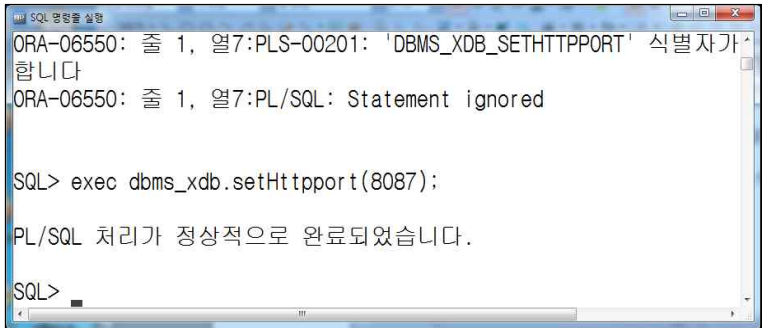
1) 시작→프로그램→Oracle Database 10g Express Edition→SQL 명령줄 실행

(또는 CMD창에서 C>sqlplus /nolog)

2) 데이터베이스 접속 : SQL> CONNECT SYSTEM/설치할때입력한패스워드 (패스워드를 oracle로 한 경우)



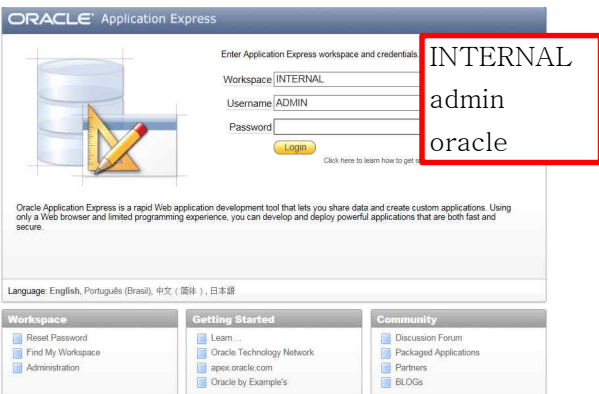
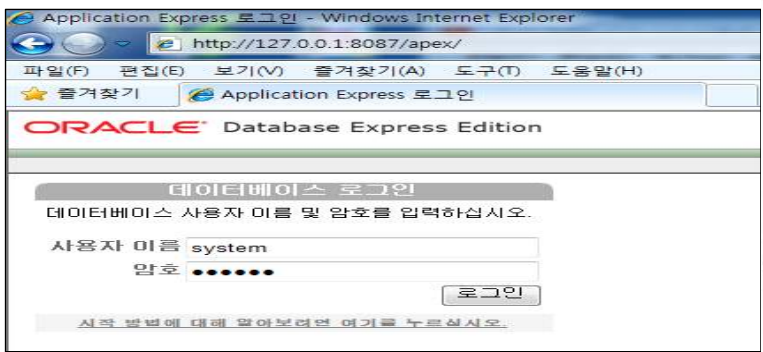
3) 포트변경 명령 수행 : SQL> EXEC DBMS\_XDB.SETHTTPPORT(8087);



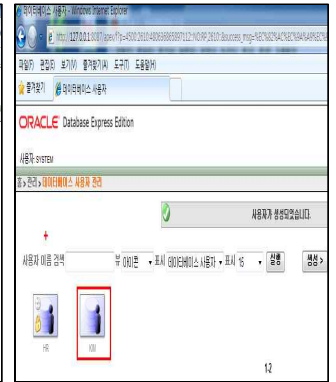
4) SQL Plus 종료 : SQL> EXIT

[오라클 데이터베이스 홈페이지 접속]

<http://127.0.0.1:8087/apex>



[사용자 계정 생성]



## 2교시-2 : 테이블 생성

```
SQL> CONN SYSTEM/ORACLE
SQL> SELECT * FROM TAB;
SQL> EXIT
```

```
SQL> CONN SYSTEM/ORACLE
SQL> DROP USER SCOTT CASCADE; (SCOTT 라는 사용자계정 삭제. 이미 존재할 경우가 있으므로)
```

```
SQL-1> CREATE USER SCOTT IDENTIFIED BY TIGER DEFAULT TABLESPACE SYSTEM;
(사용자계정:SCOTT, 비밀번호:TIGER, 기본 테이블스페이스 영역: SYSTEM)
```

※ TABLESPACE : Data File의 Block이 Server Process에 의해 읽혀져서 Instance의 논리적인 tablespace에 저장되며, 하나 이상의 데이터 파일로 구성된다.

```
SQL-2> GRANT CONNECT, RESOURCE TO SCOTT;
(SCOTT 계정에 권한부여)
```

```
SQL> CONN SCOTT/TIGER; (SCOTT계정으로 접속하기)
SQL> SELECT * FROM TAB;
SQL> CREATE TABLE STAFF
(
  ID NUMBER(5),
  NAME VARCHAR2(50),
  ADDRESS VARCHAR2(50),
  BIRTHDAY DATE,
  SENIORITY NUMBER(2) DEFAULT 1,
  DEPT_ID NUMBER(3)
);
```

```
SQL> SELECT * FROM TAB;
SQL> DESC STAFF; (테이블의 필드들의 타입보기. 즉 구조보기)
```

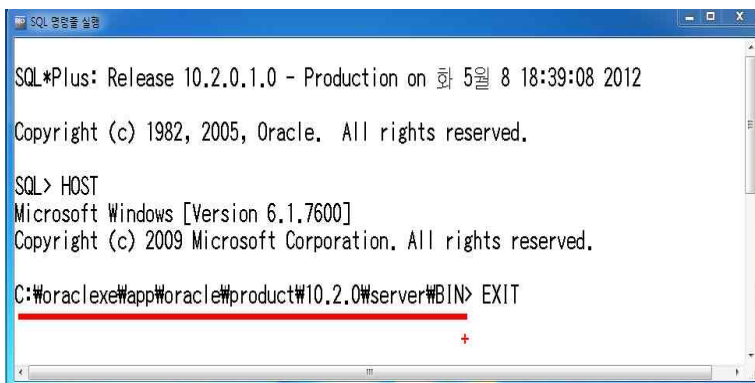
```
SQL> INSERT INTO STAFF(ID, NAME, ADDRESS) VALUES (1, '우수자', '상주');
SQL> INSERT INTO STAFF(ID, NAME, ADDRESS) VALUES (2, '보조자', '대구');
SQL> SELECT * FROM STAFF;
```

## 2교시-3 : 파일 다루기

### [파일 명령어]

- SAVE 파일명 : 버퍼에 있는 내용을 파일에 저장
- GET 파일명 : 파일의 내용을 버퍼로 불러옴
- EDIT 파일명 : 버퍼에 있는 내용을 편집기로 불러옴
- START 파일명 또는 @파일명 : 저장된 SQL스크립트를 실행
- SPOOL 파일명 : 조회 결과를 화면에 저장
- SPOOL OFF : 스푼 해제
- EXIT : SQL\*PLUS 툴을 종료
- CONNECT 사용자아이디/암호 : 다른 사용자로 접속하기 위해 사용
- HOST : 도스상태로 잠시 빠져나감

SQL> HOST (도스상태로 빠져나감)



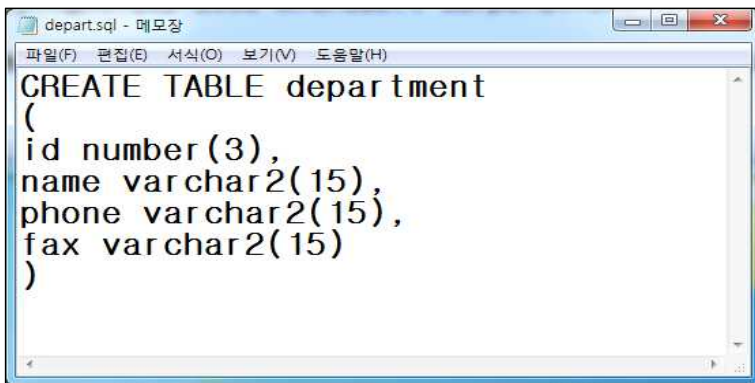
```
SQL*Plus: Release 10.2.0.1.0 - Production on 화 5월 8 18:39:08 2012
Copyright (c) 1982, 2005, Oracle. All rights reserved.

SQL> HOST
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\WoraclexeWappWoracleWproductW10.2.0WserverWBIN> EXIT
```

기본 저장 폴더는 C:\WoraclexeWappWoracleWproductW10.2.0WserverWBIN 이다  
메모장에서 작성후 `depart.sql` 이란 이름으로 C:\WoraclexeWappWoracleWproductW10.2.0WserverWBIN에 저장  
한다. (편리를 위해 C:\Wsam 이란 샘플 폴더를 만들어 그 내부에 저장할 수도 있다)

### 파일명:[depart.sql]



```
CREATE TABLE department
(
  id number(3),
  name varchar2(15),
  phone varchar2(15),
  fax varchar2(15)
)
```

SQL> GET depart.sql (sql문법을 버퍼로 가져온다. 만일 c:\Wsam폴더에 저장시 SQL> GET c:\Wsam\depart.sql)  
SQL> / 또는 run (버퍼의 내용을 실행한다)  
SQL> DESC department;

파일명: [insert.sql]

```
insert.sql - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
insert into department(id, name, phone, fax) values(123, '홍길동', '123', '123');
insert into department(id, name, phone, fax) values(124, '김순자', '123', '123');
insert into department(id, name, phone, fax) values(125, '이기자', '123', '123');
insert into department(id, name, phone, fax) values(126, '최고야', '123', '123');
insert into department(id, name, phone, fax) values(127, '박말순', '123', '123');
```

SQL> @ insert.sql (가져오면서 바로 실행된다. 만일 c:Wsam폴더에 저장시 SQL> @ c:WsamWininsert.sql)  
SQL> SELECT \* FROM department;

SQL-1> CONN SYSTEM/ORACLE  
SQL-2> GRANT CREATE ANY DIRECTORY TO SCOTT; (DIRECTORY OBJECT 생성 권한 부여)  
SQL-3> CONN SCOTT/TIGER;

SQL> SELECT \* FROM USER\_SYS\_PRIVS;

| USERNAME | PRIVILEGE            | ADM |
|----------|----------------------|-----|
| SCOTT    | CREATE ANY DIRECTORY | NO  |
| SCOTT    | UNLIMITED TABLESPACE | NO  |

SQL-4> CREATE OR REPLACE DIRECTORY SCHOOL\_DIR AS 'C:WSAM';  
(C:WSAM 폴더에 대해 “SCHOOL\_DIR” 이라는 directory object를 생성한다)

C:\Woraclexe\Wapp\Woracle\Wproduct\W10.2.0\Wserver\WRDBMS\WADMIN 폴더의 scott.sql 파일을 c:Wsam 폴더로 복사해 온다

SQL> @ c:Wsam\Wscott.sql  
SQL> select \* from tab;

SQL> SET LINESIZE 80 --> 한줄에 출력될 수 있는 라인사이즈를 변경할 수 있다.  
SQL> SET PAGESIZE 10 --> 한 페이지에 출력되는 ROW의 수  
SQL> SELECT \* FROM EMP;

SQL> SELECT ENAME FROM EMP;  
SQL> COLUMN ENAME FORMAT A20; --> 컬럼의 크기를 20으로 지정  
SQL> SELECT ENAME FROM EMP;

SQL> COLUMN SAL FORMAT 0,000,000  
SQL> SELECT EMPNO, ENAME, SAL FROM EMP;  
SQL> COLUMN SAL FORMAT 999999  
SQL> COLUMN SAL CLEAR --> SAL 컬럼에 설정된 값을 해제

## 2장

### 1교시 : sql 문법 - 데이터 추출 연산자

```
SQL> SELECT * FROM EMP;
SQL> SELECT * FROM EMP WHERE DEPTNO=10;
SQL> SELECT EMPNO, ENAME, SAL FROM EMP WHERE SAL>=2000;
SQL> SELECT EMPNO, ENAME FROM EMP WHERE ENAME='ford';  ->(x) 문자열 데이터는 대소문자 확실히
SQL> SELECT ENAME, HIREDATE FROM EMP WHERE HIREDATE >= '1982/01/01';
SQL> SELECT EMPNO, ENAME FROM EMP WHERE ENAME LIKE 'J%';
SQL> SELECT EMPNO, ENAME FROM EMP WHERE ENAME LIKE '%J%';
SQL> SELECT EMPNO, ENAME FROM EMP WHERE ENAME LIKE '_A%';
SQL> select empno, ename, comm
      from emp
      where comm=300 or comm=500 or comm=1400;

SQL> select empno, ename, comm
      from emp
      where comm IN(300,500,1400);

SQL> select empno, ename, sal
      from emp
      where sal >=500 and sal <=4000;

SQL> select empno, ename, sal
      from emp
      where sal BETWEEN 500 AND 4000;

SQL> select empno, ename, deptno
      from emp
      where NOT deptno = 10;

SQL> select empno, ename, comm
      from emp
      where comm NOT IN(300,500,1400);

SQL> select ename, mgr
      from emp
      where mgr IS NULL;

SQL> select empno, ename, sal
      from emp
      ORDER BY sal;

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

```
ORDER BY sal DESC;
SQL> select empno, ename, sal
from emp
ORDER BY sal DESC, ename ASC;
```

## 2교시 : sql 주요 함수

```
SQL> select 10*20 from DUAL;    --> DUAL 테이블은 기술할 테이블이 없을때 사용하는 가짜 테이블
SQL> select round(45.293, 2)
from DUAL;
SQL> select round(45.293, -1)
from DUAL;
SQL> select sal, MOD(sal, 100)
from emp;
SQL> select * from emp
where MOD(empno, 2)=1;
```

기타 문자, 날짜함수 등은 생략하겠습니다.

### 1) 그룹함수

count, sum, avg, max, min

```
SQL> select sum(sal)
from emp;
```

```
SQL> select avg(sal)
from emp;
```

```
SQL> select max(sal), min(sal)
from emp;
```

```
SQL> select count(comm)
from emp;
```

```
SQL> select count(comm)
from emp
where deptno=10;
```

```
SQL> select count(distinct job)
from emp;
```

```
SQL> select count(*)
from emp;
```

```
SQL> select max(sal), ename
from emp;
```

```
SQL> select deptno, ename, avg(sal)
from emp
group by deptno;
→ 예러; group by절로 묶이지 않는 단순 컬럼은 select 리스트에 사용 불가
```

```
SQL> select deptno, count(*), count(comm)
from emp
group by deptno;
```

```
SQL> select deptno, avg(sal)
from emp
group by deptno
having avg(sal) >= 2000;
```

```
SQL> select deptno, count(sal)
from emp
where sal >= 1000
group by deptno
having count(sal) >= 3;
```

### 3교시 : 조인

만일 사원번호가 7900번인 사원이 어떤 부서 소속인지 소속 부서명을 알아내기 위해서

```
SQL> select empno, ename, deptno
from emp
where empno=7900;
```

```
SQL> select dname
from dept
where deptno=30;
```

#### 1) 동일조인(equi join)

```
SQL> select *
from emp, dept
where emp.deptno = dept.deptno;
```

```
SQL> select ename, dname
from emp, dept
where emp.deptno = dept.deptno;
```

```
SQL> select e.ename, d.dname, e.deptno, d.deptno
from emp e, dept d
where e.deptno=d.deptno;
```



만일 사원이름이 SMITH 인 사원명(ename), 부서명(dname), 부서코드(deptno)를 출력하시오(join방법으로 해결)  
단) emp테이블에는 사원명(ename), 부서코드(deptno), 사원코드 등등이 존재하고,  
dept테이블에는 부서코드(deptno), 부서명(dname) 등등이 존재한다.

```
SQL> select emp.ename, dept.dname, emp.deptno  
from emp, dept  
where emp.deptno = dept.deptno and emp.ename='SMITH' ;
```

## 2) 비동일조인(non-equi join)

= 연산자 이외의 비교 연산자를 사용

```
SQL> select e.ename, e.sal, s.grade  
from emp e, salgrade s  
where e.sal >= s.losal and e.sal <= s.hisal;
```

```
SQL> select e.ename, d.dname, s.grade  
from emp e, dept d, salgrade s  
where e.deptno = d.deptno  
and e.sal between s.losal and s.hisal;
```

## 특정 사원의 매니저가 누구인지 알아내기(self join)

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

```
SQL> select e.ename || '의 매니저는' || m.ename || '입니다.'  
from emp e, emp m  
where e.mgr = m.empno;
```

## 3) 외부조인(outer join)

```
SQL> select e.ename || '의 매니저는' || m.ename || '입니다.'  
from emp e, emp m  
where e.mgr = m.empno(+);
```

→ 좌측 외부 조인

```
SQL> select e.ename, d.dname  
from emp e, dept d  
where e.deptno(+) = d.deptno ;
```

## 실습

예제1) 사원들의 이름(ename), 부서번호(deptno), 부서이름(dname)을 출력  
단) emp테이블에는 사원명(ename), 부서코드(deptno), 사원코드 등등이 존재하고,  
dept테이블에는 부서코드(deptno), 부서명(dname) 등등이 존재한다.

```
select emp.ename, dept.deptno, dept.dname  
from emp, dept
```

```
where emp.deptno = dept.deptno;
```

예제2) 부서번호가 30번인 직원들의 이름(ename), 직급(job), 부서번호(deptno), 부서위치(loc)를 출력  
단) emp테이블에는 사원명(ename), 부서코드(deptno), 직급(job) 등등이 존재하고,

dept테이블에는 부서코드(deptno), 부서위치(loc) 등등이 존재한다.

```
select emp.ename, emp.job, emp.deptno, dept.loc
from emp, dept
where emp.deptno=dept.deptno
and emp.deptno=30;
```

예제3) 커미션을 받는 직원의 이름(ename), 커미션(comm), 부서이름(dname) 및 부서위치(loc)를 출력  
단) emp테이블에는 사원명(ename), 부서코드(deptno), 커미션(comm) 등등이 존재하고

dept테이블에는 부서코드(deptno), 부서명(dname), 부서위치(loc) 등등이 존재한다.

```
select emp.ename, emp.comm, dept.dname, dept.loc
from emp, dept
where emp.deptno = dept.deptno
and emp.comm IS NOT NULL and emp.comm NOT IN(0);
```

예제4) DALLAS에서 근무하는 직원의 이름, 직급, 부서번호, 부서이름을 출력

```
select emp.ename, emp.job, dept.deptno, dept.dname
from emp, dept
where emp.deptno = dept.deptno
and dept.loc='DALLAS';
```

예제5) 이름에 A가 들어가는 직원들의 이름과 부서이름을 출력

```
select emp.ename, dept.dname
from emp, dept
where emp.deptno = dept.deptno
and emp.ename like '%A%'
```

예제6) 사원이름과 직급, 급여, 급여등급을 출력

```
select e.ename, e.job, e.sal, s.grade
from emp e, salgrade s
where e.sal BETWEEN s.losal AND s.hisal;
```

예제7) 사원이름, 부서번호와 해당 직원과 같은 부서에서 근무하는 직원을 출력 (self join)

```
select e.ename "자신", e.deptno, c.ename "동료", c.deptno
from emp e, emp c
where e.ename <> c.ename
AND e.deptno = c.deptno
ORDER BY e.ename;
```

#### 4교시 : 서브 쿼리

단일 행 서브 쿼리 / 다중 행 서브 쿼리

##### 1) 단일 행 서브 쿼리

예1)

```
select dname
from dept
where deptno = (select deptno
                 from emp
                 where ename='JONES');
```

예2) 직원들의 평균 급여보다 더 많은 급여를 받는 직원을 검색

```
select ename, sal
from emp
where sal > (select avg(sal) from emp);
```

예3) 평균 급여보다 많은 급여를 받는 직원들의 사번, 직원명, 급여를 나타내되 급여를 많이 받는 순으로 출력

```
select ename, sal
from emp
where sal > (select avg(sal) from emp)
order by sal desc;
```

예4) 부서번호가 10번인 직원 중에서 최대 급여를 받는 직원과 동일한 급여를 받는 직원 번호와 직원명을 기술하시오

```
select empno, ename
from emp
where sal = (select max(sal)
             from emp
             where deptno = 10);
```

##### 2) 다중 행 서브 쿼리

예1) BLAKE의 부서번호는 30번이므로 BLAKE와 같은 부서에서 근무하는, 즉 부서 번호가 30인 사람들의 이름과 고용 일을 출력하는 쿼리문

```
select ename, hiredate, deptno
from emp
where deptno IN ( select deptno
                  from emp
                  where ename = 'BLAKE' );
```

예2) BLAKE와 같은 부서에 있는 사람들의 이름과 고용 일을 출력하는데 BLAKE는 빼고 구하기

```
select ename, hiredate, deptno
from emp
where deptno IN ( select deptno
                  from emp
                  where ename = 'BLAKE' )
AND ename != 'BLAKE';
```

예3) IN연산자

```
select ename, sal, deptno
from emp
where deptno IN ( select sal
                  from emp
                  where sal >= 3000);
```

예4) 30번 부서에서 급여를 가장 많이 받는 사원의 급여인 2850보다 더 높은 급여를 받는 직원들만 출력(ALL 연산자)

```
select ename, sal
from emp
where sal > ALL (select sal
                 from emp
                 where deptno = 30);
```

예5) 부서번호가 30번인 직원들의 급여 중 가장 작은 값인 950보다 높은 급여를 받는 사원의 이름, 급여를 출력 (ANY 연산자)

```
select ename, sal
from emp
where sal > ANY (select sal
                 from emp
                 where deptno = 30);
```

실습

예제1) SCOTT의 급여와 동일하거나 더 원이 받는 직원명과 급여를 출력

```
select ename, sal
from emp
where sal >= ( select sal
              from emp
              where ename='SCOTT');
```

예제2) 직급(job)이 사원(CLERK)인 사람의 부서의 부서번호와 부서명과 지역을 출력

```
select deptno, dname, loc
from dept
where deptno IN (select deptno
                 from emp
                 where job='CLERK');
```

예제3) 이름이 T를 포함하고 있는 직원들과 같은 부서에서 근무하고 있는 사원의 직원번호와 이름을 출력

```
select empno, ename
from emp
where deptno IN (select deptno
                 from emp
                 where ename LIKE '%T%');
```

예제4) 부서 위치가 DALLAS인 모든 사원의 이름, 부서 번호를 출력

```
select ename, deptno
from emp
where deptno = (select deptno
                 from dept
                 where loc='DALLAS');
```

예제5) SALES부서의 모든 사원의 이름과 급여를 출력

```
select ename, sal
from emp
where deptno = (select deptno
                 from dept
                 where dname='SALES');
```

예제6) KING에게 보고하는 모든 사원의 이름과 급여를 출력.

KING에게 보고하는 사원이란 의미는 상관(mgr)이 KING인 사원을 의미한다

```
select ename, sal, mgr
from emp
where mgr IN ( select empno
                from emp
                where ename = 'KING');
```

### 3장

#### 1교시 : DDL (테이블 생성 및 변경, 삭제)

##### 1) 테이블 생성

###### 예1)

```
create table em01
  (empno number(4),
   ename varchar2(20),
   sal number(7, 2));
```

###### 예2) 서브 쿼리를 이용한 테이블 생성

```
create table em02
  as select * from emp;
```

###### 예3)

```
create table em03
  as select empno, ename from emp;
```

###### 예4) 해당되는 행이 없을 경우 빈 테이블만 생성된다.

```
create table em04
  as select * from emp where 1=0;
```

###### 예5)

```
create table em05
  as select * from emp;
```

##### 2) 테이블 구조 변경

###### 예1) 컬럼 추가하기

```
alter table em05
  add(email varchar2(10));
```

###### 예2) 컬럼 변경하기

```
alter table em05
  modify (email varchar2(40));
```

###### 예3) 컬럼 삭제하기

```
alter table em05
  drop column email;
```

###### 예3) set unused 옵션 적용 (무조건 삭제하는 것은 위험); 논리적으로 제한하게 됨

```
alter table em05
  set unused(email);
```

```
alter table em05
  drop unused column;
```

가장 사용빈도가 적은 시간에 실제적인 삭제 작업을 하는 것이 좋다.

3) 테이블 제거

```
drop table em04;
```

4) 테이블의 모든 행 제거

```
select * from em05;
```

```
truncate table em05;
```

→모든 행 제거

5) 디렉터리

DBA\_XXXX, ALL\_XXXX, USER\_XXXX

예1) USER\_ 데이터 디렉터리

```
SQL> show user
```

```
SQL> desc user_tables;
```

```
SQL> select table_name from user_tables  
order by table_name desc;
```

예2) ALL\_ 데이터 디렉터리

```
SQL> DESC all_tables;
```

```
SQL> select owner, table_name from all_tables;
```

예3) DBA\_ 데이터 디렉터리 뷰

```
SQL> conn system/oracle
```

```
SQL> select owner, table_name from dba_tables  
where owner='SYSTEM';
```

2교시 : DML (테이블 추가, 수정, 삭제)

1) insert문

예1)

```
create table dept01  
as select * from dept;
```

```
insert into dept01(deptno, dname, loc)  
values (60, '회계과', '서울');
```

```
insert into dept01  
values (70, '인사과', '수원');
```

2) 치환변수 사용

(변수 앞에 &기호 붙이면 입력 받는 값을 임시적으로 저장한다)

예1)

```
select *  
from emp
```

```
where sal > &salary_value;
```

예2) && 을 사용할 경우에는 처음 한번만 값을 묻고, 그 다음부터는 동일한 변수에 대해 묻지 않고 이전에 입력된 값을 적용

```
select &&column_name  
from &&table_name;
```

3) 서브 쿼리로 행 추가하기

```
create table dept02  
as  
select * from dept  
where 1=0;
```

```
insert into dept02  
select * from dept;
```

4) 다중 테이블에 다중 행 입력

```
create table emp_hir  
as select empno, ename, hiredate from emp where 1=0;
```

```
create table emp_mgr  
as select empno, ename, mgr from emp where 1=0;
```

```
SQL> insert all  
into emp_hir values(empno, ename, hiredate)  
into emp_mgr values(empno, ename, mgr)  
select empno, ename, hiredate, mgr  
from emp  
where deptno > 20;
```

```
SQL> select * from emp_hir;
```

```
SQL> select * from emp_mgr;
```

5) 조건(when)에 의해 다중 테이블에 다중 행 입력

```
SQL> create table emp_hir02  
as select empno, ename, hiredate from emp where 1=0;
```

```
SQL> create table emp_sal  
as select empno, ename, sal from emp where 1=0;
```

예) 1982년 이후에 입사한 사원 검색되어 emp\_hir02 테이블에 입력  
급여가 2000이상인 사원만 검색되어 emp\_sal 테이블에 입력

```
SQL> insert all  
when hiredate > '1982/01/01' then  
into emp_hir02 values(empno, ename, hiredate)  
when sal >= 2000 then  
into emp_sal values(empno, ename, sal)
```



```
select empno, ename, hiredate, sal from emp;
```

```
SQL> select * from emp_hir02;
```

```
SQL> select * from emp_sal;
```

#### 6) update문

```
SQL> create table dept02  
as select * from dept;
```

예) 10번 부서의 지역명을 서울로 변경한다.

```
SQL> update dept02  
set loc='서울'  
where deptno=10;
```

예) 모든 사원의 급여를 10% 인상

```
SQL> update em06  
set sal=sal*1.1;
```

#### 7) 서브 쿼리를 이용한 데이터 수정하기

```
SQL> create table dept03  
as  
select * from dept;
```

예) 부서번호가 20인 부서의 부서명과 지역명을 부서 번호가 40번인 부서와 동일하게 수정하기

```
SQL> update dept03  
set (dname, loc)=(select dname, loc  
from dept  
where deptno=40)  
where deptno=20;
```

#### 8) delete문

```
SQL> delete  
from dept03  
where deptno=30;
```

```
SQL> create table em07  
as select * from emp;
```

```
SQL> delete from em07  
where deptno=(select deptno from dept where dname='SALES');
```

#### 9) merge문

테이블에 기존의 행이 존재하면 갱신(update)되고, 존재하지 않으면 새로운 행으로 추가(insert)됨

```
SQL> create table em08  
as select *  
from emp;
```

```
SQL> create table em09
      as select *
      from emp
      where job='MANAGER';
```

```
SQL> update em09
      set job='TEST'
      where job='MANAGER';
```

```
SQL> insert into em09
      values (8000, 'SYJ', 'TOP', 7566, '2005/01/02', 1200, 10, 20);
```

```
SQL> select * from em08;
```

예) em08 테이블에 em09 테이블을 합병하라

```
SQL> merge into em08 e
      using em09 s
      on(e.empno=s.empno)
      when matched then
        update set
          e.name = s.name,
          e.job   = s.job,
          e.mgr   = s.mgr,
          e.hiredate = s.hiredate,
          e.sal    = s.sal,
          e.comm   = s.comm,
          e.deptno = s.deptno
      when not matched then
        insert values (s.name, s.job,
                      s.mgr, s.hiredate, s.sal,
                      s.comm, s.deptno);
```

em08 테이블과 em09 테이블의 사원 번호가 일치하면 em09테이블의 내용으로 업데이트(update) 한다.

일치하지 않으면 em09테이블의 내용을 추가(insert) 한다.

```
SQL> select * from em08;
```

## 4장

### 1교시 : 트랜잭션 관리

|    |   |                                     |
|----|---|-------------------------------------|
| 1  | 데이터베이스에서 트랜잭션(Transaction)은 데이터 처리의 한 단위이다. 오라클에서 발생하는 여러 개의 SQL                    |                                     |
| 2  | 명령문들을 하나의 논리적인 작업 단위로 처리하는데 이를 트랜잭션이라고 한다. <u>DML명령(insert, update,</u>             |                                     |
| 3  | <u>delete)</u> 을 모두 묶어 놓은 하나의 논리적인 작업 단위를 의미한다.                                     |                                     |
| 4  | 하나의 트랜잭션은 All-OR-Nothing방식으로 처리된다. 여러개의 명령어의 집합이 정상적으로 처리되면 정상                      |                                     |
| 5  | 종료하도록 하고 여러 개의 명령어 중에서 하나의 명령어라도 잘못되었다면 전체를 취소해버린다.                                 |                                     |
| 6  | 데이터베이스에서 작업의 단위로 트랜잭션이란 개념을 도입한 이유는 데이터의 일관성을 유지하면서 안정적                             |                                     |
| 7  | 으로 데이터를 복구시키기 위해서이다.  |                                     |
| 8  |   |                                     |
| 9  | [트랜잭션 제어를 위한 명령어]   |                                     |
| 10 | - commit : 저장되지 않은 모든 데이터를 데이터베이스에 저장하고 현재의 트랜잭션을 종료하라는 명령                          |                                     |
| 11 | - savepoint 이름 : 현재까지의 트랜잭션을 특정 이름으로 지정하라는 명령                                       |                                     |
| 12 | - rollback [to savepoint이름]: 저장되지 않은 모든 데이터 변경 사항을 취소하고 현재의 트랜잭션을 끝내라               |                                     |
| 13 | 는 명령  |                                     |
| 14 | [자동 커밋]   |                                     |
| 15 | DDL문에는 create, alter, drop, rename, truncate 등이 있다. 이러한 DDL문은 자동으로 커밋(auto commit)이 |                                     |
| 16 | 발생된다. 즉 rollback 으로 되살릴수 없다.  |                                     |
| 17 | [세이브 포인트]   |                                     |
| 18 | 세이브포인트는 롤백과 함께 사용해서 현재 트랜잭션 내의 특정 세이브포인트까지 롤백한다.                                    |                                     |
| 19 | SQL1> <b>commit;</b>  | 이전 수행했던 모든 데이터 변경을 모두 적용            |
| 20 | SQL> <b>select * from dept01;</b>   | 내용을 확인후                             |
| 21 | SQL> <b>delete from dept01;</b>   | 테이블 삭제                              |
| 22 | SQL> <b>select * from dept01;</b>   | 삭제된 것을 확인                           |
| 23 | SQL> <b>rollback;</b>   | 이전 상태로 되돌리기 (roll)                  |
| 24 | SQL> <b>select * from dept01;</b>   | 원상 복구되어 있는 것을 확인                    |
| 25 | SQL> <b>delete from dept01 where deptno=20;</b>                                     | 부서번호 20번 사원에 대한 정보만 삭제              |
| 26 | SQL> <b>select * from dept01;</b>   | 확인                                  |
| 27 | SQL> <b>commit;</b>   | 결과를 물리적으로 영구히 저장                    |
| 28 | SQL> <b>rollback;</b>   | 커밋을 수행한 후에 다시 롤백 수행안됨               |
| 29 | SQL> <b>select * from dept01;</b>   | 확인                                  |
| 30 | SQL>  |                                     |
| 31 | SQL>  |                                     |
| 32 | SQL2> <b>truncate table emp;</b>  | truncate table 테이블명; 은 delete문처럼 데이 |
| 33 | SQL> <b>rollback;</b>   | 터를 삭제할 때 사용. where 문을 사용할 수 없       |
| 34 | SQL> <b>select * from emp;</b>  | 음. 자동 커밋되므로 rollback 해도 복구안됨.       |
| 35 | SQL>  |                                     |
| 36 | SQL3> <b>savepoint a;</b>   | 세이브포인트 a 설정                         |
| 37 | SQL> <b>delete from dept05 where deptno=20;</b>                                     | 삭제                                  |
| 38 | SQL> <b>select * from dept05;</b>   | 확인                                  |
| 39 | SQL> <b>savepoint b;</b>  | 세이브포인트 b 설정                         |
| 40 | SQL> <b>delete from dept05 where deptno=10</b>                                      | 삭제                                  |
| 41 | SQL> <b>select * from dept05;</b>   | 확인                                  |
| 42 | SQL> <b>rollback to a;</b>  | 부서번호 20번 삭제하기 바로 전으로 복구             |
| 43 | SQL>  |                                     |

44 SQL> select \* from dept05;

## 2교시 : 무결성 제약조건(1)

1 오라클에서는 데이터 무결성 제약조건으로 5가지를 지원한다.

2 데이터 디렉터리 user\_constraints에서 constraint\_type은 P, R, U, C 4가지 값 중에 하나를 갖는다.

| constraint_type | 의미                                    |
|-----------------|---------------------------------------|
| not null        | C 해당 컬럼 값으로 null 허용하지 않음              |
| unique          | U 테이블 내에서 해당 컬럼 값은 항상 유일한 값을 가질것      |
| primary key     | P 해당 컬럼 값은 반드시 존재해야 하고 유일한 값          |
| foreign key     | R 해당 컬럼의 값이 다른 테이블의 컬럼의 값을 참조해야 함     |
| check           | C 해당 컬럼에 저장 가능한 데이터 값의 범위나 사용자 조건을 지정 |

8 [컬럼 레벨]

9 (1) deptno number(2) [primary key](#),

10 (2) deptno number(2) [constraint](#) [제약조건명](#) [primary key](#),

11 [테이블 레벨]

12 (1) [primary key](#)(deptno),

13 (2) [constraint](#) [제약조건명](#) [primary key](#)(deptno),

14 SQL1> create table dept02

[컬럼 레벨]

15 SQL> (

16 SQL> deptno number(2) [primary key](#),

17 SQL> dname varchar2(15),

18 SQL> loc varchar2(15)

19 SQL> );

20 SQL>

21 SQL> create table dept03

22 SQL> (

23 SQL> deptno number(2) [constraint](#) [dept03\\_deptno\\_pk](#) [primary key](#),

[제약조건명](#)→

24 SQL> dname varchar2(15),

테이블명\_

25 SQL> loc varchar2(15)

컬럼명\_

26 SQL> );

제약조건유형

27 SQL>

ex)

28 SQL>

dept03\_deptno\_pk

29 SQL> [select](#) constraint\_name, constraint\_type, table\_name [from](#) user\_constraints;

30 SQL> column table\_name format a20;

31 SQL> column constraint\_name format a20;

32 SQL>

33 SQL>

34 SQL2> create table dept04

[테이블 레벨]

35 SQL> (

36 SQL> deptno number(2),

37 SQL> dname varchar2(15),

38 SQL> loc varchar2(15),

39 SQL> [primary key](#)(deptno)

40 SQL> );

41 SQL>

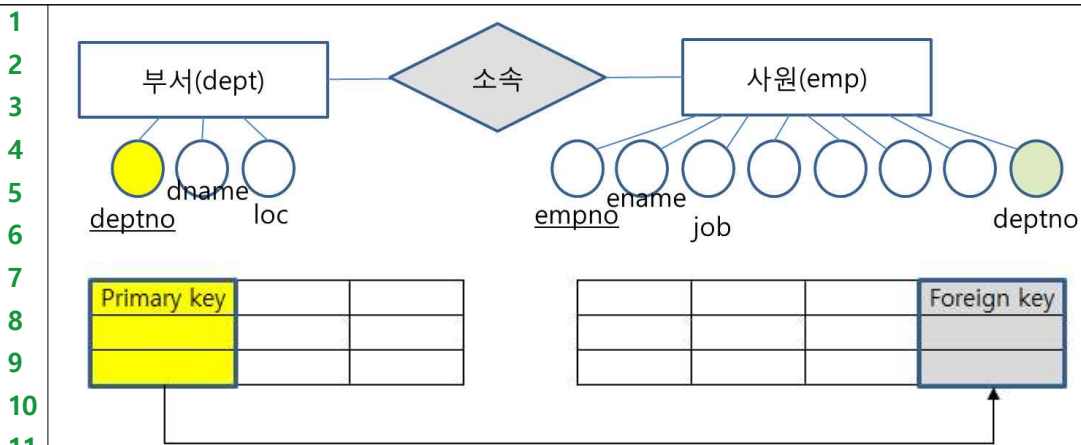
42 SQL> create table dept05

```

43 SQL> (
44 SQL>     deptno number(2),
45 SQL>     dname varchar2(15),
46 SQL>     loc varchar2(15),
47 SQL>     constraint dept05_deptno_pk primary key(deptno)
48 SQL> );
49 SQL> select constraint_name, constraint_type, table_name from user_constraints;
50 SQL>
51
52
53

```

### 3교시 : 무결성 제약조건(2)



```

12 SQL> 부서(dept)테이블이 부모테이블이 되고, 사원(emp)테이블이 자식테이블이 된
13 SQL> 다. 두 테이블과 같이 주종 관계에 있는 테이블에 대해서 부모 테이블의 기
14 SQL> 본키(primary key)가 자식 테이블의 외래키(Foreign key)가 된다
15 SQL> - 개체무결성:기본키가 유일성을 가짐
16 SQL> - 참조무결성:외래키가 참조하는 데 결함이 없어야 함
17 SQL>
18 SQL> create table dept07
19 SQL> (
20 SQL>     deptno number(2),
21 SQL>     dname varchar2(15),
22 SQL>     loc varchar2(15),
23 SQL>     constraint dept07_deptno_pk primary key(deptno)
24 SQL> );
25 SQL>
26 SQL> create table emp07
27 SQL> (
28 SQL>     empno number(4) primary key,
29 SQL>     ename varchar2(15),
30 SQL>     sal number(7,2),
31
32

```

```

33 SQL> deptno number(2) references dept07(deptno)
34 SQL> );
35 SQL>
36 SQL> 또는
37 SQL> create table emp07
38 SQL> (
39 SQL> empno number(4) primary key,
40 SQL> ename varchar2(15),
41 SQL> deptno number(2) constraint emp07_deptno_fk references dept07(deptno)
42 SQL> );
43 SQL>
44 SQL>
45 SQL> select constraint_name, constraint_type, table_name from user_constraints;
46
47 SQL> [유일키 제한 조건]
48 SQL> create table dept08
49 SQL> (
50 SQL> deptno number primary key,
51 SQL> dname varchar2(15) unique,
52 SQL> loc varchar2(15))
53 SQL> );
54 SQL>
55 SQL> select constraint_name, constraint_type, table_name from user_constraints;
56 SQL>
57 SQL> alter table dept07
58 SQL> add constraint dept07_dname_uq unique(dname);
59 SQL>
60 SQL> select constraint_name, constraint_type, table_name from user_constraints;
61 SQL>
62 SQL>
63 SQL> insert into dept07 values(1, 'A', 'AA');
64 SQL> insert into dept07 values(2, 'A', 'AA'); //
65 SQL> insert into dept07 values(3, NULL, 'AA');
66 SQL> insert into dept07 values(4, NULL, 'AA');
67 SQL>
68
69
70 [check 제한 조건]:조건에 맞는 데이터만 입력되도록 함
71 SQL> create table emp08
72 SQL> (
73 SQL> empno number(4) constraint emp08_empno_pk primary key,
74 SQL> ename varchar2(15),
75 SQL> sal number(7,2) constraint emp08_sal_ck check(sal between 500 and 5000),
76 SQL> deptno number(2) constraint emp08_deptno_fk references dept08(deptno)
77

```

기존테이블  
에 유일키  
지정

오류  
NULL값은  
중복가능

```
78 SQL> );
79
80 [not null 제한 조건]
81 SQL> create table dept09
82 SQL> (
83 SQL>     deptno number(2) constraint dept09_deptno_pk primary key,
84 SQL>     dname varchar2(15) constraint dept09_dname_nn not null,
85 SQL>     loc varchar2(15)
86 SQL> );
87
88 SQL>
```

5장

1교시 : 뷰

|    |  |   |
|----|--|---|
| 1  | 뷰(view)는 물리적인 테이블을 근거한 논리적인 가상 테이블               |   |
| 2  | 뷰를 사용하는 이유:                                      |   |
| 3  | (1) 복잡하고 긴 쿼리문을 뷰로 정의하면 접근을 단순화시킬 수 있음           |   |
| 4  | (2) 보안에 유리                                       |   |
| 5  | create, drop, alter(x)                           |   |
| 6  |  |   |
| 7  | SQL> <u>create view emp_view30</u>               | emp_view30뷰 테이블 생성                        |
| 8  | SQL> <u>as select empno, ename, deptno</u>       |   |
| 9  | SQL> <u>from emp</u>                             |   |
| 10 | SQL> <u>where deptno=30;</u>                     |   |
| 11 | SQL>   |   |
| 12 | SQL> desc emp_view30                             |   |
| 13 | SQL>   | 구조확인                                      |
| 14 | SQL> select * from emp_view30                    |   |
| 15 | SQL>   |   |
| 16 | SQL>   |   |
| 17 | SQL> desc user_views;                            |   |
| 18 | SQL> select view_name from user_views;           | 데이터디렉터리 사용자뷰의 구조확인                        |
| 19 | SQL> select view_name, text from user_views;     | text에는 서브쿼리문이 저장되어 있음                     |
| 20 | SQL>   |   |
| 21 | SQL> <u>insert into emp_view30</u>               |   |
| 22 | SQL> <u>values(1111,'AAA',30);</u>               |   |
| 23 | SQL>   |   |
| 24 | SQL> select * from emp_view30;                   | 뷰테이블 뿐만 아니라 기본테이블에도                       |
| 25 | SQL> select * from emp;                          | 추가 되었음을 확인할 수 있음                          |
| 26 | SQL>   |   |
| 27 | SQL>   |   |
| 28 | SQL> <u>create view emp_view10(사원번호,이름,부서번호)</u> | 컬럼명을 명시해서 뷰생성                             |
| 29 | SQL> <u>as select empno, ename, deptno</u>       |   |
| 30 | SQL> <u>from emp</u>                             |   |
| 31 | SQL> <u>where deptno=10;</u>                     |   |
| 32 | SQL>   |   |
| 33 | SQL> select * from emp_view10;                   |   |
| 34 | SQL> desc emp_view10;                            | 구조확인하면 명시한 컬럼명                            |
| 35 | SQL>   |   |
| 36 | SQL>   |   |
| 37 | SQL>   | 예러  |
| 38 | SQL> <u>create view dept_sum</u>                 |   |
| 39 | SQL> <u>as select deptno, sum(sal) →</u>         | as select deptno, sum(sal) <u>sum_sal</u> |
| 40 | SQL> <u>from emp</u>                             | 컬럼 별칭을 지정해야 함                             |
| 41 | SQL> <u>group by deptno;</u>                     |   |
| 42 | SQL>   |   |
| 43 | SQL>   |   |



## 2교시 : 복합 뷰

```

1  복합 뷰는 두개 이상의 기본 테이블에 의해 정의한 뷰
2  SQL> select e.empno, e.ename, d.dname
3  SQL> from emp e, dept d
4  SQL> where e.deptno = d.deptno;
5  SQL>
6  SQL> create view emp_view_join
7  SQL> as
8  SQL> select e.empno, e.ename, d.dname
9  SQL> from emp e, dept d
10 SQL> where e.deptno = d.deptno;
11 SQL>
12 SQL>
13 SQL> select * from emp_view_join;
14 SQL>
15 SQL> [뷰의 제거]
16 SQL> drop view emp_view30;
17 SQL> drop view emp_view10;
18 SQL> drop view emp_view_join;
19 SQL>
20 SQL>
21 SQL>
22 SQL> [뷰의 변경]
23 SQL> 이미 존재하는 뷰는 삭제하고 다시 생성할 수 있음(create or replace)
24 SQL> create view emp_view_join
25 SQL> as
26 SQL> select
27 SQL>
28 SQL> create or replace view emp_view_join
29 SQL> as select e.ename, d.dname, d.loc
30 SQL> from emp e, dept d
31 SQL> where e.deptno=d.deptno;
32 SQL>
33 SQL>

```

### 3교시 : 시퀀스

|    |   |           |
|----|---|-----------|
| 1  | 기본 키는 유일한 값을 가져야 하므로 직접 기본키를 생성하는 것이 부담이 클수 있다. 시퀀스는 테이블 내              |           |
| 2  | 의 유일한 숫자를 자동으로 생성하는 자동 번호 발생기이므로 시퀀스를 기본 키로 사용하게 되면 사용자의 부              |           |
| 3  | 담을 줄일 수 있다.   |           |
| 4  | - increment by 옵션 : 증가치를 지정   |           |
| 5  | - start with 옵션 : 시작 값을 지정  |           |
| 6  | - maxvalue 옵션 : 최대값을 지정   |           |
| 7  | - minvalue 옵션 : 최소값을 지정   |           |
| 8  | - cycle 옵션 : 최대값까지 증가가 완료되게 되면 다시 start with 옵션에 지정한 시작 값에서 시작          |           |
| 9  | - cache 옵션 : 메모리상에서 시퀀스 값을 관리. 기본값은 20. nocache는 원칙적으로 메모리상 관리x         |           |
| 10 |   |           |
| 11 | 시퀀스는 대부분 insert 연산과 같이 사용되어 컬럼 값을 자동으로 발생시키는 용도로 사용                     |           |
| 12 | SQL> create sequence dept_deptno_seq                                    |           |
| 13 | SQL> increment by 10  |           |
| 14 | SQL> start with 10;   |           |
| 15 | SQL>  |           |
| 16 | SQL> SELECT dept_deptno_seq.nextval FROM dual;                          | 10        |
| 17 | SQL> SELECT dept_deptno_seq.curval FROM dual;                           | 10        |
| 18 | SQL> SELECT dept_deptno_seq.nextval FROM dual;                          | 20        |
| 19 | SQL> SELECT dept_deptno_seq.nextval FROM dual;                          | 30        |
| 20 | SQL> SELECT dept_deptno_seq.curval FROM dual;                           | 30        |
| 21 | SQL>  |           |
| 22 | SQL>  |           |
| 23 | SQL> alter sequence dept_deptno_seq maxvalue 50;                        | 시퀀스의 수정   |
| 24 | SQL>  |           |
| 25 | SQL> desc user_sequences;   | 데이터딕셔너리   |
| 26 | SQL> select sequence_name, max_value, increment_by from user_sequences; | 중 사용자 시퀀스 |
| 27 | SQL>  |           |
| 28 | SQL>  |           |
| 29 | SQL> drop sequence dept_deptno_seq ;                                    | 시퀀스 삭제    |
| 30 | SQL>  |           |
| 31 | SQL>  |           |
| 32 | SQL> create table dept5   |           |
| 33 | SQL> (  |           |
| 34 | SQL> deptno number(4) primary key,                                      |           |
| 35 | SQL> dname varchar(15),   |           |
| 36 | SQL> loc varchar(15)  |           |
| 37 | SQL> );   |           |
| 38 | SQL> desc dept5;  |           |
| 39 | SQL>  |           |
| 40 | SQL> create sequence dept5_deptno_seq                                   | 시작 10 부터  |
| 41 | SQL> increment by 10  | 10씩 증가    |
| 42 | SQL> start with 10  | 다시시작안됨    |
| 43 | SQL>  |           |
| 44 | SQL> nocycle;   |           |

|      |  |
|------|--|
| SQL> |  |
| SQL> | insert into dept5                              |
| SQL> | values(dept5_deptno_seq.nextval, '인사과', '서울'); |
| SQL> | insert into dept5                              |
| SQL> | values(dept5_deptno_seq.nextval, '경리과', '서울'); |
| SQL> |  |
| SQL> | select * from dept5;                           |

#### 4교시 : 오라클의 데이터형

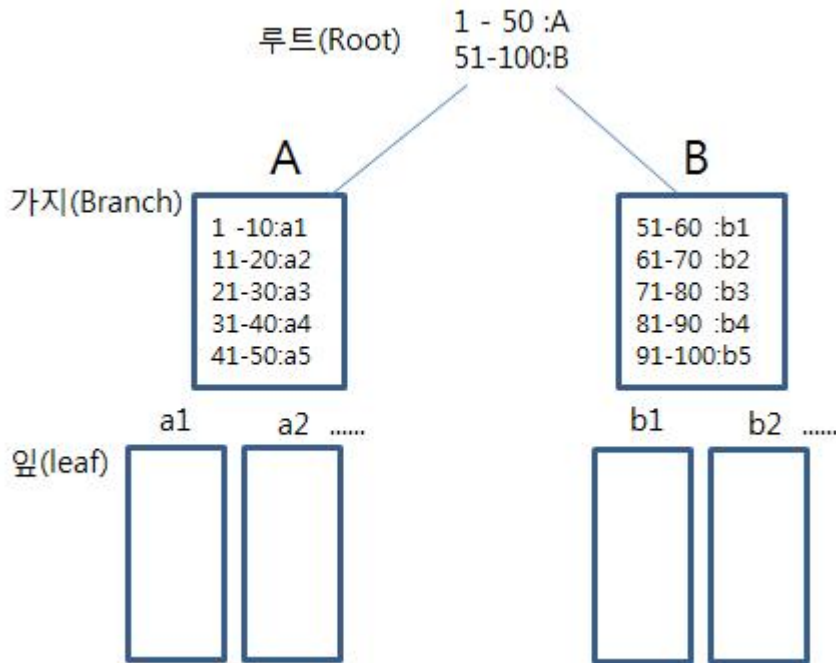
|    |   |                  |
|----|---|------------------|
| 1  | DESC 명령어로 테이블의 구조를 살펴보면 데이터의 형이 number와 varchar2 대부분이다                    |                  |
| 2  | - number(5,2) :전체 5자리에 소수이하 2자리.  |                  |
| 3  | - number(5) :전체 5자리.  |                  |
| 4  | - number : 입력한 데이터 값만큼 공간할당   |                  |
| 5  | - date : YY/MM/DD 형식으로 출력. 한글판 영문판의 형식에 따라 DD/MON/YY 형식을 출력               |                  |
| 6  | - varchar2(20) :가변길이 문자열  |                  |
| 7  | - char(20) :고정길이 문자열  |                  |
| 8  | - timestamp(n) : date형의 확장된 형태  |                  |
| 9  | - interval year(년도자리수) to month(달의자리수) : 년과 월을 이용하여 기간을 저장                |                  |
| 10 | - interval day(일수자리수) to second(초의자리수) : 일,시,분,초를 이용하여 기간을 저장. 두 날짜 값의 정확 |                  |
| 11 | 한 차이를 표현. 자리수를 지정하지 않으면 기본적으로 2자리   |                  |
| 12 | SQL> select sysdate from dual;  |                  |
| 13 | SQL>  |                  |
| 14 | SQL> create table sam02 (   | year01 필드는       |
| 15 | SQL> year01 interval year to month );                                     | 인터발 연부터 달까지      |
| 16 | SQL>  |                  |
| 17 | SQL> insert into sam02 values(interval '36' month );                      | 인터발 36개월 입력시     |
| 18 | SQL> select year01 from sam02;  | +03-00 → (3년 표현) |
| 19 | SQL>  |                  |
| 20 | SQL> insert into sam02 values(interval '20' month );                      | 인터발 20개월 입력시     |
| 21 | SQL> select year01 from sam02;  | +01-0 →(1년8개월표현) |
| 22 | SQL>  |                  |
| 23 | SQL>  |                  |
| 24 | SQL> select year01, sysdate, sysdate+year01 from sam02;                   |                  |
| 25 | SQL>  |                  |
| 26 | SQL> create table sam03 (   | day01 필드는        |
| 27 | SQL> day01 interval day(3) to second );                                   | 인터발 날짜부터 초까지     |
| 28 | SQL>  |                  |
| 29 | SQL> insert into sam03 values(interval '100' day(3) );                    | 인터발 100일         |
| 30 | SQL> select day01 from sam03;   | 100초             |
| 31 | SQL> insert into sam03 values(interval '100' second);                     | 100분             |
| 32 | SQL> insert into sam03 values(interval '100' minute);                     | 100시간            |
| 33 | SQL> insert into sam03 values(interval '100' hour);                       | +100 00:00:00.00 |
| 34 | SQL>  |                  |
| 35 | SQL>  |                  |
| 36 | SQL> select day01, sysdate, sysdate+day01 from sam03;                     |                  |

## 5교시 : 인덱스

1 오라클에서 검색을 빠르게 하기 위해 인덱스를 제공하고 있음. 내부 구조는 B\* 트리형식을 구성됨  
 2 인덱스란 sql명령문의 처리 속도를 향상시키기 위해서 컬럼에 대해서 생성하는 오라클 객체이다.  
 3 <순차적 저장>

| 1  | 2  | 3  | 4 | 5  | 6  | 7  | 8  | 9  | ... |
|----|----|----|---|----|----|----|----|----|-----|
| 50 | 61 | 25 | 5 | 11 | 47 | 49 | 82 | 15 |     |

7 <B\* 트리형식 저장>



```

24 SQL> create table e2
25 SQL> as select * from emp;
26 SQL> select * from e2;
27 SQL>
28 SQL> insert into e2 select * from e2; (여러번 반복)
29 SQL> insert into e2 select * from e2; (여러번 반복)
30 SQL> insert into e2 select * from e2; (여러번 반복)
31 SQL> insert into e2 select * from e2; (여러번 반복)
32 SQL> select * from e2;
33 SQL>
34 SQL>
35 SQL> set timing on
36 SQL> select distinct empno, ename from e2 where ename='FORD'
37 SQL>
38 SQL> create index idx_e2_ename
39 SQL> on e2(ename)
40 SQL>
41 SQL>
42 SQL> select index_name, table_name
43 SQL> from user_indexes
44 SQL> where table_name in('EMP', 'DEPT', 'E2');
```

800,000 정도의 행 생성

시간을 확인

인덱스명 idx\_e2\_ename  
e2테이블의 ename에 대  
해서 인덱스 생성

데이터 디크너리  
user\_indexes 확인

|    |  |   |
|----|--|---|
| 45 | SQL> alter index idx_e2_ename rebuild; | 자료가 많이 변경되면<br>인덱스효율은 떨어진다<br>한번씩 재생성을 해주어야 함 |
| 46 | SQL>                                   |   |
| 47 | SQL> drop index idx_e2_ename ;         |   |
| 48 | SQL>                                   |   |
| 49 | SQL> create table d2                   |   |
| 50 | SQL> as select * from dept;            |   |
| 51 | SQL>                                   |   |
| 52 | SQL>                                   |   |
| 53 | SQL> insert into d2                    |   |
| 54 | SQL> values(50, '인사과','서울');           |   |
| 55 | SQL>                                   |   |
| 56 | SQL> insert into d2                    |   |
| 57 | SQL> values(60, '총무과','대전');           |   |
| 58 | SQL>                                   |   |
| 59 | SQL> insert into d2                    |   |
| 60 | SQL> values(70, '교육팀', '대전');          |   |
| 61 | SQL>                                   |   |
| 62 | SQL>                                   |   |
| 63 | SQL> select * from d2;                 |   |
| 64 | SQL>                                   |   |
| 65 | SQL> create unique index idx_d2_deptno | 고유인덱스를 지정                                     |
| 66 | SQL> on d2(deptno);                    |   |
| 67 | SQL>                                   |   |
| 68 | SQL> create unique index idx_d2_loc    | 중복된 데이터를 갖는<br>컬럼을 인덱스 지정하면<br>오류 발생          |
| 69 | SQL> on d2(loc);                       |   |
| 70 | SQL>                                   |   |
| 71 | SQL> create index idx_d2_com           |   |
| 72 | SQL> on d2(deptno, dname);             |   |
| 73 | SQL>                                   |   |
| 74 | SQL>                                   |   |
| 75 | SQL> select index_name, table_name     |   |
| 76 | SQL> from user_indexes;                |   |