



# SQL,PL/SQL(ORACLE)

## 1. 환경설정 SELECT WHERE 연산자

## [[ Part 1 - 설치 및 접근법 ]]

### 1. 오라클 설치(11g or 11gExpress)

#### (1) 디렉토리 이름이 한글이면 X

- 1) oracle 설치파일이 존재하는 경로에 한글X
- 2) oracle 설치경로에 한글X

#### (2) 디렉토리에 공백이 있으면 X

- 1) oracle 설치파일이 존재하는 경로에 공백X
- 2) oracle 설치경로에 공백X

#### (3) 설치시에 에러 해결 방법

- 1) OS부터 밀고 다시 설치
- 2) oracle를 다시 설치
  - <1> 실행 -> regedit 에서 oracle관련 파일을 제거
  - <2> oracle 설치 디렉토리를 삭제
  - <3> 다시 설치



## 2. 오라클 접근법

### (1) sqlplus

- 1) oracle Applicatoion이용법  
(oracle -> application development -> SQL Plus)
- 2) 도스창 이용법  
(실행 -> sqlplus scott/tiger )

### (2) isqlplus 이용법 (WEB)

- (1) oracle의 Apache web server 작동  
프로그램 -> oracle -> Oracle HttpServer -> 시작
- (2) 접근 : http://ip/isqlplus

### (3) utility application 이용법

- (1) toad
- (2) orange
- (3) sqlgate
- .....



## [[ Part 2 - ORACLE의 개요 ]]

### 1. DBMS ( DataBase Management System )의 개념

#### (1) DB(DataBase)

- 지속적으로 유지관리해야 할 데이터의 집합

#### (2) DBMS

- DB를 편리하게 관리하고, 효율적으로 저장하고  
검색할 수 있는 환경을 제공하는 시스템소프트웨어  
를 의미 (ex: oracle, ms-sql, mysql, db2, ....)

### 2. SQL ( Structured Query Language )

데이터를 Access하기 위해 DBMS와 통신하는 언어

### 3. 기본 사용자 계정

(1) SYS : 오라클 super사용자 ID이며, 데이터베이스에서  
발생하는 모든 문제를 처리할 수 있는 권한

(2) SYSTEM : SYS계정과 같은데, 차이는 데이터베이스를  
생성할 수 있는 권한이 없다.

(3) SCOTT : 처음 오라클을 사용하는 user들을 위한  
SAMPLE 계정이며, 일반적인 프로그램을 작성할 때  
사용되는 계정이다.(for developer)

(4) HR : SAMPLE 사용자 계정

### [ 계정권한부여 ]

```
SQL> conn /as sysdba
```

```
SQL> alter user HR identified by hi123456 account unlock;
```



## 4. 주요 용어

- (1) **TABLE** : 관계형 DB에서 기본 데이터 저장구조로써 Entity(실체)의 집합저장소  
(ex: JOBS, DEPARTMENTS, EMPLOYEES, LOCATIONS, .. )
- (2) **ROW** : 테이블의 행 (하나의 유효 데이터)
- (3) **COLUMN** : 테이블의 열명  
(ex: DEPARTMENT\_ID, DEPARTMENT\_NAME, LOCATION\_ID)
- (4) **PRIMARY(PRIMARY-KEY)**  
테이블에서 각 ROW를 유일하게 구분하는 COLUMN  
(ex: DEPARTMENTS 테이블의 DEPTNO컬럼 )
- (5) **FOREIGN-KEY**  
다른 테이블의 COLUMN값을 참조하는 테이블의 COLUMN  
(ex: EMPLOYEES 테이블의 DEPTNO컬럼 )
- (6) **FIELD** : 테이블에서 ROW와 COLUMN이 교차하는 데이터  
(ex: DEPARTMENTS 테이블의 SALES라는 값 )
- (7) **NULL** : 데이터가 존재하지 않는 FIELD

cf) HR계정의 table

- DEPARTMENTS : 부서 테이블 (부모 테이블)
- EMPLOYEES : 사원 테이블 (자식 테이블)



## 5. SQL ( Structured Query Language )

- (1) **DQL** (Data Query Language) -> 데이터 질의어  
-> 테이블내의 데이터를 조회할 때 사용  
(ex: SELECT )
- (2) **DML** (Data Manipulation(조작) Language)  
-> 테이블의 데이터를 입력,수정,삭제할 때 사용  
(ex: INSERT, UPDATE, DELETE )
- (3) **DDL** (Data Definition Language)  
-> 테이블등의 객체를 생성, 변경, 삭제할 때 사용  
(ex: CREATE, ALTER, DROP, .... )
- (4) **TCL** (Transaction Control Language)  
-> 테이블내의 DML문을 DB에 저장 or 취소할 때 사용  
(ex: COMMIT, ROLLBACK, SAVEPOINT )
- (5) **DCL** (Data Control Language)  
-> DB사용자에게 권한을 부여하는 or 취소할 때 사용  
(ex: GRANT, REVOKE )

## 6. PL-SQL

제어문(조건문, 반복문)이 들어있는 SQL로써 오라클 DBMS  
에서 지원하는 확장된 SQL



## [[ Part 3 - SQL ]]

### 1. DQL

#### (1) 기본구조

// select 컬럼명1, 컬럼명2, ... from 테이블명  
where 조건절 order by 기준컬럼명 [ asc/desc ]

SQL> select \* from tab;

SQL> select \* from departments;

SQL> select EMPLOYEE\_ID, JOB\_ID, DEPARTMENT\_ID from employees;

<참고 >

- 테이블정보 얻어오기 : SQL> desc employees;

#### (2) DISTINCT(중복제거==unique와 같다) / ALL

SQL> select distinct JOB\_ID from employees;

SQL> select all JOB\_ID from employees;

( == select JOB\_ID from employees)

#### (3) ORDER BY (정렬)

select EMPLOYEE\_ID, SALARY from employees;

select EMPLOYEE\_ID, SALARY from employees order by SALARY;

select EMPLOYEE\_ID, SALARY from employees order by SALARY asc;

select EMPLOYEE\_ID, SALARY from employees order by SALARY desc;

select JOB\_ID, SALARY from employees order by JOB\_ID, SALARY;

select JOB\_ID, SALARY from employees order by JOB\_ID asc, SALARY asc;

select JOB\_ID, SALARY from employees order by JOB\_ID, SALARY desc;

select JOB\_ID, SALARY from employees order by JOB\_ID desc, SALARY desc;

#### (4) ALIAS (별칭)

```
SQL> select JOB_ID "직업", SALARY "봉급" from employees;
SQL> select JOB_ID 직업, SALARY "봉급" from employees;
SQL> select JOB_ID "직업", SALARY "봉급  #@" from employees;
SQL> select JOB_ID, SALARY AS "봉급 ₩천원" from employees;
SQL> select JOB_ID id, SALARY AS 봉급 from employees;
SQL> select JOB_ID id, SALARY AS "1004" from employees;
```

cf) 숫자or 특수문자 ALIAS는 반드시 ""안에 넣어 줘야 한다.

#### (5) WHERE

```
SQL> select EMPLOYEE_ID, DEPARTMENT_ID
      from employees where DEPARTMENT_ID =10;
SQL> select SALARY , EMPLOYEE_ID
      from employees where EMPLOYEE_ID =30;
SQL> SELECT SALARY , DEPARTMENT_ID
      FROM employees WHERE DEPARTMENT_ID =30
      ORDER BY SALARY DESC;
SQL> SELECT JOB_ID , FIRST_NAME
      FROM employees WHERE JOB_ID ='IT_PROG'
      ORDER BY FIRST_NAME ;
SQL> SELECT EMPLOYEE_ID, FIRST_NAME, HIRE_DATE
      FROM employees WHERE HIRE_DATE = '07/12/07';
```





## (6) 연산자 종류

### 1) 산술 연산자 ( \*, /, +, - )

```
SQL> SELECT ename, sal*1.1 FROM emp WHERE deptno=10;
```

### 2) 비교 연산자( =, !=, >, >=, <, <= )

```
SQL> SELECT ename, sal FROM emp WHERE sal=950;
```

```
SQL> SELECT ename, sal FROM emp WHERE sal>=3000;
```

```
SQL> SELECT ename, sal FROM emp WHERE sal<3000;
```

```
SQL> SELECT deptno, sal FROM emp WHERE deptno!=30;
```

### 3) 논리 연산자 ( AND, OR, NOT )

```
SQL> SELECT deptno, ename, sal FROM emp  
WHERE deptno=20 AND sal>=3000;
```

```
SQL> SELECT job, deptno FROM emp  
WHERE job='SALESMAN' AND deptno=30;
```

```
SQL> SELECT ename, sal FROM emp  
WHERE sal<1000 OR sal>=4000;
```

```
SQL> SELECT ename, sal FROM emp  
WHERE NOT (sal<1000 OR sal>=4000);
```

```
SQL> SELECT deptno, ename, sal FROM emp  
WHERE NOT deptno=30;
```



#### 4) 집합 연산자 (UNION, UNION ALL, INTERSECT, MINUS )

**UNION : 두 테이블 더해서 중복은 제거하고 출력**

SQL>

```
SELECT EMPLOYEE_ID, SALARY FROM EMPLOYEES  
WHERE SALARY >= 20000 OR SALARY <= 2100
```

**UNION**

```
SELECT EMPLOYEE_ID, SALARY FROM EMPLOYEES  
WHERE SALARY <= 2200;
```

**UNION ALL : 두 테이블 더해서 중복포함 출력**

SQL>

```
SELECT EMPLOYEE_ID, SALARY FROM EMPLOYEES  
WHERE SALARY >= 20000 OR SALARY <= 2100
```

**UNION ALL**

```
SELECT EMPLOYEE_ID, SALARY FROM EMPLOYEES  
WHERE SALARY <= 2200;
```



## **INTERSECT : 두 테이블 교집합 출력**

SQL>

```
SELECT EMPLOYEE_ID,SALARY FROM EMPLOYEES  
WHERE SALARY >=20000 OR SALARY <=2100
```

### **INTERSECT**

```
SELECT EMPLOYEE_ID,SALARY FROM EMPLOYEES  
WHERE SALARY <=2200;
```

## **MINUS : 앞 결과에서 뒤 결과 빼고 출력 앞뒤 바뀌면 결과도 다르다.**

SQL>

```
SELECT EMPLOYEE_ID,SALARY FROM EMPLOYEES  
WHERE SALARY >=20000 OR SALARY <=2100
```

### **MINUS**

```
SELECT EMPLOYEE_ID,SALARY FROM EMPLOYEES  
WHERE SALARY <=2200;
```

```
SELECT EMPLOYEE_ID,SALARY FROM EMPLOYEES  
WHERE SALARY <=2200
```

### **MINUS**

```
SELECT EMPLOYEE_ID,SALARY FROM EMPLOYEES  
WHERE SALARY >=20000 OR SALARY <=2100;
```



5) IN,OR,AND, ANY, ALL, BETWEEN, LIKE, IS NULL, IS NOT NULL, EXISTS,...

```
SQL> SELECT DEPARTMENT_ID, SALARY FROM employees  
      WHERE DEPARTMENT_ID IN(10, 20, 100);  
      // WHERE DEPARTMENT_ID =10 OR DEPARTMENT_ID =20 OR  
      DEPARTMENT_ID =100 와 같다
```

```
SQL> SELECT DEPARTMENT_ID, SALARY FROM employees  
      WHERE SALARY >= ANY(8000,12000,20000)  
      // 해당 값들로 하나하나씩 기준잡고 싶을때
```

```
SQL> SELECT JOB_ID , FIRST_NAME, SALARY FROM employees  
      WHERE SALARY >= 10000 AND SALARY <= 12000;
```

```
SQL> SELECT JOB_ID , FIRST_NAME, SALARY FROM employees  
      WHERE SALARY BETWEEN 10000 AND 12000;
```

```
SQL> SELECT last_name FROM employees WHERE last_name  
      BETWEEN 'Chen' AND 'Dilly' ORDER BY last_name;
```

```
SQL> SELECT last_name FROM employees  
      WHERE NOT last_name = 'Dilly' ORDER BY last_name;
```



```
SQL> SELECT last_name FROM employees
      WHERE last_name LIKE 't%';
SQL> SELECT last_name FROM employees
      WHERE last_name LIKE '%y%';
SQL> SELECT last_name FROM employees
      WHERE last_name LIKE '__e%';
SQL> SELECT last_name FROM employees
      WHERE last_name LIKE '%es';
SQL> SELECT last_name FROM employees
      WHERE last_name LIKE '%Di%';
SQL> SELECT last_name , COMMISSION_PCT FROM employees
      WHERE COMMISSION_PCT IS NULL;
SQL> SELECT last_name , COMMISSION_PCT FROM employees
      WHERE COMMISSION_PCT IS NOT NULL;
//대소문자 구분한다.주의할것.
```

## 6) 결합연산자 ( || )

```
SQL> SELECT first_name||" "||last_name ||'의 연봉'|| SALARY ||' '
      FROM employees ;
SQL> SELECT last_name ||'의 연봉은 '|| SALARY ||'입니다.'
      FROM employees ;
SQL> SELECT last_name || 111111 || SALARY || 22222
      FROM employees ;
```



## (7) 연산자 우선순위

- 1) 1순위 : ( ) > 산술연산자 > 연결연산자( || ) > 비교연산자
- 2) 2순위 : IS NULL, LIKE, IN > BETWEEN
- 3) 3순위 : NOT
- 4) 4순위 : AND
- 5) 5순위 : OR

```
SQL> SELECT EMPLOYEE_ID FROM employees  
      WHERE NOT (SALARY > 1000 AND SALARY < 12000);  
SQL> SELECT EMPLOYEE_ID FROM employees  
      WHERE NOT SALARY > 1000 AND SALARY < 12000;
```

## (8) sqlplus 에서 SQL문장의 실행 방법

- 1) 문장의 끝에 ;을 붙여주고 enter!
- 2) 문장의 다음 라인에 /를 붙여주고 enter!  
SQL> SELECT deptno FROM dept  
2 /
- 3) 직전의 버퍼에 담겨있는 SQL문장을 실행  
SQL> run 또는 r
- 4) 편집창 호출  
SQL> ed



(9)패턴검색시 사용되는 특수기호 : % \_(언더바)

참고 : 언더바 패턴이 이미 들어있는 데이터의 경우(예: AX\_YA)는 like '%A구분자\_%' escape '구분자' 형식으로 검색해야 한다.

-구분자는 보통 ₩와 같은 자주 안쓰이는 것으로 사용 할 수 있으며 정수도 쓸 수 있다.

```
drop table test2;  
create table test2(fname varchar2(20));
```

```
insert into test2 values('S1234');  
insert into test2 values('1234S_1234');  
insert into test2 values('$S_1234');  
insert into test2 values('sssS_1234');  
insert into test2 values('THE X₩_Y');  
insert into test2 values('THE1 X_Y');  
insert into test2 values('THE X₩&Y');  
insert into test2 values('THE X&Y');  
insert into test2 values('THE2 X_Y');
```

```
select * from test2;  
select fname from test2 where fname like '%S₩_%' escape '₩';  
select fname from test2 where fname like '%S$_%' escape '$';  
select fname from test2 where fname like '%X5_Y%' escape '5';
```

