# 2. SQL문

## 1) SQL

SQL이란 Structured Query Language(구조적 질의 언어)의 줄임말로, 관계형 데이터베이스 시스템(RDBMS)에서 자료를 관리 및 처리하기 위해 설계된 언어이다. 현재는 SQL의 표준으로 ANSI SQL이 정립되었으며, 오늘날 여러 DBMS프로그램에서 ANSI SQL을 기반으로 사용하고 있으나, 각 프로그램 별로 조금씩 차이가 있는 개별 SQL을 사용하고 있다.

또한 일반적인 프로그래밍 언어(범용 언어)와 달리 대화식 언어이기에 명령문이 짧고 간결한 것이 특징이며 SQL 자체는 범용 언어에 비해 한계가 있기 때문에, 단독으로 사용하기 보단 C#, Java, Python, PHP와 같은 고수준 언어와 함께 쓰는 것이 일반적이다.

### (1) SQL문의 종류

SQL문법은 크게 3가지 종류로 나누어 진다.

- **DML(Data Manipulation Language) :** DB를 검색, 수정, 입력, 삭제하는 데이터 조작어**.** 가장많이 사용된다.

  **INSERT** : DB 객체에 데이터를 입력
  **DELETE** : DB 객체에 데이터를 삭제
  **UPDATE** : 기존에 존재하는 DB 객체 안의 데이터수정
  **SELECT** : DB 객체로부터 데이터를 검색

- **DDL(Data Definition Language) :** DB객체를 생성, 삭제, 변경하는 데이터 정의어, DB설계단계에서 주로 사용된다.

  **CREATE :** DB 객체 생성
  **DROP** : DB 객체를 삭제
  **ALTER** : 기존에 존재하는 DB 객체를 다시 정의
  **TRUNCATE** : DB 객체 내용 삭제

- **DCL(Data Control Language) :** DB 사용자의 권한 제어,관리 하는 데이터 제어어.

  **GRANT** : DB 객체에 권한 부여
  **REVOKE** : 이미 부여된 DB 객체의 권한을 취소
  **COMMIT** : 트랜잭션(* DB의 상태를 변화시키기 해서 수행하는 작업의 단위) 확정 (TCL)
  **ROLLBACK** : 트랜잭션 취소 (TCL)
  **SAVEPOINT** : 복귀지점 설정 (TCL)

### (2) SQL문의 특성

- SQL문은 대소문자를 구별하지 않는다

  (단 서버환경이나 DBMS종류에 따라 DB또는 필드명에대해 대소문자를 구별한다.)

- SQL문은 한 줄 또는 여러 줄에 입력될 수 있다.
- 절은 보통 읽고 편집하기 쉽게 줄을 나누도록 한다.(권장)
- 고유의 값은 따옴표(' ')로 감싸준다.

  (ex. SELECT * FROM EMP WHERE NAME = 'JAMES';)

- SQL문 명령어의 끝은 반드시 세미콜론(;)으로 끝나야 한다.
- 주석처리는 --로 한다.

## 2) SELECT 문

SELECT문은 모든 DML문에서 가장 많이 쓰이는 데이터를 검색하는 문법이며 일반적인 개발자들이 가장 많이 접하고 사용하게 될 문이다

### (1) 기본 형식

```
SELECT      -- SELECT절에는 필요한 데이터의 칼럼명을 쓴다.
FROM        -- FROM 절에는 데이터를 가져올 테이블명을 쓴다
WHERE       -- WHERE 절에는 가져올 데이터의 조건을 쓴다. 조건이 없으면 생략한다.
ORDER BY    -- ORDER BY절에는 명시한 칼럼명으로 데이터를 정렬해준다. ;

```

EX)

```
SHOW USER;              -- 현재 접속된 계정을 보여준다
SELECT * FROM TAB;      -- 현 계정(scott)이 가지고 있는 테이블   * 는  모든필드를 뜻하는 특수문자
SELECT * FROM EMP;      -- EMP 테이블의 모든 열(필드), 모든 행
SELECT * FROM DEPT;     -- DEPT테이블의 모든 열, 모든 행
```

```
SELECT EMPNO, ENAME, SAL, JOB FROM EMP;  -- EMP 테이블에 있는 EMPNO, ENAME, SAL 칼럼 출력
```

결과)

| | EMPNO | ENAME | SAL | JOB |
|---|---|---|---|---|
| 1 | 7369 | SMITH | 800 | CLERK |
| 2 | 7499 | ALLEN | 1600 | SALESMAN |
| 3 | 7521 | WARD | 1250 | SALESMAN |
| 4 | 7566 | JONES | 2975 | MANAGER |
| 5 | 7654 | MARTIN | 1250 | SALESMAN |
| 6 | 7698 | BLAKE | 2850 | MANAGER |
| 7 | 7782 | CLARK | 2450 | MANAGER |
| 8 | 7788 | SCOTT | 3000 | ANALYST |
| 9 | 7839 | KING | 5000 | PRESIDENT |
| 10 | 7844 | TURNER | 1500 | SALESMAN |
| 11 | 7876 | ADAMS | 1100 | CLERK |
| 12 | 7900 | JAMES | 950 | CLERK |
| 13 | 7902 | FORD | 3000 | ANALYST |
| 14 | 7934 | MILLER | 1300 | CLERK |

### (2) 별칭(ALIAS)

실행 시 출력되는 칼럼(열)에 별칭(ALIAS)을 지정할 수도 있다.

EX)

```
SELECT EMPNO AS "사 번", ENAME AS "이름", SAL AS "급여", JOB FROM EMP; -- 별칭(alias)
SELECT EMPNO AS 사번, ENAME AS 이름, SAL AS 급여, JOB FROM EMP;
    -- ""로 지정하지 않아도 무관하나 ""를 쓰는것이 가독성이 좋다.
SELECT EMPNO 사번, ENAME 이름, SAL 급여, JOB FROM EMP;
    -- 문법상 AS명령어로 별칭이 올것을 알려주지만 생략 가능하다.
SELECT EMPNO "사 번", ENAME "이 름", SAL "급 여", JOB FROM EMP;
    -- 별칭에 공백이 있으면 ""로 지정해 줘야한다.
SELECT EMPNO NO, ENAME NAME, SAL SALARY FROM EMP;
    -- 별칭 : NO, NAME, SALARY
```

| | 사번 | 이름 | 급여 | JOB |
|---|---|---|---|---|
| 1 | 7369 | SMITH | 800 | CLERK |
| 2 | 7499 | ALLEN | 1600 | SALESMAN |
| 3 | 7521 | WARD | 1250 | SALESMAN |
| 4 | 7566 | JONES | 2975 | MANAGER |
| 5 | 7654 | MARTIN | 1250 | SALESMAN |
| 6 | 7698 | BLAKE | 2850 | MANAGER |
| 7 | 7782 | CLARK | 2450 | MANAGER |
| 8 | 7788 | SCOTT | 3000 | ANALYST |
| 9 | 7839 | KING | 5000 | PRESIDENT |
| 10 | 7844 | TURNER | 1500 | SALESMAN |
| 11 | 7876 | ADAMS | 1100 | CLERK |
| 12 | 7900 | JAMES | 950 | CLERK |

### (3) 비교연산자

특정 행을 출력하는 WHERE절에는 조건을 위한 비교연산자를 하며 ORACLE에서는 숫자, 문자, 날짜형 모두 비교연산자가 가능하며 문자는 해당 문자의 아스키코드 크기로 구분한다.

- 같다(=), 크거나 같다(>=), 작거나 같다(<=), 다르다(!=, ^=, <>)

EX)

```
SELECT EMPNO "사번", ENAME "이름", SAL "급여" FROM EMP WHERE SAL = 3000; -- 같다
  -- SAL이 3000과 같은 데이터의 EMPNO, ENAME, SAL 정보를 EMP 테이블로부터 가져온다
SELECT EMPNO "사번", ENAME "이름", SAL "급여" FROM EMP WHERE SAL != 3000; -- 다르다
SELECT EMPNO "사번", ENAME "이름", SAL "급여" FROM EMP WHERE SAL ^= 3000; -- 다르다
SELECT EMPNO "사번", ENAME "이름", SAL "급여" FROM EMP WHERE SAL <> 3000; -- 다르다
```

결과)

| | 사번 | 이름 | 급여 |
|---|---|---|---|
| 1 | 7788 | SCOTT | 3000 |
| 2 | 7902 | FORD | 3000 |

EX)

```
-- ex. 사원이름(ENAME)이 'A','B','C'로 시작하는 사원의 모든 필드
SELECT * FROM EMP WHERE ENAME < 'D';

-- ex. 81년도 이전에 입사한 사원의 모든 필드
SELECT * FROM EMP WHERE HIREDATE < '81/01/01';

-- ex. 10번 부서번호(deptno)인 사원의 모든 필드를 출력
SELECT * FROM EMP WHERE DEPTNO=10;

-- ex. 이름(ENAME)이 FORD인 직원의 EMPNO, ENAME, MGR(상사의 사번)을 출력
SELECT EMPNO, ENAME, MGR FROM EMP WHERE ENAME = 'FORD';
select empno, ename, mgr from emp where ename = 'FORD'; -- 데이터는 대소문자 구분
```

### (4) 논리연산자

ORACLE에서는 AND, OR, NOT의 논리연산자가 있다.

EX)

```
-- ex. 급여(SAL)가 2000이상 3000이하인 직원의 모든 필드
SELECT * FROM EMP WHERE SAL>=2000 AND SAL<=3000;

-- ex. 82년도에 입사한 사원의 모든 필드
SELECT * FROM EMP WHERE HIREDATE >= '82/01/01' AND HIREDATE<= '82/12/31';

-- 연봉이 2400 이상인 직원의 ENAME, SAL, 연봉 출력 (연봉 = SAL*12)
SELECT ENAME, SAL, SAL*12 "연봉" FROM EMP WHERE SAL*12 > 2400;
SELECT ENAME, SAL, SAL*12 "연봉" FROM EMP WHERE 연봉>2400;  -- 실행 오류
                                        -- WHERE절에는 별칭을 쓸 수 없다
SELECT ENAME, SAL, SAL*12 "연봉" FROM EMP WHERE SAL*12>2400 ORDER BY 연봉;
                                        -- ORDER BY절에는 별칭 사용 가능

-- 10번 부서(DEPTNO)이거나 직책(JOB)이 MANAGER인 사람의 모든 필드
SELECT * FROM EMP WHERE DEPTNO=10 OR JOB='MANAGER';

-- 부서번호가 10번이 아닌 사람의 모든 필드
SELECT * FROM EMP WHERE DEPTNO != 10;
SELECT * FROM EMP WHERE NOT DEPTNO=10;
```

### (5) 산술 표현식

ORACLE에서의 기본적인 산술 표현식은 다른 프로그램들과 비슷하다. 하지만 연산자를 포함한 식에 값이 지정되지 않은 NULL값이 있으면 결과는 NULL이 된다. 이런 경우를 위해 NVL 함수가 존재한다.

또한 ORACLE에서 기본연산자(+-/*..)는 숫자끼리만 가능하며 문자형숫자가 오면 숫자형으로 바뀌서 연산된다.

EX)

```
-- ex. 모든 사원의 이름(ENAME), 월급(SAL), 상여(COMM), 연봉(SAL*12+COMM)

   -- NVL(NULL일 수도 있는 필드명, 대치값) 반드시 매개변수 둘은 타입 같아야 함
SELECT ENAME, SAL, COMM, SAL*12+COMM FROM EMP;
   -- COMM에 NULL값이 있는 행은NULL값이 출력된다.
SELECT ENAME, SAL, COMM, SAL*12+NVL(COMM, 0) FROM EMP;

-- 모든 사원의 사원명(ENAME), 상사의 사번(MGR)을 출력(상사없으면(NULL이면) 0으로 출력)
SELECT ENAME, NVL(MGR, 0) FROM EMP;
SELECT ENAME, NVL(MGR, '없음') FROM EMP; -- NVL 매개변수 타입 불일치 에러
```

결과)

| | ENAME | SAL | COMM | SAL*12+COMM |
|---|---|---|---|---|
| 1 | SMITH | 800 | (null) | (null) |
| 2 | ALLEN | 1600 | 300 | 19500 |
| 3 | WARD | 1250 | 500 | 15500 |
| 4 | JONES | 2975 | (null) | (null) |
| 5 | MARTIN | 1250 | 1400 | 16400 |
| 6 | BLAKE | 2850 | (null) | (null) |
| 7 | CLARK | 2450 | (null) | (null) |
| 8 | SCOTT | 3000 | (null) | (null) |
| 9 | KING | 5000 | (null) | (null) |

->

| | ENAME | SAL | COMM | SAL*12+... |
|---|---|---|---|---|
| 1 | SMITH | 800 | (null) | 9600 |
| 2 | ALLEN | 1600 | 300 | 19500 |
| 3 | WARD | 1250 | 500 | 15500 |
| 4 | JONES | 2975 | (null) | 35700 |
| 5 | MARTIN | 1250 | 1400 | 16400 |
| 6 | BLAKE | 2850 | (null) | 34200 |
| 7 | CLARK | 2450 | (null) | 29400 |
| 8 | SCOTT | 3000 | (null) | 36000 |
| 9 | KING | 5000 | (null) | 60000 |

### (6) 연결연산자

문자열 연산자라고도 하며 열이나 문자를 연결하여 결과를 보여주며 연결할 명령어 사이에 '||' 기호로 사용한다. 숫자형이 오면 문자형으로 바뀌어서 연산된다.

EX)

```
    -- ex. "SMITH : 연봉 = XXX"과 같이 모든 행 출력
SELECT ENAME || ' : 연봉 = ' || (SAL*12+NVL(COMM,0)) FROM EMP;

    -- ex. "SHITH 는 CLERK" 과 같이 모든행 출력
SELECT ENAME || '은' || JOB "EMPLOYEES" FROM EMP;
```

결과)

| | ENAME\|\|':연봉='\|\|(SAL*12+NVL(COMM,0)) |
|---|---|
| 1 | SMITH : 연봉 = 9600 |
| 2 | ALLEN : 연봉 = 19500 |
| 3 | WARD : 연봉 = 15500 |
| 4 | JONES : 연봉 = 35700 |
| 5 | MARTIN : 연봉 = 16400 |
| 6 | BLAKE : 연봉 = 34200 |
| 7 | CLARK : 연봉 = 29400 |
| 8 | SCOTT : 연봉 = 36000 |
| 9 | KING : 연봉 = 60000 |

| | EMPLOYEES |
|---|---|
| 1 | SMITH은CLERK |
| 2 | ALLEN은SALESMAN |
| 3 | WARD은SALESMAN |
| 4 | JONES은MANAGER |
| 5 | MARTIN은SALESMAN |
| 6 | BLAKE은MANAGER |
| 7 | CLARK은MANAGER |
| 8 | SCOTT은ANALYST |
| 9 | KING은PRESIDENT |

### (7) 중복제거

SELECT 절 뒤에 DISTINCT 명령어를 명시하여 사용하며 출력할 결과의 중복된 값을 제거하여 나타내준다 오름차순으로 정렬하는 것이 기본값이다.

EX)

```
SELECT DEPTNO FROM EMP;              -- 모든 열의 DEPTNO 출력

SELECT DISTINCT DEPTNO FROM EMP;     -- 모든 열의 DEPTNO중 중복된 값을 제거하여 출력
```

결과)

| | DEPTNO |
|---|---|
| 1 | 20 |
| 2 | 30 |
| 3 | 30 |
| 4 | 20 |
| 5 | 30 |
| 6 | 30 |
| 7 | 10 |
| 8 | 20 |
| 9 | 10 |
| 10 | 30 |

->

| | DEPTNO |
|---|---|
| 1 | 30 |
| 2 | 20 |
| 3 | 10 |

## (8) SQL 연산자

1. BETWEEON 'A' AND 'B'

A와 B를 포함한 A와 B사이의 값 즉, A이상 B 이하의 값을 말한다

EX)

```
    -- ex. SAL이 1500이상 3000이하인 사번 이름 급여
SELECT EMPNO, ENAME, SAL FROM EMP WHERE SAL>=1500 AND SAL<=3000;
SELECT EMPNO, ENAME, SAL FROM EMP WHERE SAL BETWEEN 1500 AND 3000;
SELECT EMPNO, ENAME, SAL FROM EMP WHERE SAL BETWEEN 3000 AND 1500; -- 에러
                                                -- 왼쪽값(A)이 적은 값이 와야한다.

    -- ex. 1500미만, 3000초과하는 모든 필드
SELECT * FROM EMP WHERE SAL NOT BETWEEN 1500 AND 3000; --NOT과 같이 활용할수있다.

    -- ex.82년도에 입사한 직원의 모든 필드 출력
SELECT * FROM EMP WHERE HIREDATE BETWEEN '82/01/01' AND '82/12/31';

    -- ex. 사원이름이 'A','B'로 시작하는 사원의 모든 필드
SELECT * FROM EMP WHERE ENAME BETWEEN 'A' AND 'C' AND ENAME!='C';

    -- ex. 사원이름이 'B','C'로 시작하는 사원의 모든 필드
SELECT * FROM EMP WHERE ENAME BETWEEN 'B' AND 'D' AND ENAME<>'D';
```

2. IN(A,B,C...)

괄호안에 나열된 값 중 일치하는 값, 즉 A,B,C 의 값 중 어느 하나의 값이라도 일치하는 값을 출력한다.

EX)

```
    -- ex. DEPTNO가 10,20,40번인 부서 사원의 모든 필드
SELECT * FROM EMP WHERE DEPTNO=10 OR DEPTNO=20 OR DEPTNO=40;
SELECT * FROM EMP WHERE DEPTNO IN (10,20,40);

    -- ex. DEPTNO가 10,20,40번 부서를 제외한 사원의 모든 필드
SELECT * FROM EMP WHERE DEPTNO NOT IN (10,20,40);
SELECT * FROM EMP WHERE DEPTNO!=10 AND DEPTNO<>20 AND DEPTNO^=40;

    -- ex. EMPNO가 7902, 7788, 7566인 사원의 모든 필드
SELECT * FROM EMP WHERE EMPNO IN (7902, 7788, 7566);
```

3. LIKE

검색하고자 하는 값을 정확히 모를 경우 부분적으로 일치하는 값, 값의 패턴을 찾기 위해 사용한다.

다음 2가지의 와일드카드가 사용된다.

% : 문자가 없거나, 하나 이상의 문자를 말하며 어떤 값이 와도 상관없다.

_ : 하나의 문자를 말하며 반드시 하나의 값이 와야한다.

EX)

```
    -- ex. 이름이 M이 들어간 사원의 모든 필드
SELECT * FROM EMP WHERE ENAME LIKE '%M%';
    -- ex. 이름이 M으로 시작하는 사원의 모든 필드
SELECT * FROM EMP WHERE ENAME LIKE 'M%';
    -- ex. 이름이 S로 끝나는 사원의 모든 필드
SELECT * FROM EMP WHERE ENAME LIKE '%S';
    -- ex. SAL이 5로 끝나는 사원의 모든 필드
SELECT * FROM EMP WHERE SAL LIKE '%5';

    -- ex. 82년도에 입사한 사원의 모든 필드;
SELECT * FROM EMP WHERE HIREDATE LIKE '%82%';
SELECT * FROM EMP WHERE TO_CHAR(HIREDATE, 'RR/MM/DD') LIKE '82/%';

    -- ex. 1월에 입사한 사원의 모든 필드
SELECT * FROM EMP WHERE HIREDATE LIKE '__/01/__';
SELECT * FROM EMP WHERE TO_CHAR(HIREDATE, 'RR/MM/DD') LIKE '__/01/__';

    -- ex. 82년도가 아닌 년도에 입사한 사원의 모든 필드
SELECT * FROM EMP WHERE HIREDATE NOT LIKE '82/%';

    -- ex. 이름에 %가 들어간 사원의 모든 필드
SELECT * FROM EMP WHERE ENAME LIKE '%%%'; -- %들어간 자료가 아닌 모든 사원을 출력
SELECT * FROM EMP WHERE ENAME LIKE '%#%%' ESCAPE '#';
                        -- ESCAPE 함수는 ESCAPE ' '로 지정한 문자 다음에 오는
                        -- 와일드카드(EX. %_..)를 일반문자로 인식해준다.
```

4. IS NULL

값이 포함되는 혹은 포함되지 않는 데이터(NULL)를 추출할 때 사용하는 문이다.

EX)

```
    -- ex. 상여금이 NULL인 사원의 모든 필드
SELECT * FROM EMP WHERE COMM IS NULL;
SELECT * FROM EMP WHERE COMM=NULL; -- 오류. NULL을 비교시 반드시 IS NULL로 해야한다.

    -- ex. 상여금이 없는 사원의 모든 필드
SELECT * FROM EMP WHERE COMM IS NULL OR COMM=0;        -- COMM 이 NULL이거나 0인 데이터

    -- ex. 상여금이 있는 사원의 모든 필드
SELECT * FROM EMP WHERE NOT COMM IS NULL AND COMM!=0; -- COMM이 NULL도 0도 아닌 데이터
SELECT * FROM EMP WHERE NOT (COMM IS NULL OR COMM=0);
```

5. ORDER BY

도출된 데이터를 정렬하여 출력할 때 사용한다.

EX)

```
SELECT ENAME, SAL FROM EMP ORDER BY SAL ASC; -- SAL 오름차순 정렬
SELECT ENAME, SAL FROM EMP ORDER BY SAL DESC; -- SAL 내림차순 정렬
SELECT ENAME, SAL FROM EMP ORDER BY SAL; -- SAL 오름차순 정렬
                                        -- 차순을 명시하지않을때 기본값이 오름차순이다.

SELECT ENAME, HIREDATE FROM EMP ORDER BY HIREDATE; -- HIREDATE 순서대로 정렬
SELECT ENAME, HIREDATE FROM EMP ORDER BY HIREDATE DESC; -- HIREDATE 최신순으로 정렬

    -- 이름, 연봉(SAL*12+COMM)을 출력(연봉이 높은 순으로, 연봉이 같은 경우 이름순으로)
SELECT ENAME, SAL*12+NVL(COMM,0) "ANNUAL_SAL"
    FROM EMP
    ORDER BY SAL*12+NVL(COMM,0) DESC, ENAME;

SELECT ENAME, SAL*12+NVL(COMM,0) "ANNUAL_SAL"
    FROM EMP
    ORDER BY ANNUAL_SAL DESC, ENAME;
    -- ORDER BY 절에는 별칭 사용 가능하다(FROM->WHERE->SELECT->ORDER)
```