ch05. WHERE절과 연산자

___ 더 정확하고 다양하게 결과를 출력

05-1. 필요한 데이터만 쏙 출력하는 WHERE

WHERE절은 SELECT문으로 데이터를 조회할 때 특정 조건을 기준을 원하는 행을 출력할 때 사용

- 부서 번호가 30인 데이터만 출력하기

```
SELECT *
      FROM EMP
WHERE DEPTNO = 30;
```

WHERE절을 활용한 SELECT문의 기본형식

```
SELECT [조회할 열1 이름], [열2 이름], . . . , [열N 이름]
     FROM [조회할 테이블 이름]
WHERE [조회할 행을 선별하기 위한 조건식];
```

WHERE 조건식에 참 (true) 인 행만 출력이 되고, 거짓 (false) 인 행은 출력되지 않는다.

05-2. 여러 개 조건식을 사용하는 AND, OR 연산자

WHERE절에서 조건식을 여러 개 지정할 때 AND, OR 연산자를 사용한다.

AND 연산자

```
SELECT *
  FROM EMP
```

```
WHERE DEPTNO = 30
  AND JOB = 'SALESMAN';
```

⚠️ *WHERE절에서 비교하는 데이터가 문자열일 경우에는 '_' 으로 묶어 준다. 앞 뒤에 공백이 있으면 공백도 문자로 인식하기 때문에 주의!*

OR 연산자

```
SELECT *
  FROM EMP
WHERE DEPTNO = 30
  OR JOB = 'CLERK';
```

⚠️ *AND는 피연산자가 둘다 true여야 하고, OR는 피연산자가 둘 다 또는 둘 중 하나가 true이면 결과 값이 true가 된다.*

WHERE절 조건식의 개수

제한이 없다. 조건식을 두 개 이상 사용할 경우에도 각 조건식 사이에 AND 또는 OR 연산자를 추가하여 사용한다.

05-3. 연산자 종류와 활용 방법 알아보기

산술 연산자

- EMP테이블에서 급여 열에 12를 곱한 값이 36000인 행을 출력

```
SELECT *
  FROM EMP
WHERE SAL * 12 = 36000;
```

비교 연산자

- **대소 비교 연산자**
  - 대소 비교 연산자를 사용하여 출력

```
SELECT *
  FROM EMP
WHERE SAL >= 3000;
```

  - 대소 비교 연산자 종류

연산자	사용법	설명
>	A > B	A 값이 B 값을 초과할 경우 true
>=	A >= B	A 값이 B 값 이상일 경우 true
<	A < B	A 값이 B 값 미만일 경우 true
<=	A <= B	A 값이 B 값 이하일 경우 true

✅ 숫자 뿐만이 아니라 문자열일 때도 사용가능하다!

비교문자열이 문자 하나일 때 :

```
SELECT *
  FROM EMP
WHERE ENAME >= 'F';
```

비교문자열이 문자 여러 개일 때 :

```
SELECT *
  FROM EMP
WHERE ENAME <= 'FORZ';
```

→ 자주 사용되는 내용은 아니기에 가볍게 이해만!

- **등가 비교 연산자**
  - 종류

연산자	사용법	의미
=	A = B	A 값이 B 값과 같을 경우 true, 다를 경우 false 반환

연산자	사용법	의미
!=	A != B	A 값이 B 값과 다를 경우 true, 같을 경우 false 반환
<>	A <> B	A 값이 B 값과 다를 경우 true, 같을 경우 false 반환
^=	A ^= B	A 값이 B 값과 다를 경우 true, 같을 경우 false 반환

- != 사용하여 출력

```
SELECT *
  FROM EMP
WHERE SAL != 3000;
```

- <> 사용하여 출력

```
SELECT *
  FROM EMP
WHERE SAL <> 3000;
```

- ^= 사용하여 출력

```
SELECT *
  FROM EMP
WHERE SAL ^= 3000;
```

⇒ 실행결과가 모두 같다

- **논리 부정 연산자**
  - 위의 '같지 않다'의 의미를 가지는 연산자랑 같은 의미를 갖는 논리 부정 연산자 NOT .

```
SELECT *
  FROM EMP
WHERE NOT SAL = 3000;
```

1. NOT연산자는 IN, BETWEEN, IS NULL 과 함께 복합적으로 사용하는 경우가 많다.
2. 복잡한 여러 개 조건식이 AND, OR로 묶여 있는 상태에서 정반대의 결과를 얻고자 할 때 유용

- **IN 연산자**
  - *특정 열에 포함된 데이터를 여러 개 조회할 때 활용*

```
SELECT *
  FROM EMP
WHERE JOB = 'MANAGER'
  OR  JOB = 'SALESMAN'
  OR  JOB = 'CLERK';
```

⏬

```
SELECT *
  FROM EMP
WHERE JOB IN ('MANAGER', 'SALESMAN', 'CLERK');
```

  - NOT 연산자를 이용하여 출력

```
SELECT *
  FROM EMP
WHERE JOB != 'MANAGER'
  OR  JOB <> 'SALESMAN'
  OR  JOB ^= 'CLERK';
```

⏬

```
SELECT *
  FROM EMP
WHERE JOB NOT IN ('MANAGER', 'SALESMAN', 'CLERK');
```

- **BETWEEN A AND B 연산자**

대소 비교 연산자와 AND를 사용하여 출력

```
SELECT *
  FROM EMP
WHERE SAL >= 2000
  AND SAL <= 3000;
```

⏬

```
SELECT *
  FROM EMP
WHERE SAL BETWEEN 2000 AND 3000;
```

⇒ NOT 연산자도 앞의 실습과 비슷하게 BETWEEN 앞에 NOT을 붙여주면 된다.

- **LIKE 연산자와 와일드 카드**
  - LIKE 연산자 : 이메일이나 게시판 제목 또는 내용 검색 기능처럼 일부 문자열이 포함된
    데이터를 조회할 때 사용
  - 와일드 카드 : 특정 문자 또는 문자열을 대체하거나 문자열 데이터의 패턴을 표기하는
    특수문자

    ⇒ LIKE 연산자와 함께 사용 가능한 와일드카드 : _ 와 %

```
SELECT *
  FROM EMP
WHERE ENAME LIKE 'S%';
```

```
SELECT *
  FROM EMP
WHERE ENAME LIKE '%S';
```

```
SELECT *
  FROM EMP
WHERE ENAME LIKE '_L%';
```

```
SELECT *
  FROM EMP
WHERE ENAME LIKE '%AM%';
```

⇒ 마찬가지로 반대값을 찾으려면 LIKE 앞에 NOT을 붙인다.

- **IS NULL**

NULL : 데이터 값이 완전히 '비어 있는' 상태 (0 ≠ NULL)

특정 열 또는 연산의 결과 값이 NULL인지 여부를 확인하려면 IS NULL연산자를 사용.

```
SELECT *
  FROM EMP
WHERE COMM IS NULL;
```

⇒ NULL 값이 아닌 데이터만 조회하려면 IS NOT NULL 이라고 쓴다

- AND 연산자와 IS NULL 사용하기

```
SELECT *
  FROM EMP
WHERE SAL > NULL
  AND COMM IS NULL;
```

	true	false	NULL
true	true	false	NULL
false	false	false	false
NULL	NULL	false	NULL

- OR 연산자와 IS NULL 사용하기

```
SELECT *
  FROM EMP
WHERE SAL > NULL
  OR COMM IS NULL;
```

	true	false	NULL
true	true	true	true
false	true	false	NULL
NULL	true	NULL	NULL

- **집합 연산자**

ONION 연산자로 합집합을 의미한다.

```
SELECT EMPNO, ENAME, SAL, DEPTNO
  FROM EMP
WHERE DEPTNO = 10
UNION
SELECT EMPNO, ENAME, SAL, DEPTNO
  FROM EMP
WHERE DEPTNO = 20;
```

⚠️ 주의할 점은 집합 연산자로 두 개의 SELECT문의 결과 값을 연결할 때 각 SELECT문을 출력하려는
열의 개수와 각 열의 자료형이 순서별로 일치해야 함.

종류	설명
UNION	연결된 SELECT문의 결과 값의 합집합. 결과 값의 중복은 제거되서 출력.
UNION ALL	연결된 SELECT문의 결과 값의 합집합. 중복된 결과 값도 제거 없이 모두 출력.
MINUS	먼저 작성한 SELECT문의 결과 값에서 다음 SELECT문의 결과 값을 차집합 처리. 먼저 작성한 SELECT문의 결과 값 중 다음 SELECT문에 존재하지 않는 데이터만 출력.
INTERSECT	먼저 작성한 SELECT문과 다음 SELECT문의 결과 값이 같은 데이터만 출력. 교집합.

- UNION (출력 결과 데이터가 같을 때)

```
SELECT EMPNO, ENAME, SAL, DEPTNO
  FROM EMP
WHERE DEPTNO = 10
UNION
SELECT EMPNO, ENAME, SAL, DEPTNO
  FROM EMP
WHERE DEPTNO = 10;
```

- UNION ALL (출력 결과 데이터가 같을 때)

```
SELECT EMPNO, ENAME, SAL, DEPTNO
  FROM EMP
WHERE DEPTNO = 10
UNION ALL
SELECT EMPNO, ENAME, SAL, DEPTNO
  FROM EMP
WHERE DEPTNO = 10;
```

- MINUS

```
SELECT EMPNO, ENAME, SAL, DEPTNO
  FROM EMP
MINUS
SELECT EMPNO, ENAME, SAL, DEPTNO
  FROM EMP
WHERE DEPTNO = 10;
```

- INTERSECT

```
SELECT EMPNO, ENAME, SAL, DEPTNO
  FROM EMP
INTERSECT
SELECT EMPNO, ENAME, SAL, DEPTNO
  FROM EMP
WHERE DEPTNO = 10;
```

우선순위	연산자	설명
↑ (높음)	*, /	산술 연산자 곱하기, 나누기
	+, -	산술 연산자 더하기, 빼기
	=, !=, ^=, <>, >, >=, <, <=	대소 비교 연산자
	IS (NOT) NULL, (NOT) LIKE, (NOT) IN	(그 외) 비교 연산자
	BETWEEN A AND B	BETWEEN 연산자
	NOT	논리 부정 연산자 NOT
(낮음) ↓	AND	논리 연산자 AND
	OR	논리 연산자 OR

연습문제

Q1.

```
SELECT *
FROM EMP
WHERE ENAME LIKE '%S';
```

Q2.

```
SELECT EMPNO, ENAME, JOB, SAL, DEPTNO
  FROM EMP
 WHERE DEPTNO = 30
   AND JOB = 'SALESMAN';
```

Q3.

```
SELECT EMPNO, ENAME, JOB, SAL, DEPTNO
  FROM EMP
 WHERE DEPTNO IN (20, 30)
   AND SAL > 2000;
```

```
ELECT EMPNO, ENAME, JOB, SAL, DEPTNO
  FROM EMP
 WHERE DEPTNO = 20
   AND SAL > 2000
UNION
SELECT EMPNO, ENAME, JOB, SAL, DEPTNO
  FROM EMP
 WHERE DEPTNO = 30
   AND SAL > 2000;
```

Q4.

```
SELECT *
  FROM EMP
 WHERE SAL < 2000
    OR SAL > 3000;
```

Q5.

```
SELECT ENAME, EMPNO, SAL, DEPTNO
  FROM EMP
 WHERE DEPTNO = 30
   AND ENAME LIKE '%E%'
   AND SAL NOT BETWEEN 1000 AND 2000;
```

Q6.

```
SELECT *
  FROM EMP
 WHERE COMM IS NULL
   AND MGR IS NOT NULL
   AND JOB IN ('MANAGER', 'CLERK')
   AND ENAME NOT LIKE '_L%';
```