# sql-study-day6

## 05 - 3 연산자 종류와 활용 방법 알아보기

- 산술 연산자

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

// SAL열에 12를 곱한 값이 36000인 행을 출력하는 SQL문
```

- 비교 연산자

```
SELECT *
FROM EMP
WHERE SAL >= 3000;
// SAL이 3000이상인 사원을 조회하는 SQL문
```

이 외에도 아래 표처럼 다른 대소 비교 연산자가 있다.

| 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 |

⭐ 대소 비교 연산자는 비교 대상인 데이터가 숫자가 아닌 문자열일 때도 사용할 수 있습니다. 하지만 문자열의 대소 비교는 숫자 데이터 비교보다는 자주 사용되는 내용은 아니기 때문에 가볍게 이해하는 정도로만 기억해도 된다.

- 등가 비교 연산자

연산자의 양쪽 항목 값이 같으면 true가 반환되고, 반대로 연산자의 양쪽 값이 다를 경우 true를 반환하는 연산자도 있다.

| 연산자 | 사용법 | 의미 |
| --- | --- | --- |

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

- 논리 부정 연산자

비교 연산자는 아니지만 ! =, <>, ^=과 똑같은 결과를 출력하기 위해 사용할 수 있는 연산자가 하나 더 있다. 바로 논리 부정 연산자인 NOT 연산자이다.

```
SELECT *
FROM EMP
WHERE NOT SAL = 3000;
// WHERE SAL != 3000; WHERE SAL <> 3000; WHERE SAL ^= 3000;
// 과 똑같이 출력된다.
```

⭐ NOT 연산자를 IN, BETWEEN, IS NULL 연산자와 함께 복합적으로 사용하는 경우가 많고, 대소 · 등가 비교 연산자에 직접 사용하는 경우는 별로 없다.

- IN 연산자

```
SELECT *
FROM EMP
WHERE JOB = 'MANAGER'
OR JOB = 'SALESMAN'
OR JOB = 'CLERK';
// OR 연산자를 사용해서 출력해도 되지만, 조건이 늘어날수록
```

OR 연산자를 사용해서 출력해도 되지만, 조건이 늘어날수록 조건식을 많이 작성해야 하기 때문에 아래와 같이 IN 연산자를 사용하면 특정 열에 해당하는 조건을 여러 개 지정할 수 있다.

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

- BETWEEN A AND B 연산자

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

AND 연산자를 사용해서도 가능하지만, 특정 열 값의 최소 · 최고 범위를 지정하여 해당 범위 내의 데이터만 조회할 경우에 대소 비교 연산자 대신 BETWEEN A AND B 연산자를 사용하면 더 간단하게 표현할 수 있다.

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

- LIKE 연산자와 와일드 카드

LIKE 연산자는 이메일이나 게시판 제목 또는 내용 검색 기능처럼 일부 문자열이 포함된 데이터를 조회할 때 사용한다.

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

ENAME LIKE ‘S%’ 조건식은 ENAME 열 값이 대문자 S로 시작하는 데이터를 조회하라는 뜻이다.

위 조건식에서 사용한 %기호를 와일드 카드라고 한다. 와일드 카드는 특정 문자 또는 문자열을 대체하거나 문자열 데이터의 패턴을 표기하는 특수 문자이다.

LIKE 연산자와 함께 사용할 수 있는 와일드 카드는 _와 %이다.

| **종류** | **의미** |
| --- | --- |
| _ | 어떤 값이든 상관없이 한 개의 문자 데이터를 의미 |
| % | 길이와 상관없이(문자 없는 경우도 포함) 모든 문자 데이터를 의미 |
| | |

ENAME의  두 번째 글자가 L인 사원 데이터를 조회하고 싶다면 아래와 같이 LIKE 연산자에 와일드 카드를 활용할 수 있다.

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

---

ENAME에 AM이라는 단어를 포함하는 사원을 조회할 때 아래와 같이 작성하면 된다.

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

- IS NULL

WHERE절은 조건식의 결과 값이 true인 행만 출력하는데 연산 결과 값이 NULL이 되어 버리면 조건식의 결과 값이 false도 true도 아니게 되므로 출력 대상에서 제외된다.

그러므로 특정 열 또는 연산의 결과 값이 NULL인지 여부를 확인하려면 IS NULL 연산자를 아래와 같이 사용해야 한다.

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

IS NULL 연산자를 사용하면 추가 수당 열 값이 존재하지 않는 데이터만 출력한다.

⭐데이터가 NULL인지 아닌지를 확인하는 용도로만 사용하는 IS NULL과 IS NOT NULL 연산자는 매우 자주 사용되므로 사용법을 꼭 기억하자❗

- 집합연산자 UNION

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

집합 연산자를 사용하면서 주의할 점을 두 개의 SELECT문의 결과 값을 연결할 때 각 SELECT문이 출력하려는 열 개수와 각 열의 자료형이 순서별로 일치해야 한다.

⭐ 집합 연산자에는 아래와 같이 4가지 종류가 있다.

| 종류 | 설명 |
| --- | --- |
| UNION | 연결된 SELECT문의 결과 값을 합집합으로 묶어 준다. 결과 값의 중복은 제거된다. |
| UNION ALL | 연결된 SELECT문의 결과 값을 합집합으로 묶어 준다. 중복된 결과 값도 제거 없이 모두 출력된다. |
| MINUS | 먼저 작성한 SELECT문의 결과 값에서 다음 SELECT문의 결과 값을 차집합 처리한다. 먼저 작성한 SELECT문의 결과 값 중 다음 SELECT문에 존재하지 않는 데이터만 출력된다. |
| INTERSECT | 먼저 작성한 SELECT문과 다음 SELECT문의 결과 값이 같은 데이터만 출력된다. 교집합과 같은 의미이다. |

- 연산자 우선순위

WHERE절 조건식에 사용한 여러 연산자는 아래와 같은 우선순위(priority)를 가지고 있다.

| 우선순위 | 연산자 | 설명 |
| --- | --- | --- |
| ↑ | *, / | 산술 연산자 곱하기, 나누기 |
| (높음) | +, - | 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 |

⭐ 수학식에서와 마찬가지로 먼저 수행해야 하는 연산식을 소괄호()로 묶어 주면 연산자의 기본 우선순위와는 별개로 괄호 안의 연산식을 먼저 수행한다.

# 5장 잊기 전에 한 번 더!

- Q1. EMP 테이블을 사용하여 다음과 같이 사원 이름(ENAME)이 S로 끝나는 사원 데이터를 모두 출력하는 SQL문을 작성해 보세요.

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

- Q2. EMP 테이블을 사용하여 30번 부서(DEPTNO)에서 근무하고 있는 사원 중에 직책(JOB)이 SALESMAN인 사원의 사원 번호, 이름, 직책, 급여, 부서 번호를 출력하는 SQL문을 작성해 보세요.

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

- Q3. EMP 테이블을 사용하여 20번, 30번 부서에 근무하고 있는 사원 중 급여(SAL)가 2000 초과인 사원을 다음 두 가지 방식의 SELECT문을 사용하여 사원 번호, 이름, 급여, 부서 번호를 출력하는 SQL문을 작성해 보세요.

1. 집합 연산자를 사용하지 않은 방식

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

2. 집합 연산자를 사용한 방식

```
SELECT 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. 이번에는 NOT BETWEEN A AND B 연산자를 쓰지 않고, 급여(SAL) 열 값이 2000 이상 3000 이하 범위 이외의 값을 가진 데이터만 출력하도록 SQL문을 작성해 보세요.

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

- Q5. 사원 이름에 E가 포함되어 있는 30번 부서의 사원 중 급여가 1000~2000 사이가 아닌 사원 이름, 사원 번호, 급여, 부서 번호를 출력하는 SQL문을 작성해 보세요.

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

- Q6. 추가 수당이 존재하지 않고 상급자가 있고 직책이 MANAGER, CLERK인 사원 중에서 사원 이름의 두 번째 글자가 L이 아닌 사원의 정보를 출력하는 SQL문을 작성해 보세요.

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