

## 2. 데이터 제한 및 정렬



# 선택되는 행 제한

- FROM 절 다음에 WHERE 절을 사용하여 반환되는 행을 제한합니다.

- 구문

```
SELECT *|{[DISTINCT] column|expression [alias],...}  
FROM   table  
[WHERE condition(s)];
```

- WHERE 절은 열 값, 리터럴 값, 산술식 또는 함수를 비교할 수 있으며, 다음 세 가지 요소로 구성됩니다.
  - 열 이름
  - 비교 조건
  - 열 이름, 상수 또는 값 목록

# WHERE 절 사용

- 다음 예제의 SELECT 문은 90번 부서에 속하는 모든 사원의 사번,이름, 업무 ID 및 부서 번호를 검색합니다.

```
SELECT employee_id, last_name, job_id, department_id  
FROM   employees  
WHERE  department_id = 90 ;
```

# 문자열 및 날짜의 사용

- 문자열 및 날짜 값은 작은 따옴표로 묶습니다.
- 문자 값은 대소문자를 구분하며 날짜 값은 날짜형식을 구분 합니다.

```
SELECT last_name, job_id, department_id  
FROM   employees  
WHERE  last_name = 'Whalen';
```

```
SELECT last_name, job_id, hire_date  
FROM   employees  
WHERE  hire_date = '14/11/16';
```

# 비교 조건

- 비교 조건은 표현식을 다른 값이나 표현식과 비교하는 조건문에 사용되며 WHERE 절에서 다음 형식으로 사용됩니다.
- 구문

```
... WHERE expr operator value
```

- 작성예시
  - ... WHERE hire\_date > '95/01/01'
  - ... WHERE salary >= 6000
  - ... WHERE last\_name = 'Smith'

# 비교 연산자

연산자	의미
=	같음
>	보다 큼
>=	크거나 같음
<	보다 작음
<=	작거나 같음
<>	같지 않음

## 비교 조건 사용

- 다음은 EMPLOYEES 테이블에서 급여가 3000 이하인 사원의 이름과 급여를 검색합니다.

```
SELECT last_name, salary  
FROM   employees  
WHERE  salary <= 3000;
```

- 다음은 2010년 1월 1일 이후에 입사한 사원을 검색합니다.

```
SELECT employee_id, last_name, job_id, hire_date  
FROM   employees  
WHERE  hire_date >= '10/01/01';
```

## 기타 SQL 비교 조건 연산자

연산자	의미
BETWEEN ... AND ...	두 값 사이(상/하한값 포함)
IN(set)	괄호 안의 값 목록 중의 하나와 일치
LIKE	문자 패턴 일치
IS NULL	널값



# BETWEEN 조건 사용

- BETWEEN 조건을 사용하여 값의 범위에 따라 행을 표시합니다.
- 다음 SELECT 문은 EMPLOYEES 테이블에서 급여가 \$2,500 이상이고 \$3,500 이하인 사원의 행을 반환합니다.

```
SELECT last_name, salary  
FROM   employees  
WHERE  salary BETWEEN 2500 AND 3500;
```

하한

상한

- BETWEEN의 결과에는 하한값과 상한값이 포함됩니다.

# IN 조건 사용

- IN 멤버 조건을 사용하면 값이 목록에 있는지 확인할 수 있습니다.
- 예제는 관리자의 사원 번호가 100, 101 또는 201인 사원정보를 표시합니다.

```
SELECT employee_id, last_name, salary, manager_id
FROM   employees
WHERE  manager_id IN (100, 101, 201);
```

- 다음 예제는 EMPLOYEES 테이블에서 WHERE 절의 이름 목록에 포함된 이름을 가진 사원의 행을 반환합니다.

```
SELECT employee_id, manager_id, department_id
FROM   employees
WHERE  last_name IN ('Hartstein', 'Vargas');
```

# LIKE 조건 사용

- LIKE 조건을 사용하면 유효한 검색 문자열 값인 대체 문자를 사용하여 검색할 수 있습니다.
- 검색 문자열은 두 가지 기호를 사용하여 구성할 수 있습니다.
  - %에는 문자가 오지 않거나 여러 개 올 수 있습니다.
  - \_에는 문자가 하나만 올 수 있습니다.
- SELECT 문은 EMPLOYEES 테이블에서 이름이 S로 시작하는 모든 사원의 이름을 반환합니다

```
SELECT first_name  
FROM   employees  
WHERE  first_name LIKE 'S%';
```

# LIKE 조건 사용

- 예제는 이름의 두번째 문자가 o인 모든 사원의 이름을 표시합니다.

```
SELECT last_name  
FROM employees  
WHERE last_name LIKE '_o%';
```

- ESCAPE 식별자를 사용하여 "%" 또는 "\_" 자체를 검색할 수 있습니다.
- 'SA\_'를 포함하는 문자열을 검색하는 경우 다음 SQL 문을 사용하여 검색할 수 있습니다.

```
SELECT employee_id, last_name, job_id  
FROM employees  
WHERE job_id LIKE '%SA\_%' ESCAPE '\';
```

# NULL 조건 사용

- IS NULL 연산자를 사용하여 널 여부를 테스트합니다.
- 예제는 관리자가 없는 모든 사원의 이름과 관리자를 검색하거나 커미션을 받지 않는 업무에 해당하는 모든 사원의 이름, 업무 ID 및 커미션을 표시합니다.

```
SELECT last_name, manager_id
FROM   employees
WHERE  manager_id IS NULL;
```

```
SELECT last_name, job_id, commission_pct
FROM   employees
WHERE  commission_pct IS NULL;
```

# 논리 조건

- AND 및 OR 연산자를 사용하여 하나의 WHERE 절에 여러 조건을 지정할 수 있습니다.

연산자	의미
AND	구성 요소 조건이 모두 TRUE면 TRUE를 반환합니다.
OR	구성 요소 조건 중 하나라도 TRUE면 TRUE를 반환합니다.
NOT	뒤따르는 조건이 FALSE면 TRUE를 반환합니다.

# AND 연산자 사용

- 다음 예제는 업무 ID에 문자열 MAN이 포함되고 급여가 \$10,000 이상인 직원만 선택됩니다.
- AND는 조건이 모두 TRUE여야 합니다.

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >=10000
AND    job_id LIKE '%MAN%';
```

# OR 연산자 사용

- 업무 ID에 MAN이 포함되거나 급여가 \$10,000 이상인 사원이 선택됩니다.
- OR는 조건 중 하나가 TRUE면 됩니다.

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >= 10000
OR     job_id LIKE '%MAN%';
```



# NOT 연산자 사용

- 예제는 업무 ID가 IT\_PROG, ST\_CLERK 또는 SA\_REP가 아닌 모든 사원의 이름과 업무 ID를 표시합니다.

```
SELECT last_name, job_id  
FROM   employees  
WHERE  job_id  
       NOT IN ('IT_PROG', 'ST_CLERK', 'SA_REP');
```

- NOT 연산자는 BETWEEN, LIKE, NULL 등 다른 SQL 연산자와 함께 사용할 수 있습니다.
  - ... WHERE job\_id NOT IN ('AC\_ACCOUNT', 'AD\_VP')
  - ... WHERE salary NOT BETWEEN 10000 AND 15000
  - ... WHERE last\_name NOT LIKE '%A%'
  - ... WHERE commission\_pct IS NOT NULL

# 우선순위 규칙

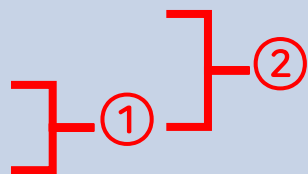
- 먼저 계산할 식의 좌우에 괄호를 사용하여 기본 순서를 무시하고 우선순위를 변경할 수 있습니다.

우선순위	연산자
1	산술연산자
2	연결연산자
3	비교조건
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	NOT 논리조건
7	AND 논리조건
8	OR 논리조건

# 우선순위 규칙

- 사장이면서 급여가 \$15,000를 넘는 직원 또는 영업 직원인 직원의 행(row)을 선택합니다.

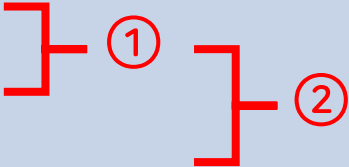
```
SELECT last_name, job_id, salary
FROM   employees
WHERE  job_id = 'SA_REP'
OR     job_id = 'AD_PRES'
AND    salary > 15000;
```



# 우선순위 규칙

- 다음 예제는 사장 또는 영업 사원이면서 급여가 \$15,000를 넘는 사원의 행을 선택합니다.
- 괄호를 사용하여 우선순위를 강제로 지정합니다.

```
SELECT last_name, job_id, salary
FROM employees
WHERE (job_id = 'SA_REP'
OR job_id = 'AD_PRES')
AND salary > 15000;
```



# ORDER BY 절

- ORDER BY 절을 사용하여 행을 정렬합니다.
  - ASC: 오름차순, 기본값
  - DESC: 내림차순
- ORDER BY 절은 SELECT 문의 가장 끝에 둡니다.

```
SELECT  last_name, job_id, department_id, hire_date
FROM    employees
ORDER BY hire_date ;
```

## 내림차순으로 정렬

- 행 표시 순서를 바꾸려면 ORDER BY 절에서 열 이름 다음에 DESC 키워드를 지정합니다.
- 예제는 가장 최근에 입사한 사원 순으로 결과를 정렬합니다.

```
SELECT  last_name, job_id, department_id, hire_date
FROM    employees
ORDER BY hire_date DESC ;
```

# 열 별칭을 기준으로 정렬

- ORDER BY 절에 열 별칭을 사용할 수 있습니다
- 다음 예제는 연봉을 기준으로 데이터를 정렬합니다.

```
SELECT employee_id, last_name, salary*12 annsal  
FROM employees  
ORDER BY annsal;
```

# 여러 열을 기준으로 정렬

- ORDER BY 목록의 순서가 정렬 순서입니다.
- 여러 열을 기준으로 질의 결과를 정렬할 수 있습니다.
- SELECT 목록에 없는 열을 기준으로 정렬할 수도 있습니다.
- 예제는 모든 사원의 이름, 부서번호, 급여를 표시하고 부서번호를 기준으로 정렬한 후 급여를 기준으로 내림차순으로 결과를 정렬합니다.

```
SELECT last_name, department_id, salary  
FROM   employees  
ORDER BY department_id, salary DESC;
```



# Position Number 의 사용

- ORDER BY 절에는 열 이름이나 별칭 대신 숫자가 올 수 있습니다.
- 이 숫자는 SELECT 목록의 열 순서에 해당되며, Position Number라고 합니다.
- 예제는 모든 사원의 이름, 부서번호, 급여를 표시하고 부서번호를 기준으로 정렬한 후 급여를 기준으로 내림차순으로 결과를 정렬합니다.

```
SELECT last_name, department_id, salary  
FROM employees  
ORDER BY 2, 3 DESC;
```

Thank you 😊