3. 검색 제한과 정렬

컴퓨터공학부

김은경

[학습 목표]

1. SELECT문의 WHERE절을 사용하여 행을 제한적으로 검색할 수 있다.
2. 대소 비교 연산자를 사용해서 기술된 검색 조건을 포함하는 질의(Query)를 작성할 수 있다.
3. 기타 비교 연산자를 사용해서 기술된 검색 조건을 포함하는 질의를 작성할 수 있다.
4. 논리 연산자를 사용해서 여러 개의 검색 조건을 연결할 수 있다.
5. SELECT문의 ORDER BY 절을 이용하여 특정 행(들)을 기준으로 검색 결과를 정렬할 수 있다.

1. 행 제한 검색

행 제한 검색(셀렉션) 방법

- WHERE 절에 검색 조건을 작성하면, 테이블에서 조건이 만족되는 특정 행(row)만 선택해서 검색할 수 있음

```
SELECT    [DISTINCT] {* | 열이름1 [별칭1], 열이름2 [별칭2], ... }
FROM      테이블이름
[WHERE    조건(들)];
```

1. 행 제한 검색

✓ 행 제한 검색의 예

```
SELECT    emp_name,  job, dept_id
FROM      emp
WHERE     job = '세일즈';
```

EMP_NAME	JOB	DEPT_ID
민동규	세일즈	400
정종철	세일즈	400
진영진	세일즈	400

2. 대소 비교 연산자

대소 비교 연산자의 종류

종류	의미
=	같다(Equal To)
>	크다(Greater Than)
>=	크거나 같다(Greater Than or Equal To)
<	작다(Less Than)
<=	작거나 같다(Less Than or Equal To)
<>	다르다(Not Equal To)

✓ 숫자 뿐만 아니라 문자열과 날짜 값을 비교할 때도 사용함

2. 대소 비교 연산자

WHERE 절 작성 형식

- 대소 비교 연산자 앞에 열 이름을, 뒤에 비교할 값을 작성해야 함

```
WHERE  열_이름  대소비교연산자  비교할_값
```

피연산자 기술 시 주의사항

① 문자열과 날짜 값은 **단일 인용부호(')**로 묶어서 표시한다.

② 문자열은 대소문자를 엄격히 구분한다.

③ 날짜 값(DATE 타입)은 날짜 형식을 구분한다.

(디폴트 형식은 'YY/MM/DD' 형식이다.)

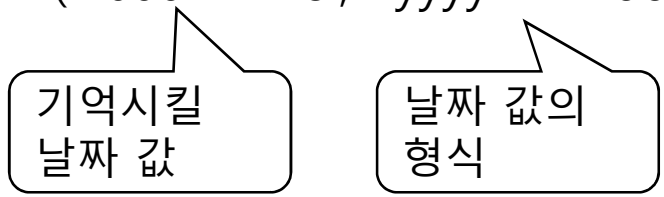
2. 대소 비교 연산자

[참고] 오라클의 DATE 타입 자료 처리 방법

1) 날짜 값을 기억시킬 때는 항상 **TO_DATE** 함수를 사용해야 함

예) TO_DATE('1999/12/18', 'yyyy/mm/dd')

예) TO_DATE('2000-12-18', 'yyyy-mm-dd')



예) emp 테이블에 새로운 데이터 삽입하기

```
INSERT INTO EMP VALUES (3262, '홍길동', '사무직', 100, 1100,
                        NULL, 6311, TO_DATE('09/01/2003', 'MM/DD/YYYY'));
```

2) 날짜 값을 비교할 때는 디폴트 형식인 '**YY/MM/DD**' 형식으로 비교해야 함

예)
```
SELECT    emp_name,   hiredate
FROM      emp
WHERE     hiredate = '03/09/01';
```

2. 대소 비교 연산자

✓ 대소 비교 연산자 사용 예

- 입사일이 "2001년 12월 3일"인 모든 사원 검색하기

```
SELECT    emp_name,  hiredate,  job
FROM      emp
WHERE     hiredate = '20/12/13';
```

EMP_NAME	HIREDATE	JOB
김마리아	20/12/03	사무직
정동길	20/12/03	연구직

3. 기타 비교 연산자

기타 비교 연산자의 종류

종류	의미	WHERE 절 기술 형식
BETWEEN a AND b	a와 b 두 값 사이에 포함되는 지 비교함	WHERE 열_이름 BETWEEN a AND b
IN (값 목록)	값 목록에 있는 어떤 값과 일치하는 지 비교함	WHERE 열_이름 IN (값 목록)
LIKE '문자패턴'	문자 패턴과 일치하는 지 비교함	WHERE 열_이름 LIKE '문자패턴'
IS NULL	널 값인 지 비교함	WHERE 열_이름 IS [NOT] NULL

3. 기타 비교 연산자

BETWEEN ... AND 연산자

- 비교 대상이 두 값 사이에 속하는지 판단하기 위한 연산자

```
WHERE  열_이름  BETWEEN  a  AND  b
```

- 열_이름의 값이 a보다 같거나 크고 b보다 같거나 작으면 참(true)이 됨

3. 기타 비교 연산자

✓ **BETWEEN ... AND 연산자 사용 예**

- 급여가 3000 이상, 5000 이하인 모든 사원 검색하기

```
SELECT    emp_name,   sal
FROM      emp
WHERE     sal   BETWEEN   3000 AND   5000;
```

```
WHERE   sal >= 3000
AND   sal <= 5000;
```

EMP_NAME	SAL
민동규	3700
정종철	4520
이영희	3200
백기수	5000
진영진	3500
정동길	4500

6 rows selected.

3. 기타 비교 연산자

IN 연산자

- 비교 대상이 괄호 안에 기술된 값의 리스트에 있는 어느 한 값과 일치하는지 판단하는 연산자

```
WHERE   열_이름  IN(값_목록)
```

- 열_이름의 값이 나열된 값 목록 가운데 하나와 일치하면 참(true)이 됨

3. 기타 비교 연산자

✓ IN 연산자 사용 예

- 관리자의 사원 번호가 6311, 6361, 6351 가운데 하나인 모든 사원의 사원번호와 이름, 관리자번호, 그리고 부서번호 검색하기

```
SELECT    emp_id, emp_name,  mgr_id, dept_id
FROM      emp
WHERE     mgr_id  IN (6311, 6361, 6351);
```

EMP_ID	EMP_NAME	MGR_ID	DEPT_ID
7910	이영희	6351	300
7878	백기수	6361	200
7872	이문정	6311	100
7920	김마리아	6351	300
7933	김철수	6361	200

```
WHERE  mgr_id = 6311
OR  mgr_id = 6361
OR  mgr_id = 6351;
```

3. 기타 비교 연산자

LIKE 연산자

- 비교 대상을 비교할 정확한 값을 모를 때, 대략의 문자열 패턴과 일치하는 지를 판단하기 위해 사용하는 연산자

```
WHERE  열_이름  LIKE  '문자패턴'
```

- 열_이름의 값이 LIKE 뒤에 기술된 '문자열' 패턴과 일치하면 참(true)이 됨

✓ **'문자열'에 사용할 수 있는 와일드카드(wildcard) 문자**

① **%** (퍼센트 기호) : 없거나 하나 이상의 문자를 나타냄

② _ (밑줄) : 임의의 한 문자를 나타냄

정확히 예측할 수 없는 문자를 대신 표현하는 문자

3. 기타 비교 연산자

✓ LIKE 연산자 사용 예

- 성이 '김'인 모든 사원의 이름과 담당업무와 근무부서번호 검색하기

```
SELECT    emp_name, job, dept_id
FROM      emp
WHERE     emp_name  LIKE  '김%';
```

EMP_NAME	JOB	DEPT_ID
김대정	부장	100
김철수	부장	200
김마리아	사무직	300
김철수	사무직	200

```
SELECT    emp_name, job, dept_id
FROM      emp
WHERE     emp_name  LIKE  '김__';
```

밑줄이 2개이므로 '김마리아'는 해당되지 않음

EMP_NAME	JOB	DEPT_ID
김대정	부장	100
김철수	부장	200
김철수	사무직	200

3. 기타 비교 연산자

✓ **LIKE 연산자를 사용한 날짜 비교 예**

- 1999년에 입사한 모든 사원의 이름과 입사일 검색하기

```
SELECT     emp_name, hiredate
FROM       emp
WHERE      hiredate   LIKE   '21%';
```

EMP_NAME	HIREDATE
이문정	21/01/12
김철수	21/01/23

3. 기타 비교 연산자

IS NULL 연산자

- 비교 대상이 널(NULL) 값을 갖는지를 판단할 때 사용하는 연산자

```
WHERE  열_이름  IS  [NOT]  NULL
```

✓ **주의 사항**

- 널 값은 어떤 값과도 같거나 다를 수 없으므로, 대소 비교 연산자인 '=' 또는 '<>' 연산자로는 비교할 수 없음

3. 기타 비교 연산자

✓ IS NULL 연산자 사용 예

- 업무상 보너스가 해당이 없는 모든 사원들 검색하기

```
SELECT    emp_name,  job,  bonus
FROM      emp
WHERE     bonus  IS  NULL;
```

EMP_NAME	JOB	BONUS
박철수	대표이사	
김대정	부장	
진대준	부장	
이영희	경리	
김철수	부장	
백기수	연구직	
이문정	사무직	
김마리아	사무직	
정동길	연구직	
김철수	사무직	

```
SELECT    emp_name,  job,  bonus
FROM      emp
WHERE     bonus  IS NOT NULL;
```

EMP_NAME	JOB	BONUS
민동규	세일즈	200
정종철	세일즈	300
이종길	부장	500
진영진	세일즈	0

숫자 0과 널(Null) 값은 다름

4. 논리 연산자

논리 연산자의 종류

종류	역할
AND	양쪽 조건이 모두 참일 때만 참이 됨
OR	한쪽 조건만 참이면 참이 됨
NOT	BETWEEN ~AND, LIKE, NULL, IN 등의 연산자와 함께 사용하여 그 결과를 부정함

논리 연산자의 우선순위

괄호 > 모든 비교 연산자 > NOT > AND > OR

4. 논리 연산자

✓ AND 연산자 사용 예

- 급여가 6000 이상인 모든 부장의 이름과 담당업무, 급여, 근무부서번호 검색하기

```
SELECT    emp_name, job, sal, dept_id
FROM      emp
WHERE     sal   >= 6000
AND       job = '부장';
```

EMP_NAME	JOB	SAL	DEPT_ID
김대정	부장	6200	100
이종길	부장	7000	400
김철수	부장	6500	200

4. 논리 연산자

✓ OR 연산자 사용 예

- 급여가 6000 이상이거나 담당업무가 부장인 모든 사원의 이름과 담당업무, 급여, 근무부서번호 검색하기

```
SELECT    emp_name, job, sal, dept_id
FROM      emp
WHERE     sal   >= 6000
OR        job = '부장';
```

EMP_NAME	JOB	SAL	DEPT_ID
박철수	대표이사	9000	200
김대정	부장	6200	100
이종길	부장	7000	400
진대준	부장	5850	300
김철수	부장	6500	200

4. 논리 연산자

✓ NOT 연산자 사용 예

- 담당업무가 세일즈나 사무직이 아닌 모든 사원의 이름과 담당업무, 급여, 근무부서번호 검색하기

```
SELECT    emp_name, job, sal, dept_id
FROM      emp
WHERE     job  NOT  IN  ('세일즈', '사무직');
```

EMP_NAME	JOB	SAL	DEPT_ID
박철수	대표이사	9000	200
김대정	부장	6200	100
이종길	부장	7000	400
진대준	부장	5850	300
이영희	경리	3200	300
김철수	부장	6500	200
백기수	연구직	5000	200
정동길	연구직	4500	

8 rows selected.

4. 논리 연산자

✓ 괄호를 이용한 검색 조건 작성 예

```
SELECT    emp_name, job, sal
FROM      emp
WHERE     job  = '경리'
OR        (job = '부장'
AND       sal  > 6000);
```

EMP_NAME	JOB	SAL
김대정	부장	6200
이종길	부장	7000
이영희	경리	3200
김철수	부장	6500

```
SELECT    emp_name, job, sal
FROM      emp
WHERE     (job  = '경리'
OR        job = '부장')
AND       sal > 6000;
```

EMP_NAME	JOB	SAL
김대정	부장	6200
이종길	부장	7000
김철수	부장	6500

괄호의 우선순위가 가장 높으므로
괄호의 위치에 따라 검색 결과가 달라짐

5. 행 정렬

ORDER BY 절을 활용한 행 정렬(오름차순/내림차순)

```
SELECT   열이름1 [열별칭1], 열이름2 [열별칭2], ...
FROM     테이블이름
WHERE    조건(들)
ORDER BY 열이름3 [열별칭3] [ASC | DESC] , ...
```

- ORDER BY 절은 SELECT 문장의 맨 마지막에 위치해야 함
- ASC : 오름차순 정렬(디폴트 값)
- DESC : 내림차순 정렬
- ORDER BY 절에서 열 이름 대신 SELECT 절에서 부여한 열 별칭을 사용할 수 있음
- ORDER BY 절에 여러 개의 열 이름을 기술한 경우, 열 이름의 순서가 정렬에 중요한 역할을 함
  - 즉, 먼저 기술된 열 이름을 기준으로 정렬한 다음, 그 값이 같은 경우 뒤에 기술한 열 이름을 기준으로 정렬함

5. 행 정렬

✓ 오름차순 정렬 예

```
SELECT    emp_name,  job,  hiredate
FROM      emp
ORDER  BY    hiredate;
```

EMP_NAME	JOB	HIREDATE
박철수	대표이사	17/12/17
정종철	세일즈	18/02/22
김대정	부장	18/12/17
민동규	세일즈	19/02/20
이종길	부장	19/04/02
진대준	부장	20/05/31
김철수	부장	20/06/09
이영희	경리	20/09/01
진영진	세일즈	20/09/08
김마리아	사무직	20/12/03
정동길	연구직	20/12/03
백기수	연구직	20/12/09
이문정	사무직	21/01/12
김철수	사무직	21/01/23

14 rows selected.

5. 행 정렬

✓ 내림차순 정렬 예

```
SELECT    emp_name,  job,  hiredate
FROM      emp
ORDER  BY    hiredate  DESC;
```

EMP_NAME	JOB	HIREDATE
김철수	사무직	21/01/23
이문정	사무직	21/01/12
백기수	연구직	20/12/09
정동길	연구직	20/12/03
김마리아	사무직	20/12/03
진영진	세일즈	20/09/08
이영희	경리	20/09/01
김철수	부장	20/06/09
진대준	부장	20/05/31
이종길	부장	19/04/02
민동규	세일즈	19/02/20
김대정	부장	18/12/17
정종철	세일즈	18/02/22
박철수	대표이사	17/12/17

14 rows selected.

5. 행 정렬

열 별칭을 활용한 행 정렬

- ORDER BY 절에서 열 이름 대신 SELECT 절에서 부여한 열 별칭을 사용할 수 있음

✓ 예 : 200번 부서에 근무하는 사원만 출력하되, 급여에 12를 곱한 연봉을 기준으로 연봉이 가장 많은 사원부터 사원번호와 이름, 연봉, 부서번호 검색하기

```
SELECT    emp_id, emp_name,  sal*12  annual, dept_id
FROM      emp
WHERE  dept_id = 200
ORDER BY  annual  DESC;
```

EMP_ID	EMP_NAME	ANNUAL	DEPT_ID
6200	박철수	108000	200
6361	김철수	78000	200
7878	백기수	60000	200
7933	김철수	30000	200

5. 행 정렬

다중 열에 의한 행 정렬

- 정렬 기준이 되는 열이 하나 이상인 경우, 먼저 기술된 열 이름을 기준으로 정렬한 후, 그 값이 같은 경우 뒤에 기술한 열 이름을 기준으로 정렬함
- 따라서 ORDER BY 절에 기술한 열 이름의 순서가 중요함

✓ 예 : 사원의 부서번호를 기준으로 오름차순으로 정렬하되, 같은 부서 안에서는 급여가 많은 사람 먼저 출력하기

```
SELECT      emp_id, dept_id, sal
FROM        emp
ORDER BY    dept_id,  sal  DESC;
```

EMP_ID	DEPT_ID	SAL
6311	100	6200
7872	100	2500
6200	200	9000
6361	200	6500
7878	200	5000
7933	200	2500
6351	300	5850
7910	300	3200
7920	300	2650
6321	400	7000
7522	400	4520
7489	400	3700
7854	400	3500
7901		4500

14 rows selected.

Q & A