4-2. 집합 쿼리 – 집합 연산자

홍형경
chariehong@gmail.com
2020.01

1. 집합 쿼리

- 집합 연산자를 사용한 쿼리

- 수학에서 배운 집합 개념과 동일 (합집합, 교집합 등)

- 하나의 SELECT 문장이 반환한 결과를 한 집합으로 보고, 한 개 이상의 SELECT 문장이 집합 연산자로 연결된 형태

- 여러 개의 SELECT 문이 연결되어 최종적으로는 하나의 결과 집합이 만들어짐

1. 집합 쿼리

SELECT …
FROM …
WHERE …
집합연산자
SELECT …
FROM …
WHERE …
집합연산자
…

하나의 집합 쿼리

<제한사항>

· 각 SELECT 절의 컬럼 수, 데이터 타입은 동일

· 최종 반환되는 컬럼 명은 맨 첫 SELECT 절의 컬럼 이름을 따름

· ORDER BY 절은 맨 마지막 SELECT 문장에서만 붙일 수 있음

2. 집합 연산자

· 집합 쿼리는 집합 연산자를 사용해 SELECT 문장을 연결하는 형태

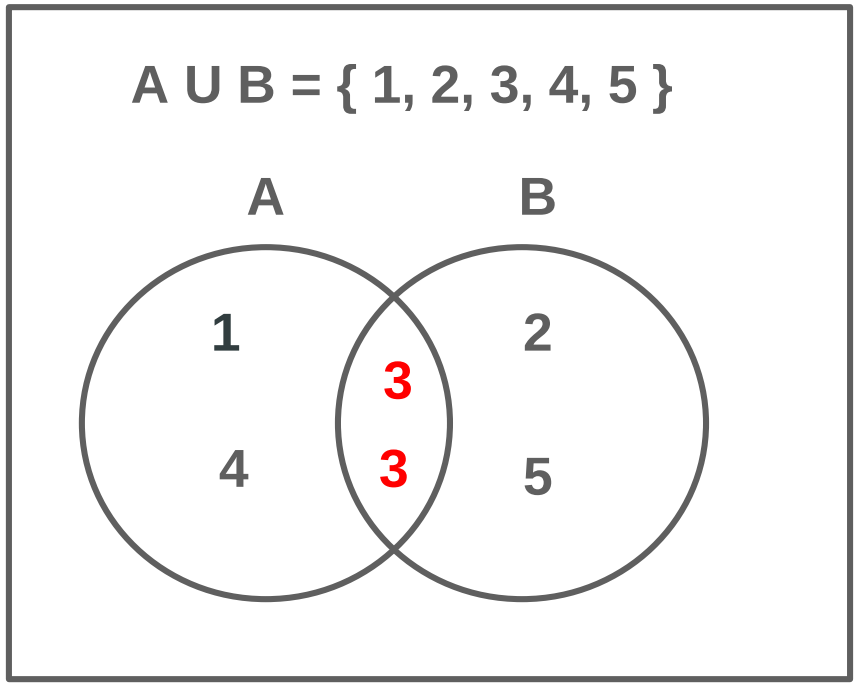
· UNION, UNION ALL, INTERSECT, MINUS 4개 연산자 존재

· 집합 연산자는 수학의 집합 개념과 유사

· 각 SELECT 문이 반환하는 결과를 하나의 집합으로 보고 집합 연산자를 통해 연결

2. 집합 연산자 - UNION

· 두 집합의 모든 원소를 가져오는 합집합 개념



테이블	col1
Tbl_A	1
	3
	4
Tbl_B	2
	3
	5

```
SELECT col1
    FROM Tbl_A
UNION
SELECT col1
    FROM Tbl_B
ORDER BY 1;
```

col1
1
2
3
4
5

2. 집합 연산자 - UNION

```
SELECT col1
    FROM Tbl_A
UNION
SELECT col1
    FROM Tbl_B
ORDER BY 1;
```

- 두 문장의 SELECT 절에 명시하는 컬럼 수, 데이터 타입은 동일해야 함
- 조회된 결과의 컬럼명은 첫 번째 SELECT 문장의 컬럼명으로 보임
- ORDER BY 절은 맨 마지막에 붙일 수 있음 (생략 가능)
- 각 결과 집합에서 조회된 중복 값은 1번만 조회됨

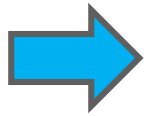
2. 집합 연산자 – UNION ALL

· UNION과 동일하나 중복 값도 모두 조회됨

· 나머지 내용은 UNION 과 동일

테이블	col1
Tbl_A	1
	3
	4
Tbl_B	2
	3
	5

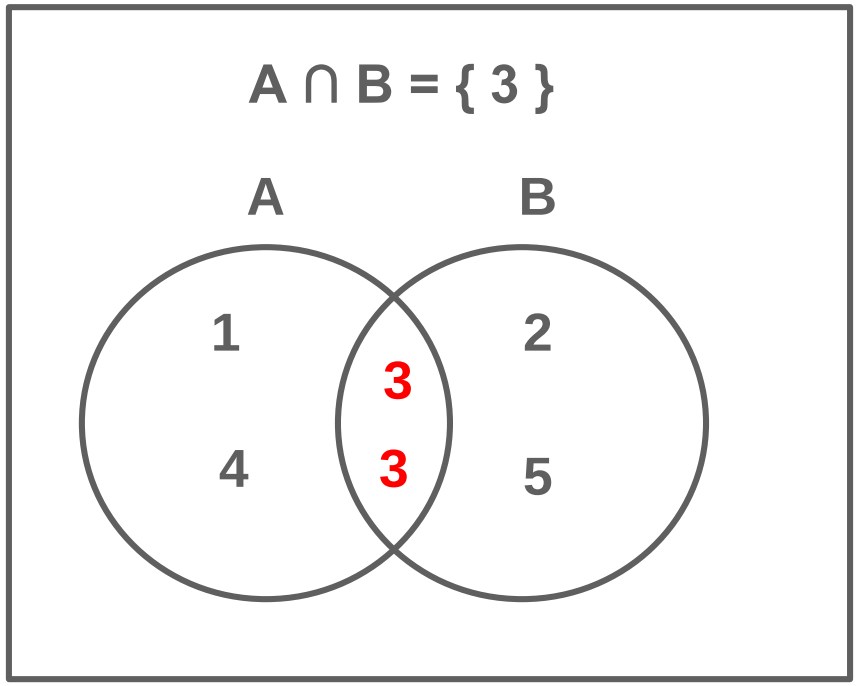
```
SELECT col1
    FROM Tbl_A
UNION ALL
SELECT col1
    FROM Tbl_B
ORDER BY 1;
```



col1
1
2
3
3
4
5

2. 집합 연산자 - INTERSECT

· 두 집합의 공통 원소를 가져오는 교집합 개념 (Distinct Row)



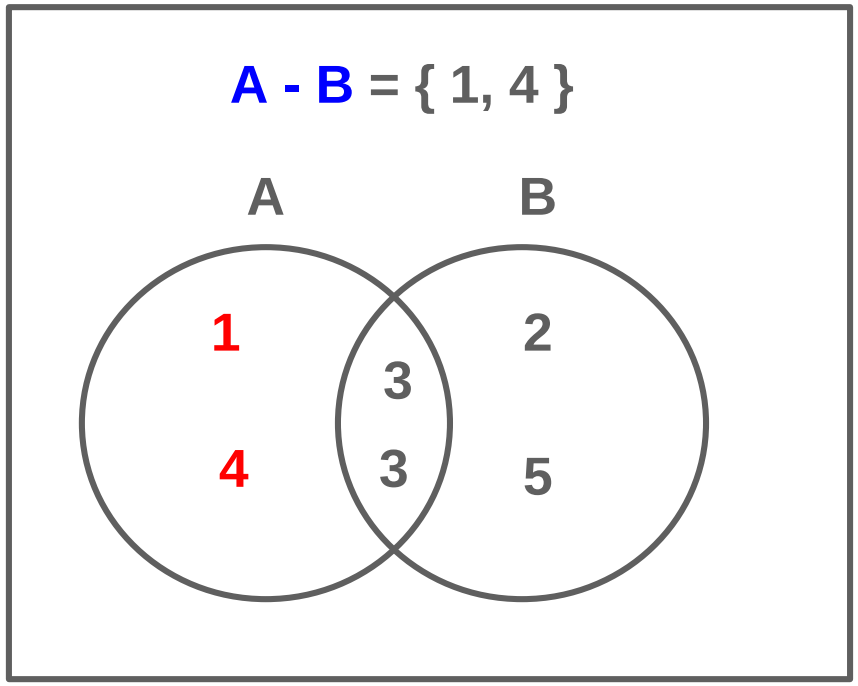
테이블	col1
Tbl_A	1
	3
	4
Tbl_B	2
	3
	5

```
SELECT col1
    FROM Tbl_A
INTERSECT
SELECT col1
    FROM Tbl_B
ORDER BY 1;
```

col1
3

2. 집합 연산자 - MINUS

· 선두 집합 에만 있는 원소를 가져오는 차집합 개념 (Distinct Row)



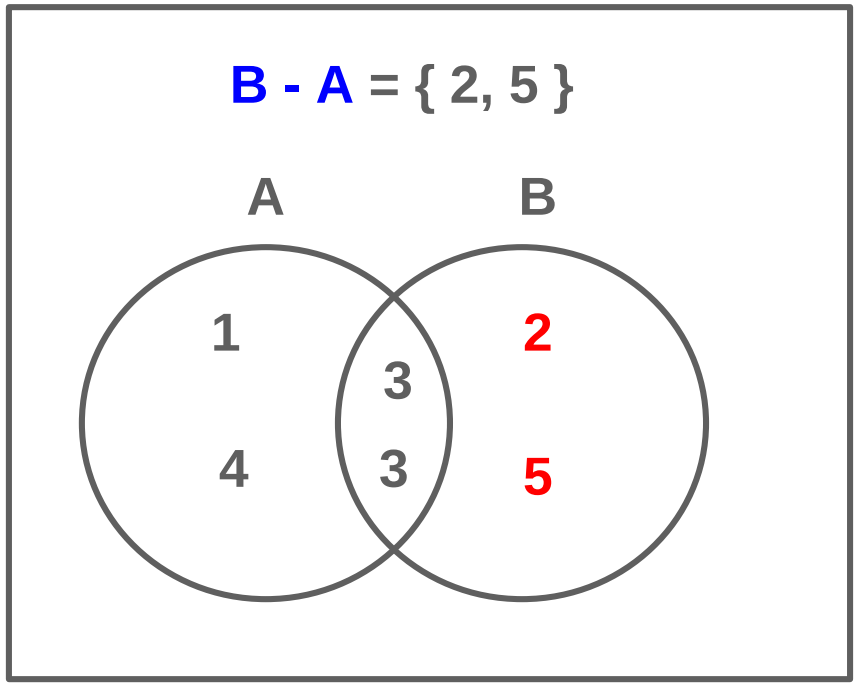
테이블	col1
Tbl_A	1
	3
	4
Tbl_B	2
	3
	5

```
SELECT col1
    FROM Tbl_A
MINUS
SELECT col1
    FROM Tbl_B
ORDER BY 1;
```

col1
1
4

2. 집합 연산자 - MINUS

· 먼저 명시한 SELECT 문의 결과 집합이 기준 (Distinct Row)



테이블	col1
Tbl_A	1
	3
	4
Tbl_B	2
	3
	5

```
SELECT col1
    FROM Tbl_B
MINUS
SELECT col1
    FROM Tbl_A
ORDER BY 1;
```

col1
2
5

3. 집합연산자 실습

(1) UNION

```
-- A 집합
SELECT job_id
  FROM employees
 WHERE 1=1
   AND salary BETWEEN 2000 and 5000
 ORDER BY job_id;

SELECT DISTINCT job_id
  FROM employees
 WHERE salary BETWEEN 2000 and 5000
 ORDER BY job_id;
```

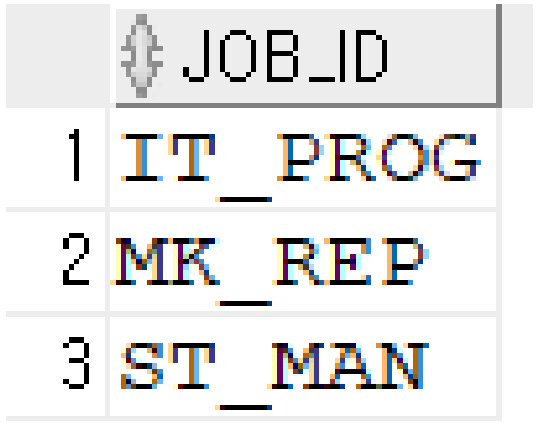
	JOB_ID
1	AD_ASST
2	IT_PROG
3	IT_PROG
4	IT_PROG
5	PU_CLERK
6	PU_CLERK
7	PU_CLERK
8	PU_CLERK
9	PU_CLERK
10	SH_CLERK
11	SH_CLERK
12	SH_CLERK
13	SH_CLERK
14	SH_CLERK
15	SH_CLERK
16	SH_CLERK

	JOB_ID
1	AD_ASST
2	IT_PROG
3	PU_CLERK
4	SH_CLERK
5	ST_CLERK

3. 집합연산자 실습

(1) UNION

```
-- B 집합
SELECT job_id
  FROM employees
 WHERE 1=1
   AND salary BETWEEN 5001 AND 6000
 ORDER BY job_id;
```



	JOB_ID
1	IT_PROG
2	MK_REP
3	ST_MAN

3. 집합연산자 실습

(1) UNION

```
-- A집합
SELECT job_id
  FROM employees
 WHERE 1=1
   AND salary BETWEEN 2000 and 5000
 UNION
-- B 집합
SELECT job_id
  FROM employees
 WHERE 1=1
   AND salary BETWEEN 5001 AND 6000
 ORDER BY job_id;
```

	JOB_ID
1	AD_ASST
2	IT_PROG
3	PU_CLERK
4	SH_CLERK
5	ST_CLERK

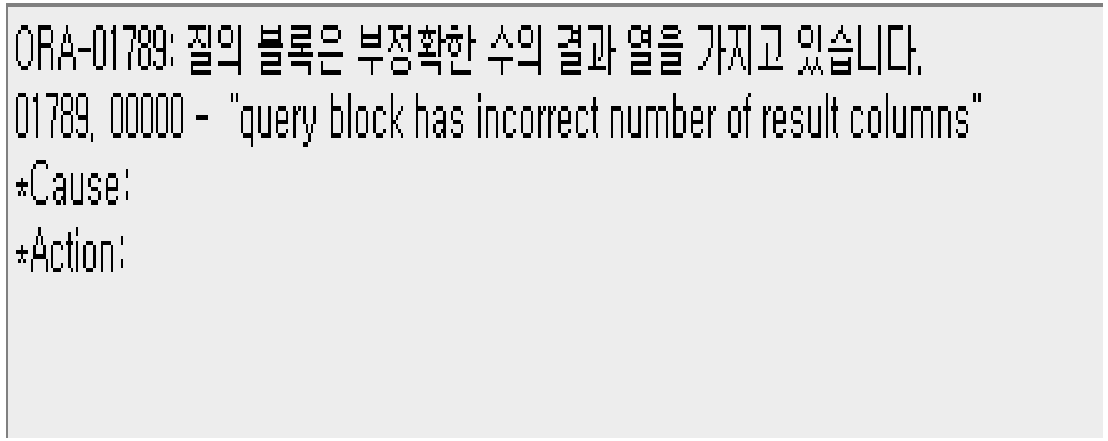
	JOB_ID
1	IT_PROG
2	MK_REP
3	ST_MAN

	JOB_ID
1	AD_ASST
2	IT_PROG
3	MK_REP
4	PU_CLERK
5	SH_CLERK
6	ST_CLERK
7	ST_MAN

3. 집합연산자 실습

(1) UNION

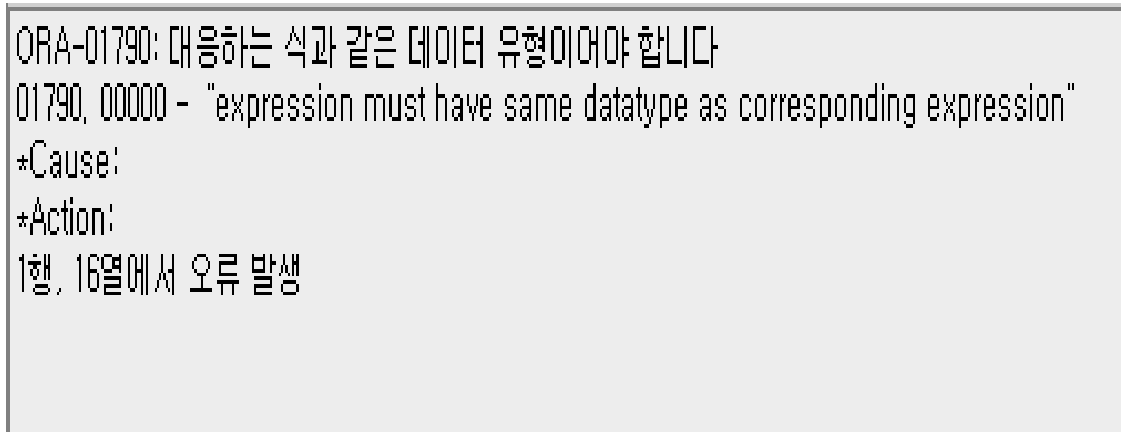
```
SELECT job_id, salary
  FROM employees
 WHERE 1=1
   AND salary BETWEEN 2000 and 5000
 UNION
SELECT job_id
  FROM employees
 WHERE 1=1
   AND salary BETWEEN 5001 AND 6000
 ORDER BY job_id;
```



3. 집합연산자 실습

(1) UNION

```
SELECT job_id, salary
  FROM employees
 WHERE 1=1
   AND salary BETWEEN 2000 and 5000
 UNION
SELECT job_id, phone_number
  FROM employees
 WHERE 1=1
   AND salary BETWEEN 5001 AND 6000
 ORDER BY job_id;
```



3. 집합연산자 실습

(1) UNION

```
SELECT job_id, salary
  FROM employees
 WHERE 1=1
   AND salary BETWEEN 2000 and 5000
 UNION
SELECT job_id, department_id
  FROM employees
 WHERE 1=1
   AND salary BETWEEN 5001 AND 6000
 ORDER BY job_id;
```

	JOB_ID	SALARY
1	AD_ASST	4400
2	IT_PROG	60
3	IT_PROG	4200
4	IT_PROG	4800
5	MK_REP	20
6	PU_CLERK	2500
7	PU_CLERK	2600
8	PU_CLERK	2800
9	PU_CLERK	2900
10	PU_CLERK	3100
11	SH_CLERK	2500
12	SH_CLERK	2600
13	SH_CLERK	2800
14	SH_CLERK	2900

구문 오류는 없으나, 의미상 오류

3. 집합연산자 실습

(2) UNION ALL

```
SELECT job_id
  FROM employees
 WHERE 1=1
   AND salary BETWEEN 2000 and 5000
 UNION ALL
SELECT job_id
  FROM employees
 WHERE 1=1
   AND salary BETWEEN 5001 AND 6000
 ORDER BY job_id;
```

	JOB_ID
1	AD_ASST
2	IT_PROG
3	IT_PROG
4	IT_PROG
5	IT_PROG
6	MK_REP
7	PU_CLERK
8	PU_CLERK
9	PU_CLERK
10	PU_CLERK
11	PU_CLERK
12	SH_CLERK
13	SH_CLERK
14	SH_CLERK

3. 집합연산자 실습

(3) INTERSECT

```
SELECT job_id
  FROM employees
 WHERE 1=1
   AND salary BETWEEN 2000 and 5000
 INTERSECT
SELECT job_id
  FROM employees
 WHERE 1=1
   AND salary BETWEEN 5001 AND 6000
 ORDER BY job_id;
```

	JOB_ID
1	AD_ASST
2	IT_PROG
3	PU_CLERK
4	SH_CLERK
5	ST_CLERK

	JOB_ID
1	IT_PROG
2	MK_REP
3	ST_MAN

	JOB_ID
1	IT_PROG

3. 집합연산자 실습

(4) MINUS (A – B)

```
SELECT job_id
  FROM employees
 WHERE 1=1
   AND salary BETWEEN 2000 and 5000
 MINUS
SELECT job_id
  FROM employees
 WHERE 1=1
   AND salary BETWEEN 5001 AND 6000
 ORDER BY job_id;
```

	JOB_ID
1	AD_ASST
2	IT_PROG
3	PU_CLERK
4	SH_CLERK
5	ST_CLERK

	JOB_ID
1	IT_PROG
2	MK_REP
3	ST_MAN

	JOB_ID
1	AD_ASST
2	PU_CLERK
3	SH_CLERK
4	ST_CLERK

3. 집합연산자 실습

(4) MINUS (B – A)

```
SELECT job_id
  FROM employees
 WHERE 1=1
   AND salary BETWEEN 5001 AND 6000
 MINUS
SELECT job_id
  FROM employees
 WHERE 1=1
   AND salary BETWEEN 2000 and 5000
 ORDER BY job_id;
```

	JOB_ID
1	AD_ASST
2	IT_PROG
3	PU_CLERK
4	SH_CLERK
5	ST_CLERK

	JOB_ID
1	IT_PROG
2	MK_REP
3	ST_MAN

	JOB_ID
1	MK_REP
2	ST_MAN

학습정리

· 집합 쿼리는 독립적인 여러 개의 쿼리가 집합 연산자로 연결된 형태의 쿼리이다.

· 집합 연산자는 UNION, UNION ALL, INTERSECT, MINUS 총 4개가 있는데, UNION은 합집합, INTERSECT는 교집합, MINUS는 차집합과 같은 동작을 한다.

· UNION, MINUS는 중복 값이 한 번만 조회되는 반면, UNION ALL은 모든 값이 조회된다.

· MINUS는 선두 쿼리 결과 집합에서 후행 쿼리 결과 집합을 빼는 동작을 하므로 어떤 쿼리를 선두에 놓는지에 따라 결과가 달라진다. 반면 UNION, UNION ALL, INTERSECT는 쿼리 순서에 상관 조회되는 결과는 같다. (순서는 다를 수 있다)

Quiz

1. 다음 쿼리를 실행하면 오류가 발생하는데 그 이유는 무엇일까요?

```
SELECT job_id jobs
  FROM employees
 WHERE department_id = 60
UNION
SELECT job_id
  FROM employees
 WHERE department_id = 90
 ORDER BY job_id;
```


Quiz

2. 집합 연산자를 사용해 employees 테이블에서 2001과 2003년에 입사한 사원의 사원번호와 입사일자를 조회하는 쿼리를 작성해 보세요.

Quiz

3. employees 테이블에서 job_id 별로 급여(salary)의 합계를 구하고, 마지막에 전체 급여 합계를 구하는 쿼리를 UNION 연산자를 사용해 작성해 보세요.

