

database-8

☰ 태그	
🕒 날짜	@2023년 6월 13일

두 개 이상의 테이블을 합쳐서 조회하고 싶을 때 사용되는 join 문법을 설명

내용

- join 의미
- implicit join vs explicit join
- inner join vs outer join
- left outer join
- right outer join
- full outer join
- equi join
- using 키워드를 사용한 조인
- natural join
- cross join

SQL에서 JOIN이란?

- 두개 이상의 table들에 있는 데이터를 한번에 조회하는 것
- 여러 종류의 join이 존재한다.

implicit join과 explicit join의 차이

implicit join :

- from 절에는 table들만 나열하고 where 절에 join condition을 명시하는 방식
- old-style join syntax
- where 절에 selection condition과 join condition이 같이 있기 때문에 가독성이 떨어진다.
- 복잡한 join 쿼리를 작성하다 보면 실수로 잘못된 쿼리를 작성할 가능성이 크다.

explicit join :

```
SELECT D.name
FROM employee AS E, department AS D
WHERE E.id = 1 and E.dept_id = D.id;

// 아래로 변경

SELECT D.name
FROM employee AS E JOIN department AS D ON E.dept_id = D.id
WHERE E.id = 1;
```

- from 절에 join 키워드와 함께 joined table들을 명시 하는 방식
- from 절에 on 뒤에 join condition이 명시된다.
- 가독성이 좋다.
- 복잡한 join 쿼리 작성중에도 실수할 가능성이 적다.

inner join

```
SELECT *
FROM employee E JOIN department D ON E.dept_id = D.id;

// JOIN앞에 INNER 키워드가 생략된 것이다.

SELECT *
FROM employee E INNER JOIN department D ON E.dept_id = D.id;
```

joined table							
E attributes				D attributes			
id	name	dept_id	id	name	leader_id
1	MESSI	1003	1003	DEV	1
2	JANE	1004	1004	DESIGN	3
3	JENNY	1003	1003	DEV	1
4	BROWN	1001	1001	HQ	4
13	JISUNG	1005	1005	PRODUCT	13

INNER JOIN

- 두 table에서 join condition을 만족하는 tuple들로 result table을 만드는 join
- FROM table1 [INNER] JOIN table2 ON join_condition
- join condition에 사용 가능한 연산자 (operator) : =, <, >, != 등등 여러 비교 연산자가 가능
- join condition에서 null 값을 가지는 tuple은 result table에 포함되지 못한다.

OUTER JOIN

- outer join : 두 table에서 join condition을 만족하지 않는 tuple들도 result table에 포함하는 join
- FROM table1 LEFT [OUTER] JOIN table2 ON join_condition
- FROM table1 RIGHT [OUTER] JOIN table2 ON join_condition
- FROM table1 FULL [OUTER] JOIN table2 ON join_condition
- join condition에 사용 가능한 연산자(operator) : =, <, >, != 등등 여러 비교 연산자가 가능

```
left outer join
SELECT *
FROM employee E LEFT OUTER JOIN department D ON E.dept_id = D.id;
```

mysql> SELECT *
-> FROM employee E LEFT OUTER JOIN department D ON E.dept_id = D.id;

joined table							
E attributes				D attributes			
id	name	dept_id	id	name	leader_id
1	MESSI	1003	1003	DEV	1
2	JANE	1004	1004	DESIGN	3
3	JENNY	1003	1003	DEV	1
4	BROWN	1001	1001	HQ	4
13	JISUNG	1005	1005	PRODUCT	13
15	SIMON	null	null	null	null

- SIMON 이 포함됨

```
right outer join
SELECT *
FROM employee E right OUTER JOIN department D ON E.dept_id = D.id;
```

joined table							
id	name	dept_id	id	name	leader_id
4	BROWN	1001	1001	HQ	4
null	null	null	null	null	1002	HR	null
3	JENNY	1003	1003	DEV	1
1	MESSI	1003	1003	DEV	1
2	JANE	1004	1004	DESIGN	3
13	JISUNG	1005	1005	PRODUCT	13

E attributes D attributes

- simon의 대한 정보는 null 로 표시됨

```
full outer join // mysql에서는 지원하지 않음

SELECT *
FROM employee E FULL OUTER JOIN department D ON E.dept_id = D.id;
```

joined table

id	name	dept_id	id	name	leader_id
1	MESSI	1003	1003	DEV	1
2	JANE	1004	1004	DESIGN	3
3	JENNY	1003	1003	DEV	1
4	BROWN	1001	1001	HQ	4
13	JISUNG	1005	1005	PRODUCT	13
15	SIMON	null	null	null	null
null	null	null	null	null	1002	HR	null

E attributes

D attributes

equi join : join condition에서 = (equality comparator)를 사용하는 join

inner join

```
SELECT *
FROM employee E INNER JOIN department D ON E.dept_id = D.id;
```

left outer join

```
SELECT *
FROM employee E LEFT OUTER JOIN department D ON E.dept_id = D.id;
```

right outer join

```
SELECT *
FROM employee E right OUTER JOIN department D ON E.dept_id = D.id;
```

full outer join // mysql에서는 지원하지 않음

```
SELECT *
FROM employee E FULL OUTER JOIN department D ON E.dept_id = D.id;
```

= 연산자를 사용했기 때문에 네가지 모두 equi join이다.

equi join에 대한 두가지 시각

- inner join outer join 상관없이 = 를 사용한 join이라면 equi join으로 보는 경우
- inner join으로 한정해서 = 를 사용한 경우에 equi join으로 보는 경우

using

joined table							
EMPLOYEE				DEPARTMENT			
<u>dept_id</u>	<u>name</u>	...	<u>salary</u>	<u>dept_id</u>	<u>dept_id</u>	<u>name</u>	<u>leader_id</u>
1	MESSI	1003	1003	DEV	1
2	JANE	1004	1004	DESIGN	3
3	JENNY	1003	1003	DEV	1
4	BROWN	1001	1001	HQ	4
13	JISUNG	1005	1005	PRODUCT	13
15	SIMON	null			

joined table							
E attributes				D attributes			
<u>id</u>	<u>name</u>	...	<u>salary</u>	<u>dept_id</u>	<u>dept_id</u>	<u>name</u>	<u>leader_id</u>
1	MESSI	1003	1003	DEV	1
2	JANE	1004	1004	DESIGN	3
3	JENNY	1003	1003	DEV	1
4	BROWN	1001	1001	HQ	4
13	JISUNG	1005	1005	PRODUCT	13
15	SIMON				

dept_id 가 중복되어서 출력이 되는데 굳이 두개를 출력을 해야하나 싶을때 using을 사용

```
mysql> SELECT *
-> FROM employee E INNER JOIN department D USING (dept_id);
```

using 사용 후

joined table							
USING attribute		E attributes			D attributes		
<u>dept_id</u>	<u>id</u>	<u>name</u>	...	<u>salary</u>	<u>name</u>	<u>leader_id</u>	
1003	1	MESSI	DEV	1	
1004	2	JANE	DESIGN	3	
1003	3	JENNY	DEV	1	
1001	4	BROWN	HQ	4	
1005	13	JISUNG	PRODUCT	13	
15		SIMON	...				

using

- 두 table이 equi join 할 때 join하는 attribute의 이름이 같다면, USING으로 간단하게 작성할 수 있다.
- 이 때 같은 이름의 attribute는 result table에서 한번만 표시된다.
- FROM table1 [INNER] JOIN table2 USING (attribute(s))
- FROM table1 LEFT [OUTER] JOIN table2 USING (attribute(s))
- FROM table1 RIGHT [OUTER] JOIN table2 USING (attribute(s))
- FROM table1 FULL [OUTER] JOIN table2 USING (attribute(s))

natural join

- 두 table에서 같은 이름을 가지는 모든 attribute pair에 대해서 equi join을 수행
- join condition을 따로 명시하지 않는다.
- FROM table1 NATURAL [INNER] JOIN table2
- FROM table1 NATURAL LEFT [OUTER] JOIN table2
- FROM table1 NATURAL RIGHT [OUTER] JOIN table2
- FROM table1 NATURAL FULL [OUTER] JOIN table2

The diagram illustrates a natural inner join between the EMPLOYEE and DEPARTMENT tables. The EMPLOYEE table has columns: id, name, ..., salary, dept_id. The DEPARTMENT table has columns: dept_id, dept_name, leader_id. A green oval highlights the 'dept_id' column in both tables, indicating it is the common attribute used for joining. A blue oval highlights the 'dept_id' column in the DEPARTMENT table, which is being joined with the 'dept_id' column from the EMPLOYEE table. A yellow oval highlights the 'dept_id' column in the EMPLOYEE table, which is being joined with the 'dept_id' column from the DEPARTMENT table. The resulting joined table is shown below.

joined table						
dept_id	id	name	...	salary	dept_name	leader_id
1003	1	MESSI	DEV	1
1004	2	JANE	DESIGN	3
1003	3	JENNY	DEV	1
1001	4	BROWN	HQ	4
1005	13	JISUNG	PRODUCT	13

The diagram shows the natural inner join of the EMPLOYEE and DEPARTMENT tables. The joined table is labeled 'joined table' at the top. Below it, the columns are grouped into three categories: 'common attribute', 'E attributes', and 'D attributes'. The 'common attribute' group contains the 'dept_id' column. The 'E attributes' group contains the 'id', 'name', '...', 'salary' columns. The 'D attributes' group contains the 'dept_name' and 'leader_id' columns. The data in the joined table is identical to the one in the previous diagram.

joined table						
dept_id	id	name	...	salary	dept_name	leader_id
1003	1	MESSI	DEV	1
1004	2	JANE	DESIGN	3
1003	3	JENNY	DEV	1
1001	4	BROWN	HQ	4
1005	13	JISUNG	PRODUCT	13

같은 attribute가 두 개 일 경우

EMPLOYEE					DEPARTMENT		
<u>id</u>	name	...	salary	dept_id	<u>dept_id</u>	name	leader_id
1	MESSI	1003	1001	HQ	4
2	JANE	1004	1002	HR	null
3	JENNY	1003	1003	DEV	1
4	BROWN	1001	1004	DESIGN	3
13	JISUNG	1005	1005	PRODUCT	13
15	SIMON	null			

- 반환되는 결과는 아무것도 없게 된다.

cross join

- 두 table의 tuple pair 로 만들 수 있는 모든 조합 (= Cartesian product)을 result table로 반환
- join condition이 없다.
- implicit cross join : FROM table1, table2
- explicit cross join : FROM table1 CROSS JOIN table2

mysql> SELECT * -> FROM employee CROSS JOIN department;							
EMPLOYEE				DEPARTMENT			
<u>id</u>	name	...	salary	dept_id	<u>dept_id</u>	name	leader_id
1	MESSI	1003	1001	HQ	4
2	JANE	1004	1002	HR	null
3	JENNY	1003	1003	DEV	1
4	BROWN	1001	1004	DESIGN	3
13	JISUNG	1005	1005	PRODUCT	13
15	SIMON	null			

EMPLOYEE					DEPARTMENT		
<u><u>id</u></u>	<u><u>name</u></u>	<u><u>...</u></u>	<u><u>salary</u></u>	<u><u>dept_id</u></u>	<u><u>id</u></u>	<u><u>name</u></u>	<u><u>leader_id</u></u>
1	MESSI	1003	1001	HQ	4
2	JANE	1004	1002	HR	null
3	JENNY	1003	1003	DEV	1
4	BROWN	1001	1004	DESIGN	3
13	JISUNG	1005	1005	PRODUCT	13
15	SIMON	null			

mysql> SELECT *
-> FROM employee CROSS JOIN department;

result table

<u><u>id</u></u>	<u><u>name</u></u>	<u><u>...</u></u>	<u><u>salary</u></u>	<u><u>dept_id</u></u>	<u><u>id</u></u>	<u><u>name</u></u>	<u><u>leader_id</u></u>
1	MESSI	1003	1001	HQ	4
1	MESSI	1003	1002	HR	null
1	MESSI	1003	1003	DEV	1
1	MESSI	1003	1004	DESIGN	3
1	MESSI	1003	1005	PRODUCT	13
2	JANE	1004	1001	HQ	4
2	JANE	1004	1002	HR	null
...

cross join @MySQL

- MySQL에서는 cross join = inner join = join 이다
- cross join에 on(or USING) 을 같이 쓰면 INNER JOIN으로 동작한다.
- INNER JOIN(or JOIN)| ON(or USING) 없이 사용되면 cross join으로 동작한다.

self join

- table0| 자기 자신에게 join하는 경우