

database-4

☰ 태그	
🕒 날짜	@2023년 6월 12일

SQL로 테이블에 데이터를 추가, 수정, 삭제하는 문법

내용

- 데이터 추가하기 (insert)
- 데이터 수정하기 (update)
- 데이터 삭제하기 (delete)
- 조건절과 논리/비교 연산자 (where)

데이터 추가 문법

```
INSERT INTO 테이블명
-> VALUES(입력값, 입력값, 입력값...) // 순서는 테이블을 처음만들 때 attribute 순서

query ok, 1 row affected (0.05sec)
```

```
mysql> INSERT INTO employee
-> VALUES (1, 'MESSI', '1987-02-01', 'M', 'DEV_BACK', 100000000, null);

Query OK, 1 row affected (0.05 sec)

mysql> INSERT INTO employee
-> VALUES (1, 'JANE', '1996-05-05', 'F', 'DSGN', 90000000, null);

ERROR 1062 (23000): Duplicate entry '1' for key 'employee.PRIMARY'
```

두번째 입력은 1이라는 primary 키가 동일하게 있어서 실패하게 된다.

primary key 중복 실패

```
mysql> SHOW CREATE TABLE employee;
...
... CONSTRAINT `employee_chk_2` CHECK ((`salary` >= 50000000))
...
```

- SHOW CREATE TABLE 테이블명
- 테이블 제약 확인

```
mysql> INSERT INTO employee (name, birth_date, sex, position, id)
-> VALUES ('JENNY', '2000-10-12', 'F', 'DEV_BACK', 3);
```

- 작성한 attribute에 값을 넣겠다.
- 작성한 순서대로 입력값을 입력하면 되겠다.

전체 attribute에 대한 값을 가져오기

- SELECT * FROM employee;

```
+-----+-----+-----+-----+-----+-----+
| id | name  | birth_date | sex | position | salary | dept_id |
+-----+-----+-----+-----+-----+-----+
| 1 | MESSI | 1987-02-01 | M  | DEV_BACK | 100000000 | NULL   |
| 2 | JANE  | 1996-05-05 | F  | DSGN    | 90000000  | NULL   |
| 3 | JENNY | 2000-10-12 | F  | DEV_BACK | 50000000  | NULL   |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

jenny에 연봉정보를 넣지 않았어도 default값으로 50000000로 설정해놓았기 때문에
50000000으로 설정됨

INSERT statement

데이터를 하나만 추가하는 경우.

처음 CREATE TABLE을 할때 attribute 순서대로 모든 attribute에 대응하는 값을 넣어주는 쿼리

- `INSERT INTO table_name VALUES(comma-separated all values);`
넣으려는 값. 모든 attribute에 대한 값을 지정해주어야한다.

내가 원하는 일부 attribute의 대해서만 원하는 순서대로 넣고 싶을 때

- `INSERT INTO table_name (attributes list)`
내가 넣으려는 attribute 순서
`VALUES (attributes list 순서와 동일하게 comma-separated values);`

위 두개는 테이블에 하나에 데이터를 넣을 때 해당

한번에 한 테이블에 여러개의 데이터를 넣어줄 때 사용

- `INSERT INTO table_name VALUES(., .), (., .), (., .);`

```
mysql> insert into employee values
-> (4, 'BROWN', '1996-03-13', 'M', 'CEO', 120000000, null),
-> (5, 'DINGYO', '1990-11-05', 'M', 'CTO', 120000000, null),
-> (6, 'JULIA', '1986-12-11', 'F', 'CFO', 120000000, null),
-> (7, 'MINA', '1993-06-17', 'F', 'DSGN', 80000000, null),
-> (8, 'JOHN', '1999-10-22', 'M', 'DEV_FRONT', 65000000, null),
-> (9, 'HENRY', '1982-05-20', 'M', 'HR', 82000000, null),
-> (10, 'NICOLE', '1991-03-26', 'F', 'DEV_FRONT', 90000000, null),
-> (11, 'SUZANNE', '1993-03-23', 'F', 'PO', 75000000, null),
-> (12, 'CURRY', '1998-01-15', 'M', 'PLN', 85000000, null),
-> (13, 'JISUNG', '1989-07-07', 'M', 'PO', 90000000, null),
```

```
mysql> insert into department values
-> (1001, 'headquarter', 4),
-> (1002, 'HR', 6),
-> (1003, 'development', 1),
-> (1004, 'design', 3),
-> (1005, 'product', 13);
```

Query OK, 5 rows affected (0.00 sec)
Records: 5 Duplicates: 0 Warnings: 0

(1001, 'headquarter', 4)

(id, 'depart_id', emp_id)

```
mysql> insert into project values
-> (2001, '쿠폰 구매/선물 서비스 개발', 13, '2022-03-10', '2022-07-09'),
-> (2002, '확장성 있게 백엔드 리팩토링', 13, '2022-01-23', '2022-03-23'),
-> (2003, '홈페이지 UI 개선', 11, '2022-05-09', '2022-06-11');
```

Query OK, 3 rows affected (0.01 sec)
Records: 3 Duplicates: 0 Warnings: 0

(proj_id, proj_name, emp_id, start_date, end_date)

```
mysql> insert into works_on values
-> (5, 2001),
-> (13, 2001),
-> (1, 2001),
-> (8, 2001),
...
-> (7, 2003),
-> (2, 2003),
-> (12, 2003);
```

데이터 수정

```
UPDATE statemenst
```

- employee ID가 1인 Messi는 개발(development) 팀 소속이다.
- 개발팀 ID는 1003이다
- Messi의 소속팀 정보를 업데이트 해주자

```
UPDATE employee SET dept_id = 1003 WHERE id = 1;  
업데이트해주려는 attribute이름 = 업데이트할 값 where messi의 아이디인 id =1;
```

업데이트가 되었는지 확인

```
SELECT * FROM employee WHERE id = 1;  
id 1에서만 확인하고 싶으니 where id = 1;을 넣어준다.
```

```
mysql> SELECT * FROM employee WHERE id = 1;  
+----+-----+-----+-----+-----+  
| id | name | birth_date | sex | position | salary | dept_id |  
+----+-----+-----+-----+-----+  
| 1 | MESSI | 1987-02-01 | M | DEV_BACK | 100000000 | 1003 |  
+----+-----+-----+-----+-----+
```

```
mysql> UPDATE employee SET dept_id = 1004 WHERE id = 2;  
mysql> UPDATE employee SET dept_id = 1003 WHERE id = 3;  
...  
...  
mysql> UPDATE employee SET dept_id = 1005 WHERE id = 13;  
mysql> UPDATE employee SET dept_id = 1003 WHERE id = 14;
```

```
UPDATE statement
```

- 개발팀 연봉을 두배로 인상하고 싶다.

- 개발팀 id는 1003이다

EMPLOYEE						
<u>id</u>	name	birth_date	sex	position	salary	dept_id
1	100000000	1003
2	90000000	1004
3	50000000	1003
...
12	85000000	1005
13	90000000	1005

```
UPDATE employee SET salary = salary*2 WHERE dept_id = 1003;
```

- 프로젝트가 ID 2003에 참여한 임직원의 연봉을 두 배로 인상하고 싶다.

```
UPDATE employee, work_on SET salary = salary*2 WHERE id = empl_id and proj_id = 2003;
```

더 직관적으로 표시하고 싶다면

```
UPDATE employee, work_on SET salary = salary*2 WHERE employee.id = works_on.empl_id
and work_on.proj_id = 2003;
```

EMPLOYEE				WORKS_ON	
<u>id</u>	...	salary	...	empl_id	proj_id
1	...	100000000	...	1	2001
2	...	90000000
3	...	50000000	...	5	2002
...	2	2003
11	...	75000000
12	...	85000000	...	11	2003

회사의 모든 구성원의 연봉을 두배로 인상하고 싶다.

```
UPDATE employee SET salary = salary*2;
```

UPDATE statement

```
UPDATE table_name(s) // UPDATE 키워드 시작, 필요시 하나이상의 테이블명  
SET attribute = value[, attribute = value, .. ]  
[WHERE condition(s)]; // 필요시. 어떤 튜플의 대해서만 업데이트할건지. 업데이트하고 싶은  
// 튜플들이 만족해야하는 조건을 하나 이상 WHERE에 작성  
// WHERE가 없으면 모든 튜플들이 업데이트 된다.
```

데이터 삭제

DELETE statement

- john이 회사를 하게 되면서 employee 테이블에서 john 정보를 삭제해야한다.
- john의 employee ID는 8이다.
- 현재 john은 project 2001에 참여하고 있었다.

아래는 존과 관련 정보

EMPLOYEE				WORKS_ON	
<u>id</u>	name	...	dept_id	<u>empl_id</u>	<u>proj_id</u>
...
8	JOHN	...	1003	8	2001
...

```
DELETE FROM employee WHERE id = 8;
```

employee에서는 삭제가 될텐데 work_on에서도 삭제를 해주어야 할까?
아니다.

empl_id와 proj_id를 foreignkey로 설정해 두었기 때문에

FOREIGN KEY (`empl_id`) references EMPLOYEE(`id`)
on delete CASCADE on update CASCADE,

- jane이 휴직하게 되면서 현재 진행 중인 프로젝트에서 중도하차하게 됐다.
- jane의 id는 2다

```
DELETE FROM work_on WHERE impl_id = 2;
```

WORKS_ON	
<code>empl_id</code>	<code>proj_id</code>
...	...
2	2002
2	2003
...	...

튜플이 삭제됨

- 현재 딩요가 두 개의 프로젝트에 참여하고 있었는데 프로젝트2001 에 선택과 집중을 하기로 하고 프로젝트 2002에서는 빠지기로 했다.
- 딩요의 id는 5다

```
DELETE FROM works_on WHERE impl_id = 5 and proj_id = 2002;
```

프로젝트가 더 많다면

```
DELETE FROM works_on WHERE impl_id = 5 and proj_id <> 2001;  
<> 제외한다는 의미 , != 과 동일
```

- 회사에 큰 문제가 생겨서 진행중인 모든 프로젝트들이 중단됐다.

```
DELETE FROM project; // where x 모든 투플 삭제
```