Chapter 4. 데이터의 추가, 삭제, 갱신

16강. 행 추가하기 - INSERT

1) INSERT로 행 추가하기

INSERT 명령을 사용하여 테이블의 행 단위로 데이터를 추가

- 예제 4-2. sample41 테이블의 열 구성 확인하기

```
mysql> DESC sample41;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| no    | int         | NO   |     | NULL    |       |
| a     | varchar(30) | YES  |     | NULL    |       |
| b     | date        | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
```

- 예제 4-3. sample41 테이블에 행 추가하기

```
mysql> INSERT INTO sample41 VALUES(1, 'ABC', '2014-01-25');
Query OK, 1 row affected (0.074 sec)
```

  - INSERT 명령은 데이터가 클라이언트에서 서버로 전송→반환되는 결과X

- 예제 4-4. sample41 테이블

```
mysql> SELECT * FROM sample41;
+----+------+------------+
| no | a    | b          |
+----+------+------------+
|  1 | ABC  | 2014-01-25 |
+----+------+------------+
```

2) 값을 저장할 열 지정하기

```
-- INSERT의 열 지정
INSERT INTO 테이블명(열1, 열2,...) VALUES(값1, 값2,...)
```

- 열을 지정할 경우 테이블 뒤에 괄호로 묶어 열명 나열
- VALUES 구에 값을 지정 시, 지정한 열과 동일한 개수로 값 지정
- 예제 4-5. 열을 지정해 행 추가하기

```
mysql> INSERT INTO sample41(a, no) VALUES('XYZ', 2);
Query OK, 1 row affected (0.133 sec)

mysql> SELECT * FROM sample41;
+----+------+------------+
| no | a    | b          |
+----+------+------------+
|  1 | ABC  | 2014-01-25 |
|  2 | XYZ  | NULL       |
+----+------+------------+
```

3) NOT NULL 제약

- 행을 추가할 때 NULL값을 지정하여 추가 가능
- sample41의 모든 열의 값이 NULL인 행 추가 시...

  → 에러 발생!

  → no 열에 대해 **NOT NULL 제약**이 걸려있기 때문
- 제약: 테이블에 저장하는 데이터를 설정으로 제한하는 것
- 예제 4-6. NOT NULL 제약 회피하기

```
mysql> INSERT INTO sample41(no, a, b) VALUES(3, NULL, NULL);
Query OK, 1 row affected (0.035 sec)

mysql> SELECT * FROM sample41;
+----+------+------------+
| no | a    | b          |
+----+------+------------+
|  1 | ABC  | 2014-01-25 |
|  2 | XYZ  | NULL       |
|  3 | NULL | NULL       |
+----+------+------------+
```

NULL 간단 정리!

1. NULL 조건 비교 시, IS NULL 사용
2. NULL 포함 연산 시, 결과는 NULL
3. NULL 허용을 원치 않으면 NOT NULL 제약

4) DEFAULT

값을 지정하지 않았을 경우 사용하는 초깃값

- 테이블 정의할 때 지정 가능
- 열을 지정해 행을 추가할 때, 지정하지 않은 열은 Default 값을 사용해 저장됨
- 예제 4-7. sample411의 열 구성 확인하기

```
mysql> DESC sample411;
+-------+------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+-------+------+------+-----+---------+-------+
| no    | int  | NO   |     | NULL    |       |
| d     | int  | YES  |     | 0       |       |
+-------+------+------+-----+---------+-------+
```

- 예제 4-9. 명시적으로 디폴트 지정

```
mysql> INSERT INTO sample411(no,d) VALUES(2, DEFAULT);
Query OK, 1 row affected (0.155 sec)

mysql> SELECT * FROM sample411;
+----+------+
| no | d    |
+----+------+
|  1 |    1 |
|  2 |    0 |
+----+------+
```

암묵적으로 디폴트 저장

디폴트값을 지정할 열을 INSERT명령문에서 별도로 지정하지 않는 것

- 예제 4-10. 암묵적으로 디폴트 지정

```
mysql> INSERT INTO sample411(no) VALUES(3);
Query OK, 1 row affected (0.139 sec)

mysql> SELECT * FROM sample411;
+----+------+
| no | d    |
+----+------+
|  1 |    1 |
|  2 |    0 |
|  3 |    0 |
+----+------+
```

17강. 삭제하기 - DELETE

1) DELETE로 행 삭제하기

DELETE 명령으로 행 단위의 데이터를 삭제한다

```
-- DELETE 명령
DELETE FROM 테이블명 WHERE 조건식
```

- WHERE 구 생략 시, 모든 행을 대상으로 동작 → 모든 행 삭제
- WHERE 구 지정 시, 해당 조건식에 맞는 행만 삭제
- 예제 4-11. sample41 테이블

```
mysql> SELECT * FROM sample41;
+----+------+------------+
| no | a    | b          |
+----+------+------------+
|  1 | ABC  | 2014-01-25 |
|  2 | XYZ  | NULL       |
|  3 | NULL | NULL       |
+----+------+------------+
```

- 예제 4-12. sample41에서 행 삭제하기

```
mysql> DELETE FROM sample41 WHERE no=3;
Query OK, 1 row affected (0.170 sec)

mysql> SELECT * FROM sample41;
+----+------+------------+
| no | a    | b          |
+----+------+------------+
|  1 | ABC  | 2014-01-25 |
|  2 | XYZ  | NULL       |
+----+------+------------+
```

2) DELECTE 명령 구

- WHERE 구 역할
  - SELECT, DELETE 모두 조건에 맞는 행을 선택
  - SELECT → 조건에 맞는 결과 반환
  - DELETE → 조건에 맞는 행 삭제
- 조건 변경 가능
  - 예:
    - `WHERE no = 3` → no가 3인 행 삭제
    - `WHERE no = 1 OR no = 2` → no가 1 또는 2인 행 삭제
- ORDER BY 구 사용 불가

  → 삭제 순서는 중요하지 않으므로 의미 없음

18강. 데이터 갱신하기 - UPDATE

1) UPDATE로 데이터 갱신하기

UPDATE 명령으로 데이터를 갱신한다(테이블의 셀 값을 갱신한다)

```
-- UPDATE 명령
UPDATE 테이블명 SET 열1=값1, 열2=값2,... WHERE 조건식
```

- DELETE와 다르게 셀 단위로 데이터 갱신 가능

- WHERE 구에 조건 지정 시, 그에 일치하는 행 갱신

  WHERE 구 생략 시, 테이블의 모든 행이 갱신
- SET 구를 사용하여 갱신할 열과 값을 지정
  - 이때 = 은 비교 연산자X, 대입 연산자
- 값은 상수로 표기, 자료형에 맞게 지정
- 예제 4-14. sample41의 셀 값을 갱신하기

```
mysql> UPDATE sample41 SET b='2014-09-07' WHERE no=2;
Query OK, 1 row affected (0.166 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM sample41;
+----+------+------------+
| no | a    | b          |
+----+------+------------+
|  1 | ABC  | 2014-01-25 |
|  2 | XYZ  | 2014-09-07 |
+----+------+------------+
```

- UPDATE명령에서는 WHERE조건에 일치하는 모든 행이 갱신

2) UPDATE로 갱신할 경우 주의사항

- 예제 4-15. UPDATE명령으로 증가 연산하기

```
mysql> UPDATE sample41 SET no=no+1;
Query OK, 2 rows affected (0.157 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql> SELECT * FROM sample41;
+----+------+------------+
| no | a    | b          |
+----+------+------------+
|  2 | ABC  | 2014-01-25 |
|  3 | XYZ  | 2014-09-07 |
+----+------+------------+
```

- 갱신할 값을 열이 포함된 식으로도 표기 가능

3) 복수열 갱신

SET구에서는 필요에 따라 콤마(,)로 구분하여 갱신할 열을 여러 개 지정 가능

- SET 구의 실행 순서는 데이터베이스 제품에 따라 처리 방식이 다름
- **MySQL**
  - 예제 4-17. MYSQL에서 UPDATE 명령1 실행

```
mysql> UPDATE sample41 SET no=no+1, a=no;
Query OK, 2 rows affected (0.169 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql> SELECT * FROM sample41;
+----+------+------------+
| no | a    | b          |
+----+------+------------+
|  3 | 3    | 2014-01-25 |
|  4 | 4    | 2014-09-07 |
+----+------+------------+
```

  - 예제 4-18. MySQL에서 UPDATE 명령2 실행

```
mysql> UPDATE sample41 SET a=no, no=no+1;
Query OK, 2 rows affected (0.098 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql> SELECT * FROM sample41;
+----+------+------------+
| no | a    | b          |
+----+------+------------+
|  4 | 3    | 2014-01-25 |
|  5 | 4    | 2014-09-07 |
+----+------+------------+
```

  - MySQL에서는 SET구에 기술된 순서로 갱신 처리

    ⇒ 갱신식 안에 열을 참조할 때 처리 순서 고려 필요

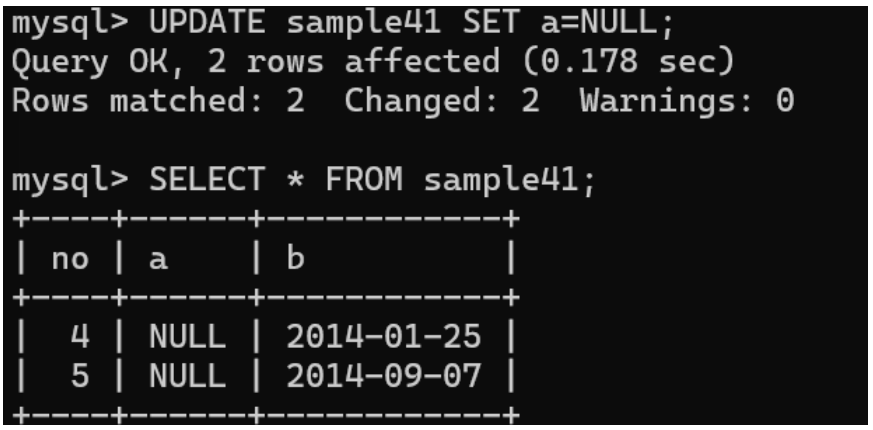
- **Oracle**
  - SET 구에 기술한 식의 순서가 처리에 영향을 주지 않음

    ⇒ 처리 순서 바꿔도 결과가 동일

4) NULL로 갱신하기

UPDATE 명령으로 셀 값을 NULL로 갱신시킬 수 있다

- 예제 4-20. NULL 초기화



- 그러나 NOT NULL 제약이 설정되어 있는 열은 NULL 허용X

19강. 물리삭제와 논리삭제

1) 두 종류의 삭제방법

데이터베이스에서 데이터를 삭제할 때 물리삭제와 논리삭제의 방법이 있다.

(단, SQL명령이 두 가지 존재한다는 의미X, 두 가지 사고 방식이 있다로 이해)

물리삭제

SQL의 DELETE 명령을 사용해 직접 데이터를 삭제하자는 사고 방식

논리삭제

테이블에 '삭제플래그'와 같은 열을 미리 준비하여, 테이블에서 실제로 행을 삭제하는 대신, UPDATE명령을 이용해 '삭제플래그'의 값을 유효하게 갱신하자는 사고 방식

- 실제 테이블 안에 데이터가 있지만, 참조할 때에는 '삭제플래그'가 삭제로 설정된 행을 제외하는 SELECT 명령 실행 ⇒ 결과적으로는 해당 행이 삭제된 것처럼 보임

- 삭제플래그를 이용하는 방법 외에도 여러가지 방법
- 장점: 삭제되기 전 상태로 되돌리기 쉬움
- 단점: 삭제해도 데이터베이스 저장공간이 늘지 않음, 검색속도가 떨어짐, 삭제를 하는 건데 UPDATE 명령을 실행하여 혼란 야기

2) 삭제방법 선택하기

상황에 맞게 선택한다

- 예1) SNS에서 사용자 탈퇴 → 물리 삭제

  : 개인정보 유출 방지를 하는 측면에서 좋은 선택
- 예2) 쇼핑 사이트에서 사용자가 주문 삭제 → 논리 삭제

  : 주문 통계를 낼 때 유용
- 예3) 하드웨어의 제한 → 물리 삭제

  : 논리삭제로는 데이터베이스 사용량이 줄지 않음, 오히려 늘어남