8253jang@daum.net
장희정

MySQL의 데이터 유형

1) 숫자형

-정수(일반적으로 int, bigint를 주로 사용한다)

데이터 유형	데이터 크기(byte)	숫자 범위(SIGNED)	숫자 범위(UNSIGNED)
TINYINT	1	-128 ~ 127	0 ~ 255
SMALLINT	2	-32,768 ~ 32,767	0 ~ 65535
MEDINUMINT	3	-8388608 ~ 8388607	0 ~ 16777215
INT	4	-2^31(약 -21억) ~ 2^31-1(약 21억)	0 ~ 2^32-1(약 42억)
BIGINT	8	-2^63 ~ 2^63-1	0 ~ 2^64-1

-실수

부동 소수점 숫자를 저장하기 위해 FLOAT와 DOUBLE을 사용할 수 있음

데이터 유형	데이터 크기(byte)	숫자 범위
FLOAT(n)	4 ~ 8	-3.40E+30 ~ 1.17E-38
DOUBLE	8	-1.22E-308 ~ 1.79E+308

고정 소수점 숫자를 저장하기 위해서는 DECIMAL와 NUMBER을 사용할 수 있음

데이터 유형	데이터 크기(byte)	숫자 범위
DECIMAL(p,s)	5 ~ 17	-10^38+1 ~ 10^38+1
NUMERIC(p,s)	5 ~ 17	-10^38+1 ~ 10^38+1

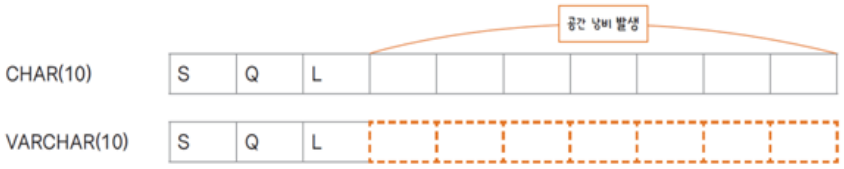
Ex) DECIMAL(20, 5)라고 하면 정수 15자리, 소수 5자리를 표현함

2) 문자형

-고정길이(실제 값을 입력하지 않아도 지정한 만큼 저장 공간을 사용)

-가변길이(실제 입력한 값의 크기만큼만 저장 공간을 사용)

8253jang@daum.net
장희정



데이터 유형		데이터 크기(byte)	설명
CHAR(n)		1 ~ 255	고정 길이 문자열로, 0 ~ 255 크기의 문자열까지 저장 가능 기본값
VARCHAR(n)		1 ~ 65535	가변 길이 문자열로, n만큼의 크기를 지정. 0 ~ 16383 크기의 문자열까지 저장 가능(한 문자당 최대 4바이트 사용)
BINARY(n)		1 ~ 255	고정 길이 문자열로, 0 ~ 255 크기의 문자열까지 저장 가능
VARBINARY(n)		1 ~ 65535	가변 길이 문자열로, 0 ~ 16383 크기의 문자열까지 저장 가능
TEXT 형식	TINYTEXT	1 ~ 255	최대 길이가 255자인 문자열을 저장
	TEXT(n)	1 ~ 65535	최대 길이가 65,535바이트인 문자열 데이터 저장
	MEDIUMTEXT	1 ~ 16777215	최대 길이가 16,777,215바이트인 문자열 데이터 저장
	LONGTEXT	1 ~ 4294967295	최대 길이가 4,294,967,295바이트인 문자열 데이터 저장
BLOB 형식	TINYBLOB	1 ~ 255	최대 길이가 255바이트인 문자열 데이터 저장
	BLOB(n)	0 ~ 65535	최대 길이가 65,535바이트인 데이터 저장
	MEDIUMBLOB	1 ~ 16777215	최대 길이가 16,777,215바이트인 문자열 데이터 저장
	LONGBLOB	1 ~ 4294967295	최대 길이가 4,294,967,295바이트인 문자열 데이터 저장
ENUM		1 ~ 2	가능한 값 목록에서 선택한 하나의 값만 가질 수 있는 문자열 개체로, ENUM 목록에 최대 65535개의 값을 나열할 수 있다.
SET		1, 2, 3, 4, 8	가능한 값 목록에서 선택한 0개 이상의 값을 가질 수 있는 문자열 개체로, 최대 64개의 값을 나열할 수 있다.

- MySQL의 경우 문자열 데이터 유형을 사용할 때,
  CHAR 또는 VARCHAR 뒤에 인자(괄호 안의 숫자)로 사용하는
  숫잣값의 의미는 다른 DBMS과 달라서 정확히 알고 사용해야 함
- MySQL에서는 CHAR 또는 VARCAHR 뒤에 나오는 숫자는
  바이트가 아닌 문자열의 문자 수임
- 즉, CHAR(5) 또는 VARCHAR(5)이면 5byte를 저장할 수 있는 것이 아니라
  문자 5개를 저장할 수 있다는 뜻임
- 그래서 CHAR(5)을 사용하더라도
  실제 저장 공간은 입력되는 데이터에 따라 크기가 다른 공간을 사용함

8253jang@daum.net
장희정

3) 날짜형

데이터 유형	데이터 크기(byte)	설명
TIME	3	HH:MM:SS(시:분:초) 형태의 데이터에 사용하고, 범위는 -838:59:59 ~838:59:59이다.
DATE	3	날짜만 있는 데이터에 사용하고, YYYY-MM-DD(연-월-일) 형태이다. 범위는 1000-01-01 ~ 9999-12-31이다.
DATETIME	8	날짜와 시간을 모두 포함하는 데이터에 사용하고, YYYY-MM-DD hh:mm:ss [.fraction] 형식으로, 범위는 1000-01-01 00:00:00.000000 ~ 9999-12-31 23:59:59.000000이다.
TIMESTAMP	4	날짜와 시간 부분을 모두 포함하는 데이터에 사용하고, 범위는 1970-01-01 00:00:01 UTC ~ 2038-01-19 03:14:07 UTC이다.

4) ENUM & SET 데이터 타입

데이터 타입	설명
`ENUM('value1', 'value2', ...)`	선택 가능한 값 중 하나 (예: `ENUM('남', '여')`)
`SET('value1', 'value2', ...)`	여러 개 선택 가능 (예: `SET('축구', '농구', '야구')`)

예시) ENUM은 **하나의 값만 선택** 가능,

```
CREATE TABLE employees (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(50) NOT NULL,
    gender ENUM('남', '여') NOT NULL
);

INSERT INTO employees (name, gender) VALUES ('홍길동', '남');
INSERT INTO employees (name, gender) VALUES ('김영희', '여');
INSERT INTO employees (name, gender) VALUES ('이철수', '남');

INSERT INTO employees (name, gender) VALUES ('박지수', '기타'); -- 에러
```

예시) SET은 **여러 개 선택** 가능

8253jang@daum.net
장희정

```
CREATE TABLE hobbies (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(50) NOT NULL,
    interests SET('축구', '농구', '야구', '독서', '게임')
);

INSERT INTO hobbies (name, interests) VALUES ('홍길동', '축구');
INSERT INTO hobbies (name, interests) VALUES ('김영희', '농구,야구');
INSERT INTO hobbies (name, interests) VALUES ('이철수', '독서,게임,축구');

INSERT INTO hobbies (name, interests) VALUES ('박지수', '테니스'); --에러
```

MySQL 의 SET 타입은 데이터베이스 정규화(Normalization) 원칙에 위배될 수 있다..
왜 SET 이 정규화 원칙에 위배될까?

- **첫 번째 정규화(1NF)**:
  - 1NF 에서는 **각 컬럼이 단일 값(Atomic Value)을 가져야 한다**는 원칙이 있다..
  - 하지만 SET 타입은 **한 개의 컬럼에 여러 개의 값을 저장**하기 때문에 **1NF 를 위반**된다.
- **데이터 중복과 무결성 문제**:
  - SET 컬럼에 들어가는 값이 콤마(,)로 구분되어 있기 때문에 검색이 어렵다.
  - SET 값이 변경되면 **데이터를 정리하는 것이 복잡**해진다.
  - 개별 값을 수정할 때 **비효율적**이다.

자주 사용하는 MySQL 데이터 타입

- **숫자**: INT, BIGINT, DECIMAL, FLOAT
- **문자열**: VARCHAR(N), TEXT
- **날짜/시간**: DATE, DATETIME, TIMESTAMP
- **특수 데이터 타입**: ENUM, SET

MySQL 테이블 생성 방법

MySQL 에서 CREATE TABLE 문을 사용하여 테이블을 생성한다.
테이블 이름: 테이블의 고유한 이름을 설정
열(Column) 정의: 각 열의 데이터 타입과 제약 조건을 설정

8253jang@daum.net
장희정

제약 조건(Constraints): 기본 키(Primary Key), 외래 키(Foreign Key), NOT NULL, UNIQUE 등의 제약 조건을 지정할 수 있다.

테이블 생성방법

```
create table 테이블이름(
 컬럼명 datatype [ null | not null ] [제약조건 ] ,
 컬럼명 datatype [ null | not null ] [제약조건 ]  ,
 ..
)
```

예시)

```
CREATE TABLE employees (
    emp_id INT PRIMARY KEY,              -- 사원 ID (기본 키)
    name VARCHAR(50) NOT NULL,           -- 이름 (NULL 허용 안 됨)
    age INT CHECK (age >= 18),           -- 나이 (18세 이상만 허용)
    department VARCHAR(30),              -- 부서
    salary DECIMAL(10,2),                -- 급여 (소수점 2자리까지 저장)
    hire_date DATE                       -- 입사일
);
```

- **emp_id INT PRIMARY KEY** : emp_id 를 기본 키로 설정. 중복 불가능, NULL 값 허용 안 됨.
- **name VARCHAR(50) NOT NULL** : 이름은 50 자까지 가능하며, NULL 값을 허용하지 않음.
- **age INT CHECK (age >= 18)** : 나이는 18 세 이상이어야 함 (MySQL 8.0 이상에서 CHECK 지원).
- **department VARCHAR(30)** : 부서명은 30 자까지 저장 가능.
- **salary DECIMAL(10,2)** : 급여는 최대 10 자리(소수점 2 자리 포함)까지 저장 가능.
- **hire_date DATE** : 입사 날짜를 DATE 형식으로 저장.

MySQL 제약 조건 특징

8253jang@daum.net
장희정

제약 조건	설명
`PRIMARY KEY`	중복 & NULL 불가, 테이블 내 유일한 값
`FOREIGN KEY`	다른 테이블의 `PRIMARY KEY`를 참조
`NOT NULL`	NULL 값 입력 불가
`UNIQUE`	중복 불가 (단, NULL 허용)
`DEFAULT`	값이 없을 때 기본값 설정
`CHECK`	특정 조건을 만족해야 입력 가능 (MySQL 8.0 이상)
`AUTO_INCREMENT`	자동 증가 (숫자 타입에서만 사용 가능)

PRIMARY KEY (기본 키)

- 한 테이블에서 각 행을 **유일하게 식별**하는 키.
- **NULL 값을 허용하지 않으며 중복될 수 없음**.
- PRIMARY KEY 가 설정된 열은 자동으로 **UNIQUE** 와 **NOT NULL** 속성을 가짐.

```
CREATE TABLE users (
    id INT PRIMARY KEY,   -- 기본 키 설정
    name VARCHAR(50) NOT NULL
);
```

FOREIGN KEY (외래 키)

- 다른 테이블의 PRIMARY KEY 를 참조하여 **테이블 간 관계**를 설정.
- 외래 키가 참조하는 기본 키 값이 존재해야 삽입 가능.
- **ON DELETE CASCADE** 등을 설정하여 연관된 데이터가 함께 삭제될 수도 있음.
- **ON DELTE SET NULL** 등을 설정하면 부모레코드 삭제 될 때 참조되는 자식 레코드값 NULL 로 변경

```
CREATE TABLE orders (
    order_id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);
```

8253jang@daum.net
장희정

ON DELETE CASCADE: users 테이블에서 사용자가 삭제되면 orders 테이블에서도 해당 사용자의 주문이 자동 삭제됨.

UNIQUE (유니크)

- 중복 값을 허용하지 않음.
- 단, NULL 값은 허용됨 (PRIMARY KEY 와 차이점).

```
CREATE TABLE customers (
    id INT PRIMARY KEY AUTO_INCREMENT,
    email VARCHAR(100) UNIQUE  -- 이메일은 중복 불가
);
```

email 컬럼에 중복된 값이 들어가면 오류 발생.

DEFAULT (기본값)

- 컬럼에 값이 입력되지 않았을 때 자동으로 설정할 **기본값**.

```
CREATE TABLE products (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(50) NOT NULL,
    stock INT DEFAULT 10  -- 기본 재고는 10
);
```

stock 값을 입력하지 않으면 자동으로 10 이 설정됨.

CHECK (제약 조건)

- 특정 컬럼의 값이 **지정된 조건을 만족해야 함**.
- MySQL 8.0 이상에서 지원됨.

```
CREATE TABLE employees (
    id INT PRIMARY KEY AUTO_INCREMENT,
    age INT CHECK (age >= 18)  -- 나이는 18 세 이상이어야 함
);
```

8253jang@daum.net
장희정

age 값이 18 미만이면 입력 불가.

AUTO_INCREMENT (자동 증가)

- INT 타입에서 사용 가능하며 **새로운 레코드가 추가될 때 자동 증가**.
- PRIMARY KEY 또는 UNIQUE 가 필요함.

```
CREATE TABLE users (
    id INT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(50) NOT NULL
);
```

id 값이 자동으로 1 부터 증가.

AUTO_INCREMENT 는 하나의 테이블에서 오직 하나의 컬럼에만 사용할 수 있다..

이유

- AUTO_INCREMENT 는 **한 테이블에서 단 하나의 컬럼**에만 적용된다.
- 보통 PRIMARY KEY 또는 UNIQUE 컬럼에서 사용된다.
- 자동 증가하는 값은 테이블 내에서 **하나의 시퀀스**로 관리되므로, 여러 개의 AUTO_INCREMENT 가 있으면 충돌이 발생할 수 있기 때문에 하나의 테이블에 한 개만 가능하다.

알아두기

MySQL 에서는 AUTO_INCREMENT 가 설정된 컬럼에도 **직접 값을 넣을 수 있다.**
MySQL 의 AUTO_INCREMENT 는 기본적으로 자동 증가 값을 제공할 뿐, 직접 입력을 막지는 않는다.

직접 삽입한 값이 AUTO_INCREMENT 보다 크면, 다음 증가 값이 자동 조정된다.

예를 들어, 현재 AUTO_INCREMENT 값이 1 인데 직접 100 을 삽입하면, 이후의 AUTO_INCREMENT 값은 101 이 된다.

즉, 직접 삽입된 값이 AUTO_INCREMENT 값보다 크면, 자동 증가 값이 이에 맞춰 조정된다.

8253jang@daum.net
장희정

- AUTO_INCREMENT 값을 강제로 변경

```
ALTER TABLE 테이블이름 AUTO_INCREMENT = 변경값;
```

복합 키(Composite Key)와 AUTO_INCREMENT 조합

테이블에서 **하나의 AUTO_INCREMENT 만 허용되므로**, 복합 키(PRIMARY KEY + AUTO_INCREMENT)를 활용할 수 있다.

```
CREATE TABLE orders (
    order_id INT AUTO_INCREMENT,   -- 자동 증가 키
    product_id INT,                -- 다른 키
    PRIMARY KEY (order_id, product_id)   -- 복합 키 설정
);
```

order_id 는 자동 증가하지만, product_id 는 사용자가 직접 입력해야 한다.

두 개의 자동 증가 값이 필요한 경우

AUTO_INCREMENT 를 하나만 유지하면서, **별도의 시퀀스 테이블을 만들어 관리**하는 방법도 있다.

```
CREATE TABLE sequence_table (
    seq_id INT AUTO_INCREMENT PRIMARY KEY
);

CREATE TABLE products (
    product_id INT PRIMARY KEY,
    batch_number INT NOT NULL
);

INSERT INTO sequence_table () VALUES ();   -- 새로운 시퀀스 값 생성
SET @new_id = LAST_INSERT_ID();            -- 방금 생성된 ID 가져오기

INSERT INTO products (product_id, batch_number) VALUES (@new_id, 101);
```

8253jang@daum.net
장희정

```
select * from sequence_table;
select * from products;
```

@를 붙이는 이유
세션 변수(Session Variable)임을 나타낸다.
@변수명 형식의 변수는 세션 단위로 유지되는 사용자 변수(User Variable) 이다.
세션이 유지되는 동안 같은 값을 여러 번 사용할 수 있다.
SQL 내부에서 사용 가능하다.
@변수는 어떤 SQL 문장에서도 바로 사용할 수 있다.
연속적인 쿼리에서 값 유지 할 때 사용한다.

일반적인 DECLARE 로 선언한 변수는 스토어드 프로시저나 함수 안에서만 사용된다.

sequence_table 에서 seq_id 를 가져와 product_id 로 설정하면, 하나의 AUTO_INCREMENT 만 사용하면서도 여러 개의 증가하는 값을 관리할 수 있다.

테이블 수정

① 컬럼추가
alter table 테이블이름 add (컬럼명 자료형 [제약조건] , 컬럼명 자료형 [제약조건] , ...);

② 컬럼삭제
alter table 테이블이름 drop column 컬럼이름;

③ datatype 변경
alter table 테이블이름 modify 컬럼이름 변경자료형 [not null | null];

④ 컬럼이름 변경
alter table 테이블이름 rename column 기존컬럼명 to 변경컬럼명;

⑤ 제약조건 추가
alter table 테이블이름 ADD CONSTRAINT 별칭 제약조건종류;

-제약조건 삭제
ALTER TABLE 테이블이름 DROP 제약조건;

8253jang@daum.net
장희정

MySQL 에서 제약 조건을 추가할 때는 ADD CONSTRAINT, **삭제할 때는** DROP PRIMARY KEY, DROP FOREIGN KEY, DROP INDEX **등 적절한 명령어를 사용해야 한다.**

제약 조건	추가 명령어	삭제 명령어
PRIMARY KEY	`ALTER TABLE test ADD CONSTRAINT pk_test PRIMARY KEY(id);`	`ALTER TABLE test DROP PRIMARY KEY;`
FOREIGN KEY	`ALTER TABLE orders ADD CONSTRAINT fk_orders_customer FOREIGN KEY (customer_id) REFERENCES customers(id);`	`ALTER TABLE orders DROP FOREIGN KEY fk_orders_customer;`
UNIQUE	`ALTER TABLE users ADD CONSTRAINT unique_email UNIQUE(email);`	`ALTER TABLE users DROP INDEX unique_email;`
CHECK (MySQL 8.0 이상)	`ALTER TABLE employees ADD CONSTRAINT chk_salary CHECK (salary > 0);`	`ALTER TABLE employees DROP CONSTRAINT chk_salary;`
NOT NULL	`ALTER TABLE users MODIFY name VARCHAR(50) NOT NULL;`	`ALTER TABLE users MODIFY name VARCHAR(50) NULL;`

- 테이블 삭제

drop table 테이블이름;

SQL 의 종류

- DDL 문장(CREATE, DROP, ALTER, TRUNCATE)

- DML 문장(INSERT ,UPDATE, DELETE)

데이터 조작 : DML(INSERT , UPDATE, DELETE)

- ROLLBACK OR COMMIT 가능

1) INSERT 문장

- INSERT INTO 테이블이름(컬럼명, 컬럼명,....) VALUES(값, 값,값,....);
- INSERT INTO 테이블이름 VALUES(값, 값,값,....); -- 모든 컬럼에 순서대로 값을 넣을때!!!
- INSERT INTO 테이블이름(컬럼명, 컬럼명,....)

8253jang@daum.net
장희정

VALUES(값, 값,값,....) ,(값, 값,값,....) , (값, 값,값,....) ..; -- 여러데이터를 한번에 삽입

2) DELETE 문장

DELETE [FROM] 테이블이름
[WHERE 조건식]

3) UPDATE 문장

UPDATE 테이블이름
SET 컬럼명=변경값 , 컬럼명=변경값,....
[WHERE 조건식]

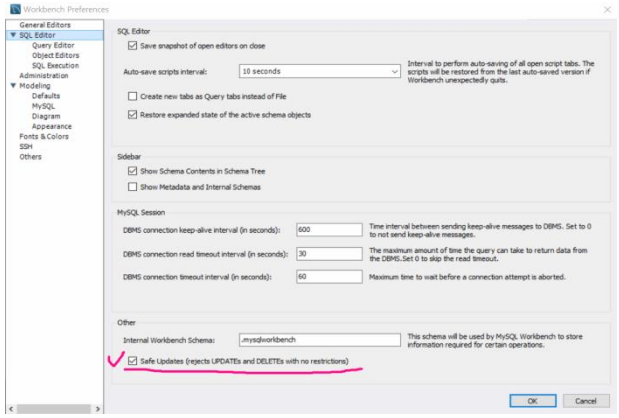
알아두기

where 절 없이 delete or update 를 MySQL Workbench 등에서 실행할 때,
safe update mode(안전 업데이트 모드)가 활성화되어 있으면 기본 키(PK) 또는 인덱스(KEY)가 없는 DELETE 나 UPDATE 문을 허용하지 않음. 에러 발생한다.

즉, WHERE 절 없이 DELETE 를 실행하면 전체 데이터가 삭제되므로, 실수로 데이터를 삭제하는 것을 방지하기 위해 제한이 걸려 있는 것이다.

SET SQL_SAFE_UPDATES = 0; -- 일시적으로 해제
SET SQL_SAFE_UPDATES = 1; -- 다시 활성화

MySQL Workbench 에서 영구적으로 Safe Mode 를 끄려면
MySQL Workbench 실행
Edit → Preferences → SQL Editor 선택
"Safe Updates" 체크 해제
MySQL Workbench 를 다시 연결(Reconnect)



기본적으로 **MySQL 은 AUTO COMMIT 모드가 활성화 되어 있다.**

8253jang@daum.net
장희정

```
SET AUTOCOMMIT = 0;   -- 자동 커밋 비활성화

START TRANSACTION;      -- 트랜잭션 시작
DELETE FROM userlist WHERE id = 1;   -- 레코드 삭제
ROLLBACK;   -- 변경 사항 되돌리기
```

테이블 복사하기

```
CREATE TABLE 테이블이름
  AS 복사할테이블정보;
```

주의 : 테이블을 복사하면 제약조건은 복사 안된다!!! - 복사한후에 제약조건을 ALTER 를 이용해서 추가한다.