MySQL 사용

- **루트 패스워드 지정**

```
[root@Client~ ~]# mysqladmin -u root password 1
```

  - mysqladmin -u root password '새로운 패스워드'

MySQL의 기본 설정을 그대로 두면 누구나 사용자 이름과 패스워드를 입력하지 않고 데이터베이스에 접근할 수 있다
이 상태는 아주 위험하므로 보안을 위해 익명 사용자를 반드지 삭제 해야한다

```
[root@Client~ ~]# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 5.0.77 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use mysql;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> delete from user where User='';
Query OK, 2 rows affected (0.00 sec)

mysql> quit
Bye
[root@Client~ ~]# mysqladmin -u root -p reload
Enter password:
[root@Client~ ~]#
```

- **MySQL 사용하기**
  - MySQL 명령어들은 ";" 으로 끝난다, ";" 은 MySQL에게 명령을 실행시키라고 명령한다는 뜻이다
    따라서 ";" 을 빠뜨리면 아무 것도 실행되지 않는다
    ";"을 생략하면 명령어 중간에 다음 줄로 넘어 간다, 긴 명령어를 입력할 때는 일부러 줄을 넘겨가면서 입력하는 것이 더 효율적일 수 있다

```
mysql> show databases
    -> 
    -> ;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| test               |
+--------------------+
3 rows in set (0.00 sec)
```

    명령어 끝에 ";"을 입력하지 않으면 다음 줄로 넘어가게 되고 ";" 을 입력하면 그때서야 명령어가 실행된다

    MySQL 명령어는 대소문자를 구별하지 않지만 데이터베이스와 테이블 이름에는 대소문자가 구별된다

- **MySQL 접속하기**
  - mysql -h *hostname* -u *username* -p
    -h : MySQL 서버가 실행되고 있어 접속해야 하는 주소, 만약 로컬에 MySQL 서버가 사용중이라면 생략가능하다
    -u : 접속해야하는 사용자 이름을 적으면된다 "-u" 옵션이 없으면, 기본적으로는 운영체제에 접속한 사용자 입력이 대신 들어가게 된다
    -p : 패스워드를 입력한다, 비밀번호를 설정하지 않았다면 생략하고 접속하면 된다, 옵션 뒤에 반드시 비밀번호를 입력할 필요는 없다

MySQL 사용자권한

- **MySQL의 권한 시스템**
  - MySQL의 뛰어난 장점 중 하나는 정교한 권한 시스템이다, **권한**이란 특정 사용자가 특정한 곳에 접근하고 실행 시킬 수 있는 권리 , 한도를 뜻한다
    파일에서의 퍼미션(permission)과 같은 개념이라 할 수 있다
    MySQL에서 사용자를 처음 설정할 때 이 사용자가 이 시스템에서 어떤 일들은 할 수 없는지, 할 수 있는 일은 어디까지인지 등의 권한을 설정해주어야 한다

  - 최소 권한의 원칙
    "사용자(혹은 프로세스)는 할당된 일을 수행하기 위해 가장 최소의 권한만을 가져야 한다"
    웹에서 받은 쿼리를 처리할 때에도, 사용자가 root처럼 모든 권한을 다 가져야 할 필요는 없으므로,
    데이터베이스에 접근할 필요한 만큼의 권한만을 가진 다른 사용자를 새로 만드는 것이 좋다

  - **GRANT** - 사용자를 새로 만들고, 권한을 주는 명령어
    : 명령어 형식
    GRANT *privileges* [columns]
    ON *item*
    TO *user_name* [IDENTIFIED BY 'password']
    [REQUIRE ssl_options]
    [WITH [GRANT OPTION | limit_options]]
    -->> [ ] 에 있는 것은 옵션

    - 'privileages' - 허용해줄 권한들이 들어간다(권한이 여러 개 일 때에는 ' , ' 로 구분해서 입력한다)
    - [columns] - 이 권한을 열 단위로 지정해줄 수 있게 한다, 권한이 적용되는 열 이름들을 이 안에 써주면되고,
      여러 개 일 때에는 ' , '로 구분해준다

    <사용자에 대한 권한>

권한	적용 대상	설명
SELECT	테이블 , 열	사용자가 테이블에서 어떤 행을 선택할 수 있게 해준다
INSERT	테이블 , 열	사용자가 테이블에 새로운 행을 삽입할 수 있게 한다
UPDATE	테이블 , 열	사용자가 이미 존재하고 있는 행을 변경할 수 있게 한다
DELETE	테이블	사용자가 테이블의 행을 지울 수 있게 한다
INDEX	테이블	사용자가 특정 테이블에 인덱스를 만들거나 없앨 수 있게 한다
ALTER	테이블	사용자가 테이블의 구조를 바꿀 수 있게 한다 (열을 하나 추가하거나, 열 또는 테이블의 이름을 바꾼다거나, 열의 데이터 형을 변경할 때 쓰인다)
CREATE	데이터베이스 , 테이블	사용자가 새로운 데이터베이스나 테이블을 만들 수 있게 한다 GRANT 명령에 특정 DB나 테이블이 명시되어 있다면, 이 사용자는 명시된 DB 또는 테이블만 만들 수 있다 (이때 DB , 테이블이 존재하고 있다면 이 CREATE를 실행하기 전에 DROP부터 해야 함을 의미한다)
DROP	데이터베이스 , 테이블	사용자가 데이터베이스나 테이블을 지울 수 있게 한다

    - 'item' - 이 권한들이 적용되는 데이터베이스나 테이블의 이름을 넣어주면 된다
      모든 DB에 적용되게 하려면 ' *.* ' 라고 입력한다 , 이를 **전역적**인 권한이라고 한다, 이를 ' * ' 라고 쓸 수도 있다
      어떤 DB에 있는 모든 테이블들의 권한을 설정할 때에는 *dbname.** 라고 표현
      특정 DB의 특정 테이블은 *dbname.tablename* 라고 표현 , 또한 columns도 적어줄 수 있다
      이 명령을 사용할 때에 어떤 특정한 DB안에 들어가 사용하고 있는 중이었다면, *tablename*만 적어서
      현재의 DB의 그 테이블에 대한 권한을 설정할 수도 있다

    - 'user_name' - MySQL에 접속할 사용자의 이름을 넣는다
      이는 시스템에 로그인하는 이름과는 다르다, 여기서의 사용자 이름은 호스트이름까지 포함해야한다

예를 들어, Kim이라고 하거나(이는 Kim@localhost 와 같다) Kim@somewhere.com 이라고 해야 한다
사용자가 같은 이름이지만 도메인이 다른 곳에서 들어올 수 있기 때문이다
접속하는 곳까지 명시할 수 있고, 특정 위치에서 접속할 수 있는 테이블과 DB를 지정할 수 있기 때문에 보안에 좋다

- 'password' - 사용자의 비밀번호가 들어간다

- 'REQUIRE' - 사용자가 SSL(Secure Sockets Layer)을 통해서만 접속할 수 있다고 정하는 데 사용한다

- 'WITH GRANT OPTION' - 어떤 사용자가 다른 사용자에 대한 권한을 설정할 수 있게 해준다
  Ex) WITH 절은 다음과 같이 사용할 수 있다

  MAX_QURIES_PER_HOUR *n*
  한 사용자의 n시간 동안 쿼리 수를 제한
  MAX_UPDATE_PER_HOUR *n*
  한 사용자의 n시간 동안 업데이트를 제한
  MAX_CONNECTIONS_PER_HOUR *n*
  한 사용자의 n시간 동안 접속 수를 제한

  --> 한 사용자가 시스템에 주는 부하를 제한하는 데 유용하게 사용된다

  <관리자를 위한 권한>

권한	설명
CREATE TEMPORARY TABLES	관리자가 CREATE TABLES 문에 TEMPORARY 키워드를 사용할 수 있게 한다
FILE	파일의 데이터가 테이블로 읽혀지는 것을 가능하게 한다(반대도 가능)
LOCK TABLES	LOCK TABLES 문을 직접 사용할 수 있게 한다
PROCESS	관리자가 사용자에게 속한 서버에서 실행되는 프로세스들에 대해서 볼 수 있게 한다
RELOAD	관리자가 grant 테이블을 다시 불러들여 권한 , 호스트 , 로그 , 테이블 등을 flush할 수 있게한다
REPLICATION CLIENT	복사 마스터와 슬레이브에 대해 SHOW STATUS를 할 수 있다
REPLICATION SLABE	복사 슬레이브 서버가 마스터 서버에 접속할 수 있게 한다
SHOW DATABASES	SHOW DATABASES를 통해 모든 데이터베이스를 볼 수 있다
SHUTDOWN	관리자가 MySQL 서버를 정지시킬 수 있게 한다
SUPER	관리자가 사용자에게 속한 쓰레드를 중지 시킬 수 있다

  <특별한 권한>

권한	설명
ALL	사용자와 관리자에 대한 모든 권한을 다 준다, ALL PRIVILEGES 라고 해도 된다
USAGE	어떤 권한도 주지 않는다 , 접속은 할 수 있지만 아무 일도 할 수 없다, 나중에 권한을 더 줄 수 있다

- 권한은 mysql이라는 데이터베이스 안의 4개의 시스템 테이블(mysql.user , mysql.db , mysql.tables_priv , mysql.columns_priv)
  안에 저장 된다 권한을 바꾸기 위해 이 테이블들을 직접 편집할 수도 있다

○ **REVOKE** - 사용자의 권한을 뺏는다

: 명령어 형식
REVOKE *privileges* [(columns)]
ON *item*
FROM *user_name*

WITH GRANT OPTION 구문으로 권한을 부여할 수 있는 권한을 주었다면, 이를 뺏을 때는 아래와 같다
- REVOKE All PRIVILEGES, GRANT
  FROM user_name

○ GRANT 와 REVOKE 를 사용한 예제 :-)

```
mysql> grant usage
    -> on books.*
    -> to kim identified by 'aa111';
Query OK, 0 rows affected (0.01 sec)
```

- 기본적으로 Books 라는 DB에 아무 권한이 없는 사용자이름은 'kim' 비밀번호는 'aa111' 인 사용자 생성

```
mysql> grant insert, update, delete, index, alter,create, drop
    -> on books.*
    -> to kim;
Query OK, 0 rows affected (0.00 sec)
```

- 'kim' 이라는 사용자에게 Books 라는 DB를 대상으로 필요한 권한을 준다

```
mysql> revoke alter, create, drop
    -> on books.*
    -> from kim;
Query OK, 0 rows affected (0.00 sec)
```

- 'kim' 이라는 사용자에게 books 라는 DB를 대상으로 권한을 뺏어 간다

```
mysql> revoke all
    -> on books.*
    -> from kim;
Query OK, 0 rows affected (0.00 sec)
```

- 'kim' 이라는 사용자가 더 이상 books 라는 DB에 접근하지 못하게 모든 권한을 뺏어 간다

MySQL 사용하기

- **데이터베이스 사용하기**
  - 제일 먼저 해야 할 일은 지금부터 사용하고자 하는 데이터베이스의 이름이 무엇인지 MySQL 에게 알려주는 것이다

    - use *dbname;*
      - dbname에는 자신이 사용할 데이터베이스의 이름을 넣으면 된다

    - DB 생성
      - create database *dbname;*

    - DB 삭제
      - drop database *dbname;*

- **데이터베이스 테이블 만들기**

    - CREATE TABLE *tablename(columns)*
      - tablename : 만들 테이블의 이름
      - columns : 이 테이블의 열이 될 항목들이 들어간다(여러 개 일때에는 ',' 로 구분)

Ex) '서점'의 스키마
- 고객관리(고객ID, 이름, 주소, 도시)
- 주문(주문ID, 고객ID, 가격, 날짜)
- 책(일련번호, 작가, 제목, 가격)
- 책_주문(주문ID, 일련번호, 수량)

□ **고객관리 TABLE**

```
mysql> create table customer(
    -> customerid int unsigned not null auto_increment primary key,
    -> name char(50) not null,
    -> address char(100) not null,
    -> city char(30) not null
    -> );
Query OK, 0 rows affected (0.06 sec)
```

□ 각 키워드들의 의미

AUTO_INCREMENT - 정수형의 열에서 사용할 수 있는 MySQL 의 특징적인 기능
이 필드에 아무 값도 없는 채로 테이블에 한 행의 데이터를 입력한다면, 지금까지의 숫자 중
가장 큰 것보다 1만큼 증가한 수를 자동으로 여기에 입력해준다

PRIMARY KEY - 이 열이 테이블의 기본 키라는 뜻

NOT NULL - 이 테이블의 모든 행이 반드시 값을 가져야 함을 의미한다

INT UNSIGNED - int 는 데이터 형을 정의, unsigned 는 이 열에 들어가는 데이터는 0 이상의 양수여야 함을 뜻한다

CHAR(Byte) - 데이터 형을 char로 ( ) 안의 Byte만큼 공간을 만든다

□ **주문 TABLE**

```
mysql> create table orders(
    -> orderid int unsigned not null auto_increment primary key,
    -> customerid int unsigned not null,
    -> amount float(6,2),
    -> date date not null
    -> );
Query OK, 0 rows affected (0.01 sec)
```

amount 열은 float 이라는 실수형으로 정의되어 다, 모든 실수형을 정의할 때에는 몇자리의 숫자를 보여줄지와
그 중 몇 자리가 소수점 아래 자리인지를 명시해준다
전체가 6자리 그 중 2자리가 소수점 아래 자리

date 열의 형은 date(날짜)이다

이 테이블에서는 모두 NOT NULL 인데 amount 는 예외 이다
--> 어떤 주문이 DB에 입력될 때, 이는 '주문' 테이블에 저장되며, 어떤 상품들을 주문했는지에 대해서는 '책_주문'에 저장
되고 난 후에 금액이 계산되기 때문이다
주문을 저장할 때에는 금액이 얼마인지 알지 못하기 때문에 이 항목은 NULL 이 될 수 있도록 허용한다

□ **책 TABLE**

```
mysql> create table books(
    -> isbn char(13) not null primary key,
    -> author char(50),
    -> title char(100),
    -> price float(4,2)
    -> );
Query OK, 0 rows affected (0.01 sec)
```

ISBN 는 책마다 유일하게 정의된 번호이기 때문에 기본 키로 사용하고 있다
나머지 항목들은 NULL이 되어도 상관없도록 허용하고 있다 , 서점에서 책의 저자와 제목, 가격 은 모르고 책의 ISBN(일련번호) 번호만 알고 있는 경우도 있기 때문이다

□ **책_주문 TABLE**

```
mysql> create table book_order(
    -> orderid int unsigned not null,
    -> isbn char(13) not null,
    -> quantity tinyint unsigned,
    -> primary key (orderid,isbn)
    -> );
Query OK, 0 rows affected (0.00 sec)
```

여러 개의 열이 기본 키가 되는 예를 보여 준다, 두 개 이상의 열이 함께 기본 키를 이룰 때에는, 별도의 구문과 괄호로 선언해주도록 한다

quantity열을 TINYINT UNSIGNED 라는 데이터 형으로 정의 했다(0~255 사이의 정수를 가진다)

- **데이터베이스 살펴보기**
  - 데이터베이스 확인하기
    - show databases;

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| book               |
| mysql              |
| test               |
+--------------------+
4 rows in set (0.00 sec)
```

  - 테이블 확인하기
    - show tables;

```
mysql> use book;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+----------------+
| Tables_in_book |
+----------------+
| book_order     |
| books          |
| customer       |
| orders         |
+----------------+
4 rows in set (0.00 sec)
```

먼저 어떤 DB를 볼 것인가에 대해 DB를 입력하고 테이블을 확인한다

  - 특정 테이블 자세한 정보 확인
    - describe *table_name;*

```
mysql> describe books;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| isbn   | char(13)    | NO   | PRI | NULL    |       |
| author | char(50)    | YES  |     | NULL    |       |
| title  | char(100)   | YES  |     | NULL    |       |
| price  | float(4,2)  | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
4 rows in set (0.08 sec)
```

--> 이런 명령어들은 열의 형이 무엇인지 알아볼 때, MySQL 안에 어떤 DB 들이 있는지 찾아볼 때 유용하다

- INDEX 만들기
  - INDEX 란
    - 테이블에 저장된 데이터를 빠르게 조회하기 위한 데이터베이스 객체
    - B -Tree 구조를 가진다 (B - Tree Index 의 경우)
  - 명령어 형식
    CREATE [UNIQUE | FULLTEXT] INDEX *index_name*
    ON *table_name*(index_column_name) [(length)] [ASC | DESC], ...])

    length 필드를 사용하여 필드의 앞에서부터 일정 자리 수만큼만 인덱스로 사용할 수 있다
    인덱스를 오름차순(ASC) 혹은 내림차순 (DESC)으로 정렬할지 정할 수 있다 , 기본적으로는 오름차순 이다

MySQL 데이터 형

- **열의 데이터 형**
  - 열(column)이 가질 수 있는 데이터 형은 크게 숫자, 날짜와 시간 문자열 이렇게 세 가지로 구분할 수 있다
    - 일단 어떠한 데이터 형을 선택하느냐에 따라 저장 공간 낭비를 줄일 수 있다는 점을 염두에 두자
    그리고 데이터를 저장할 때에는 꼭 맞는 데이터 형을 사용하도록 한다

    : 많은 데이터 형들이, 얼마만큼(몇 자리가) 보이게 할 것인지를 처음에 정할 수 있다, 앞으로 표에서는 이것을 'M' 이라 하겠다
    'M' 값을 지정하는 것이 옵션이면, [ ] 안에 M을 표시하겠다
    ('M'에는 최대 255까지 쓸 수 있다, 모든 옵션들은 [ ] 로 표시하겠다)

- **숫자 데이터 형**
  - 숫자 데이터 형은 정수형(intger)이거나 실수형이다
  - 실수형에서는 정수부분과 소수부분 숫자의 개수를 각각 지정할 수 있다 , 이 소수부분 숫자의 개수를 앞으로 'D'라고 표기
    ('D'에는 최대값으로 30 까지 쓸 수 있다)

<정수 데이터 형>

종류	범위	저장 공간(Byte)	설명
TINYINT[(M)]	-127 ~ 128 / 0 ~ 255	1	굉장히 작은 수
BIT			TINYINT 와 동일
BOOL			TINYINT 와 동일
SMALLINT[(M)]	-32768 ~ 32767 / 0 ~ 65535	2	작은 정수
MEDIUMINT[(M)]	-8388608 ~ 8388607 / 0 ~ 16777215	3	보통 크기
INT[(M)]	-2^31 ~ 2^31-1 / 0 ~ 2^32-1	4	보통의 정수
INTEGER[(M)]			INT 와 동일
BIGINT[(M)]	-2^63 ~ 2^63-1 / 0 ~ 2^64-1	8	큰 정수

범위 중 앞의 것은 SIGNED , 뒤의 것은 UNSIGNED 일 때의 값

<부동소수 데이터 형>

종류	범위	저장 공간(Byte)	설명
FLOAT(*precision*)	precision 값에 따라 다르다	다양	32/64비트 부동소수를 나타낸다
FLOAT[(M,D)]	±1.175494351E-38 / ±3.402823466E+38	4	32비트 부동소수로 FLOAT(4)와 같은데, 전체 자릿수와 소수점자릿수를 지정해 줄 수 있다
DOUBLE[(M,D)]	±1.7976931348623157E+308 / ±2.2250738585072014E-308	8	FLOAT(8)과 같은데, 전체 자릿수와 소수점 자릿수를 지정해줄 수 있다
DOUBLE PRECISION[(M,D)]	위와 같다		DOUBLE[(M,D)]와 같다
REAL[(M,D)]	위와 같다		DOUBLE[(M,D)]와 같다
DECIMAL[(M[,D])]	가변	M+2	char 형으로 저장된다 , M에 의해 범위가 결정된다
NUMERIC[(M,D)]	위와 같다		DECIMAL과 같다
DEC[(M,D)]	위와 같다		DECIMAL과 같다
FIXED[(M,D)]	위와 같다		DECIMAL과 같다

- **날짜와 시간에 대한 데이터 형**
  - 이 형들은 문자와 숫자가 섞인 형식의 데이터를 입력받을 수 있다

- TIMESTAMP 형의 열은 사용자가 특별한 다른 입력을 주지 않으면, 데이터를 최종적으로 입력/변경한 시간이 저장된다
- 트랜잭션 기록에 유용하다

<날짜와 시간에 대한 데이터 형>

형	범위	설명
DATE	1000-01-01 / 9999-12-31	날짜, YYYY-MM-DD 의 형식
TIME	-838:59:59 / 838:59:59	시간, HH:MM:SS 의 형식으로 표기 되며, 실생활에서의 시간보다 값의 범위가 훨씬 크다
DATETIME	1000-01-01 0000:00:00 / 9999-12-31 23:59:59	날짜와 시간, YYYY-MM-DD HH:MM:SS 의형식
TIMESTAMP[(M)]	1970-01-01 00:00:00	타임스탬프, 데이터의 최종 변경 시각을 저장하는 데 유용하다 M 값에 따라 표기 형식이 달라지며, 유닉스에서의 한계 값은 2037이다
YEAR[(2|4)]	70~69(1970~2069) / 1901 ~ 2155	연도, 2 또는 4로 표기한다(예를 들어, 03년으로 할지 2003년으로 할지)

<TIMESTAMP 표기 형식>

TIMESTAMP 종류	표기 형식
TIMESTAMP	YYYYMMDDHHMMSS
TIMESTAMP(14)	YYYYMMDDHHMMSS
TIMESTAMP(12)	YYMMDDHHMMSS
TIMESTAMP(10)	YYMMDDHHMM
TIMESTAMP(8)	YYYYMMDD
TIMESTAMP(6)	YYMMDD
TIMESTAMP(4)	YYMM
TIMESTAMP(2)	YY

- **문자열 데이터 형**
  - 문자열 데이터 형은 세 가지로 분류할 수 있다

1. 일반적인 문자열(String), 즉 짧은 길이의 구문이다
char(고정된 길이의 문자)와 varchar(가변 길이의 문자) 두 가지가 있고, 각각의 범위의 한도는 지정해줄 수 있다
char 형으로 지정된 열에 데이터가 입력될 때, 실제 데이터 길이가 지정된 범위에 미치지 못하면 특별한 문자가 나머지 부족한 분량에 채워진다
varchar 형은 데이터 길이에 맞게 가변적이다
각각 처리 속도와 저장 공간의 절약이라는 면에서 장단점이 있다

2.TEXT 와 BLOB 형, 상대적으로 긴 길이의 문장이나 이진 데이터를 저장할 때 쓰인다
BLOB이란 'binary large object'이라는 뜻이다, 이미지나 사운드 데이터들도 저장할 수 있다
TEXT가 대소문자를 구별한다는 것만 제외하면 BLOB와 TEXT는 동일하다

3.SET과 ENUM 형
SET 형으로 지정된 열에 들어가는 데이터는, 미리 정의 해놓은 집합 안의 원소들로만 구성되어야 한다, 집합에는 원소가 64개까지 들어갈 수 있다
ENUM은 SET과 비슷하게, 미리 정의된 집합 안의 원소에 있는 값만 데이터로 저장한다
SET 형으로 선언되어 있으면 집합 안의 여러 원소가 데이터에 들어가는데 반해, ENUM은 하나만 들어가거나 NULL이어야 한다는 차이가 있다, ENUM에 쓰이는 집합에는 65,535 개까지 들어갈 수 있다

<문자열 데이터 형>

연산자	사용법	동일식
[NATIONAL] CHAR(M) [BINARY|ASCII|UNICODE]	0 ~ 255 개의 문자	길이는 M이다 (M은 0 ~ 255 사이) 옵션은 아래에 설명
CHAR		CHAR(1)과 같다
[NATIONAL] VARCHAR(M) [BINARY]	1 ~ 255 개의 문자	가변적인 길이도 지원된다는 점만 빼면 위의 CHAR 와 같다

-NATIONAL 이라고 쓰면, 기본 문자집합들이 사용된다는 것을 뜻한다, MySQL에서는 이것이 기본적인 설정이지만.
이 '기본 문자 집합'이라는 것은 ANSI SQL 표준의 일부분일 뿐이다
- BINARY 라는 키워드는, 데이터가 대소문자를 구분하지 않는다는 의미이다(별도로 지정이 없을 때에는 대소문자를 구분한다)
- ASCII 키워드는 'latin1' 문자집합을 사용한다는 뜻이다
- UNICODE 키워드는 'ucs' 문자집합을 사용한다는 뜻이다

<TEXT 와 BLOB 형>

형	최대 길이 (문자 수)	설명
TINYBLOB	2^8-1(255)	작은 크기의 이진 데이터 저장
TINYTEXT	2^8-1(255)	작은 크기의 텍스트 데이터 저장
BLOB	2^16-1(65,535)	보통 크기의 이진 데이터 저장
TEXT	2^16-1(65,535)	보통 크기의 텍스트 데이터 저장
MEDIUMBLOB	2^24-1(16,777,215)	중간 크기의 이진 데이터 저장
MEDIUMTEXT	2^24-1(16,777,215)	중간 크기의 텍스트 데이터 저장
LONGBLOB	2^32-1(4,294,967,295)	큰 크기의 이진 데이터 저장
LONGTEXT	2^32-1(4,294,967,295)	큰 크기의 텍스트 데이터 저장

<ENUM과 SET 형>

형	집합에 들어갈 수 있는 최대 개수	설명
ENUM('value1','value2',....)	65,535	정의된 value 중에서 하나만 들어가거나 NULL 이어야 한다
SET('value1','value2,...)	64	정의된 value를 여러 개 사용한 데이터가 들어가거나 NULL 이어야 한다