Mysql - Basic syntax

Create, Alter, Drop, Insert, Select

1. Create

1.1 Database

데이터 베이스 생성

CREATE DATABASE <database_name>;

test 데이터 베이스 생성

CREATE DATABASE test;

데이터 베이스 선택

USE <database_name>;

test 데이터 베이스 선택

USE test;

현재 데이터 베이스 확인

SELECT DATABASE()

1.2 Table

- 테이블 생성시 컬럼이름, 데이터타입, 제약조건에 대한 내용이 들어갑니다.

테이블 생성

CREATE TABLE <table_name> (

    column_name_1 column_data_type_1 column_constraint_1,

    column_name_2 column_data_type_2 column_constraint_2,

```
    ...
)

# 제약조건이 없는 user1 테이블 생성
CREATE TABLE user1(
    user_id INT,
    name Varchar(20),
    email Varchar(30),
    age INT(3),
    rdate DATE
)

# 제약조건이 있는 user2 테이블 생성
CREATE TABLE user2(
    user_id INT PRIMARY KEY AUTO_INCREMENT,
    name Varchar(20) NOT NULL,
    email Varchar(30) UNIQUE NOT NULL,
    age INT(3) DEFAULT '30',
    rdate TIMESTAMP
)
```

2. Alter

2.1 Database

ALTER를 이용하여 데이터베이스 encoding을 변경할수 있습니다.

```
# 현재 문자열 인코딩 확인
```

```
show variables like "character_set_database";

# test 데이터 베이스의 문자열 인코딩을 utf8으로 변경
ALTER DATABASE test CHARACTER SET = utf8;
ALTER DATABASE test CHARACTER SET = ascii;

# 문자열 인코딩 변경 결과 확인
show variables like "character_set_database";
```

2.2 Table

ALTER를 이용하여 Table의 컬럼을 추가하거나 삭제하거나 수정할수 있습니다.

2.2.1 ADD

```
# user2 테이블에 TEXT 데이터 타입을 갖는 tmp 컬럼을 추가
sql> ALTER TABLE user2 ADD tmp TEXT;
```

2.2.2 MODIFY

```
# user2 테이블에 INT(3) 데이터 타입을 갖는 tmp 컬럼으로 수정
sql> ALTER TABLE user2 MODIFY COLUMN tmp INT(3);
```

2.2.3 DROP

```
# user2 테이블의 tmp 컬럼을 삭제
sql> ALTER TABLE user2 DROP tmp;
```

3. DROP

DROP을 시용하여 데이터베이스나 테이블를 삭제할수 있습니다.

3.1 Database

```
# tmp 데이터 베이스 생성
CREATE DATABASE tmp;
SHOW DATABASES;

# tmp 데이터 베이스 삭제
DROP DATABASE tmp;
SHOW DATABASES;
```

3.2 Table

```
# tmp 데이터 베이스 생성
CREATE DATABASE tmp;
# tmp 데이터 베이스 선택
USE tmp;
# tmp 테이블 생성
CREATE TABLE tmp( id INT );
# tmp 테이블 삭제
DROP TABLE tmp;
```

4. INSERT

4.1 syntax

테이블 이름 뒤에 오는 컬럼이름은 생략이 가능하며 대신에 VALUES 뒤에 value 값이 순서대로 와야 합니다.

INSERT INTO <table_name>(<column_name_1>, <column_name_2>, ...)

VALUES(<value_1>, <value_2>, ...)

여러개의 row를 한꺼번에 insert

INSERT INTO <table_name>(<column_name_1>, <column_name_2>, ...)

VALUES (<value_1>, <value_2>, ...),

(<value_1>, <value_2>, ...),

...

(<value_1>, <value_2>, ...);

4.2 example

test 데이터 베이스 선택

sql> USE test;

user1 테이블에 user_id, name, email, age, rdate를 입력

INSERT INTO user1(user_id, name, email, age, rdate)

VALUES (1, "jin", "pdj@gmail.com", 30, now()),

(2, "peter", "peter@daum.net", 33, '2017-02-20'),

(3, "alice", "alice@naver.com", 23, '2018-01-05'),

(4, "po", "po@gmail.com", 43, '2002-09-16'),

(5, "andy", "andy@gmail.com", 17, '2016-04-28'),

(6, "jin", "jin1224@gmail.com", 33, '2013-09-02');

5. SELECT

5.1 basic

SELECT <column_name_1>, <column_name_2>, ...

FROM <table_name>

전체 컬럼 데이터 조회

sql> SELECT *

FROM user1

user_id, name, rdate 세개의 컬럼 데이터 조회

sql> SELECT user_id, name, rdate

FROM user1

5.2 ALIAS

alias(as)를 이용하여 컬럼명을 변경할수 있습니다. as는 생략이 가능합니다.

user_id, name, rdate 세개의 컬럼 데이터 조회

sql> SELECT user_id as "아이디", name as "이름", rdate as "등록일"

FROM user1

5.3 DISTINCT

DISTINCT를 이용하여 특정 컬럼의 중복 데이터를 제거할수 있습니다.

name 컬럼을 중복 제거하여 조회

```
sql> SELECT DISTINCT(name)
        FROM user1
```

5.4 WHERE

WHERE절을 이용해서 검색 조건을 추가할수 있습니다. WHERE절에는 AND, OR, 연산자 등의 기능을 사용할수 있습니다.

```
# 나이가 30살 이상인 user를 조회
sql> SELECT *
        FROM user1
        WHERE age >= 30

# 등록일이 2016-01-01일 이후의 user를 조회
sql> SELECT *
        FROM user1
        WHERE rdate >= "2016-01-01"

# 등록일이 2010-01-01에서 2017-12-31인 user를 조회 (AND는 둘다 true이면 true, OR도 사용가능)
sql> SELECT *
        FROM user1
        WHERE rdate >= "2010-01-01" AND rdate <= "2017-12-13"

# BETWEEN을 사용하여 기간 조회 (숫자데이터도 사용 가능)
sql> SELECT *
        FROM user1
        WHERE rdate BETWEEN "2010-01-01" AND "2017-12-13"
```

5.5 ORDER BY

ORDER BY를 이용하여 특정 컬럼으로 데이터 정렬이 가능합니다.

```
# age로 오름차순 정렬 (ASC는 생략이 가능)
sql> SELECT *
     FROM user1
     ORDER BY age ASC

# age로 내림차순 정렬
sql> SELECT *
     FROM user1
     ORDER BY age DESC

# age로 내림차순 정렬하고 rdate를 오름차순 정렬
sql> SELECT *
     FROM user1
     ORDER BY age DESC, rdate
```

5.6 CONCAT

CONCAT을 사용하여 select한 데이터를 합쳐서 새로운 컬럼으로 보여주는 것이 가능합니다.

```
# name과 age를 같이 보여주도록 조회
sql> SELECT email, CONCAT(name, "(", age, ")") AS "name_age"
     FROM user1
```

5.7 LIKE

LIKE를 이용하여 특정 문자열이 들어간 데이터 조회가 가능합니다. %는 어떤 문자나와 같은 의미로 "%gmail" 은 gmail 문자열 앞에 아무 문자나 올수 있음을 의미 합니다. 또한 NOT LIKE를 사용하여 특정 문자가 들어가지 않는 데이터를 조회할수 있습니다.

```
# email에 gmail이 들어간 데이터 조회
sql> SELECT *
     FROM user1
     WHERE email LIKE "%@gmail%"

# email에 gmail이 들어가지 않는 데이터 조회
sql> SELECT *
     FROM user1
     WHERE email NOT LIKE "%@gmail%"
```

5.8 IN

IN은 여러개의 조건을 만족하는 데이터를 조회하고 싶을때 사용합니다. WHERE 절의 조건을 여러개 사용하는것을 간단하게 만들수 있습니다.

```
# name이 peter와 alice 조회
sql> SELECT *
     FROM user1
     WHERE name="peter" OR name="alice"

# name이 peter와 alice 조회
sql> SELECT *
     FROM user1
     WHERE name IN ("peter", "alice")
```

5.9 LIMIT

LIMIT은 조회하는 데이터의 수를 제한할수 있습니다. 데이터가 너무 많은 경우에는 항상 limit을 사용하여 적은 데이터를 조회해야 조회 시간이나 업데이트시의 실수를 줄일수 있습니다.

```
# user1 테이블에서 3개의 데이터 조회
sql> SELECT *
     FROM user1
     LIMIT 3

# user1 테이블에서 두번째에서 네번째까지의 3개의 데이터 조회
sql> SELECT *
     FROM user1
     LIMIT 1,3
```

6. UPDATE

업데이트시에는 항상 select-where로 변경할 데이터를 확인하고 update 해줘야 실수를 줄일수 있습니다. 또한 limit도 함께 사용해주면 좋습니다.

6.1 syntax

```
UPDATE <table_name>
SET <column_name_1> = <value_1>, <column_name_2> = <value_2>,
WHERE <condition>
```

6.2 example

```
# jin 이름을 가지고 있는 사람의 나이를 20, 이메일을 pdj@daum.net으로 변경
sql> UPDATE user1
```

SET age=20, email="pdj@daum.net"

WHERE name="jin"

7. DELETE

7.1 syntax

DELETE FROM <table_name>

WHERE <condition>

7.2 example

2016-01-01 이전 데이터 삭제

sql> DELETE FROM user1

WHERE rdate < "2016-01-01"

8. GROUP

8.1 GROUP BY

GROUP BY는 여러개의 동일한 데이터를 가지는 특정 컬럼을 합쳐주는 역할을 하는 명령입니다.

sql에는 아래와 같은 그룹함수가 있습니다.

count, min, max, sum, avg

world 데이터 베이스로 이동

world 데이터 베이스는 city, country, countrylanguage 테이블이 있는 데이터 베이스 입니다.

sql> use world

8.1.1 COUNT

```
# city 테이블의 CountryCode를 묶고 각 코드마다 몇개의 데이터가 있는지 확인
sql> SELECT CountryCode, COUNT(CountryCode)
        FROM city
        GROUP BY CountryCode

# countrylanguage 테이블에서 전체 언어가 몇개 있는지 구하시오.
sql > SELECT COUNT(DISTINCT(Language)) as language_count
         FROM countrylanguage
```

8.1.2 MAX

```
# 대륙별 인구수와 GNP 최대 값을 조회
sql> SELECT continent, MAX(Population) as Population, MAX(GNP) as GNP
        FROM country
        GROUP BY continent
```

8.1.3 MIN

```
# 대륙별 인구수와 GNP 최소 값을 조회 (GNP와 인구수가 0이 아닌 데이터 중에서)
sql> SELECT continent, MIN(Population) as Population, MIN(GNP) as GNP
        FROM country
        WHERE GNP != 0 AND Population != 0
        GROUP BY continent
```

8.1.4 SUM

```
# 대륙별 총 인구수와 총 GNP
sql> SELECT continent, SUM(Population) as Population, SUM(GNP) as GNP
```

```
    FROM country
    WHERE GNP != 0 AND Population != 0
    GROUP BY continent
```

8.1.5 AVG

```
# 대륙별 평균 인구수와 평균 GNP 결과를 인구수로 내림차순 정렬
sql> SELECT continent, AVG(Population) as Population, AVG(GNP) as GNP
    FROM country
    WHERE GNP != 0 AND Population != 0
    GROUP BY continent
    ORDER BY Population DESC
```

8.2 HAVING

GROUP BY에서 반환되는 결과에 조건을 줄수 있습니다.

```
# 대륙별 전체인구를 구하고 5억이상인 대륙만 조회
sql> SELECT continent, SUM(Population) as Population
    FROM country
    GROUP BY continent
    HAVING Population > 500000000
```

```
# 대륙별 평균 인구수, 평균 GNP,  1인당 GNP한 결과를 1인당 GNP가 0.01 이상인 데이터를 조회하고 1인당 GNP를 내림차순으로 정렬
sql> SELECT continent, AVG(Population) as Population, AVG(GNP) as GNP,
            AVG(GNP) / AVG(Population) * 1000 as AVG
    FROM country
    WHERE GNP != 0 AND Population != 0
    GROUP BY continent
```

HAVING AVG > 0.01

ORDER BY AVG DESC

9. Data Type

데이터 베이스의 테이블을 생성할때 각 컬럼은 데이터 타입을 가집니다.

reference

https://dev.mysql.com/doc/refman/5.7/en/data-types.html

9.1 Numberic

reference

https://dev.mysql.com/doc/refman/5.7/en/numeric-types.html

9.1.1 정수 타입 (integer types)

Type	Storage (Bytes)	Minimum Value Signed	Minimum Value Unsigned	Maximum Value Signed	Maximum Value Unsigned
TINYINT	1	-128	0	127	255
SMALLINT	2	-32768	0	32767	65535
MEDIUMINT	3	-8388608	0	8388607	16777215
INT	4	-2147483648	0	2147483647	4294967295
BIGINT	8	-2^63	0	2^63-1	2^64-1

9.1.2 고정 소수점 타입 (fixed-point types)

DECIMAL(M, D)

M : 소수점을 포함한 전체 자리수

D : 소수 부분 자리수

```
sql> ALTER TABLE user1
     ADD COLUMN deci DECIMAL(5,2)
```

NUMERIC(M, D)

9.1.3 실수 (floating-point types)

소수점을 나타내기 위한 데이터 타입으로 아래의 두가지 데이터 타입이 있습니다. 두가지의 데이터 타입은 데이터 저장공간의 차이가 있습니다.

FLOAT (4byte), DOUBLE (8byte)

또한 아래와 같이 고정 소수점 타입으로도 사용이 가능합니다.

FLOAT(M,D), DOUBLE(M,D)

9.1.4 비트 값 타입 (bit value type)

0과 1로 구성이 되는 2진수(binary) 데이터를 나타냅니다.

BIT(M)

- M은 비트의 범위를 나타냅니다. 예를들어 M을 5로 작성하면 00000(2) ~ 11111(2) 까지 표현이 가능합니다.

9.2 Date & Time

reference

https://dev.mysql.com/doc/refman/5.7/en/date-and-time-types.html

9.2.1 DATE

DATE는 날짜를 저장하는 데이터 타입이며, 기본 포멧은 "년-월-일" 입니다.

9.2.2 DATETIME

DATETIME은 날짜와 시간을 저장하는 데이터 타입이며, 기본 포멧은 "년-월-일 시:분:초" 입니다.

9.2.3 TIMESTAME

TIMESTAME는 날짜와 시간을 저장하는 데이터 타입이며, DATETIME과 다른점은 날짜를 입력하지 않으면 현재 날짜와 시간을 자동으로 저장할수 있는 특징이 있습니다.

9.2.4 TIME

TIME은 시간을 저장하는 데이터 타입이며, 기본 포멧은 "시:분:초" 입니다.

9.2.5 YEAR

YEAR는 연도를 저장할수 있는 데이터 타입입니다.

YEAR(2)는 2자리의 연도를 저장할수 있으며 YEAR(4)는 4자리의 연도를 저장할수 있습니다.

9.3 String

reference

https://dev.mysql.com/doc/refman/5.7/en/string-types.html

9.3.1 CHAR & VARCHAR

Value	CHAR(4)	Storage Required	VARCHAR(4)	Storage Required
''	' '	4 bytes	''	1 byte
'ab'	'ab '	4 bytes	'ab'	3 bytes
'abcd'	'abcd'	4 bytes	'abcd'	5 bytes
'abcdefgh'	'abcd'	4 bytes	'abcd'	5 bytes

9.3.2 TEXT

CHAR와 VARCHAR는 대체로 크기가 작은 문자열을 저장할때 사용되며 크기가 큰 문자열을 저장할 때는 TEXT를 사용합니다. TEXT의 타입에 따라서 아래와 같이 크기를 가집니다.

Type	Maximum length
TINYTEXT	255 (2^8−1) bytes
TEXT	65,535 (2^{16}−1) bytes = 64 KiB
MEDIUMTEXT	16,777,215 (2^{24}−1) bytes = 16 MiB
LONGTEXT	4,294,967,295 (2^{32}−1) bytes = 4 GiB

10. Constraint

데이터 베이스의 테이블을 생성할때 각 컬럼은 각각의 제약조건을 갖습니다.

10.1 NOT NULL

NOT NULL 제약조건이 있는 컬럼에 NULL 값 (비어있는 값)을 저장할수 없습니다.

10.2 UNIQUE

UNIQUE 제약조건이 있는 컬럼에 같은 값을 저장할수 없습니다.

10.3 PRIMARY KEY

NOT NULL과 UNIQUE 의 제약조건을 동시에 만족해야 합니다. 그러므로 컬럼에 비어 있는 값과 동일한 값을 저장할수 없습니다. 하나의 테이블에 하나의 컬럼만 조건을 설정할수 있습니다.

10.4 FOREIGN KEY

다른 테이블과 연결되는 값이 저장됩니다.

10.5 DEFAULT

데이터를 저장할때 해당 컬럼에 별도의 저장값이 없으면 DEFAULT로 설정된 값이 저장됩니다.

10.6 AUTO_INCREMENT

주로 테이블의 PRIMARY KEY 데이터를 저장할때 자동으로 숫자를 1씩 증가시켜 주는 기능으로 사용합니다.