

DBMS Basic 2: MySQL CRUD

CodeSquad Master
Hoyoung Jung



실습 준비

docker 다운로드 및 설치

터미널 (또는 powershell)에서

```
$ docker pull honux77/mysql  
$ docker run -d -p 3306:3306 --name dbserver \  
honux77/mysql mysqld  
$ docker exec -it dbserver mysql -u root
```

이후 설정은 필요 없음

한글 utf-8 설정

<https://github.com/honux77/practice/wiki/mysql-ko-utf8>

/etc/mysql/my.cnf에 추가

client 부분밑에 추가

[client]

default-character-set = utf8

mysqld 부분밑에 추가

[mysqld]

init_connect = SET collation_connection = utf8_general_ci

init_connect = SET NAMES utf8

character-set-server = utf8

collation-server = utf8_general_ci

mysqldump 부분밑에 추가

[mysqldump]

default-character-set = utf8

mysql 부분밑에 추가

[mysql]

default-character-set = utf8

일반사용자 외부 접속 허용

```
$ cd /etc/mysql  
$ grep -r 'bind'  
# bind-adress=127.0.0.1 내용 주석처리  
$ sevice mysql restart
```

재부팅시 mysqld 자동 실행

```
$ sudo update-rc.d mysql defaults  
# sudo update-rc.d mysql remove
```

데이터베이스 및 일반 사용자 생성

---- 데이터베이스 생성

CREATE DATABASE mydb;

---- 아이디 생성 및 패스워드 지정

CREATE USER 'honux'@'%' IDENTIFIED BY 'my_password';

---- 사용자에게 디비 권한 지정

GRANT ALL ON honuxdb.* TO 'honux'@'%;

FLUSH PRIVILEGES;

GUI 이용 접속

macOS

<https://www.sequelpro.com/>

windows

HeidiSQL

<http://www.heidisql.com/>

그런데 접속이 안 되면?

테이블 생성

```
DROP TABLE IF EXISTS USER;  
CREATE TABLE USER (  
    UID INT PRIMARY KEY AUTO_INCREMENT,  
    EMAIL VARCHAR(64),  
    NAME VARCHAR(64),  
    PW VARCHAR(64)  
);
```

INSERT

```
INSERT INTO USER VALUES (NULL, ...);  
INSERT INTO USER(NAME, PW) VALUES ('honux', 'asdf');
```


SELECT

```
SELECT * FROM USER;  
SELECT ID, NAME FROM USER;  
SELECT ID, NAME FROM USER WHERE ID=3;
```

DELETE

팁: DELETE 문과 SELECT 문은 거의 비슷함, SELECT를 먼저 수행해 본다.

```
DELETE FROM USER; #모든 레코드 삭제  
DELETE FROM USER WHERE ID = 4;
```

UPDATE

```
UPDATE USER SET MONEY = 0;  
UPDATE USER SET NAME = 'Honux' WHERE ID=2;
```

CRUD 실습

- 테이블 생성

```
USER(ID, EMAIL, NAME)
```

```
BOARD(ID, USER_ID, TITLE, BODY, POST_DATE)
```

- 데이터 삽입
- 갱신
- 삭제

테이블 생성

- PRIMARY KEY
- AUTO_INCREMENT
- ENGINE
- CHARSET
- NOT NULL
- DEFAULT
- UNIQUE
- <http://dev.mysql.com/doc/refman/5.7/en/create-table.html>

테이블 정보보기

```
DESC 테이블이름;  
SHOW CREATE TABLE 테이블이름;
```

데이터 타입

- 용도에 맞춰 적절한 타입을 선택해야 함

```
INT  
DEC(5,2)  
DOUBLE  
CHAR(8)  
VARCHAR(64)  
DATETIME
```

- <http://dev.mysql.com/doc/refman/5.7/en/data-types.html>

날짜 관련 타입

DATE: '2000-01-01'
DATETIME: '9999-11-27 15:37:24.5'
TIMESTAMP: DATETIME과 동일

- 단 TIMESTAMP는 1970 - 2038년까지만 저장 가능

```
DROP TABLE IF EXISTS TEST;  
CREATE TEMPORARY TABLE TEST(  
    A INT PRIMARY KEY AUTO_INCREMENT,  
    B TIMESTAMP, C DATE, D DATETIME);  
  
INSERT INTO TEST (C) VALUES ('1977-12-09');  
SELECT * FROM TEST;  
SET @now = NOW();  
SELECT @now;  
INSERT INTO TEST (C) VALUES (@now);  
SELECT * FROM TEST;
```


날짜 처리하기

```
NOW(), CURTIME(), CURDATE()  
SELECT NOW();  
SELECT NOW() AS NOW,  
       DATE_ADD(NOW(), INTERVAL 2 DAY) AS FUTURE;  
SELECT NOW() AS NOW,  
       DATE_SUB(NOW(), INTERVAL 30 MINUTE)  
       AS PAST;
```

PRIMARY KEY 제약 조건

PRIMARY KEY: 반드시 고유해야 하는 대표값

AUTO_INCREMENT: mysql 전용 예약어

```
DROP TABLE IF EXISTS USER;  
CREATE TABLE USER (  
        UID INT PRIMARY KEY AUTO_INCREMENT  
);  
INSERT INTO USER VALUES(NULL);
```

NOT NULL

반드시 값이 존재해야 하는 필드를 위한 제약조건

```
DROP TABLE IF EXISTS USER;
CREATE TABLE USER (
    UID INT PRIMARY KEY AUTO_INCREMENT,
    EMAIL VARCHAR(64) NOT NULL,
    NAME VARCHAR(64)
);
# INSERT INTO USER VALUES(NULL); error
INSERT INTO USER VALUES (NULL, 'admin@cs.kr', 'Honux');
SELECT * FROM USER;
```

DEFAULT

기본값 지정 가능

```
DROP TABLE IF EXISTS USER;
CREATE TABLE USER (
    UID INT PRIMARY KEY AUTO_INCREMENT,
    EMAIL VARCHAR(64) NOT NULL,
    NAME VARCHAR(64),
    MONEY DEC(10,2) DEFAULT 4.99 NOT NULL
);
INSERT INTO USER(EMAIL, NAME)
VALUES ('admin@cs.kr', 'Honux');
SELECT * FROM USER;
```

ENUM 타입

- 프로그래밍의 enum과 동일
- 잘 사용하지 않는 경향이 있음

```
DROP TABLE IF EXISTS USER;  
CREATE TABLE USER (  
    UID INT PRIMARY KEY AUTO_INCREMENT,  
    EMAIL VARCHAR(64) NOT NULL,  
    NAME VARCHAR(64),  
    MONEY DEC(10,2) DEFAULT 4.99 NOT NULL,  
    GRADE ENUM('bronze', 'silver', 'gold', 'platinum')  
        DEFAULT 'bronze' NOT NULL  
);
```

INSERT

테이블에 데이터 삽입시 사용

```
INSERT INTO USER(EMAIL, NAME)
VALUES ( 'admin@cs.kr', 'Honux' );
SELECT * FROM USER;
```

MULTI INSERT

```
INSERT INTO USER(EMAIL, NAME, GRADE)
VALUES ('honux@cs.kr', 'Honux', 'platinum'),
       ('crong@cs.kr', 'Crong', 'bronze'),
       ('pobi@cs.kr', 'Pobi', 'gold'),
       ('pobi2@cs.kr', 'Pobi2', 'gold'),
;
```

CSV LOAD

```
DROP TABLE IF EXISTS TEST;  
CREATE TABLE TEST (A INT, B VARCHAR(16));  
LOAD DATA LOCAL INFILE '/Users/honux/data.csv'  
    INTO TABLE TEST  
    FIELDS TERMINATED BY ',';  
  
SELECT * FROM TEST;  
SELECT COUNT(B) FROM TEST;
```

자주 사용하게 되는 명령

여러가지 최적화를 수행해야 함

SELECT

```
SELECT * FROM USER;  
SELECT UID, NAME FROM USER;  
SELECT UID, NAME, GRADE FROM USER WHERE GRADE = 'bronze';
```

WHERE 조건절

- >, =, <, >= 와 같은 비교 연산자
- AND, OR와 같은 논리 연산자
- LIKE 를 이용한 문자열 매칭

LIKE

```
SELECT * FROM USER WHERE EMAIL LIKE 'h%';
```

매치 스트링

% : 0개 이상의 임의 문자
_ : 1개의 임의 문자

http://www.w3schools.com/sql/sql_like.asp

셀렉트의 결과 및 중첩 쿼리

- 셀렉트 쿼리의 결과는 결과에 따라 값, 레코드 또는 테이블로 간주됨
- 결과값을 다른 쿼리의 매개변수처럼 사용 가능

```
SELECT * FROM USER WHERE UID =  
(SELECT UID FROM USER WHERE EMAIL = 'pobi@cs.kr');
```

IN

```
SELECT * FROM USER WHERE UID IN (1, 2, 3);  
SELECT UID FROM USER WHERE NAME LIKE '%n%';  
SELECT * FROM USER WHERE UID IN  
    (SELECT UID FROM USER WHERE NAME LIKE '%n%');
```

Aggregation Function

COUNT, AVG, SUM, MAX, MIN, ...

```
SELECT COUNT(*) FROM USER;  
SELECT AVG(MONEY) FROM USER;  
SELECT SUM(MONEY) FROM USER;
```

GROUP BY

```
SELECT COUNT(*) FROM USER WHERE GRADE='bronze';  
SELECT COUNT(*) FROM USER WHERE GRADE='silver';  
SELECT COUNT(*) FROM USER WHERE GRADE='gold';  
SELECT GRADE, COUNT(*) FROM USER GROUP BY GRADE;
```

HAVING

- HAVING 절에는 그룹 함수를 조건으로 사용 가능

```
SELECT GRADE, COUNT(*) AS NUM FROM USER  
      GROUP BY GRADE HAVING NUM >= 2;  
# WHERE 절로는 불가능
```

ORDER BY

```
SELECT * FROM USER ORDER BY GRADE;  
SELECT * FROM USER ORDER BY GRADE DESC;
```


ALTER

```
ALTER TABLE USER ADD COLUMN LOGIN DATETIME FIRST;  
ALTER TABLE USER ADD COLUMN LAST DATETIME AFTER NAME;  
DESC USER;
```

UPDATE

```
UPDATE USER SET LOGIN = NOW();  
UPDATE USER SET MONEY = MONEY + 100 WHERE NAME = 'Pobi';  
UPDATE USER SET LAST = DATE_SUB(LOGIN, INTERVAL 1 DAY)  
    WHERE GRADE='bronze';  
SELECT * FROM USER;
```

DROP COLUMN, TABLE

```
ALTER TABLE USER DROP COLUMN LAST;  
DESC USER;  
DROP TABLE USER;  
SHOW TABLES;  
SHOW DATABASES;
```