

MYSQL

<http://jobtc.kr>
훈련교사 박원기
ver 1.1

1 개요

mysql은 rdbms(relational database management system:관계형 데이터 베이스 시스템)으로 1994년에 처음 개발되어졌다. 그 후 2008년에 썬마이크로시스템즈에 의해 인수되었고 2010년 오라클사에서 썬마이크로시스템을 인수함으로 mysql 또한 오라클사에 귀속되었다. 이 때 오픈 소스를 지향하던 기존 개발자들이 퇴사하여 mariaDB를 개발, 출시하여 mysql과 같은 생태계를 유지하고 있다.

1.1 쿼리 작성 규칙

- 대소문자를 구별하지 않는다. 단, 비교문의 문자열은 대소문자를 구별하여 비교한다.
- 문자열은 작은 따옴표, 큰 따옴표를 구분하지 않지만 작은 따옴표 사용을 권장한다.
- 날짜는 문자열로 비교해도 된다.
- 숫자 비교는 문자열로 비교해도 된다.
- 같다란 의미는 '=' 기호를 사용한다.
- 논리 연산자는 not, and , or를 사용한다.
- 쿼리 문장의 끝은 ';' 로 끝낸다.

1.2 SQL 명령의 분류

구분	내요
DDL(Data Definition Language)	CREATE, ALTER, DROP
DCL(Data Control Language)	GRANT, ROVOKE
DML(Data Manipulation Language)	INSERT, UPDATE, DELETE, MERGE, CALL
DQL(Data Query Language)	SELECT

1.3 설치 및 환경설정

설치 과정은 일반 프로그램들의 설치 과정과 별반 다르지 않지만 설치 단계가 꽤나 많다. 또한 설치중 Visual C++의 64비트용 라이브러리가 없어 설치중 오류가 발생할 수 있으며, 이는 installer를 재 실행하여 관련 라이브러리를 선택하여 추가 설치한 후 진행해야 한다.

mysql은 DB가 설치되면서 기본적으로 슈퍼유저인 root가 추가되며, 설치시 암호를 입력하게 된다. 이 암호를 반드시 기억히에 한다.

기본 유저 : [root@localhost](#)
DB : mysql

mysqldump 등과 같은 응용 프로그램을 원활하게 사용하기 위해 **시스템환경 설정에 PATH를 추가**해 놓는 것이 여러모로 유용하다. 설치 경로는 "mysql의 설치 경로/bin"까지를 등록해 둔다.

1.4 유저 등록 및 권한 부여

1.4.1 유저 생성

유저 등록 : CREATE USER 아이디@호스트 [IDENTIFIED BY 암호];

mysql이 8.0으로 버전업이 되면서 클라이언트 툴의 버전에 따라서 암호를 처리 하는 방법이 달라져 사용자를 등록하거나 암호를 지정할 때 이전 버전 방식으로 암호를 지정할 수 있도록 조치할 필요도 있다.

암호지정 방식이 mysql_native_password 에서 caching_sha2_password 형식으로 바뀐.

CREATE USER 아이디@호스트 IDENTIFIED WITH MYSQL_NATIVE_PASSWORD BY '암호';

ex) 유저 이름 : hong 암호 : 1111을 사용하여 유저 등록

```
CREATE USER hong99@localhost IDENTIFIED BY '1111';  
or  
CREATE USER hong99@localhost IDENTIFIED WITH MYSQL_NATIVE_PASSWORD BY  
'1111';
```

1.4.2 유저 제거

DROP USER 유저명@호스트

ex) hong99 유저 삭제

```
DROP USER hong99@localhost;
```

1.4.3 사용자 권한

대표적인 권한 몇가지만을 언급하도록 하겠다.

권한	내용
all	모든 권한
alter	테이블의 구조를 변경할 수 있는 권한
create	DB와 Table을 생성할 수 있는 권한
create user	사용자 관리에 관한 권한
create view	view를 관리할 수 있는 권한
select, insert, update, delete	Table 데이터 조작 권한
drop	DB, Table, View등을 삭제할 수 있는 권한

1.4.4 권한부여

```
권한부여 : GRANT [권한종류 | ALL] PRIVILEGES ON db명.테이블명 TO 아이디@호스트;
```

ex) kim@localhost에게 test DB에 있는 모든 테이블에 관한 모든 권한 부여

```
GRANT ALL PRIVILEGES ON test.* TO kim@localhost
flush privileges;
```

1.4.5 권한 제거

```
REVOKE [권한] ON [DB].[TABLE] FROM 유저@호스트
```

ex) kim@localhost에게 주웠던 권한 제거

```
REVOKE ALL ON test.* FROM kim@localhost
flush privileges;
```

2 DATABASE 생성

- DB명은 5.1 버전이후에는 변경할 수 없게 됨.
- **USE db_name** 으로 사용할 db를 선택한다.
- DB 목록 확인 : SHOW DATABASES;

2.1 생성

```
CREATE DATABASE db명;
```

2.2 삭제

```
DROP DATABASE db명;
```

2.3 sample table install

mysql 설치시 샘플로 사용될 테이블 데이터들이 설치되지 않아 따로 설치해야 한다. sample table들은 깃허브에 공유되어 있어 내려 받을 수 있으며 손쉽게 import 할 수 있다.

깃허브 : https://github.com/datacharmer/test_db

[mysql_sample_data.sql] 활용하기

- 1) 깃허브에 있는 자료를 좀 더 간단히 설치하기 위해서 만들어진 파일임.
- 2) 탐색기를 사용하여 압축을 푼 파일을 mysql 설치 폴더 아래에 있는 bin 폴더에 복사한다.
- 3) mysql\bin 폴더로 이동한 후 탐색기>파일>Power Shell을 열어 아래와 같이 작업한다.

```
mysql -u root -p < mysql_sample_data.sql
```

- 4) import 작업이 완료되면 classmodels 데이터베이스가 새롭게 생성된다. 이 때 사용자는 root가 된다.

* 다른 유저로도 작업이 가능하다.

3 TABLE

- 테이블 목록 확인 : USE 명령을 사용하여 database를 먼저 선택한 후 SHOW TABLES 로 확인한다.
- 테이블 구조 확인 : DESC 테이블명
- 테이블명 변경 : RENAME TABLE 원본 테이블명 TO 변경 테이블명

3.1 데이터 유형

3.1.1 숫자형

TYPE	BYTES	SIGNED		UNSIGNED	
		MIN	MAX	MIN	MAX
TINYINT	1	-128	127	0	255
SMALINT	2	-32,768	32,767	0	65,535
MEDIUMINT	3	-8,388,608	8,388,607	0	16,777,215
INT	4	-2,147,483,648	2,147,483,647	0	4,294,967,295
BIGINT	8			0	

3.1.2 문자형

TYPE	BYTES	비고
CHAR	255	- TEXT열에는 INDEX를 사용할 수 없음. - TEXT열에는 DEFAULT 속성을 사용할 수 없음.
VARCHAR	65,535	
TINYTEXT	255	
TEXT	65,535	
MEDIUMTEXT	16MB	
LONGTEXT	4GB	

3.1.3 날짜형

TYPE	FORMAT	비고
DATE	CCYY-MM-DD	

TIME	hh:si:ss	
DATETIME	CCYY-MM-DD hh:si:ss	5BYTE
TIMESTEP	CCYY-MM-DD hh:si:ss	4BYTE
YEAR	CCYY or YY	

3.2 테이블 생성

```
CREATE TABLE 테이블명( 컬럼명1 타입(크기) 제약조건, ...)
```

[예] 학번(mid), 성명(irum), 연락처(phone)을 갖는 student 테이블 생성.

```
CREATE TABLE student(
  mid VARCHAR(10),
  irum VARCHAR(50),
  phone VARCHAR(20)
);
```

3.3 테이블 제거

```
DROP TABLE 테이블명;
```

3.4 테이블 복사

```
CREATE TABLE 테이블명 AS SELECT ...;
```

SELECT절에 의해 선택된 컬럼이나 테이터를 사용하여 새로운 테이블을 생성할 수 있다.

3.5 테이블명 변경

```
RENAME TABLE 원본 TO 변경
```

3.6 테이블 구조 변경

[컬럼추가]

ALTER TABLE 테이블명 ADD [COLUMN] 컬러명 컬럼유형 [제약조건];

[컬럼 삭제]

ALTER TABLE 테이블명 DROP [COLUMN] 필드명;

[컬럼유형 변경]

ALTER TABLE 테이블명 MODIFY [COLUMN] 필드명 새데이터형;

[필드명 수정]

ALTER TABLE 테이블명 CHANGE [COLUMN] 이전 필드명 수정필드명 데이터 유형 [제약조건];

4 무결성 제약 조건

4.1 제약 조건의 목적

- 데이터의 무결성 유지
- relation을 보다 견고하게

4.2 무결성 제약조건의 종류

제약 조건명	설명
PRIMARY KEY(PK)	- NOT NULL, UNIQUE의 특성을 갖고 있고 테이블당 하나만 존재. - 둘 이상의 컬럼을 묶어 하나의 PK를 선언할 수 있음. - 자동으로 INDEX가 만들어진다.
FOREIGN KEY(FK)	- 외래키. 자식테이블의 데이터가 존재했을 때 부모 테이블의 데이터가 삭제되거나 수정되는 것을 예방. ON CASCADE UPDATE나 ON CASCADE DELETE 옵션을 사용하여 부모 테이블의 데이터가 삭제되거나 수정되었을 때 자식 테이블의 데이터를 삭제하거나 수정할 수 있게 할수는 있다. - 부모 테이블의 컬럼은 PK나 UNIQUE 제약 조건이 설정되어 있어야 한다.
NOT NULL	- NULL 값을 허용하지 않는다. 컬럼 생성시 기본이 NULL이기 때문에 컬럼에 제약 조건을 수정할 때 ADD를 사용하지 않고 MODIFY를 사용한다.
UNIQUE	- 하나의 NULL은 허용하고, 중복되지 않는 데이터를 유지하기 위해 사용됨. - 자동으로 INDEX가 만들어진다.
CHECK	- 허용되는 값을 일정하게 지정하거나 범위를 줄수 있다. - MariaDB 10, Mysql 8 버전 이상부터 지원
DEFAULT	- 입력되는 값이 NULL값일때 특정값이 입력값이 되도록 설정할 수 있다. - 제약조건을 수정할 때 ADD를 사용하지 않고 MODIFY를 사용한다.

4.3 제약조건 확인

데이터 베이스명을 information_schema로 변경한 뒤 작업한다.

```
USE information_schema
```

```
SHOW TABLES
SELECT * FROM table_constraints
```

4.4 제약 조건 설정하기

4.4.1 테이블 생성시 제약조건 지정하기

[TYPE 1]

```
CREATE TABLE student(
    sno INT PRIMARY KEY,
    mname VARCHAR(20) NOT NULL,
    phone VARCHAR(30) UNIQUE,
    address VARCHAR(50) DEFAULT 'seoul' ,
    gender VARCHAR(2) CHECK( gender IN('f','m') ),

    zipcode INT,
    score INT CHECK( score BETWEEN 0 AND 100),

    FOREIGN KEY(zipcode) REFERENCES member(sno) [ON DELETE CASCADE] [ON UPDATE
CASCADE]
);
```

[TYPE 2]

```
CREATE TABLE student(
    sno INT,
    mname VARCHAR(20) NOT NULL,
    phone VARCHAR(30),
    address VARCHAR(50) DEFAULT 'seoul' ,
    gender VARCHAR(2),

    zipcode INT,
    score INT,

    CONSTRAINT std_sno_pk PRIMARY KEY(sno),
    CONSTRAINT std_gender CHECK(gender IN('f','m')),
    CONSTRAINT std_phone_uk UNIQUE(phone),
    CONSTRAINT std_zipcode_fk FOREIGN KEY(zipcode) REFERENCES member(sno)
        [ON DELETE CASCADE] [ON UPDATE CASCADE]
);
```

4.4.2 테이블 생성 후 제약 조건 수정

[PRIMARY KEY]

추가	ALTER TABLE 테이블명 ADD [CONSTRAINT 제약조건명] PRIMARY KEY(컬럼명) ALTER TABLE 테이블명 MODIFY COLUMN 필드명 필드타입 PRIMARY KEY ALTER TABLE 테이블명 MODIFY COLUMN [CONSTRAINT 제약조건명] PRIMARY KEY(필드명)
삭제	ALTER TABLE 테이블명 DROP PRIMARY KEY

[FOREIGN KEY]

추가	ALTER TABLE 테이블명 ADD CONSTRAINT 제약조건이름 FOREIGN KEY(컬럼명) REFERENCES 부모 테이블명(PK 컬럼명) [ON DELETE CASCADE] [ON UPDATE CASCADE]
삭제	ALTER TABLE 테이블명 DROP FOREIGN KEY 제약 조건명

[NULL | NOT NULL]

수정	ALTER TABLE 테이블명 MODIFY COLUMN 컬럼명 자료형 [NULL OR NOT NULL]
----	---

* null과 not null 제약 조건은 ADD로 하지 않고 MODIFY로 수정한다. 따라서 DROP 을 하지 않고 NULL 또는 NOT NULL로 수정한다.

[UNIQUE]

추가	ALTER TABLE 테이블명 ADD CONSTRAINT 제약조건명 UNIQUE(컬럼명1, [컬럼명2...])
삭제	ALTER TABLE 테이블명 DROP INDEX 제약조건명 DROP INDEX 제약조건명 ON 테이블명

[CHECK]

추가	ALTER TABLE 테이블명 ADD CONSTRAINT 제약조건명 CHECK(조건)
삭제	ALTER TABLE 테이블명 DROP CONSTRAINT 제약 조건명

[DEFAULT]

추가	ALTER TABLE 테이블명 ALTER COLUMN 컬럼명 SET DEFAULT 값
삭제	ALTER TABLE 테이블명 ALTER COLUMN 컬럼명 SET DEFAULT NULL

* 제약조건 삭제시 오류가 발생한다면 외래키 제약조건 때문일 가능성이 높다. 따라서 외래키 제약조건을 먼저 삭제한 후 진행한다.

4.5 제약 조건의 활성화 / 비활성화

설정된 제약 조건을 삭제하지 않고 임시로 활성화 비활성화를 통해 제약조건을 제어 할 수 있다.

활성화	ALTER TABLE 테이블명 ENABLE CONSTRAINT 제약조건
비 활성화	ALTER TABLE 테이블명 DISABLE CONSTRAINT 제약조건

5 SELECT

저장된 데이터를 선택하는 문장이다.

5.1 기본구조

select절 이외의 모든 절은 생략할 수 있다.

```
SELECT * | 컬럼명들  
[ FROM 테이블명 ]  
[ WHERE 조건 ]  
[ GROUP BY 컬럼명 ]  
[WITH ROLLUP ]  
[ HAVING 조건 ]  
[ ORDER BY 컬럼명 [ASC | DESC] ]  
[ LIMIT 시작위치, 개수 ]
```

[SELECT 절]

- SELECT 절에 사용되는 컬럼명에는 연산식, 함수 등을 사용할 수 있다.
- 컬럼명에 AS “별칭” 과 같이 별칭을 사용할 수 있다.
- ‘*’는 모든 컬럼을 의미하지만 보안이나 성능 저하 등의 이유로 사용을 지양해야 한다.

[FROM 절]

- 데이터가 들어 있는 테이블명을 작성한다.
- 서브 쿼리를 테이블명 대신 사용할 수 있다.
- 생략된다면 SELECT절에는 단순 함수나 문자열, 수식만을 작성할 수 있다.

[WHERE 절]

- 관계 연산자, 논리연산자등을 사용하여 선택될 데이터들의 조건을 작성한다.

[GROUP BY 절]

- 데이터들 특정 컬럼의 그룹별로 취합하여 결과를 보여준다.
- GROUP BY 절을 사용하면 SELECT 절에는 GROUP BY절에서 사용된 컬럼이나 집계함수만을 사용할 수 있다.
- WITH ROLLUP을 사용하여 소계등을 쉽게 계산할 수 있다.

[HAVING 절]

- GROUP BY절에 의해 취합된 결과를 선택하는 조건절이다.

[ORDER BY 절]

- 선택된 데이터를 정렬한다.

- 2개 이상의 컬럼을 사용하여 1차로 정렬된 값들 중에서 2차로 정렬순서를 정할 수 있다.

[LIMIT 절]

- 보여줄 행의 시작 위치와 갯수를 지정한다. 시작 위치는 0 부터 시작된다.
- 출력 결과를 분리하여(페이징) 보여주어야 할 때 많이 사용된다.

[예] 연산식을 계산하여 출력.

```
SELECT 10+10, 10*10;
```

[예] customers 테이블의 모든 값 조회.

```
SELECT * FROM customers
```

[예] 컬럼명을 지정하여 값 조회

```
SELECT customerNumber, phone, city FROM customers;
```

[예] 조건을 설정하여 값 조회.

```
SELECT * FROM customers WHERE customerNumber = 112;
```

[예] customerNumber가 112이거나 114 인 자료 조회

```
SELECT * FROM customers WHERE customerNumber=112 OR customerNumber=114;
```

[예] group by절을 사용한 state별 인원수

```
SELECT state, COUNT(*) FROM customers GROUP BY state;
```

[예] state별 인원수가 5명 이상인 자료 조회

```
SELECT state, COUNT(*) cnt FROM customers  
GROUP BY state HAVING cnt>=5 ;
```

[예] 이름순으로 오름차 정렬

```
SELECT * FROM customers ORDER BY customerName DESC;
```

[예] 사원의 이름을 5번째 항목에서부터 10개 조회

```
SELECT * FROM customers LIMIT 4, 10;
```

5.2 with rollup

group by된 그룹별 소계를 다시 계산해줌.

ex) 년도별, 월별 급여 합계(단, 년도별 소계 계산)

```
SELECT date_format(paymentDate, '%y'),  
       date_format(paymentDate, '%y-%m'), SUM(amount)  
FROM payments  
GROUP BY DATE_FORMAT(paymentDate, '%y'), DATE_FORMAT(paymentDate, '%y-%m')  
WITH ROLLUP;
```

5.3 CASE

SELECT절에서 CASE문을 사용할 수 있는데, 이는 다른 언어에서 SWITCH문과 매우 유사하게 사용할 수 있는 제어문이다.

[기본 구조] type 1

```
CASE
  WHEN 조건1 THEN 처리1
  WHEN 조건2 THEN 처리2
  ...
  [ ELSE 기타 처리 ]
END AS 닉네임
```

- case문에 컬럼명이 없으면 when절의 조건1, 조건2에서 범위를 줄 수 있다.

[기본 구조] type 2

```
CASE 컬럼명
  WHEN 상수1 THEN 처리1
  WHEN 상수2 THEN 처리2
  ...
  [ ELSE 기타 처리 ]
END AS 닉네임
```

- case문에 컬럼명이 있으면 when절에서는 상수만을 사용할 수 있다.

[예] officeCode에 따른 도시이름을 한글로 출력

```
SELECT officeCode,
       case
         when officeCode=1 then '샌프란시스코'
         when officeCode=2 then '보스턴'
       END city
FROM offices;

SELECT officeCode,
       case officeCode
         when 1 then '샌프란시스코'
         when 2 then '보스턴'
```



```
END city
FROM offices;
```

6 JOIN

데이터 정규화에 의해 필연적으로 하나의 테이블이 두 개 이상의 테이블로 분리되었을 때 SELECT문을 사용하여 마치 하나의 테이블에서 데이터가 조회되는 것처럼 만들 수 있는 문장이다. 참고로 MERGE는 행 병합이라 하고 JOIN은 열 병합이라 말한다.

조인을 실행하면 일반적으로 카디션 곱으로 나타난다. 카디션 곱이란 A 테이블에 데이터가 10행이 있고, B 테이블 데이터가 5행이 있을 때 이를 조인하면 10x5행의 결과로 50행이 나타나는 것을 카디션 곱이라 한다.

6.1 JOIN의 종류

- cross join
- equi join
- non-equi join
- inner join(self join)
- outer join
 - left outer join
 - right outer join
 - full outer join

6.2 cross join

- 두 개 이상의 테이블이 아무런 기준키 없이 연결되는 join
- 연결된 테이블의 연관 관계는 없다.

[예] offices 테이블과 employees 테이블을 cross join 하기

```
SELECT * FROM offices JOIN employees
or
SELECT * FROM offices , employees
```

6.3 equi join

- 일반적인 조인을 의미한다.
- on절에 두 테이블의 공통조건을 기술하여 조건이 같은 것만 조회한다.

[예] 사번이 1002인 직원의 사무실 도시 이름을 조회

```
SELECT city FROM employees e JOIN offices o
ON e.officeCode = o.officeCode
WHERE employeeNumber = 1002;
```

[예] 사무실의 위치(city)가 Paris인 직원들의 이름 조회

```
SELECT lastName, firstName
FROM offices o JOIN employees e
ON o.officeCode = e.officeCode
WHERE o.city = 'paris';
```

6.4 non-equi join

equi join이 서로 같은 데이터를 연결하여 조회하는 것과 달리 서로 다른 값을 찾아 조회할 때 사용된다. on 절에서 비교 대상을 같지 않은것으로 설정하면 equi join과 사용 방법이 동일하다.

6.5 inner join(self join)

하나의 테이블을 마치 서로 다른 테이블인것처럼 설정하여 자신과 조인하여 결과를 조회하게 된다.

[예] Patterson William과 같은 부서 직원의 이름을 조회

```
SELECT e2.* FROM employees e1 JOIN employees e2
ON e1.officeCode = e2.officeCode
WHERE e1.lastName = 'Patterson' AND e1.firstName = 'William';
```

6.6 outer join

join은 키 값이 서로 같거나 다른 값을 연결하여 조회한다. 따라서 비교되는 키에 누락되는 데이터는 조회되지 않는다. 예를 들어 학생 테이블이 있고, 성적 테이블이 있다면 일반적인 join을 하게 되면 시험을 한번도 보지 않은 학생 명단은 누락된다. 이런 현상을 제거하기 위해 어느 한쪽의 테이블 자료는 무조건 조회되도록 만드는 것이 outer join 이다.

- left outer join : 무조건 출력되어야 하는 테이블이 join 왼쪽에 있는 테이블이 된다.
- right outer join : 무조건 출력되어야 하는 테이블이 join 오른쪽에 있는 테이블이 된다.
- full outer join : join양쪽에 있는 테이블이 키와 연결되는 것과 관계없이 무조건 조회된다. 그러나 이 방법은 DBMS의 종류에 따라 지원되지 않을 수 있다.

[예] 아래 두개의 쿼리 실행 결과를 비교해 보자

```
SELECT c.customerNumber, o.orderNumber
FROM customers c LEFT OUTER JOIN orders o
ON c.customerNumber = o.customerNumber
ORDER BY c.customerNumber;

SELECT c.customerNumber, o.orderNumber
FROM customers c JOIN orders o
ON c.customerNumber = o.customerNumber
ORDER BY c.customerNumber;
```

첫번째 쿼리의 실행 결과엔 주문 정보가 없는 고객번호가 조회되지만, 두번째 쿼리에는 주문정보가 없는 고객번호는 조회되지 않는다.

7 sub query

- ()안에 기술된다.
- 독립적인 실행이 되어야 한다.
- select절, where절, from 절에서 사용 가능하다.

7.1 where절에서 사용하기

```
... where amt = ( select max(c_name) from t_name )
... where amt > ( select max(c_name) from t_name )
... where amt in ( select distinct c_name from t_name)
... where amt not in ( select distinct c_name from t_name)
... where exists( sub query )
... where not exists ( sub query )
... where amt > any( sub query)
... where amt > all(sub query)
```

ex) 생산코드가 S10_1678인 제품의 생산라인의 상세 설명을 조회하시오.

[join 사용]

```
SELECT textDescription FROM products p JOIN productlines pl
ON p.productline = pl.productline
WHERE productCode='S10_1678';
```

[sub query 사용]

```
SELECT textDescription FROM productlines
WHERE productline = ( SELECT productline FROM products WHERE productCode='S10_1678'
);
```

7.2 from 절에서 사용하기

sub query 결과를 하나의 가상 테이블로 사용함.

```
... FROM ( SELECT ... ) AS alias_table_name
```

ex) 코드별 구매단가의 합이 100이상인 자료 조회.

```
SELECT a.* FROM (SELECT productCode, SUM(buyPrice) buy FROM products GROUP BY  
productCode) a  
WHERE a.buy>=100;
```

위의 코드를 having절로 표현하면 아래와 같다.

```
SELECT productCode, SUM(buyPrice) buy FROM products GROUP BY productCode HAVING  
buy>=100;
```

8 INDEX

PK나 unique 제약조건을 설정하면 자동으로 index가 만들어진다.

전문가들마다 약간의 차이는 보이지만 index로 설정된 데이터는 전체 데이터의 5% 내외 일때 좋은 효율을 보인다고 알려져있다.

DML 유형의 작업이 많은 경우 index의 rebuild로 인한 서버의 과부를 초래할 수 있다.

8.1 INDEX 의 종류

종류	설명
NON UNIQUE INDEX	기본인덱스
UNIQUE INDEX	중복값이 없는 값을 갖는 인덱스
FULLTEXT INDEX	컬럼내의 모든 텍스트 필드를 검색

8.2 효율적인 INDEX

- WHERE, ORDER BY 절에 자주 사용되는 컬럼
- JOIN이 자주 사용되는 컬럼.

8.3 생성

```
CREATE INDEX 인덱스명 ON 테이블명(컬럼들 [ASC | DESC])
CREATE UNIQUE INDEX 인덱스명 ON 테이블명(컬럼들 [ASC | DESC])
CREATE FULLTEXT INDEX 인덱스명 ON 테이블명(컬럼들)
```

컬럼들 항목에서 2개 이상의 컬럼을 지정한 경우 복합 INDEX로 처리된다.

테이블에 INDEX를 추가할수도 있다.

```
ALTER TABLE 테이블명 ADD INDEX 인덱스명(필드명)
ALTER TABLE 테이블명 ADD UNIQUE 인덱스명(필드명)
ALTER TABLE 테이블명 ADD FULLTEXT 인덱스명(필드명)
```

8.4 INDEX 지정사용

```
SELECT 컬럼명들 FROM 테이블명  
USE INDEX 인덱스명  
WHERE ...
```

8.5 INDEX 확인

```
SHOW INDEX FROM 테이블명
```

9 VIEW

VIEW은 일종의 가상 테이블이다. 보안이나 복잡한 쿼리를 단순하게 사용할 수 있도록 설정해 둘 수 있다.

9.1 특징

- 한번 지정된 VIEW는 변경되지 않는다.
- 삽입 및 수정, 삭제에 많은 제한 사항이 있다.
- 자신만의 인덱스를 가질 수 없다.

9.2 생성 및 삭제

생성	CREATE [OR REPLACE] VIEW 뷰명 AS SELECT ...
삭제	DROP VIEW 뷰명

10 내장함수

10.1 숫자 관련 함수

함수명	설명
abs(n)	n의 절대값 출력
ceiling(n) or ceil(n)	n의 절상값
floor(n)	n의 절삭값
round(n, p)	p가 양수이면 숫수점 이하, 음수이면 소수점 이상에서 반올림
truncate(n, p)	p의 자리에서 버림
pow(x,y) or power(x,y)	x의 y승
mod(x,y)	x를 y로 나눈 나머지
greatest(n1, n2, n3, ...)	가장 큰수
least(n1, n2, n3, ...)	가장 작은수

ex)

```
/* 수학함수 */
select abs(-10), ceil(10.1), ceil(-10.1), ceil(10), ceil(-10);
select floor(10.1), floor(-10.1), floor(10), floor(-10);

select round(12345.123456, 3), round(12345.123456, -3), round(12345.123456, 0);
select round(12345.123956, 3), round(12995.123456, -3), round(12345.923456, 0);

/*1) 수량이 1234개이고, 단가가 123원 제품의 금액을 계산하고 1000원 미만의 금액은 절삭하여 표시
*/
select (1234*123.4) "원래", floor(1234*123.4) "floor", floor(1234*123.4/1000)*1000;

/*2) 12345원의 예금액에 이자율을 25% 적용하여 지급액을 조회. 단 원단위 이하, 10원단위이하는
절상 */
select (12345*.25), ceil(12345*.25), ceil(12345*.25/10)*10, ceil(12345*.25/100)*100;

/*3) 12311원의 예금액에 이자율을 11% 적용하여 지급액을 조회. 단 원단위이하, 10원 단위이하는
반올림 */
select (12311*.11) "원금", round(12311*.11, -1) "원단위 반올림", round(12311*.11, -2) "10
원단위 반올림";

select truncate(12345.678,1), truncate(12345.678,2), truncate(12345.678,-1),
```

```
truncate(12345.678,-2);

select power(2,10), mod(10,3);

select GREATEST(345,2,2,35,4,68,6,78,3,53,6,-100,7,7,657,4,7654),
       least(345,2,2,35,4,68,6,78,3,53,6,-100,7,7,657,4,7654);
```

10.2 문자관련 함수

함수명	설명
ascii(str)	str의 아스키 코드값
concat(str1, str2, str3, ...)	문자열 연결
insert(str, start, length, newStr)	str 문자열에서 start에서 length길이 만큼의 문자열을 new Str로 바꿈. <pre>select insert('abcdef', 2, 1, '1234567'); → a1234567cdef</pre>
replace(str, oldStr, newStr)	str에서 oldStr의 문자열을 newStr로 수정 <pre>select replace('abcdef', 'bc', '123'); → a123def</pre>
instr(str, findStr)	str에서 findStr을 찾아 위치 반환 <pre>select instr('abcdef', 'cd'); → 3</pre>
left(str, length)	str에서 왼쪽부터 length 만큼 추출 <pre>select left('abcdef', 3); → abc</pre>
right(str, length)	str에서 오른쪽부터 length 만큼 추출 <pre>select right('abcdef', 3); → def</pre>
mid(str, start, length) or substring(str, start, length)	str에서 start위치에서 length 만큼 추출 <pre>select mid('abcdef', 2,3), substring('abcdef', 2,3); → def def</pre>
ltrim(str), rtrim(str), trim(str)	공백 제거
lcase(str) or lower(str)	모두 소문자로
ucase(str) or upper(str)	모두 대문자로
reverse(str)	str을 반대로 나열
format(숫자, 소수점 자리수)	천단위 소숫점 표시 <pre>select format(1234567.12345,3); → 1,234,567.123</pre>

```
/* 문자 함수 */

select ascii('a'), ascii('A'), ascii('B');
select concat('a','b','c'), 'a'+ 'b', 'a' || 'b';
select insert('abcdef', 2, 1, '123'); /* abcdef문자열의 두번째에서 한개의 문자를 123으로 바꾸어라 */
select insert('abcdef', 2, 5, '123'); /* abcdef문자열의 두번째에서 다섯개의 문자를 123으로
```

```

바꾸어라 */
select replace('abcdef', 'cd', '12'); /* abcdef문자열에서 cd를 12로 바꾸어라 */
select left('abcdef', 3), right('abcdef', 3), mid('abcdef', 2,3);
select ltrim('    abc    '), rtrim('    abc    '), trim('    abc    ');
select lower('agdaDFSDFSfdgdf'), upper('asfdDFDFadfasdf');
select 'abcdef', reverse('abcdef');

select format(12345.6789,2);

/*1) 고객정보중에서 city가 nyc인 자료를 검색, nyc모두 소문자로 변경한 후 */
select * from customers where lower(city) = lower('NyC');

/*2) 금액(payments.amount)에 부가세 10% 부과하고, 부가세와 금액 더한 총액을 계산하고
    금액, 부가세, 총액순으로 조회. 단, 모든 금액엔 천단위 분리기호표시 */

select amount as '금액', format(amount*0.1,2) as '부가세', format(amount*1.1,2) as '총액'
from payments;

SELECT
    format(amount, 2) "금액"
    , format(amount * 0.1, 2) "부가세"
    , format(amount * 1.1, 2) "총액"
FROM payments;

SELECT FORMAT(amount, 2), FORMAT(amount*0.1, 2) AS 'tax', FORMAT(amount*1.1, 2) AS 'total'
FROM payments;

select format (amount, 2) as "금액",
       format ((amount*0.1), 2) as "부가세",
       format (amount+(amount*0.1), 2) as "총액"
from payments;

select format (amount, 2) as '금액',
       format(amount * 0.1, 2) as '부가세',
       format(amount * 1.10, 2) as '총액'
from payments;

```

10.3 논리함수

함수명	설명
if(논리식, 참일때, 거짓일때)	<code>select if(10<4, 't', 'f');</code> → f
ifnull(v1, v2)	v1이 null 이면 v2를 반환, 아니면 v1를 반환 <code>select ifnull(null, '1');</code> → 1 <code>select ifnull('a', '1');</code> → a

```

/* 논리함수 */

select if(10>1, 'T', 'F'), if(10<1, 'T', 'F');
select ifnull(null, 'THIS'), ifnull('v', 'THIS');

/*1) 금액+부가세의 결과가 50만원 이상일때는 총액과 '우수고객'을 출력하고,
    그렇지 않으면 총액만 출력 */
SELECT amount AS "금액", amount*1.1 AS "총액", IF(amount*1.1 > 50000, '우수고객', NULL) AS
"우수고객 여부"
FROM classicmodels.payments;

select format(amount*1.1, 2) as "총액",
       if(amount*1.1>50000, '우수고객', '')
from payments;

select if(amount*1.1 >= 50000, concat(format(amount*1.1,2), "우수고객"),
format(amount*1.1,2))
from payments;

SELECT FORMAT(amount, 2) AS '금액',
FORMAT(amount*.1, 2) AS '부가세',
FORMAT(amount*1.1, 2) AS '총액',
IF(amount*1.1>=50000, '우수고객', '') AS "etc"
FROM payments
ORDER BY etc DESC;

```

10.4 집계함수

함수명	설명
count(fn)	null값이 아닌 row갯수
sum(fn)	합계
avg(fn)	평균
max(fn)	최대값
min(fn)	최소값

* fn : field name

```

/* 집계함수 */

select count(customerNumber), format(sum(amount), 0), format(avg(amount),0),
       format(max(amount), 0) , format(min(amount), 0)
from payments;

```

```

desc payments;

/*1) 고객별(customerNumber), 금액의 평균을 조회*/
select customerNumber as '고객 번호', format(avg(amount), 2) as '평균 금액'
from payments group by customerNumber;

SELECT customerNumber AS "CN", FORMAT(AVG(amount), 0) AS "AVG"
FROM classicmodels.payments
GROUP BY customerNumber ORDER BY CN;

/*2) 지불날짜(paymentDate)가 2004인 금액의 총액을 조회 */

select format(sum(if(left(paymentDate, 4)=2004, amount, NULL)), 2) as '2004년 총액' from
payments;

SELECT
    format(sum(amount), 0) "2004년 지불총액"
FROM payments
WHERE LEFT(paymentDate, 4) = '2004';

select format(sum(amount), 0) as "총액" from payments where paymentDate like '2004%';

```

10.5 날짜함수

함수명	설명
now() or sysdate() or current_timestamp()	년월일 시분초
curdate() or current_date() or date(날짜)	년월일
curtime() or current_time() or time(날짜)	시분초
date_add(날짜, interval 기준값)	날짜를 더함.
date_sub(날짜, interval 기준값)	날짜를 뺌
year(날짜), month(날짜), dayofmonth(날짜)	select year(now()), month(now()), dayofmonth(now()); → 2021 1 1
monthname(날짜), dayname(날짜)	
dayofweek(날짜), weekday(날짜)	weekday → 0:월요일, dayofweek → 1:일요일
dayofyear(날짜)	일년중 오늘이 몇일째날
week(날짜)	몇번째 주
date_format(날짜, 형식)	

10.5.1 date functions

- now() : 쿼리 블록에서 최초로 시작되는 시간
- sysdate() : 쿼리가 시작되는 시점의 시간

10.5.2 date_add(start_date, interval expr unit)

- unit : hour, day, month, minute_second, day_hour, second_microsecond, hour_minute,

ex) select date_add(now(), interval 100 day)

10.5.3 date_format(date, format)

format 문자열 : %y, %m, %d, (%h, %H), %i, %s, %W

ex) select date_format(now(), '%y-%m-%d %h:%i:%s (%W)');

```
/*날짜함수 */
/*
now() : 쿼리 블록에서 최초로 시작되는 시간.
sysdate() : 쿼리가 시작되는 시점의 시간.
*/
select now(), sysdate();
select curdate(), current_date(), date(now());
select curtime(), current_time(), time(now());

/* unit : hour, day, month, minute_second, */
select now(), date_add(now(), interval 100 hour);
select now(), date_sub(now(), interval 100 month);

select year(now()), year('2004-01-01');
select month(now()), month('2004-12-01');
select dayofmonth(now()), dayofmonth('2004-12-25');

select monthname(now()), dayname(now());
select dayofweek(now()), weekday(now()); /* dayofweek -> 1:일요일, weekday -> 0:월요일 */
select dayofyear(now()); /* 1년중 오늘이 몇일째인가 */
select week(now()); /* 1년중 이번주가 몇번째 주인가 */

select date_format(now(), '%y년%m월%d일(%W) %H:%i:%s');
```

11 function

```
CREATE FUNCTION 함수명(인자값들)
RETURNS 반환타입
BEGIN
    ...
    RETURN 반환값;
END;
```

함수생성시 1418 오류가 발생하면

```
SET GLOBAL log_bin_trust_function_creators = 1;
```

을 한번 설정해 주고 함수를 생성하면된다.

```
CREATE FUNCTION f1(a int)
RETURNS varchar(50)
BEGIN
    declare rvalue varchar(20);

    return 'kim';

end;
```

```
CREATE FUNCTION f1(a int)
RETURNS varchar(50)
BEGIN
    declare rvalue varchar(20);
    select firstName into rvalue
    from employees where employeeNumber = 1002;

    return rvalue;

end;
```

12 procedure

- dbms별 유사한 문법 구조를 갖고 있으나 호환되지 않는다.
- 컴파일되어서 해당 dbms내부에 저장된다. 따라서 저장형 프로시저라 불려짐.
- 반복해서 컴파일 하려면 먼저 기본 procedure를 drop procedure로 삭제해야 한다.(추후 create or replace명령을 지원할 예정이라 함)

```
CREATE PROCEDURE pro_test1()
BEGIN
  declare cnt int default 0;
  start_rtn : LOOP
    INSERT INTO test(s_name) VALUES('HONG');
    if(cnt>100) then
      leave start_rtn;
    end if;
    set cnt = cnt +1;
  END LOOP;
END
```

12.1 procedure 삭제

```
drop procedure [ if exists ] 프로시저명
```

12.2 parameters

- mode : in, out, inout
 - in : 외부에서 값을 전달 받을 때
 - out : 프로시저 내부에서 처리된 결과를 프로시저 밖에서 사용하고자 할때
 - inout : in + out 기능을 모두 갖음.

ex1) in

```
create procedure in_test(su1 int, su2 int)
BEGIN
    select su1 + su2;
end;
```

ex2) out

```
create procedure out_test1(out str varchar(50))
BEGIN
    set str = 'park';
end;
```

[console에서 테스트]

```
call out_test1(@str);
select @str;
```

ex3) inout

```
create procedure inout_test1(inout su int, inout su2 int)
BEGIN
    set su = su + 100;
    set su2 = su2 + 200;
end;
```

[console에서 테스트]

```
set @su = 10;
set @su2 = 20;
call inout_test1(@su, @su2);

select @su, @su2;
```

12.3 if - then

```
create procedure if_test1()
BEGIN
    declare a int;
    declare b int;

    set a=10;
```

```
set b=20;

if(a>b) THEN
    select a;
end if;

if(a<b) THEN
    select b;

    end if;

end;
```

12.4 if-then-else

```
create procedure if_test2()
BEGIN
    declare a int;
    declare b int;

    set a=10;
    set b=20;

    if(a>b) THEN
        select a;
    else
        select b;

        end if;

end;
```

12.5 if-then-elseif-else

```
create procedure if_test3()
BEGIN
    declare a int;
    declare b int;
    declare c int;
```

```

set a=10;
set b=20;
set c=30;

if(a>b) THEN
    select a;
elseif a>c then
    select b;
ELSE
    select c;
end if;

end;

```

12.6 case

[첫번째 유형]

```

CASE value
    WHEN c1 THEN p1;
    WHEN c2 THEN p2;
    ...
    ELSE
        p3;

END CASE;

```

- c1, c2는 value에 대응되는 상수값
- p1, p2, p3은 처리 내용

ex)

```

create procedure case_test1()
BEGIN
    declare score int default 0;
    declare str varchar(100);

    set score = 80;
    case score
        when 80 then set str = '팔십';
        when 70 then set str = '칠십';

```

```
end case;

select score, str;
end;
```

[두 번째 유형]

```
CASE
  WHEN c1 THEN p1;
  WHEN c2 THEN p2;
  ELSE p3
END CASE;
```

- c1, c2는 범위를 갖는 조건식

ex)

```
create procedure case_test2()
BEGIN
  declare score int default 0;
  declare str varchar(100);

  set score = 80;
  case
    when score >= 90 then set str = 'A';
    when score >= 80 then set str = 'B';
    when score >= 70 then set str = 'C';
    else set str = 'F';

  end case;

  select score, str;
end;
```

12.7 LOOP

- 반복 처리와 같은 경우 client tool에 따라 표시 되지 않을 수 있음.

```
[begin_label:] LOOP
    statement_list
END LOOP [end_label]
```

```
[label]: LOOP
    ...
    -- terminate the loop
    IF condition THEN
        LEAVE [label]; - exit
        ITERATE [label]; - 반복
    END IF;
    ...
END LOOP;
```

- LEAVE label : 지정된 label의 반복문을 벗어나게 함.
- ITERATE label : 지정된 label를 반복하게 함.

ex) 1~10까지 출력

```
create procedure loop_test1()
BEGIN
    declare a int default 0;

    start_a : loop
        set a = a+1;
        if a>10 THEN
            leave start_a;
        end if;
        select a;
    end loop;

end ;
```

12.8 while

```
begin_label: WHILE 조건 DO
    ...
END WHILE end_label;
```

ex)

```
create procedure while_test1()
BEGIN
  declare cnt int default 0;
  declare str varchar(100) default '';
  begin_label : while cnt<10 do
    set cnt=cnt+1;
    set str = concat(str, ' ', cnt);
  end while;
  select str;
end;
```

12.9 cursor

12.9.1 특징

- Read-only
- Non-scrollable
- Asensitive

12.9.2 커서 선언하기

```
DECLARE cursor_name CURSOR FOR select_statement;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1; *커서에 데이터가 없는 경우*
OPEN cursor_name;
FETCH cursor_name INTO variables list;
...
CLOSE cursor_name;
```

ex) 사원번호와 이름 출력하기

```
create procedure cursor_test1()
BEGIN
  declare m_no varchar(30);
  declare m_name varchar(30);
  declare finished int default 0;
```

```
declare cur cursor for select employeeNumber, lastName from employees;
declare CONTINUE HANDLER for not found set finished=1;

open cur;
my_job: LOOP
    fetch cur into m_no, m_name;
    if finished=1 THEN
        leave my_job;
    end if;
    select m_no, m_name;
end loop;
close cur;
end;
```

13 backup | restore

13.1 backup

[Console 창에서 하는 방법]

- mysql 설치 폴더는 path에 등록되어 있어야 함.

데이터베이스 전체 backup

```
mysql>mysqldump -u 사용자이름 -p 암호 원본데이터베이스명 > 복사할 데이터베이스명.sql
```

테이블 backup

```
mysql>mysqldump -u 사용자이름 -p 암호 원본데이터베이스명 테이블명 > 복사할 데이터 테이블명.sql
```

- 데이터베이스명과 테이블명은 띄어쓰기 해야 함.
- 모든 DB를 백업하려면 `mysqldump -all-databases` 옵션을 사용한다.

13.2 restore

[DB restore]

```
mysql -u 사용자이름 -p 암호 복원할 DB명 < 백업 파일명.sql
```

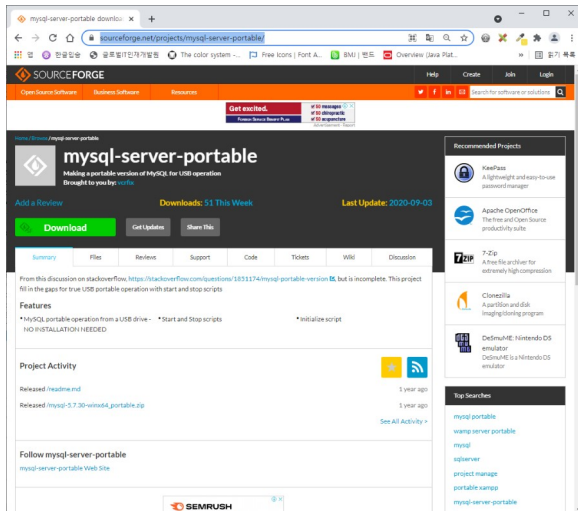
- 모든 DB를 restore 하려면 `mysql -all-databases` 옵션을 사용한다.

14 TIP

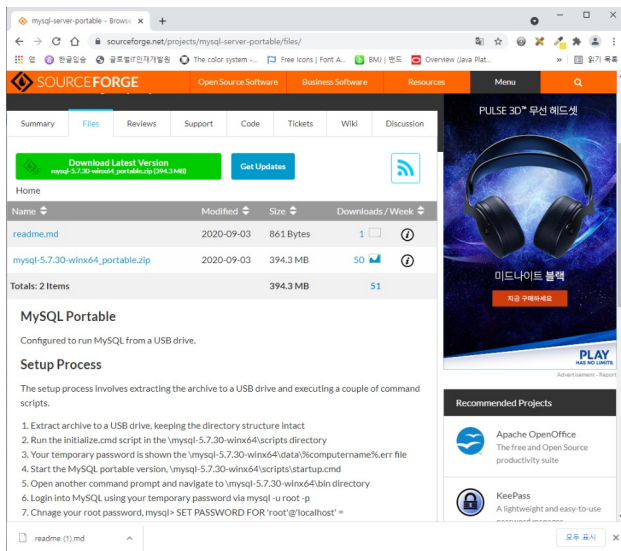
14.1 mysql 5. 7.30 portable 설치

<https://sourceforge.net/projects/mysql-server-portable/>

사이트를 방문하여 파일을 다운로드 받는다.



Files 탭을 클릭하면 간단한 사용설명서를 볼 수 있다.



윈도우의 서비스에는 등록되지 않는다. 윈도우를 재 부팅할 때 마다 따로 실행시켜 주어야 한다. 간혹 mysql 데몬 프로그램이 정상적으로 종료되지 않아 서비스 항목에 남아 있는 경우가 있다. 이 때는 윈도우 작업관리자를 열어 mysqld 서비스를 강제로 종료하고 다시 시작하면 된다.

14.2 MySQL 8.0 portable 설치

14.2.1 portable mysql install

database 의 사용 용도에 따라서 시스템에 설치하여 사용하는 경우도 있지만 이동용 저장 장치에 설치하여 이동하면서 사용해야 하는 경우도 있다. mysql database를 portable 타입으로 설치해 보자.

* 윈도우 서비스에 데몬으로 등록하여 사용할 수도 있다.

step 1.

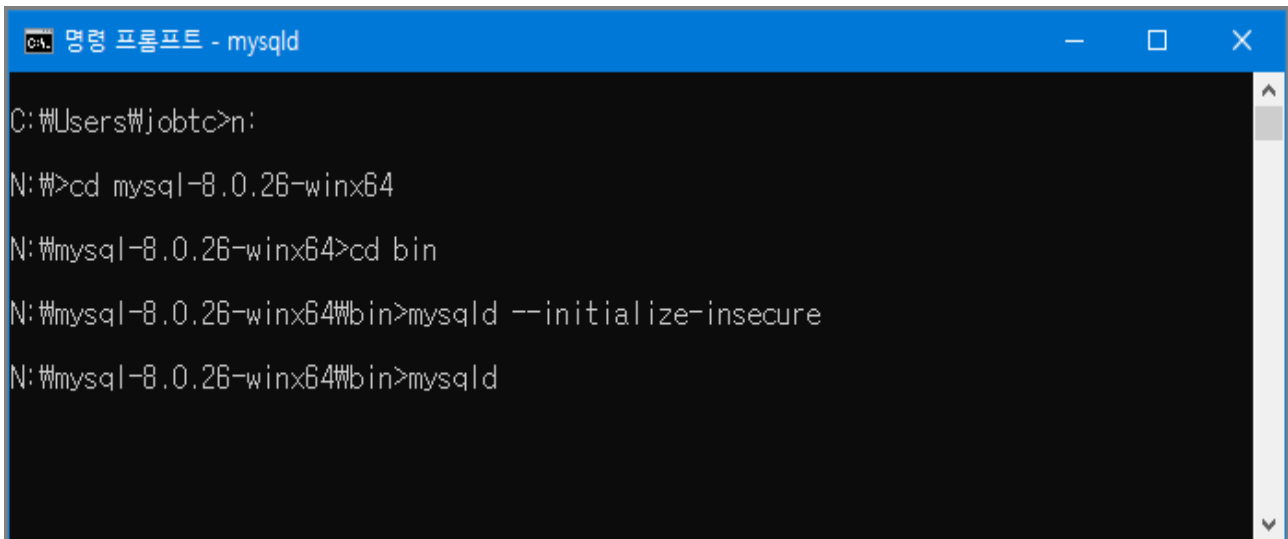
먼저 mysql.com 사이트를 방문하여 zip 형태의 mysql community server를 다운로드 하자. 이 글이 쓰여지는 시점에서는 8.0.26버전이 최신 버전이지만 너무 낮은 버전만 아니면 상관없을 것이다.

다운로드 받은 파일을 드라이브의 최상단 경로에서 압축을 해제한다. 아래는 N 드라이브에 압축을 해제한 경우이다.

```
N:\mysql-8.0.26-winx64>
```

step 2.

윈도우의 콘솔창을 관리자 모드로 열어 아래와 같은 명령을 실행한다. 기본적으로 콘솔창에서 작업하려면 도스 명령을 알아야 한다. 콘솔창을 연 후 압축이 해제된 경로로 이동한 후 아래와 같은 명령을 수행하면 백그라운드로 mysql 데몬이 실행되어 다른 콘솔창이나 DB 클라이언트용 프로그램을 통해 접속할 수 있다.



```
명령 프롬프트 - mysqld
C:\Users\jobtc>n:
N:>cd mysql-8.0.26-winx64
N:\mysql-8.0.26-winx64>cd bin
N:\mysql-8.0.26-winx64\bin>mysqld --initialize-insecure
N:\mysql-8.0.26-winx64\bin>mysqld
```

- mysqld --initialize-insecure 옵션은 root 암호없이 초기화 하는 명령이다. 압축이 해제된 경로에 data 폴더가 추가된다.

- mysqld 는 mysql database의 데몬파일로 백그라운드에서 mysql 서버를 실행시켜준다. 윈도우의 서비스 목록에 등록하지 않고 실행되기 때문에 동일한 콘솔창에서 다른 명령을 실행시킬 수는 없다. 때문에 콘솔창에서 다른 명령을 수행하려면 다른 콘솔창에서 작업해야 한다.

step 3.

다른 콘솔창을 띄워 mysql에 root 계정으로 접속한다.

```
cmd 명령 프롬프트 - mysql -u root
Microsoft Windows [Version 10.0.19042.1348]
(c) Microsoft Corporation. All rights reserved.

C:\Users\jobtc>n:
N:\>cd mysql-8.0.26-winx64
N:\mysql-8.0.26-winx64>cd bin
N:\mysql-8.0.26-winx64\bin>mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.26 MySQL Community Server - GPL

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> _
```

step 4.

root 계정 암호를 수정한다.(물론 반드시 필요한 작업은 아니지만 root는 DB에 관한 모든 권한을 갖고 있는 계정이기 때문에 반드시 암호를 설정해 두어야 한다고 생각하자.)

```
mysql> alter user root@localhost identified by '1111';
Query OK, 0 rows affected (0.01 sec)

mysql>
```

- 계정명 : root@localhost

- 암호 : 1111(임의로 설정하세요)

14.2.2 추가 작업(데몬등록)

윈도우의 서비스에 mysql 데몬을 등록하여 작업할 수도 있다. 서비스에 등록해 두면 컴퓨터가 부팅될 때 자동으로 mysql 서버가 실행된다.

```
N:\mysql-8.0.26-winx64\bin>mysqld -install
N:\mysql-8.0.26-winx64\bin>net start mysql
```

서비스가 시작되면 차후 윈도우를 부팅할 때 자동으로 데몬이 실행된다.

14.2.3 설치 초기에 암호를 지정한 경우

mysqld -initialize 옵션만으로 초기화했다면 N:\mysql-8.0.26-winx64\data 안에 컴퓨터이름.err 파일에 임시 암호가 저장된다.

```
...
2021-10-09T06:08:37.963849Z 6 [Note] [MY-010454] [Server] A temporary password is generated for
root@localhost: qe2oCIPsDc.Q
2021-10-09T06:09:26.003406Z 0 [System] [MY-010116] [Server] N:\mysql-8.0.26-winx64\bin\mysqld.exe
(mysqld 8.0.26) starting as process 13988
...
```

14.2.4 설치정보 제거

```
mysqld --remove
```

sc delete mysql 를 사용하여 mysql 서비스를 삭제할 수 있음.

14.3 한글이 깨질 때

step 1. mysql의 설치 경로에서 my.ini 파일을 아래와 같이 편집하거나 생성한다.

```
[mysqld]
# set basedir to your installation path
basedir="N:/mysql-8.0.26-winx64/"
# set datadir to the location of your data directory
datadir="N:/mysql-8.0.26-winx64/data/"

[client]
default-character-set=utf8

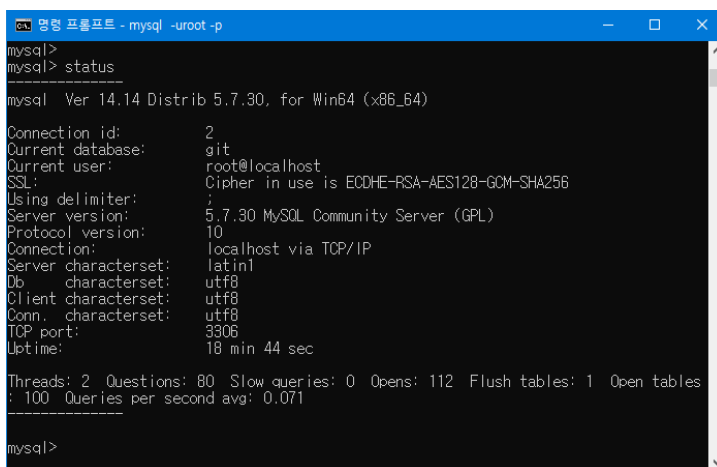
[mysql]
default-character-set=utf8
```

```
[mysqld]
collation-server = utf8_unicode_ci
init-connect='SET NAMES utf8'
character-set-server = utf8
```

step 2. 서버를 중지했다가 다시 가동한다.

step 3. status로 인코딩 상태를 체크해 본다.

```
mysql>status
```



```
mysql>
mysql> status
mysql Ver 14.14 Distrib 5.7.30, for Win64 (x86_64)

Connection id:          2
Current database:       git
Current user:           root@localhost
SSL:                    Cipher in use is ECDHE-RSA-AES128-GCM-SHA256
Using delimiter:        ;
Server version:         5.7.30 MySQL Community Server (GPL)
Protocol version:       10
Connection:             localhost via TCP/IP
Server characterset:    latin1
Db characterset:        utf8
Client characterset:    utf8
Conn. characterset:     utf8
TCP port:               3306
Uptime:                 18 min 44 sec

Threads: 2  Questions: 80  Slow queries: 0  Opens: 112  Flush tables: 1  Open tables:
: 100  Queries per second avg: 0.071

mysql>
```

step 4. 이미 존재하는 테이블에 한글이 입력되지 않을 경우

```
ALTER TABLE 테이블이름 convert to charset utf8;
```

14.4 JDBC에서 한글 깨짐 설정

```
String path = "jdbc:mysql://localhost:3306/git?useUnicode=true&characterEncoding=UTF-8";
```

와 같이 데이터베이스명 뒤에 파라미터를 붙여 준다.

14.5 계층형 데이터 구성하기

답변글이 입력될 때 시퀀스를 증가하여 글의 순서를 정하는 방법을 사용했다.

컬럼	의미	본문글 입력시	답변글 입력시
serial	자신의 순번	증가된 번호	

grp	원본글, 답변글의 묶음	자신의 serial	부모글의 grp
sequence	답변글 순서	0	1) 부모글의 sequence보다 큰 값들을 +1 증가 2) 자신의 글은 부모의 sequence+1값 입력
deep	답변글 단계	0	부모글의 deep보다 +1 증가한 값

* 이 방법의 최대 단점은 답변글을 입력할 때 마다 원본글 보다 큰 sequence값들을 증가시켜야 한다는 점이다.

* last_increment_id() 함수를 사용하여 마지막에 저장된 증가값을 가져올수는 있지만 서버가 재 시작되면 값이 사라질 수 있기 때문에 serial 테이블을 생성하여 최종값을 저장한뒤 가져와 사용하도록 하였다.

[serial값 테이블 생성]

```
create table serial(
  serial int
);

insert into serial values(0);
```

한개의 컬럼과 한줄을 갖는 테이블이다. 초기값으로 0을 저장한 한개의 row을 만들어 주어야 한다.

[serial값 가져오는 함수 정의]

매개변수 flag는 증가된 값을 가져올것인가, 이미 증가된 번호를 가져올것인가를 비교하는 변수이다. i값이 입력되면 번호를 증가한 후 값을 리턴하게 되고, 아무런 값이 없으면 기존 번호를 가져 오도록 작성되었다.

```
CREATE FUNCTION getSerial(flag char) RETURNS int(11)
begin
  declare no int;
  if flag='i' then
    update serial set serial = serial+1;
  end if;
  select serial into no from serial;
  return no;
end;
```

select getSerial('i') : 증가된 값을 가져옴.

select getSerial('') : 기존값을 가져옴.

[게시판 테이블 생성]

```
CREATE TABLE `board`
(
  `serial` int(11) NOT NULL unique,
  `worker` varchar(20) ,
  `pwd` varchar(200) ,
  `subject` varchar(100) ,
  `content` varchar(3000) ,
```

```

`mdate`    date,
`hit`      int(11),
`grp`      int(11),
`seq`      int(11),
`deep`     int(11)

```

```
);
```

[테이블에 게시판 내용 입력예]

```

-- 원본글
insert into board(serial, grp, seq, deep, subject)
values(getSerial('i'), getSerial(''), 0,0, '반갑~');

-- 답변글
insert into board(serial, grp, seq, deep, subject)
values(getSerial('i'), 1, 1,1, '답변 반갑~');

insert into board(serial, grp, seq, deep, subject)
values(getSerial('i'), 1, 2,1, '답변 반갑~');

-- 답변의 답변글
insert into board(serial, grp, seq, deep, subject)
values(getSerial('i'), 1, 3,2, '답변의 답변 반갑~');

commit;

```

[테이블 게시판 조회 예]

```

select serial, grp, seq, deep, hit, worker, mdate,
(select count(serial) from boardatt where pserial= bd.serial) attCnt,
concat(lpad('L' , deep*6*deep, '&nbsp;'), subject) as subject
from board bd
where content like '%%' or subject like '%%'
order by grp desc, seq
limit 0,3

```

* 기타 자세한 게시판 내용은 별도의 문서에서 다룬다.

14.6 root 암호 분실

실행되고 있는 데몬을 종료하고 mysql이 설치된 경로로 이동하여 임의의 파일에 아래와 같은 내용으로 파일을 생성한다.

파일명 : my_pwd.ini

```
alter user root@localhost identified by '1111';
```

데몬을 시작할 때 위에서 만든 my_pwd.ini을 초기파일로 선택하여 실행한다.

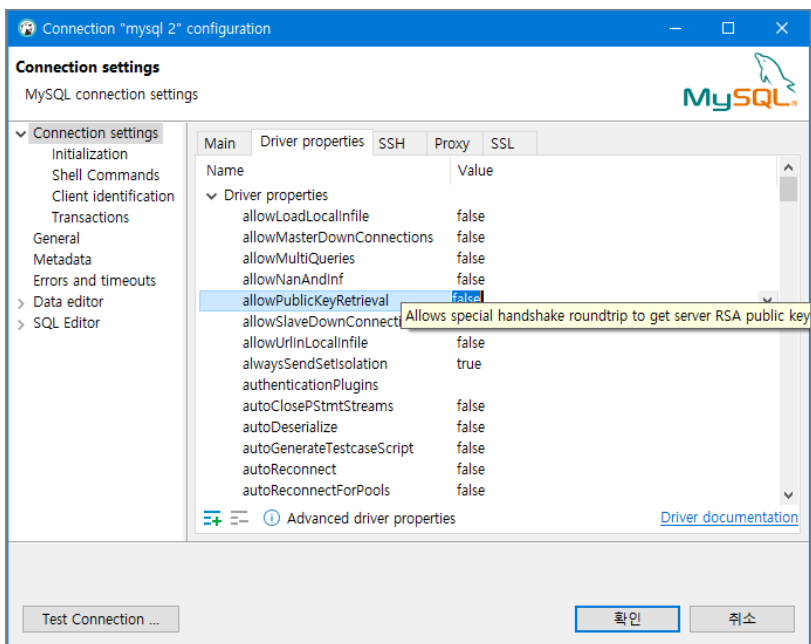
```
prompt > mysqld -init-file=my_pwd.ini
```

```
prompt> mysqld
```

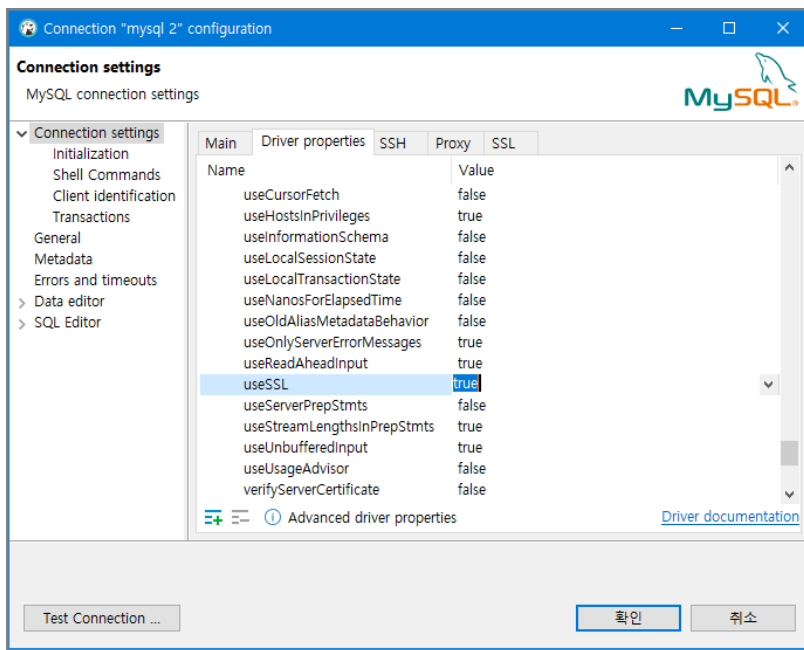
14.7 DBeaver 사용시 오류

14.7.1 db 연결시 오류 발생

콘솔 모드에서는 정상적으로 연결되나 DBeaver에서 연결 오류가 발생하는 경우에는 connections 목록에서 마우스 우클릭후 아래와 같이 정보를 수정한다.



allowPublicKeyRetrieval 정보를 true로 변경한다.



useSSL의 정보를 **false**로 수정한다.

수정된 내용을 저장한 후 다시 연결해 본다.