

DataBase & MySQL

▼ 데이터베이스 기초 개념

데이터베이스 기초 개념

데이터베이스 : 방대한 기능(데이터 관련해서 일어날 수 있는 일들이 많음)을 가지고 있는 정보 도구

데이터베이스의 핵심 : 데이터베이스의 데이터를 어떻게 입력하고 출력하는가 ?

Input -

Create, **U**pdate(수정), **D**elete

Output -

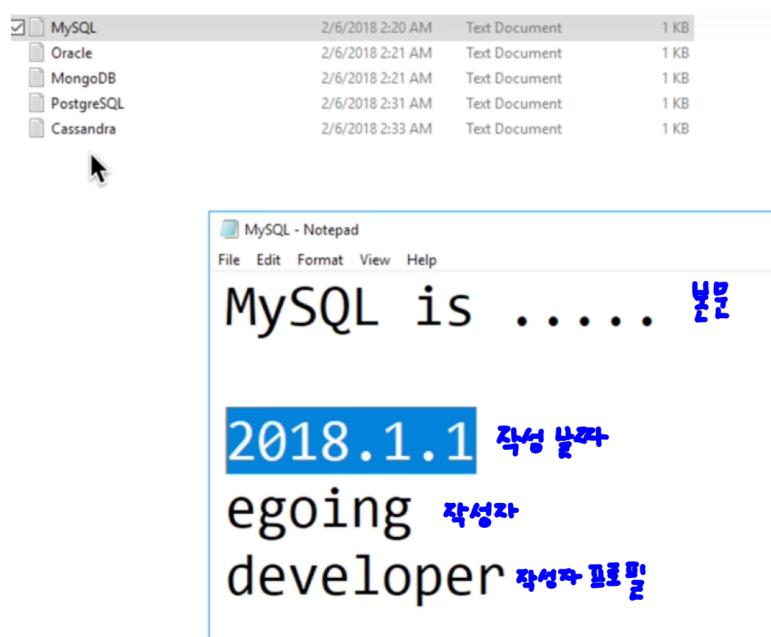
Read

∴ CRUD

→ CRUD가 본질이다. 그 외의 복잡한 기능들은 CRUD를 보좌하는 부가적인 기능이다.

file : 가장 간단한 데이터를 저장하는 방법

file이 어떻게 데이터베이스화가 될까?



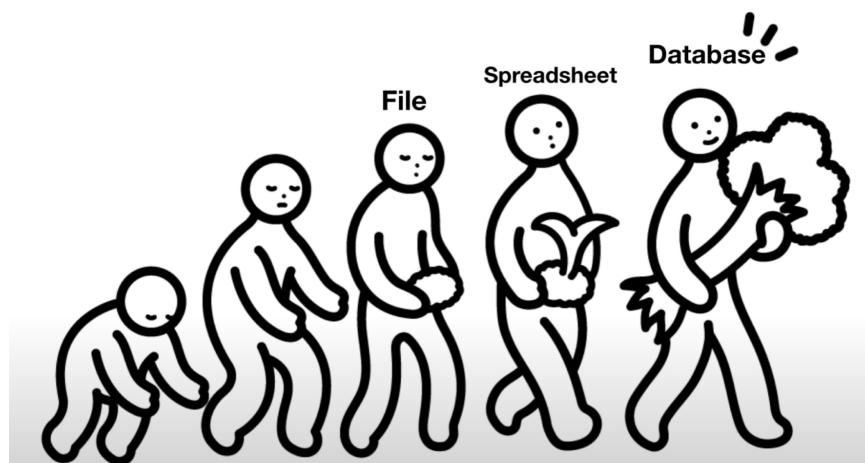
위 사진과 같이 Text 형식으로 저장한다면, 날짜 순이나 작성자에 따라 정렬하려면 어렵다. 그리고 아래의 부가 내용 없이 본문만 보려고 하는 것은 불가능하다.

→ 이런 경우 스프레드시트(excel)를 이용하면 해결이 가능하다. (구조적 데이터 저장)

	A	B	C	D	E	F	G
1	id	title	description	created	author	profile	
2	1	MySQL	MySQL is ...	01/01/18	egoing	developer	
3	2	Oracle	Oracle is ...	01/03/18	egoing	developer	
4	3	MongoDB	MongoDB is ...	01/10/18	duru	data engineer	
5	4	PostgreSQL	PostgreSQL is ...	01/12/18	taeho	data engineer, developer	
6	5	Cassandra	Cassandra is ...	01/20/18	egoing	developer	
7							

→ 저자 별, 날짜 순 등으로 정렬 가능

∴ 메모장이 아닌 스프레드시트에 저장(구조적 데이터 저장)을 하니 (= 정리 정돈을 하니), 데이터 가공이 훨씬 더 쉬워졌다.



스프레드시트는 데이터베이스는 아니고 파일에서 데이터베이스로 가는 길에 있는 것이라고 볼 수 있다. 이는 넓게 본다면 데이터베이스의 특성을 가지고 있다.

스프레드시트와 데이터베이스의 차이점은 뭘까?

데이터베이스는 컴퓨터 언어를 이용해서 데이터를 추가하고 수정하고 삭제하고 읽을 수 있다. 이의 중요한 장점은 자동화(사람이 일일이 작성하지 않고도 어떠한 조건에 따라서 자동으로 데이터를 수정하고 삭제하고 읽을 수 있음.)가 가능한 것이다.

▼ MySQL

MySQL - 1. 소개

인류는 오래전부터 컴퓨터를 이용해서 정보를 관리하고 싶었다. 그래서 파일이라고 하는 위대한 정보 관리 도구를 고안했다. 파일은 사용하기 쉽고 어디에나 있으면서 전송하

기 편하기 때문에 지금도 사용되고 있고, 미래에도 사용될 것이다. 심지어 데이터베이스도 그 정보를 파일에 저장한다.

정보가 폭발적으로 증가하고 다양해지면서 파일만으로는 정보를 효과적으로 입력, 저장, 출력하는 것이 어려워졌다. 그래서 데이터를 정리정돈을 하여 필요할 때 쉽게 꺼내고 싶어졌다.

1960년부터 파일의 한계를 극복하기 위한 시도가 본격적으로 시작되었다. 큰 기업과 천재적인 엔지니어들은 누구나 쉽게 데이터를 정리정돈 할 수 있는 전문적인 소프트웨어를 고안하기 시작하였다. 이러한 맥락에서 만들어진 소프트웨어들을 데이터베이스라고 부르기 시작하였다.

1970년 IBM에서 Relation database(관계형 데이터베이스)라는 새로운 데이터베이스를 고안하였다. 관계형 데이터베이스를 이용하면, 데이터를 표의 형태로 정리정돈 할 수 있고 정렬, 검색과 같은 작업을 빠르고 편리하고 안전하게 할 수 있다.

1994년부터 스웨덴에서 개발하기 시작한 MySQL은 무료이고 오픈 소스이면서 관계형 데이터베이스의 주요한 기능을 대부분 갖추고 있는 준수한 관계형 데이터베이스 시스템이다. 웹이 폭발적으로 성장하면서 웹 개발자들은 웹 페이지를 통해 표현할 정보를 저장할 데이터베이스를 찾게 되었다. 무료이며 오픈소스였던 MySQL은 웹 개발자에게는 매우 좋은 대안이였다.

MySQL - 2. 데이터베이스의 목적

“데이터베이스는 무엇이고 왜 쓰는가?”를 스프레드시트와 관계형 데이터베이스를 비교해서 살펴보자.

A1:F1	A	B	C	D	E	F
1	id	title	description	created	author	profile
2	1	MySQL	My SQL is ...	2018-01-10	egoing	developer
3	2	ORACLE	Oracle is ...	2018-01-15	egoing	developer
4	3	SQL Server	SQL Server is ...	2018-01-18	duru	database administrator
5	4	PostgreSQL	Postgre SQL is ...	2018-01-20	taeho	data scientist, developer
6	5	MongoDB	MongoDB is ...	2018-01-30	egoing	developer
7						
8						
9						


```
mysql> SELECT * FROM topic;
```

id	title	description	created	author	profile
1	MySQL	My SQL is ...	2018-01-01 00:00:00	egoing	developer
2	ORACLE	Oracle is ...	2018-01-15 00:00:00	egoing	developer
3	SQL Server	SQL Server is ...	2018-01-18 00:00:00	duru	database administrator
4	PostgreSQL	Postgre SQL is ...	2018-01-20 00:00:00	taeho	data scientist, developer
5	MongoDB	MongoDB is ...	2018-01-30 00:00:00	egoing	developer

5 rows in set (0.00 sec)

스프레드시트와 관계형 데이터베이스의 공통점 :

- 관계형 데이터베이스는 스트레드시트와 마찬가지로 데이터가 표의 형태로 기록된다. (관계형 데이터베이스의 중요한 특징)
- 스트레드시트와 관계형 데이터베이스는 기능이 거의 비슷하다.
 - 예를 들어, 엑셀에서 author의 이름이 egoing인 사람만 출력하려면 클릭을 통해 author이 egoing인 행만 출력하게 할 수 있다.
MySQL에서 “SELECT * FROM topic WHERE author = ‘egoing’;”(topic 테이블에서 author이 eging인 데이터만을 select해서 가져와라)을 입력하면, author이 egoing인 행만 출력하게 할 수 있다.

- 예를 들어, 엑셀에서 id값을 큰 숫자부터 나오게 하려면 클릭을 통해 정렬 순서를 바꿔주면 된다.
MySQL에서는 "SELECT * FROM topic WHERE author = 'egoing' ORDER BY id DESC;"을 입력하면 된다.

스프레드시트와 관계형 데이터베이스의 차이점 :

- 스프레드시트는 클릭을 통해 데이터를 조작한다.
하지만 관계형 데이터베이스는 SQL이라고 하는 컴퓨터 언어를 이용해서 데이터를 제어할 수 있다. 즉, 코드를 통해 데이터를 조작할 수 있다. → 이런 특성을 이용해 데이터베이스에 저장된 데이터를 웹, 앱을 통해서 공유할 수 있고, 인공지능을 통해서 분석할 수 있다.

MySQL - 3. MySQL 설치

1. MySQL 설치

```
$ brew install mysql
```

2. MySQL 설정

```
$ mysql.server start
```

위 명령어로 MySQL을 시작한다.

```
$ mysql_secure_installation
```

위 명령어로 기본 설정을 시작한다.

```
$ brew services start mysql
```

기본 설정을 마친 뒤 위 명령어로 MySQL server가 재부팅과 상관없이 켜져있을 수 있도록 brew services를 이용하여 서버를 켜둔다.

3. MySQL 사용

```
$ mysql -u root -p
```

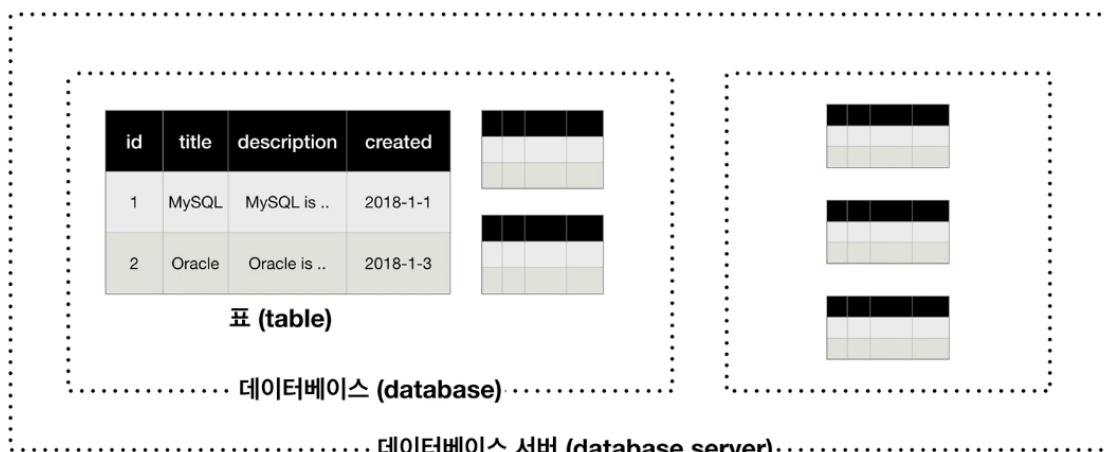
위 명령어 입력 후 루트 비밀번호를 입력하여 MySQL을 사용할 수 있게 된다.
(root 비밀번호 : min93355729@)

MySQL - 4. MySQL의 구조

MySQL은 3개의 구성 요소가 있다. (표, 데이터베이스(스키마), 데이터베이스 서버)

우리가 데이터를 기록하는 곳은 “표”이다. 관계형 데이터베이스(MySQL)에서는 스프레드시트처럼 데이터가 표의 형태로 기록된다.

예를 들어, 어떤 웹 사이트를 운영하는데 사용되는 데이터를 데이터베이스에 저장한다면 글을 저장하는 표, 댓글을 저장하는 표, 회원 정보를 저장하는 표 등 여러 표들이 생겨 날 것이다. 이렇게 표가 늘어나면 많아진 표들을 정리정돈을 해야한다.



그래서 MySQL에서는 “데이터베이스”라는 서로 연관된 표들을 그룹핑해서 연관되어있지 않은 표들과 분리하는데 사용되는 파일의 폴더 같은 것이 있다. 근데 우리가 배우는 것 자체가 데이터베이스인데 표들을 그룹핑하는 것을 데이터베이스라고 하니까 말이 너무 헷갈린다. 그래서 MySQL에서는 데이터베이스라고 하는 표현과 함께 “스키마”라는 표현을 같이 쓴다. “스키마”는 표들을 서로 그룹핑할 때 쓰는 일종의 폴더이다. 즉, 스키마는 서로 연관된 데이터들을 그룹핑한다.

이러한 스키마가 많아지면 표와 마찬가지로 어딘가에 저장이 되어야한다. 이것이 바로 “데이터베이스 서버”이다.

우리가 MySQL을 설치한 행위는 데이터베이스 서버라는 프로그램을 설치한 것이고, 우리는 이제 그 프로그램이 가지고 있는 기능을 활용해서 데이터와 관련된 여러가지 작업을 할 것이다.

MySQL - 5. 서버 접속

데이터베이스를 사용할 때 여러가지 효용이 있다.

그 중 첫번째는 보안이다. 운영체제가 뚫린다면 내부에 있는 파일은 다 뚫린다. 반면에 데이터베이스는 자체적인 보안 체계를 가지고 있기 때문에 좀 더 안전하게 데이터를 보관할 수 있다는 장점이 있다.

이 외에도 권한 기능이라는 것이 있다. 이 기능은 MySQL에 여러 user를 등록한 후 권한을 주는 기능이다. 예를 들어, minsung이라는 사람은 모든 테이블과 모든 스키마에 대해서 읽기, 쓰기, 수정, 삭제를 할 수 있고, jungmin이라는 사람은 A 스키마의 총 3개의 테이블(표) 중 1개의 테이블만 제어할 수 있고, 또 다른 사람은 A 스키마의 총 3개의 테이블 중 1개의 테이블만 읽을 수 있게 차등적으로 권한을 줄 수 있는 직접 구현하기는 힘든 기능을 자체적으로 가지고 있다.

```
./mysql -u root -p
```

-u root는 root 사용자로 접속하겠다라는 뜻이고, 만약 minsung 사용자로 접속하려면 -u minsung이라고 하면 된다. 사용하려면 기본 user가 있어야 할텐데 root는 기본 유저이며 관리자이기에 모든 권한이 주어진다. 우리가 실무에서 개발을 할 때 root의 권한으로 데이터베이스를 다루는 것은 위험하기 때문에 별도의 사용자를 만들어 평소에는 별도의 사용자로 작업을 하다가 중요한 작업이 있을때 root로 작업을 하는 것이 권장된다.

-p이라고 하면 MySQL이 비밀번호를 물어볼것이고 비밀번호를 치면 된다.

```
[minsung@iminsseocBookAir ~ $ mysql -u root -p
[Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.3.0 Homebrew

Copyright (c) 2000, 2024, 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>
```

위의 명령어를 입력한 뒤 비밀번호를 쳤다면 데이터베이스 서버에 접속한 것이다. 이제 스키마에 접속하여 보자.

MySQL - 6. 스키마의 사용

데이터베이스 서버에 접속하였으니 스키마를 만들어서 표를 만들 준비를 하자.

```
CREATE DATABASE opentutorials;
```

위 명령어로 opentutorials이라는 데이터베이스(스키마)를 만들 수 있다.

```
[mysql> CREATE DATABASE opentutorials;
Query OK, 1 row affected (0.06 sec)
```

```
DROP DATABASE opentutorials;
```

위 명령어로 opentutorials이라는 데이터베이스(스키마)를 삭제 할 수 있다.

```
[mysql> DROP DATABASE opentutorials;
Query OK, 0 rows affected (0.02 sec)
```

```
SHOW DATABASES;
```

위 명령어로 만들어진 데이터베이스들을 확인 할 수 있다.

```
[mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| opentutorials |
| performance_schema |
| sys |
+-----+
5 rows in set (0.02 sec)
```

opentutorials이라는 데이터베이스를 생성했으니 이제 이 안에 테이블을 만들어보자. opentutorials이라는 데이터베이스를 사용하려면 MySQL에게 이 데이터베이스를 사용한다고 알려줘야한다.

```
USE opentutorials;
```

위 명령어로 opentutorials이라는 데이터베이스를 사용할 수 있다. MySQL은 위 명령어를 입력한 뒤 부터 내리는 명령을 opentutorials이라는 데이터베이스에 있는 테이블을 대상으로 명령을 실행하게 된다.

```
[mysql> USE opentutorials;
Database changed
```

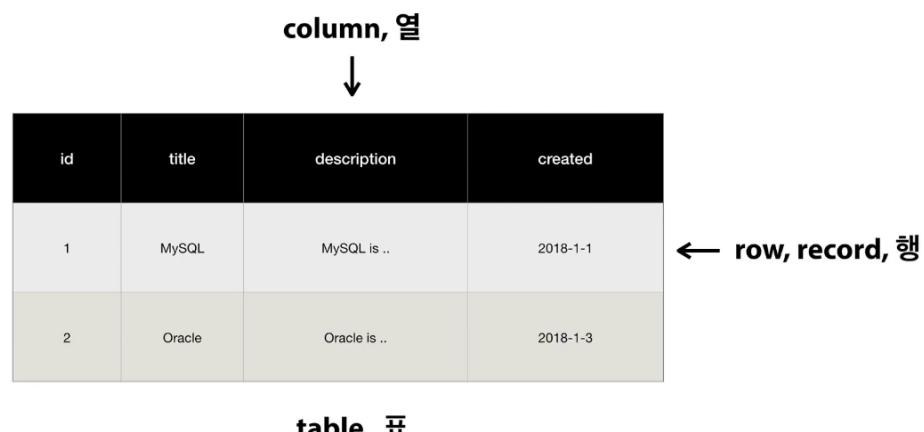
MySQL - 7. SQL과 테이블의 구조

본격적으로 테이블을 다루기 전에 기본 개념을 살펴보자.

SQL은 Structured Query Language의 줄임말로 Structured 구조화 된, Query 데이터베이스에게 요청(질의)하는, Language 데이터베이스와 User 둘다 이해하는 언어이다.

SQL의 2가지 특징은

1. 어떤 컴퓨터 언어보다 쉽다.
2. 중요하다. (SQL은 관계형 데이터베이스 카테고리에 속하는 제품들이 공통적으로 데이터베이스 서버를 제어할 때 사용하는 언어라 압도적인 다수의 데이터베이스 시스템이 SQL을 통해서 동작하므로 중요하다.)
이다.



table(표)의 구조를 살펴보면, x축은 row(행)이고 y축은 column(열)이라고 한다.
(위 사진에서는 행 2개 열 4개이다.) 데이터베이스에서 column은 데이터의 타입(구조)
이라고 하고, row는 데이터 하나(자체)라고 한다.

MySQL - 8. 테이블의 생성

이제 테이블을 생성해보자. 테이블을 생성하기 위해서는 테이블에 각각의 column 별로
column의 이름, 순서 등을 정의하고 데이터를 넣어야한다.

```
CREATE TABLE <테이블 명> (
    <column 이름> <데이터 타입> (<길이>)
    <column 이름> <데이터 타입> (<길이>)
    <column 이름> <데이터 타입> (<길이>)
    ...
    PRIMARY KEY(<첫번째 column 이름>)
);
```

위 코드를 이용하여 테이블을 만들 수 있다. 우리가 개발을 하면서 데이터를 입력하고 출력하는 것은 많이 하겠지만 테이블을 만들 일은 거의 없다. 왜냐하면 소프트웨어를 사용하면 테이블을 아주 편리하고 직관적으로 만들 수 있기 때문이다.

하지만 지금 우리는 SQL을 배우는 과정에 있고, 왜 데이터베이스가 데이터의 양이 많아지고 데이터가 다양해질때 장점이 되는지를 알아보는 것도 중요하기 때문에 테이블을 만드는 코드를 하나하나 짜보면서 데이터베이스의 여러 측면을 살펴보자.

A	B	C	D	E	F
id	title	description	created	author	profile
1	MySQL	My SQL is	2018-01-10	egoing	developer
2	ORACLE	Oracle is .	2018-01-15	egoing	developer
3	SQL Server	SQL Serv	2018-01-18	duru	database administrator
4	PostgreSQL	Postgre S	2018-01-20	taeho	data scientist, developer
5	MongoDB	MongoDB	2018-01-30	egoing	developer

위와 같은 테이블(표)을 만들기 위해서는 아래의 코드를 작성하면 된다.

```
CREATE TABLE topic (
    id INT(11) NOT NULL AUTO_INCREMENT,
    title VARCHAR(100) NOT NULL,
    description TEXT NULL,
    created DATETIME NOT NULL,
    author VARCHAR(15) NULL,
```

```
profile VARCHAR(200) NULL,  
PRIMARY KEY(id)  
);
```

코드를 하나하나 살펴보자.

```
CREATE TABLE topic (
```

위 코드는 topic이라는 테이블을 만들자라는 뜻이다. 이제 column을 만들자.

```
    id INT(11) NOT NULL AUTO_INCREMENT,
```

위 코드는 id라는 column을 만들자라는 뜻이다. 이는 INT 타입이고 길이는 11이며 값이 없는 것을 허용하지 않고 column에 다음 id를 추가할 때 자동으로 id 값이 1씩 증가한다.

스프레드시트와 MySQL의 아주 중요한 차이는 스트레드시트는 column에 어떤 형태의 데이터도 넣을 수 있지만 MySQL은 입력 받는 데이터 타입을 강제할 수 있다.

데이터가 많지 않을 때는 스트레드시트가 편리하다. 하지만 데이터가 엄청 많고, 전문화된 사람이 아니고 아무나 데이터를 입력할 수 있게 된다면 MySQL이 편리하다. (id 항목에는 숫자만 들어가야하는데 문자열을 넣을 수도 있기 때문이다.) MySQL의 column의 데이터에 INT를 지정했는데 문자열이 들어온다면 입력을 거절하고 에러를 발생시킨다. 또 하나의 장점은 id는 숫자로 지정이 되어있으므로 데이터를 꺼낼 때 신경 쓸 필요가 없다.

위 테이블(표)에서 description(본문)은 나중에 작성할 수도 있지만 id 값이나 제목이나 날짜 등은 꼭 있어야된다. 여기서 NOT NULL은 값이 없는 것을 허용하지 않겠다라는 뜻이다. 만약 NULL이면 값이 없는 것을 허용하겠다라는 뜻이다.

우리가 여기서 3번째 행을 삭제한다고 하면 id 값이 3인 행을 삭제하라고 해야한다. id 값이 중복되지 않는 이상 id 값이 3인 행이 삭제 될 것이다. id 값의 중복을 방지하기 위해 AUTO_INCREMENT를 쓴다. AUTO_INCREMENT는 추가하는 id 값이 자동으로 1씩 증가한다라는 뜻이다.

String Datatypes

The following are the **String Datatypes** in MySQL:

Data Type Syntax	Maximum Size	Explanation
CHAR(size)	Maximum size of 255 characters.	Where size is the number of characters to store. Fixed-length strings. Space padded on right to equal size characters.
VARCHAR(size)	Maximum size of 255 characters.	Where size is the number of characters to store. Variable-length string.
TINYTEXT(size)	Maximum size of 255 characters.	Where size is the number of characters to store.
TEXT(size)	Maximum size of 65,535 characters.	Where size is the number of characters to store.
MEDIUMTEXT(size)	Maximum size of 16,777,215 characters.	Where size is the number of characters to store.
LONGTEXT(size)	Maximum size of 4GB or 4,294,967,295 characters.	Where size is the number of characters to store.
BINARY(size)	Maximum size of 255 characters.	Where size is the number of binary characters to store. Fixed-length strings. Space padded on right to equal size characters. (Introduced in MySQL 4.1.2)
VARBINARY(size)	Maximum size of 255 characters.	Where size is the number of characters to store. Variable-length string. (Introduced in MySQL 4.1.2)

```
title VARCHAR(100) NOT NULL,
```

위 코드는 title이라는 column을 만들자라는 뜻이다. 이는 VARCHAR 타입이고 길이는 100이며 값이 없는 것을 허용하지 않는다.

VARCHAR : Variable Charcter, (size) : 글자 수를 size 이상 입력하면 size에 맞는 글자만 입력하고 나머지는 버림

NOT NULL : 공백(값이 없는 것) 허용 X

```
description TEXT NULL,
```

위 코드는 description(본문)이라는 column을 만들자라는 뜻이다. 이는 TEXT 타입이고 값이 없는 것을 허용한다.

본문이다 보니 글자수가 많아야 할 것이다. VARCHAR은 글자 수를 255까지 협용하는데 TEXT는 65,535까지 허용하여 TEXT로 하였다.

NULL : 공백(값이 없는 것) 허용

Date/Time Datatypes

The following are the Date/Time Datatypes in MySQL:

Data Type Syntax	Maximum Size	Explanation
DATE	Values range from '1000-01-01' to '9999-12-31'.	Displayed as 'YYYY-MM-DD'.
DATETIME	Values range from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'.	Displayed as 'YYYY-MM-DD HH:MM:SS'.
TIMESTAMP(<i>m</i>)	Values range from '1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' UTC.	Displayed as 'YYYY-MM-DD HH:MM:SS'.
TIME	Values range from '-838:59:59' to '838:59:59'.	Displayed as 'HH:MM:SS'.
YEAR[(2 4)]	Year value as 2 digits or 4 digits.	Default is 4 digits.

```
created DATETIME NOT NULL,
```

위 코드는 created이라는 column을 만들자라는 뜻이다. 이는 DATETIME 타입이고 값이 없는 것을 허용하지 않는다.

DATETIME은 날짜와 시간 모두 표현 가능하다.

```
author VARCHAR(15) NULL,
```

위 코드는 author이라는 column을 만들자라는 뜻이다. 이는 VARCHAR 타입이고 길이는 15이며 값이 없는 것을 허용한다.

```
profile VARCHAR(200) NULL,
```

위 코드는 profile이라는 column을 만들자라는 뜻이다. 이는 VARCHAR 타입이고 길이는 200이며 값이 없는 것을 허용한다.

```
PRIMARY KEY(id)
```

PRIMARY KEY : 메인 키 (중복되면 안됨)

우리가 생성하는 topic 테이블의 id column이 메인 키이다. (= 순서를 나타내는 키)

```
mysql> CREATE TABLE topic (
    → id INT(11) NOT NULL AUTO_INCREMENT,
    → title VARCHAR(100) NOT NULL,
    → description TEXT NULL,
    → created DATETIME NOT NULL,
    → author VARCHAR(15) NULL,
    → profile VARCHAR(200) NULL,
    → PRIMARY KEY(id)
    → );
Query OK, 0 rows affected, 1 warning (0.11 sec)
```

MySQL - 9. CRUD

Create - 생성하지 않으면 나머지 것들을 할 수 있으므로 중요함.

Read - 읽지 않으면 Create를 할 필요가 없으니 중요함.

Update

Delete

Create와 Read는 데이터베이스라면 반드시 가지고 있는 기능이다.

Update와 Delete는 없을 수도 있다. 예를 들어, 역사의 내용이나 회계의 거래 내역을 수정하거나 지우면 안되기 때문에 필수 기능은 아니다.

∴ 이 중에서 가장 중요한 것은 Create와 Read이다.

MySQL - 10. INSERT

Create(데이터 추가)는 SQL로 어떻게 할까?

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

```

mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| opentutorials |
| performance_schema |
| sys |
+-----+
5 rows in set (0.03 sec)

mysql> USE opentutorials
Database changed

```

데이터베이스 목록을 불러와 원하는 데이터베이스에 진입한다.

```

[mysql> SHOW TABLES;
+-----+
| Tables_in_opentutorials |
+-----+
| topic |
+-----+
1 row in set (0.01 sec)

```

진입한 데이터베이스의 TABLE 목록을 불러온다.

```

[mysql> DESC topic;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | int    | NO   | PRI | NULL    | auto_increment |
| title | varchar(100) | NO   |     | NULL    |           |
| description | text | YES  |     | NULL    |           |
| created | datetime | NO   |     | NULL    |           |
| author | varchar(15) | YES  |     | NULL    |           |
| profile | varchar(200) | YES  |     | NULL    |           |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.02 sec)

```

topic이라는 TABLE의 구조를 불러온다.

topic이라는 TABLE에 데이터를 추가하자.

```

INSERT INTO topic (title, description, created, author,
profile) VALUES('MySQL', 'MySQL is ...', NOW(), 'egoin
g', 'developer');

```

우리가 id 값에 auto_increment 옵션을 주었으므로 id 값을 지정해주지 않으면 auto_increment가 된다. id 값을 auto_increment 되게끔 하자. → id 값을 특별히 언급하지 말자.

NOW() : 현재 시간을 자동으로 불러와주는 함수

```
[mysql]> INSERT INTO topic (title, description, created, author, profile) VALUES('MySQL', 'MySQL is ...', NOW(), 'egoing', 'developer');
Query OK, 1 row affected (0.01 sec)
```

Read(추가한 데이터를 보려면)는 SQL로 어떻게 할까?

```
SELECT * FROM topic;
```

topic TABLE로부터 데이터를 가져온다. 특별한 언급이 없으므로 모든 데이터를 가져온다.

```
[mysql]> SELECT * FROM topic;
+---+---+---+---+---+
| id | title | description | created | author | profile |
+---+---+---+---+---+
| 1 | MySQL | MySQL is ... | 2024-07-02 15:23:11 | egoing | developer |
+---+---+---+---+---+
1 row in set (0.01 sec)
```

topic이라는 TABLE에 나머지 데이터를 추가하자.

```
INSERT INTO topic (title, description, created, author, profile) VALUES('ORACLE', 'ORACLE is ...', NOW(), 'egoing', 'developer');
```

```
INSERT INTO topic (title, description, created, author, profile) VALUES('SQL Server', 'SQL Server is ...', NOW(), 'duru', 'data administrator');
```

```
INSERT INTO topic (title, description, created, author, profile) VALUES('PostgreSQL', 'PostgreSQL is ...', NOW(), 'taeho', 'data scientist, developer');
```

```
INSERT INTO topic (title, description, created, author, profile) VALUES('MongoDB', 'MongoDB is ...', NOW(), 'egoing', 'developer');
```

```
mysql> INSERT INTO topic (title, description, created, author, profile) VALUES('ORACLE', 'ORACLE is ...', NOW(), 'egoing', 'developer');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM topic;
+----+-----+-----+-----+-----+
| id | title | description | created | author | profile |
+----+-----+-----+-----+-----+
| 1 | MySQL | MySQL is ... | 2024-07-02 15:23:11 | egoing | developer |
| 2 | ORACLE | ORACLE is ... | 2024-07-02 15:27:07 | egoing | developer |
+----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> INSERT INTO topic (title, description, created, author, profile) VALUES('SQL Server', 'SQL Server is ...', NOW(), 'duru', 'data administrator');
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO topic (title, description, created, author, profile) VALUES('PostgreSQL', 'PostgreSQL is ...', NOW(), 'taeho', 'data scientist, developer');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO topic (title, description, created, author, profile) VALUES('MongoDB', 'MongoDB is ...', NOW(), 'egoing', 'developer');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM topic;
+----+-----+-----+-----+-----+
| id | title | description | created | author | profile |
+----+-----+-----+-----+-----+
| 1 | MySQL | MySQL is ... | 2024-07-02 15:23:11 | egoing | developer |
| 2 | ORACLE | ORACLE is ... | 2024-07-02 15:27:07 | egoing | developer |
| 3 | SQL Server | SQL Server is ... | 2024-07-02 15:31:30 | duru | data administrator |
| 4 | PostgreSQL | PostgreSQL is ... | 2024-07-02 15:31:32 | taeho | data scientist, developer |
| 5 | MongoDB | MongoDB is ... | 2024-07-02 15:31:34 | egoing | developer |
+----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

MySQL - 11. SELECT

```
SELECT * FROM topic;
```

topic TABLE에 있는 모든 데이터 출력

```
mysql> SELECT * FROM topic;
+----+-----+-----+-----+-----+
| id | title | description | created | author | profile |
+----+-----+-----+-----+-----+
| 1 | MySQL | MySQL is ... | 2024-07-02 15:23:11 | egoing | developer |
| 2 | ORACLE | ORACLE is ... | 2024-07-02 15:27:07 | egoing | developer |
| 3 | SQL Server | SQL Server is ... | 2024-07-02 15:31:30 | duru | data administrator |
| 4 | PostgreSQL | PostgreSQL is ... | 2024-07-02 15:31:32 | taeho | data scientist, developer |
| 5 | MongoDB | MongoDB is ... | 2024-07-02 15:31:34 | egoing | developer |
+----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

```
SELECT id, title, created, author FROM topic;
```

topic TABLE에 있는 id, title, created, author 열만 출력 (나머지 열은 숨김)

```
[mysql]> SELECT id, title, created, author FROM topic;
+---+-----+-----+-----+
| id | title      | created          | author |
+---+-----+-----+-----+
| 1  | MySQL      | 2024-07-02 15:23:11 | egoing |
| 2  | ORACLE     | 2024-07-02 15:27:07 | egoing |
| 3  | SQL Server | 2024-07-02 15:31:30 | duru   |
| 4  | PostgreSQL | 2024-07-02 15:31:32 | taeho  |
| 5  | MongoDB    | 2024-07-02 15:31:34 | egoing |
+---+-----+-----+-----+
5 rows in set (0.00 sec)
```

```
SELECT id, title, created, author FROM topic WHERE author = 'egoing';
```

topic TABLE에 있는 id, title, created, author 열만 출력 + author이 egoing인 행만 출력

```
[mysql]> SELECT id, title, created, author FROM topic WHERE author = 'egoing';
+---+-----+-----+-----+
| id | title      | created          | author |
+---+-----+-----+-----+
| 1  | MySQL      | 2024-07-02 15:23:11 | egoing |
| 2  | ORACLE     | 2024-07-02 15:27:07 | egoing |
| 5  | MongoDB    | 2024-07-02 15:31:34 | egoing |
+---+-----+-----+-----+
3 rows in set (0.00 sec)
```

```
SELECT id, title, created, author FROM topic WHERE author = 'egoing' ORDER BY id DESC;
```

topic TABLE에 있는 id, title, created, author 열만 출력 + author이 egoing인 행만 출력 + id 값을 기준으로 내림차순 정렬

```
[mysql]> SELECT id, title, created, author FROM topic WHERE author = 'egoing' ORDER BY id DESC;
+---+---+-----+---+
| id | title | created | author |
+---+---+-----+---+
| 5 | MongoDB | 2024-07-02 15:31:34 | egoing |
| 2 | ORACLE | 2024-07-02 15:27:07 | egoing |
| 1 | MySQL | 2024-07-02 15:23:11 | egoing |
+---+---+-----+---+
3 rows in set (0.02 sec)
```

엑셀(스프레드시트)의 경우에는 약 65,000개의 데이터만 넣을 수 있다. 하지만 MySQL은 2^{64} -1개의 데이터를 넣을 수 있다.
데이터가 2^{64} -1개가 저장되어있는 상태에서 SELECT * FROM topic;을 해버리면 컴퓨터가 터질 것이다. 그래서 데이터를 가져올때 제약을 걸어야만한다.

```
SELECT id, title, created, author FROM topic WHERE author = 'egoing' ORDER BY id DESC LIMIT 2;
```

topic TABLE에 있는 id, title, created, author 열만 출력 + author이 egoing인 행만 출력 + id 값을 기준으로 내림차순 정렬 + 데이터를 2개만 출력

```
[mysql]> SELECT id, title, created, author FROM topic WHERE author = 'egoing' ORDER BY id DESC LIMIT 2;
+---+---+-----+---+
| id | title | created | author |
+---+---+-----+---+
| 5 | MongoDB | 2024-07-02 15:31:34 | egoing |
| 2 | ORACLE | 2024-07-02 15:27:07 | egoing |
+---+---+-----+---+
2 rows in set (0.05 sec)
```

MySQL - 12. UPDATE

```
UPDATE [LOW_PRIORITY] [IGNORE] table_reference
    SET assignment_list
    [WHERE where_condition] # 모든 행이 UPDATE 되므로 꼭 써줘야 함
    [ORDER BY ...]
    [LIMIT row_count]
```

value:
`{expr | DEFAULT}`

assignment:

```

col_name = value

assignment_list:
    assignment [, assignment] ...

```

```

[mysql]> DESC topic;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id   | int  | NO  | PRI | NULL   | auto_increment |
| title | varchar(100) | NO |     | NULL   |           |
| description | text | YES |     | NULL   |           |
| created | datetime | NO |     | NULL   |           |
| author | varchar(15) | YES |     | NULL   |           |
| profile | varchar(200) | YES |     | NULL   |           |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.07 sec)

[mysql]> SELECT * FROM topic;
+---+-----+-----+-----+-----+-----+
| id | title | description | created | author | profile |
+---+-----+-----+-----+-----+-----+
| 1 | MySQL | MySQL is ... | 2024-07-02 15:23:11 | egoing | developer |
| 2 | ORACLE | ORACLE is ... | 2024-07-02 15:27:07 | egoing | developer |
| 3 | SQL Server | SQL Server is ... | 2024-07-02 15:31:30 | duru | data administrator |
| 4 | PostgreSQL | PostgreSQL is ... | 2024-07-02 15:31:32 | taeho | data scientist, developer |
| 5 | MongoDB | MongoDB is ... | 2024-07-02 15:31:34 | egoing | developer |
+---+-----+-----+-----+-----+-----+
5 rows in set (0.01 sec)

```

id = 2인 ORACLE의 title과 description을 수정하자.

```

UPDATE topic SET title = 'ORacle', description = 'ORacle
is ...' WHERE id = 2;

```

```

[mysql]> UPDATE topic SET title = 'ORacle', description = 'ORacle is ...' WHERE id = 2;
Query OK, 1 row affected (0.02 sec)
Rows matched: 1 Changed: 1 Warnings: 0

[mysql]> SELECT * FROM topic;
+---+-----+-----+-----+-----+-----+
| id | title | description | created | author | profile |
+---+-----+-----+-----+-----+-----+
| 1 | MySQL | MySQL is ... | 2024-07-02 15:23:11 | egoing | developer |
| 2 | ORacle | ORacle is ... | 2024-07-02 15:27:07 | egoing | developer |
| 3 | SQL Server | SQL Server is ... | 2024-07-02 15:31:30 | duru | data administrator |
| 4 | PostgreSQL | PostgreSQL is ... | 2024-07-02 15:31:32 | taeho | data scientist, developer |
| 5 | MongoDB | MongoDB is ... | 2024-07-02 15:31:34 | egoing | developer |
+---+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

```

MySQL - 13. DELETE

```

DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM tbl_name [[AS] tbl_alias]
    [PARTITION (partition_name [, partition_name] ...)]
    [WHERE where_condition] # 모든 행이 DELETE 되므로 꼭 써
    줘야 함
    [ORDER BY ...]
    [LIMIT row_count]

```

id = 5인 MongoDB 행을 삭제하자.

```
DELETE FROM topic WHERE id = 5;
```

```

[mysql]> SELECT * FROM topic;
+---+-----+-----+-----+-----+-----+
| id | title | description | created | author | profile |
+---+-----+-----+-----+-----+-----+
| 1 | MySQL | MySQL is ... | 2024-07-02 15:23:11 | egoing | developer |
| 2 | ORACLE | Oracle is ... | 2024-07-02 15:27:07 | egoing | developer |
| 3 | SQL Server | SQL Server is ... | 2024-07-02 15:31:30 | duru | data administrator |
| 4 | PostgreSQL | PostgreSQL is ... | 2024-07-02 15:31:32 | taeho | data scientist, developer |
| 5 | MongoDB | MongoDB is ... | 2024-07-02 15:31:34 | egoing | developer |
+---+-----+-----+-----+-----+-----+
5 rows in set (0.01 sec)

[mysql]> DELETE FROM topic WHERE id = 5;
Query OK, 1 row affected (0.01 sec)

[mysql]> SELECT * FROM topic;
+---+-----+-----+-----+-----+-----+
| id | title | description | created | author | profile |
+---+-----+-----+-----+-----+-----+
| 1 | MySQL | MySQL is ... | 2024-07-02 15:23:11 | egoing | developer |
| 2 | ORACLE | Oracle is ... | 2024-07-02 15:27:07 | egoing | developer |
| 3 | SQL Server | SQL Server is ... | 2024-07-02 15:31:30 | duru | data administrator |
| 4 | PostgreSQL | PostgreSQL is ... | 2024-07-02 15:31:32 | taeho | data scientist, developer |
+---+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

```

MySQL - 15. 관계형 데이터베이스의 필요성

관계형 데이터베이스를 제어하기 전에 왜 관계형 데이터베이스가 필요한가 알아보자.

	A	B	C	D	E	F
1				topic		
2	id	title	description	created	author	profile
3	1	MySQL	My SQL is ...	2018-01-10	egoring	developer
4	2	ORACLE	Oracle is ...	2018-01-15	egoring	developer
5	3	SQL Server	SQL Server is ▶	2018-01-18	duru	database administrator
6	4	PostgreSQL	PostgreSQL	2018-01-20	taeho	data scientist, developer
7	5	MongoDB	MongoDB is ▶	2018-01-30	egoring	developer
8						

위 표에서 egoing / developer가 중복되고 있는 것을 볼 수 있다.

데이터가 중복되고 있다면 굉장히 중요한 신호이다. → 개선할 것이 있다는 강력한 증거이다.

만약 행이 1억개이고 천만개의 데이터들이 중복된다면, 중복되고 있는 데이터들이 여러 가지 문제를 발생시킨다. (굉장히 복잡하고 용량이 큰 데이터가 천만번 등장한다면 손해가 크다.)

각각의 데이터들의 용량이 크다면(데이터가 많다면) 데이터가 서로 같다는 것을 확신하기 어렵다. 그리고 만약 egoing이라는 이름이 같고 profile도 같은 사람이 있다면 1, 2, 5 번행의 저자가 같은 사람인지 다른 사람인지 알 수 없다. → 지금 스프레드시트 상태로는 이러한 문제를 해결하기 어렵다.

author과 profile을 따로 빼서 author이라는 TABLE을 만들자.

author		
id	name	profile
1	egoing	developer
2	duru	database administrator
3	taeho	data scientist, developer

나머지 표를 이용하여 topic TABLE도 따로 만들자. author/profile을 author TABLE의 식별자인 id값으로 대체하자.

topic				
id	title	description	created	author_id
1	MySQL	MySQL is ...	2018-01-10	1
2	ORACLE	Oracle is ...	2018-01-15	1
3	SQL Server	SQL Server is ↗	2018-01-18	2
4	PostgreSQL	PostgreSQL	2018-01-20	3
5	MongoDB	MongoDB is ↗	2018-01-30	1

이렇게 하면 TABLE이 복잡해지긴 했지만 중복된 데이터는 사라졌다. (author_id를 보면 되기 때문)

author TABLE의 값을 바꾸면 author TABLE을 참고 하고 있는 topic TABLE의 모든 행에서 데이터가 바뀐다. 만약 egoing을 바꾸면 예전 표에서는 1,2,5행을 바꿔줬어야 했는데 새 TABLE에서는 author의 1번 행만 바꾸면 된다. 즉, 유지보수 하기가 훨씬 더 편해졌다.

그리고 예전 표에서는 egoing이라는 사람이 이름이 똑같고 profile이 똑같다면 이 사람들이 같은 사람인지 동명이인인지 알 수 없었는데 새로 만든 표에서는 id로 구분하기 때문에 알 수 있다.

→ 위와 같이 topic이라고 하는 TABLE이 갖고 있는 부분에서 중복을 제거하여 별도의 author라고 하는 TABLE을 만들면 여러가지 장점이 생기는 것을 확인할 수 있다.

하지만 역시 단점도 생긴다. 기존의 TABLE은 author의 이름과 profile이 하나의 표에 다 들어있기 때문에 굉장히 직관적으로(하나의 TABLE만 쳐다보면 됨) 데이터를 볼 수 있었다. 근데 새로 만든(2개로 나눈) TABLE은 데이터를 볼 때 그 데이터에 해당되는 행의 식별자에 해당되는 별도의 표를 열어서 그 표를 비교해가면서 봐야하는 큰 불편함이 있다.(author_id를 통하여 author 표를 열어야한다.)

이러한 불편함은 우리가 데이터를 볼 때 MySQL이 나눠진 표를 합침으로써 해결해준다.

```
SELECT * FROM topic LEFT JOIN author ON topic.author_id  
= author.id;
```

위와 같은 명령어를 치면 MySQL이 알아서 결합하여 보여준다.

MySQL - 16. 테이블 분리하기

```
mysql> SHOW DATABASES;  
+  
| Database |  
+  
| information_schema |  
| mysql |  
| opentutorials |  
| performance_schema |  
| sys |  
+  
5 rows in set (0.08 sec)  
  
mysql> USE opentutorials;  
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_opentutorials |  
+  
| topic |  
+  
1 row in set (0.00 sec)
```

```
mysql> RENAME TABLE topic TO topic_backup;  
Query OK, 0 rows affected (0.04 sec)
```

topic TABLE을 백업을 위해 topic_backup이라고 이름을 바꿔주었다.

```
[mysql]> SELECT * FROM topic_backup;
+---+---+---+---+---+
| id | title      | description          | created           | author   | profile        |
+---+---+---+---+---+
| 1  | MySQL       | MySQL is ...          | 2024-07-02 15:23:11 | egoing   | developer     |
| 2  | ORacle      | ORacle is ...         | 2024-07-02 15:27:07 | egoing   | developer     |
| 3  | SQL Server  | SQL Server is ...    | 2024-07-02 15:31:30 | duru    | data administrator |
| 4  | PostgreSQL  | PostgreSQL is ...   | 2024-07-02 15:31:32 | taeho   | data scientist, developer |
+---+---+---+---+---+
4 rows in set (0.01 sec)
```

```
[mysql]> CREATE TABLE topic (
[   →   id INT(11) NOT NULL AUTO_INCREMENT,
[   →   title VARCHAR(30) NOT NULL,
[   →   description TEXT NULL,
[   →   created DATETIME NOT NULL,
[   →   author_id INT(11) NULL,
[   →   PRIMARY KEY(id)
[   → );
Query OK, 0 rows affected, 2 warnings (0.05 sec)
```

```
[mysql]> DESC topic;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra      |
+-----+-----+-----+-----+-----+-----+
| id         | int        | NO  | PRI | NULL    | auto_increment |
| title      | varchar(30) | NO  |     | NULL    |             |
| description | text       | YES |     | NULL    |             |
| created    | datetime   | NO  |     | NULL    |             |
| author_id  | int        | YES |     | NULL    |             |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.02 sec)
```

```
[mysql]> CREATE TABLE author (
[   →   id INT(11) NOT NULL AUTO_INCREMENT,
[   →   name VARCHAR(20) NOT NULL,
[   →   profile VARCHAR(200) NULL,
[   →   PRIMARY KEY(id)
[   → );
Query OK, 0 rows affected, 1 warning (0.01 sec)
```

```
[mysql]> DESC author;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra      |
+-----+-----+-----+-----+-----+-----+
| id         | int        | NO  | PRI | NULL    | auto_increment |
| name       | varchar(20) | NO  |     | NULL    |             |
| profile    | varchar(200) | YES |     | NULL    |             |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)
```

기존 TABLE을 topic과 author TABLE로 나누었다. 이제 새로운 TABLE에 데이터를 넣자.

```
[mysql> INSERT INTO author (name, profile) VALUES ('egoing', 'developer');
Query OK, 1 row affected (0.07 sec)

[mysql> SELECT * FROM author;
+---+---+---+
| id | name | profile |
+---+---+---+
| 1 | egoing | developer |
+---+---+---+
1 row in set (0.00 sec)
```

```
[mysql> INSERT INTO topic(title, description, created, author_id) VALUES (1, 'MySQL', 'MySQL is ...', '2018-1-1 12:10:11', 1);
ERROR 1136 (21S01): Column count doesn't match value count at row 1
[mysql> INSERT INTO topic(title, description, created, author_id) VALUES ('MySQL', 'MySQL is ...', '2018-1-1 12:10:11', 1);
Query OK, 1 row affected (0.01 sec)

[mysql> SELECT * FROM topic;
+---+---+---+---+
| id | title | description | created | author_id |
+---+---+---+---+
| 1 | MySQL | MySQL is ... | 2018-01-01 12:10:11 | 1 |
+---+---+---+---+
1 row in set (0.00 sec)
```

```
[mysql> INSERT INTO topic VALUES (2, 'Oracle', 'Oracle is ...', '2018-01-03 13:01:10', 1);
Query OK, 1 row affected (0.00 sec)
```

```
[mysql> INSERT INTO author (name, profile) VALUES ('duru', 'data administrator');
Query OK, 1 row affected (0.01 sec)

[mysql> INSERT INTO topic (title, description, created, author_id) VALUES ('SQL Server', 'SQL Server is ...', '2018-01-20 11:01:10', 2);
Query OK, 1 row affected (0.00 sec)

[mysql> INSERT INTO author (name, profile) VALUES ('taeho', 'data scientist, developer');
Query OK, 1 row affected (0.01 sec)

[mysql> INSERT INTO topic (title, description, created, author_id) VALUES ('PostgreSQL', 'PostgreSQL is ...', '2018-01-23 1:3:3', 3);
Query OK, 1 row affected (0.02 sec)

[mysql> INSERT INTO topic (title, description, created, author_id) VALUES ('MongoDB', 'MongoDB is ...', '2018-01-30 12:31:3', 1);
Query OK, 1 row affected (0.00 sec)
```

```

[mysql]> SELECT * FROM author;
+---+-----+-----+
| id | name | profile |
+---+-----+-----+
| 1 | egoing | developer |
| 2 | duru | data administrator |
| 3 | taeho | data scientist, developer |
+---+-----+-----+
3 rows in set (0.00 sec)

[mysql]> SELECT * FROM topic;
+---+-----+-----+-----+-----+
| id | title | description | created | author_id |
+---+-----+-----+-----+-----+
| 1 | MySQL | MySQL is ... | 2018-01-01 12:10:11 | 1 |
| 2 | Oracle | Oracle is ... | 2018-01-03 13:01:10 | 1 |
| 3 | SQL Server | SQL Server is ... | 2018-01-20 11:01:10 | 2 |
| 4 | PostgreSQL | PostgreSQL is ... | 2018-01-23 01:03:03 | 3 |
| 5 | MongoDB | MongoDB is ... | 2018-01-30 12:31:03 | 1 |
+---+-----+-----+-----+-----+
5 rows in set (0.00 sec)

```

TABLE을 분리한 후 모든 데이터를 넣어줬다.

MySQL - 17. JOIN

topic TABLE과 author TABLE을 JOIN 하기 위해서는 결합 고리는 topic TABLE에서는 author_id 값이고 author TABLE에서는 id 값이다.

```

SELECT * FROM topic LEFT JOIN author ON topic.author_id
= author.id;

```

topic TABLE에 있는 모든 행을 출력하는데 그때 author_id 값과 author TABLE의 id 값이 같은 행을 가져와서 topic TABLE에 붙인다.

```

[mysql]> SELECT * FROM topic LEFT JOIN author ON topic.author_id = author.id;
+---+-----+-----+-----+-----+-----+-----+
| id | title | description | created | author_id | id | name | profile |
+---+-----+-----+-----+-----+-----+-----+
| 1 | MySQL | MySQL is ... | 2018-01-01 12:10:11 | 1 | 1 | egoing | developer |
| 2 | Oracle | Oracle is ... | 2018-01-03 13:01:10 | 1 | 1 | egoing | developer |
| 3 | SQL Server | SQL Server is ... | 2018-01-20 11:01:10 | 2 | 2 | duru | data administrator |
| 4 | PostgreSQL | PostgreSQL is ... | 2018-01-23 01:03:03 | 3 | 3 | taeho | data scientist, developer |
| 5 | MongoDB | MongoDB is ... | 2018-01-30 12:31:03 | 1 | 1 | egoing | developer |
+---+-----+-----+-----+-----+-----+-----+
5 rows in set (0.06 sec)

```

위 TABLE에서 author_id와 id는 보기 안좋으니 없애자.

```

SELECT topic.id, title, description, created, name, profile
FROM topic LEFT JOIN author ON topic.author_id = aut

```

```
hor.id;
```

유의할 점 : id가 topic TABLE과 author TABLE에 두 개가 있으니 id를 칠 때 topic.id라고 해야한다. (맨 앞 topic.id 참조)

```
mysql> SELECT topic.id, title, description, created, name, profile FROM topic LEFT JOIN author ON topic.author_id = author.id;
+----+-----+-----+-----+-----+-----+
| id | title | description | created | name | profile |
+----+-----+-----+-----+-----+-----+
| 1 | MySQL | MySQL is ... | 2018-01-01 12:10:11 | egoing | developer |
| 2 | Oracle | Oracle is ... | 2018-01-03 13:01:10 | egoing | developer |
| 3 | SQL Server | SQL Server is ... | 2018-01-20 11:01:10 | duru | data administrator |
| 4 | PostgreSQL | PostgreSQL is ... | 2018-01-23 01:03:03 | taeho | data scientist, developer |
| 5 | MongoDB | MongoDB is ... | 2018-01-30 12:31:03 | egoing | developer |
+----+-----+-----+-----+-----+-----+
5 rows in set (0.02 sec)
```

id가 topic TABLE과 author TABLE에 두 개가 있으니 구분을 위해 TABLE에서의 id를 topic_id라고 하자.

```
SELECT topic.id AS topic_id, title, description, created, name, profile FROM topic LEFT JOIN author ON topic.author_id = author.id;
```

AS 명령어를 사용하면 된다.

```
mysql> SELECT topic.id AS topic_id, title, description, created, name, profile FROM topic LEFT JOIN author ON topic.author_id = author.id;
+----+-----+-----+-----+-----+-----+
| topic_id | title | description | created | name | profile |
+----+-----+-----+-----+-----+-----+
| 1 | MySQL | MySQL is ... | 2018-01-01 12:10:11 | egoing | developer |
| 2 | Oracle | Oracle is ... | 2018-01-03 13:01:10 | egoing | developer |
| 3 | SQL Server | SQL Server is ... | 2018-01-20 11:01:10 | duru | data administrator |
| 4 | PostgreSQL | PostgreSQL is ... | 2018-01-23 01:03:03 | taeho | data scientist, developer |
| 5 | MongoDB | MongoDB is ... | 2018-01-30 12:31:03 | egoing | developer |
+----+-----+-----+-----+-----+-----+
5 rows in set (0.01 sec)
```

author TABLE의 duru(id 2번)의 profile을 data administrator에서 database administrator로 바꾸자.

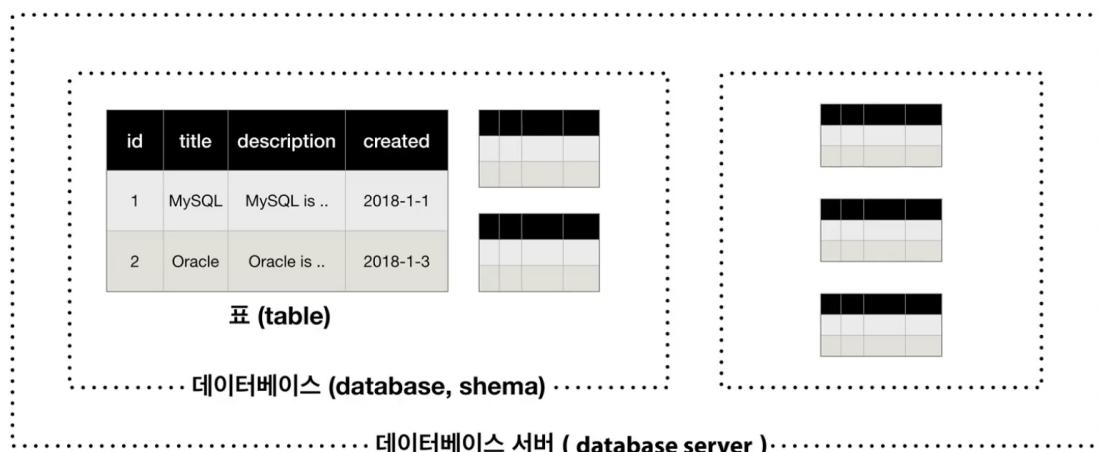
```
[mysql]> UPDATE author SET profile = 'database administrator' WHERE id = 2;
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
mysql> SELECT topic.id AS topic_id, title, description, created, name, profile FROM topic LEFT JOIN author ON topic.author_id = author.id;
+----+-----+-----+-----+-----+-----+
| topic_id | title | description | created | name | profile |
+----+-----+-----+-----+-----+-----+
| 1 | MySQL | MySQL is ... | 2018-01-01 12:10:11 | egoing | developer |
| 2 | Oracle | Oracle is ... | 2018-01-03 13:01:10 | egoing | developer |
| 3 | SQL Server | SQL Server is ... | 2018-01-20 11:01:10 | duru | database administrator |
| 4 | PostgreSQL | PostgreSQL is ... | 2018-01-23 01:03:03 | taeho | data scientist, developer |
| 5 | MongoDB | MongoDB is ... | 2018-01-30 12:31:03 | egoing | developer |
+----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

MySQL - 18. 인터넷과 데이터베이스

정보의 바다라고 하는 인터넷 위에서 데이터베이스가 동작하게 되면 굉장한 효과를 낼 수 있다.

MySQL은 내부적으로 인터넷을 활용하도록 고안된 시스템이다.



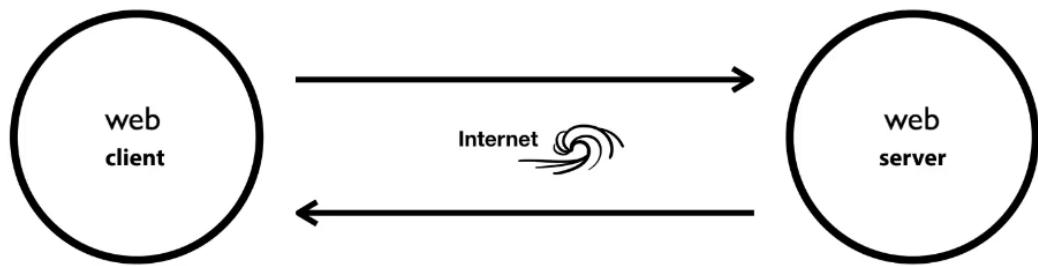
데이터베이스 서버에서 서버라는 것의 의미는 뭘까?

이 얘기를 하기 위해 인터넷에 대한 얘기를 해보자. 인터넷을 사용하기 위해서는 컴퓨터가 최소 2 대가 필요하다. 인터넷의 의미는 각자 흩어져있는 컴퓨터들이 인터넷으로 연결되면서 컴퓨터들간의 사회가 만들어지는 것이다. 인터넷을 통하여 1대의 컴퓨터가 가지고 있는 한계를 초월하게 되었다.

이 2대의 컴퓨터들 사이에서는 어떠한 일이 벌어질까? 1대의 컴퓨터는 다른 컴퓨터에게 정보를 요청한다. 또 한대의 컴퓨터는 요청한 정보를 응답한다.

예를 들어, 웹이 동작하기 위해서는 인터넷이 필요하고 인터넷 위에서 동작하기 때문에 2대의 컴퓨터가 필요하다. 1대의 컴퓨터에는 웹 브라우저가 설치되어 있다. 이 웹 브라우저를 통해 google.com, naver.com 등을 접속하면 웹 브라우저가 설치되어 있는 컴퓨터가 내가 입력한 도메인 네임 주소에 해당되는 어떤 컴퓨터를 찾아간다. 찾아간 컴퓨터가 요청한 컴퓨터에게 요청한 정보를 전송하고 이것을 웹 브라우저에 표시하면 웹에 접속이 된다.

이와 같은 방식으로 인터넷 위에서 동작하는 컴퓨터들은 정보를 요청하는 쪽과 응답하는 쪽으로 나뉜다. 정보를 요청하는 쪽은 클라이언트(고객)이라고 하고, 응답하는 쪽은 서버(사업자, 서비스 제공자)라고 한다.



클라이언트 컴퓨터는 서버 컴퓨터에게 정보를 요청하는데 이 관계가 웹이라면 정보를 요청하는 컴퓨터에 설치되어 있는 프로그램은 웹 브라우저(= 웹 클라이언트)라고 하고 그리고 응답하는 서버 컴퓨터에 설치 되어있는 프로그램은 웹 서버라고 한다.

만약 게임을 한다면 게임 클라이언트 ↔ 게임 서버.

만약 채팅을 한다면 채팅 클라이언트 ↔ 채팅 서버.

이 클라이언트와 서버라는 것은 인터넷을 이해하는 핵심적인 열쇠이다.

우리가 MySQL을 설치하면 MySQL이 데이터베이스 클라이언트와 데이터베이스 서버를 동시에 설치해 준다. 데이터베이스 서버에는 실제로 데이터가 저장이 되고 우리는 데 이터베이스 클라이언트를 통해서 데이터베이스 서버에 접속할 수 있다.

우리가 지금까지 데이터베이스를 다루면서 데이터베이스 서버를 직접 다룬 것처럼 보일 수 있겠지만 사실은 우리는 데이터베이스 서버를 직접 다룰 수 없다. 데이터베이스 서버는 반드시 데이터베이스 클라이언트를 사용해야 한다.

그럼 우리가 사용한 데이터베이스 클라이언트가 무엇이었을까?

```
./mysql -u root -p
```

```

minsung@iminsseocBookAir ~ $ mysql -u root -p
[Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 13
Server version: 8.3.0 Homebrew

Copyright (c) 2000, 2024, 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.

```

위 명령어를 입력했을 때 실행되는 CLI 기반의 프로그램이다. 이 프로그램을 실행하면 "Welcome to the MySQL monitor."이라고 써있는데 이 MySQL monitor가 데이터베이스 클라이언트 중 하나였던 것이고 우리가 MySQL을 설치하면 MySQL을 만든 사람들이 MySQL 서버에 접속할 수 있도록 기본적으로 제공하는 클라이언트가 MySQL monitor이었던 것이다. MySQL monitor은 명령어를 통해서 데이터베이스 서버를 제어하는 프로그램이다.

우리가 곧 MySQL workbench라는 것도 배울 것인데 이것은 GUI 환경에서 엑셀을 다루듯이 데이터베이스를 다룰 수 있는 MySQL 클라이언트이다.

우리가 데이터베이스 서버와 클라이언트에 대해 살펴보았다. 즉, 이 둘의 관계는 데이터베이스 서버에 데이터를 저장하고 전 세계에 있는 수 많은 데이터베이스 클라이언트들이 데이터베이스 서버를 중심으로 해서 데이터를 넣고 빼고 하는 것이 가능한 것이다.
→ 웹이나 앱과 같은 UI를 사용하지 않고도 지금까지 우리가 사용해왔던 MySQL monitor 또는 다음 시간에 살펴볼 MySQL workbench와 같은 데이터베이스 클라이언트를 통해 전 세계에 있는 수 많은 사람들이 하나의 데이터베이스 서버를 이용해서 여러가지 정보를 주고받고, 관리하는 것이 가능한 것이다.

MySQL - 19. MySQL Client

기본 클라이언트 : MySQL monitor

MySQL monitor의 장점 :

1. MySQL server를 설치하면 같이 설치되므로 MySQL server가 있는 곳에는 MySQL monotor가 있다.
2. GUI가 아닌 CLI 기반의 프로그램이라 가볍고 어디서든지 실행할 수 있다.

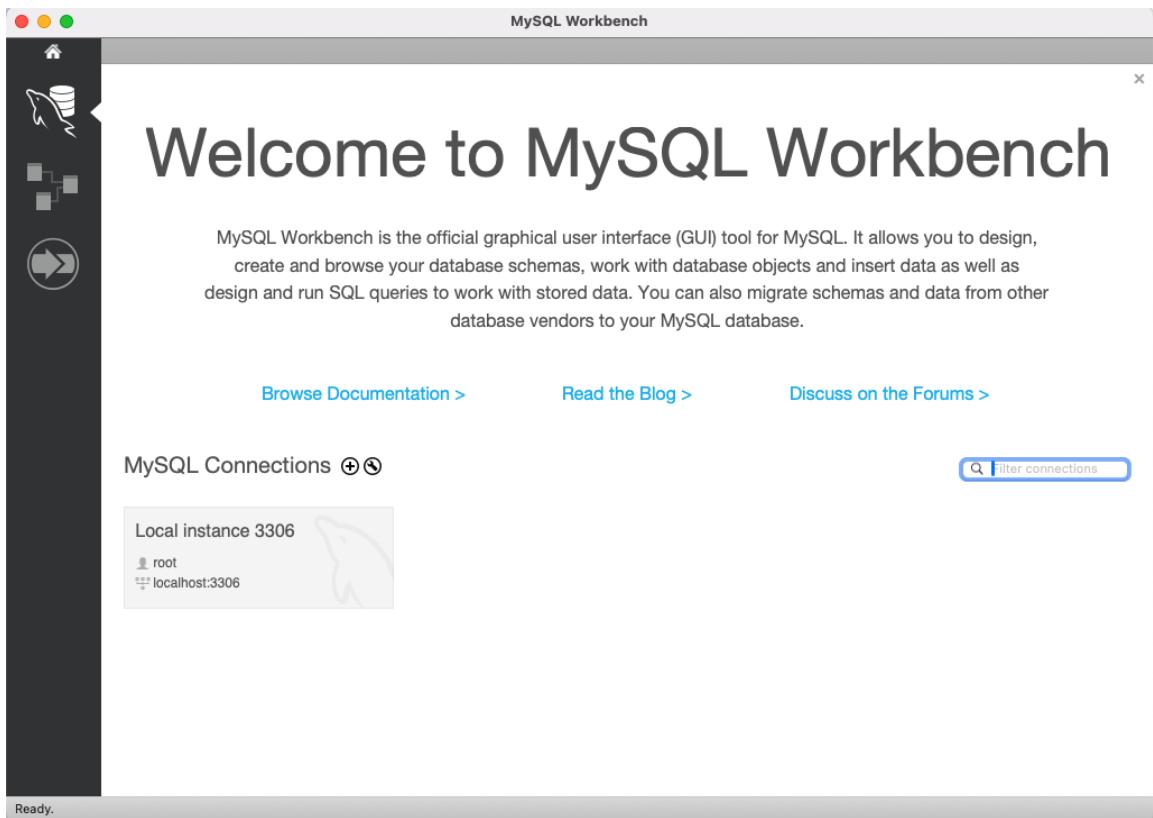
MySQL monitor의 단점 :

1. CLI 기반이라 명령어를 알아야만 한다.

MySQL monitor 외에도 MySQL Workbench, Sequel Pro, HeidiSQL, phpMyAdmin 등 많은 클라이언트가 있다.

MySQL - 20. MySQL Workbench

MySQL Workbench : MySQL에서 공식적으로 제공하는 GUI 기반의 클라이언트



MySQL monitor과 MySQL Workbench를 비교해보자.

MySQL monitor 접속 방법 :

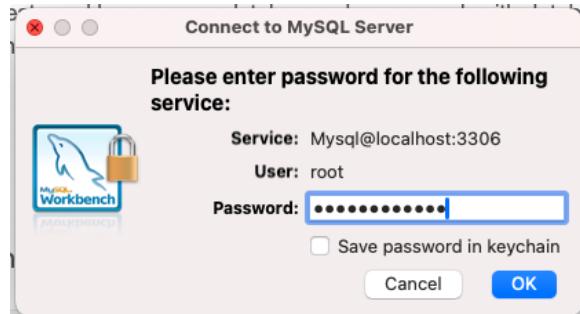
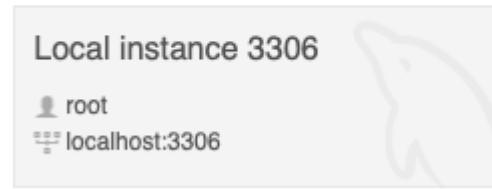
```
./mysql -u root -p -h localhost
```

-u (사용자), -p (비밀번호) -h (호스트)

인터넷을 통해 다른 컴퓨터에 있는 서버에 접속하려고 하면 -h 뒤에 서버에 해당되는 컴퓨터의 주소를 입력해주면 된다. -h를 입력하지 않는다면 MySQL monitor가 설치되어 있는 컴퓨터(내 컴퓨터)의 MySQL 서버에 접속한다.

MySQL Workbench 접속 방법 :

MySQL Connections



MySQL monitor을 쓰던, MySQL Workbench를 쓰던 모든 클라이언트들은 결국에는 SQL(쿼리)를 MySQL 서버에 전송함으로써 데이터베이스 서버를 제어하게 된다.