

# DATABASE\_MySQL

## DB 1. DB 소개 및 본질 알아보기

### 1. DB1 수업 소개.

**file**은 너무 좋지만

성능, 보안, 편의성에 한계

**그래서 DB**

파일의 한계 극복을 위해 고안된 전문화된 소프트웨어가 데이터베이스이다.  
소중한 데이터를 안전, 편리, 빠르게 보관 및 사용 가능

다양한 DB 제품이 존재.

### DB 1.에서는

다양한 DB를 관통하는 가장 본질적인 기능들을 살펴보겠다.  
MySQL, MongoDB에서 사용할 사례.

### 우리에게 필요한건

상상력

사용할 예제를 크고 복잡하고 위험하게 간주해보자.

DB의 복잡성에 대한 공감

LET'S GO

## 2. DB의 본질.

**DB는**

매우 방대한 기능을 가지고 있는 정보 도구

why? 데이터 관련해서 일어날 수 있는 일들이 참으로 많다.

그걸 일일이 다 배워야 할까?

**좋은 방법**

아무리 복잡한 기술도 그 중심에 있는 핵심은 복잡하지 않다. 어렵지도 않다.

**어떤 DB를 만나더라도**

여기서는 데이터를 어떻게 입력하고, 어떻게 출력하는가를 따져보자.  
그러면 우리는 그 DB의 **반**을 안 것이다.

## 입력은?

데이터를 생성, 수정, 삭제

## 출력은?

데이터를 읽기.

## 이 네 가지가

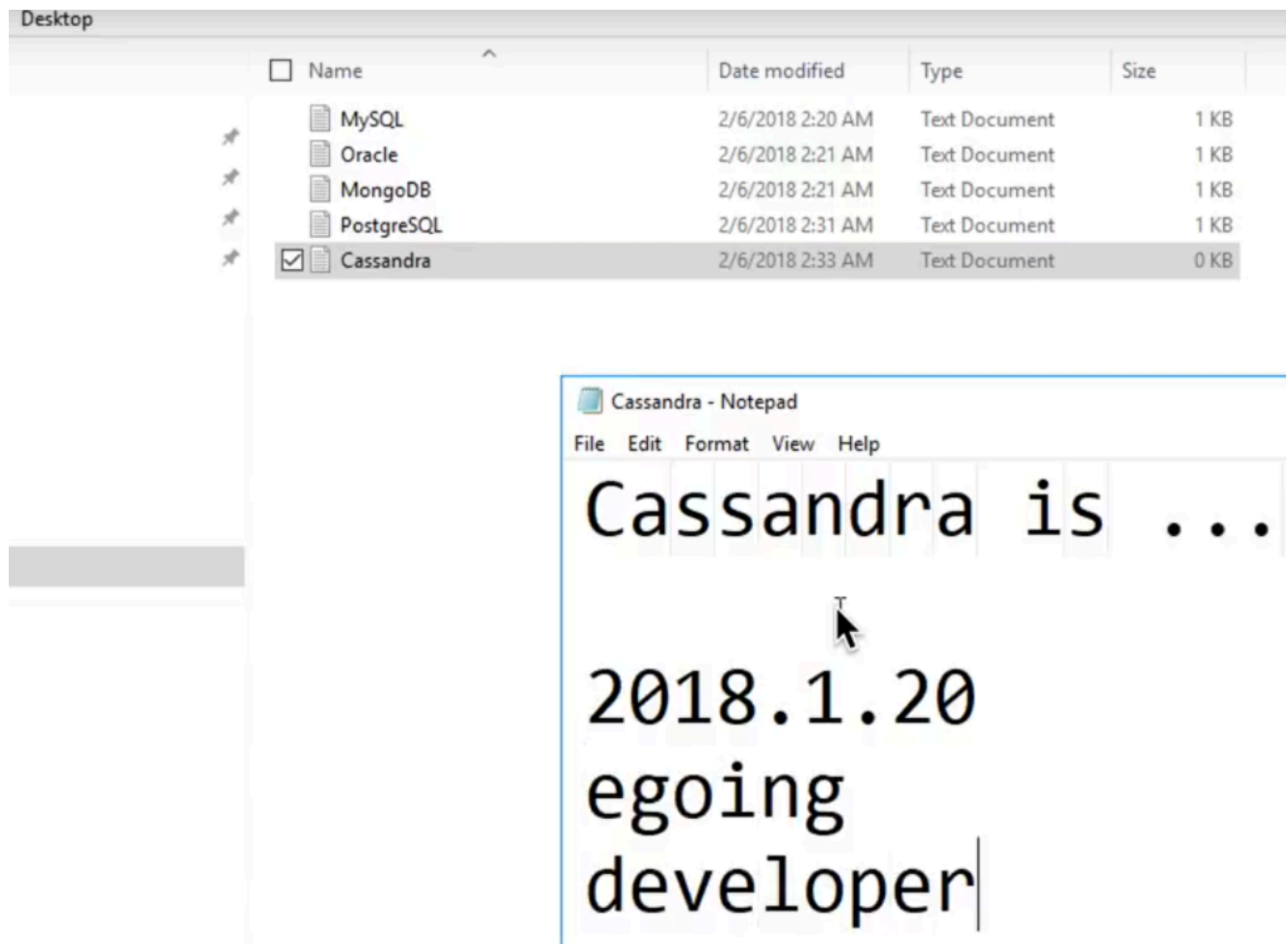
데이터 관련해서 우리에게 필요한 거의 모든 것.

그리고 이것을 **CRUD** 라고 한다.

## 3. FILE vs DATABASE.

### CRUD

file이 어떻게 DB화 되어가는가?



아주 좋아요.  
근데 이게 1억개면?

egoing이 쓴 글만 보고싶다면?

본문에 egoing이 있나 검색?

근데 저자가 아니라 본문 내용에 있는 노이즈까지 나올텐데?

작성 날짜로 정렬하고 싶다면?

반대 정렬을 하고 싶다면?

본문만 보고싶고 나머지 날짜, 저자, 직업 정보는 숨기고 싶다면?

## 다~ 안된다. file은

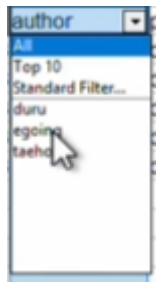
그러면...

## 스프레드시트?

	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							

중요한 효과.

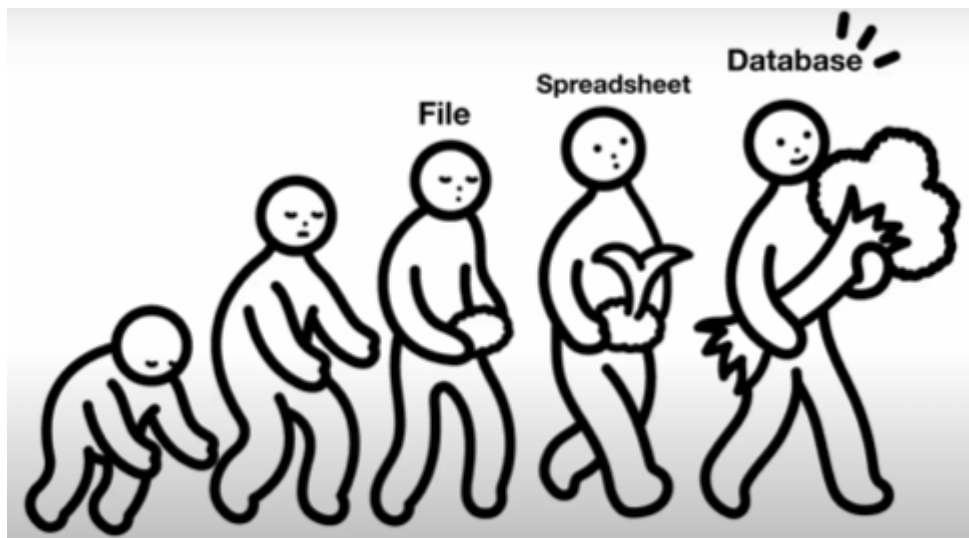
자 1억개 글이 있다고 생각해보자.



원하는 저자만 볼 수 있다!

정렬도 된다!

구조적으로 데이터를 저장했을 때, 가공하는 것이 훨씬 쉬워진다.



Spreadsheet은 File에서 DB로 가는 길목에 있다. 물론 스프레드시트가 DB는 아니다.

근데 그런 특성을 갖고 있다.































## Spreadsheets 와 DB의 차이점은?

프로그래밍적으로, 컴퓨터 언어를 이용해서 데이터를 Create, Read, Update, Delete.  
DB SW들은 다 이러한 점을 가진다.

## 그것의 장점은?

자동화 할 수 있다. 사람이 일일이 하지 않아도 CRUD 할 수 있다.

## 4. 수업을 마치며.

Rank			DBMS	Database Model	Score		
Jul 2025	Jun 2025	Jul 2024			Jul 2025	Jun 2025	Jul 2024
1.	1.	1.	Oracle	Relational, Multi-model 	1217.05	-13.33	-23.31
2.	2.	2.	MySQL	Relational, Multi-model 	940.73	-12.85	-98.73
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model 	771.14	-5.61	-36.51
4.	4.	4.	PostgreSQL	Relational, Multi-model 	680.89	+0.23	+41.98
5.	5.	5.	MongoDB 	Document, Multi-model 	403.83	+0.99	-25.99
6.	6.	 7.	Snowflake	Relational	176.17	+1.68	+39.64
7.	7.	 6.	Redis	Key-value, Multi-model 	149.72	-2.01	-7.05
8.	8.	 9.	IBM Db2	Relational, Multi-model 	127.51	+2.38	+3.11
9.	9.	 8.	Elasticsearch	Multi-model 	118.83	-2.45	-12.00
10.	10.	10.	SQLite	Relational	115.44	-1.60	+5.49
11.	11.	 12.	Apache Cassandra	Wide column, Multi-model 	108.76	+0.49	+9.63
12.	12.	 15.	Databricks	Multi-model 	108.04	+3.36	+24.74
13.	13.	 14.	MariaDB 	Relational, Multi-model 	95.44	+0.91	+4.86
14.	14.	 11.	Microsoft Access	Relational	90.47	+2.19	-10.17
15.	15.	 17.	Amazon DynamoDB	Multi-model 	84.15	+0.81	+13.20
16.	 17.	16.	Microsoft Azure SQL Database	Relational, Multi-model 	76.51	+1.20	-0.24
17.	 16.	 19.	Apache Hive	Relational	75.74	-0.94	+18.45
18.	18.	 13.	Splunk	Search engine	69.52	-0.10	-23.40
19.	19.	 18.	Google BigQuery	Relational	64.19	-0.35	+6.38
20.	20.	 21.	Neo4j	Graph	54.64	+3.30	+8.88

이미 직장에서 쓰고 있는게 있다면 그것을 공부하시고, 그게 아니라면... 위 표를 참고하세요

## 관계형 DB, 그 외 DB 모두 경험하면 좋다.

MySQL을 잘하면 Oracle을 잘하고, 그 반대도 마찬가지.

근데

MongoDB는 형식이 다른 DB다.

둘 다 해보면 정말 좋다.

## Oracle?

돈 많은 곳에서 사용  
개인, 작은 회사, 큰 회사여도 금융 등이 아니라면 비추

## MySQL?

무료, 오픈소스

SNS 같이 대규모 데이터 생성 but 데이터의 신뢰성은 아주 중요하지는 않은  
그런 기업에서는 아주 좋은 제품이다.

## MongoDB?

Document. 중요한 건 관계형DB가 아니다라는 것.

SNS, IoT, 프로그래밍 인력 증가 등등...

수많은 데이터, 다양한 종류의 데이터의 등장

**2010~ NoSQL이라는 흐름.**

## DB 2. MySQL

### 1. MySQL 강의 소개.

#### DB조차도 결국

그 정보를 파일에 저장

#### 파일에 만족 못함

정보의 폭발로 파일 만으로는 정보를 효과적으로 입력, 저장, 출력하는 것이 어려워짐.  
즉, 데이터를 잘 정리정돈해서 필요 시 쉽게 꺼내 쓰고 싶다는 생각으로.

#### DB의 탄생

1960~ 데이터 누구나 쉽게 정리 정돈하는 SW 만들기 시작.

이런 SW를 Database라 부르기 시작.

1970~ Relational database 탄생

#### RDB

지금까지 DB 분야에서의 절대 강자

#### MySQL

무료, 오픈소스, RDB 주요 기능 대부분 갖춤.

준수한 DB System이다.

#### WEB

폭발적 성장.

웹개발자는 웹페이지를 통해 표현할 정보를 저장할 DB를 찾게됨.

MySQL --- WEB

웹과 함께 폭발적인 동반 성장.

오늘도 MySQL이라는 웹의 심장에 의해 동작하고 있는 수많은 웹사이트를 이용했을 것이다.

## MySQL이라는 캐비닛으로 데이터를 깔끔하게 정리정돈 해보자~

## 2. DB의 목적.

### Spreadsheets와 DB 비교해보자

차이점과 공통점 파악해보자.

#### 공통점

표의 형태로 표현해준다.

따라서 두 개 기능 비슷하다.

#### 가장 중요한 차이점

DB는 코딩을 통해서, 컴퓨터 언어를 통해서 제어할 수 있다는 것이다.

Spreadsheets는 클릭클릭

특히 관계형 데이터베이스는 SQL이라는 언어로. 코드로.

### 저장한 데이터를 이제 다양한 목적으로...

웹, 앱, 빅데이터, 인공지능, ...

#### WEB

1. [MySQL](#)
2. [ORACLE](#)
3. [SQL Server](#)
4. [PostgreSQL](#)
5. [MongoDB](#)
6. [DB2](#)

[create](#)

Welcome

Hello, PHP

```
mysql>
mysql>
mysql>
mysql> SELECT * FROM topic;
+----+-----+-----+-----+-----+-----+
| id | title | description | created | author | p |
+----+-----+-----+-----+-----+-----+
| 1 | MySQL | MySQL is ... | 2018-01-01 00:00:00 | egoing | d |
| 2 | ORACLE | Oracle is ... | 2018-01-15 00:00:00 | egoing | d |
| 3 | SQL Server | SQL Server is ... | 2018-01-18 00:00:00 | duru | d |
| 4 | PostgreSQL | PostgreSQL is ... | 2018-01-20 00:00:00 | taeho | d |
| 5 | MongoDB | MongoDB is ... | 2018-01-30 00:00:00 | egoing | d |
+----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> SELECT * FROM topic;
+----+-----+-----+-----+-----+-----+
| id | title | description | created | author | p |
+----+-----+-----+-----+-----+-----+
| 1 | MySQL | MySQL is ... | 2018-01-01 00:00:00 | egoing | d |
| 2 | ORACLE | Oracle is ... | 2018-01-15 00:00:00 | egoing | d |
| 3 | SQL Server | SQL Server is ... | 2018-01-18 00:00:00 | duru | d |
| 4 | PostgreSQL | PostgreSQL is ... | 2018-01-20 00:00:00 | taeho | d |
| 5 | MongoDB | MongoDB is ... | 2018-01-30 00:00:00 | egoing | d |
| 6 | DB2 | DB2 is ... | 2018-02-08 18:25:24 | NULL | N |
+----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

### 우리가 직접 DB를 제어하지 않아도

이렇게 누구나 웹에 접속, 그 정보를 읽을 수 있다.

이렇게 누구나 글을 쓰면, 그 정보는 결과적으로 이렇게 DB에 저장되고 있다.

DB로 너무 많은 일을 할 수 있다.

### 3. MySQL 설치.

(codeanywhere.com 방법도 존재)

#### 윈도우에 설치

(bitnami wamp 단종)

```
C:\Users\juhwa>cd C:\Program Files\MySQL\MySQL Server 9.4\bin

C:\Program Files\MySQL\MySQL Server 9.4\bin>mysql -uroot -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 9.4.0 MySQL Community Server - GPL

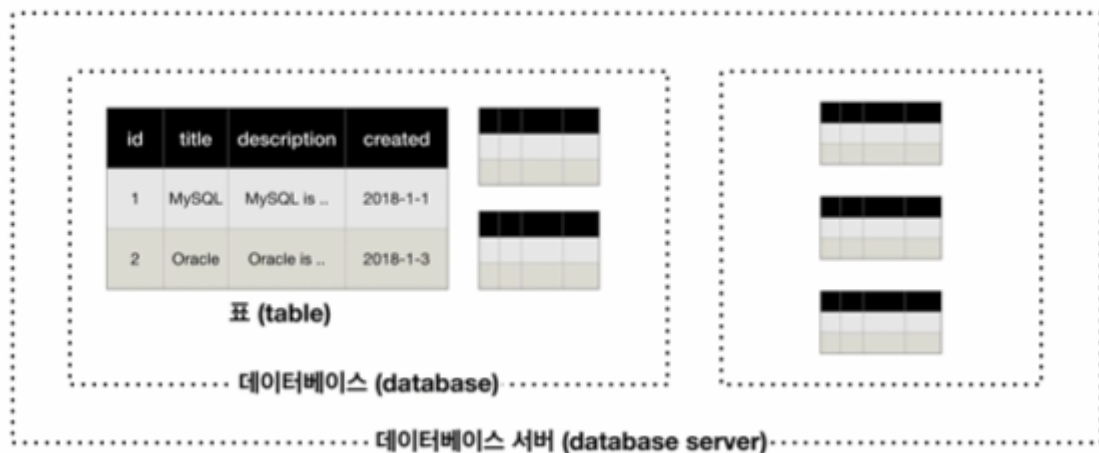
Copyright (c) 2000, 2025, Oracle and/or its affiliates.

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> |
```

### 4. MySQL 의 구조.



### MySQL은 3개의 구성요소가 있다

1. **표**: 최종적으로 데이터를 기록하는 곳. 정보는 결국 표에 저장이 된다.  
ex) 웹사이트 운영하는 DB  
글들을 저장하는 표, 댓글들을 저장하는 표, 회원 정보들을 저장하는 표 ...

=> 표들을 정리정돈 할 필요성이 생긴다. (파일에서 디렉토리의 필요성과 유사)

**테이블**이라는 표현을 많이 쓴다.

2. **데이터베이스**: 연관된 표들끼리 그룹핑하고, 연관되지 않은 표들과 분리하는 데 사용하는 파일의 폴더.

즉, 표들을 그룹핑 한 것. 표들을 그룹핑한 폴더.

지금 배우는 것 전체가 DB인데 이것도 DB?

=> **스키마**라는 표현도 같이 쓴다.

3. **데이터베이스 서버**: 스키마들을 저장하는 곳.

MySQL을 설치한 것은, 데이터베이스 서버라는 프로그램을 설치한 것이고, 그 프로그램이 갖고 있는 기능을 이용해서 우리는 데이터와 관련된 여러 작업을 하게 된다.

## 5. MySQL 서버 접속.

### DB 사용 시의 효용 중 첫번째는

보안이다.

FILE은 운영체제만 뚫으면 그 안의 파일은 무주공산이다.

1. DB는 자체적 보안 체계가 있다.

2. 권한 기능이 있다.

ex) 이 사람은 모든 테이블, 스키마에 대해서 읽기 쓰기 수정 삭제가 가능하게 한다.

다른 사람은 이 스키마 중 이 테이블만 제어할 수 있도록 한다. 등등

### 그래서 이제 코드를 보면,

```
mysql -uroot -p
```

1. **-u**: 유저의 약자, **-uroot**: root라는 유저로, **-ukim**: kim이라는 유저로.  
root는 기본 유저를 의미한다. 관리자로 모든 권한이 열려있다.  
실제로는 평소엔 위험해서 root 권한을 쓰지 않고, 별도의 사용자를 만든다.
2. **-p**: **-p비밀번호**로 사용해도 되지만, 안 적어도 MySQL이 물어본다.  
비밀번호 잊었을 때는 mysql password forget이라고 쳐보세요.

### 접속 성공 시,

이제 아래 그림처럼 데이터베이스 서버의 담장을 넘은 것이다.





이제 데이터베이스를 만나러 가보자.

## 6. MySQL 스키마의 사용.

이제 스키마를 만들고,

그걸로 표를 만들 준비를 해야죠.

### 검색해보자

how to create database(schema) in mysql

how to show database list in mysql

아하.

### 데이터베이스 만들기

```
CREATE DATABASE tutorial;
```

세미콜론은 필수다.

### ~~데이터베이스 삭제~~

```
DROP DATABASE tutorial;
```

사실 이 명령어를 외우는 게 중요한 건 아니다.

### 데이터베이스 확인

```
SHOW DATABASES;
```

이제 데이터베이스 안에 들어가려면,

(들어가서 표를 만들거니까,)

이 데이터베이스를 사용하겠다는 것을 먼저 알려줘야 한다.

```
USE tutorial;
```

이제부터 MySQL은

내가 내리는 명령을 tutorial이라고 하는 스키마에 있는 표를 대상으로 실행하게 된다.

이제 표를 만들어 가보자.

id	title	description	created
1	MySQL	MySQL is ...	2018-1-1
2	Oracle	Oracle is ...	2018-1-3

표 (table)

## 7. SQL 과 테이블 구조.

### SQL?

Structured Query Language

Structured: 표, 정리정돈

Query: 데이터베이스에게 하는 요청 (데이터 넣어줘, 읽어줘, 수정해줘, 삭제해줘, 스키마 만들어줘, ...)

Language: 공통의 약속 (프로그램과 프로그래머 간)

### SQL의 두가지 특징

1. 어떤 언어보다도 쉽다. (html과 동급)
2. 중요하다.

관계형 데이터베이스 라는 카테고리에 속하는 제품들이 공통적으로 데이터베이스 서버를 제어할 때 사용하는 언어이기 때문이다. 표준화가 되어있다. 압도적인 다수의 DB 시스템이 sql을 통해 동작.

즉, 쉽고 중요하다. 가성비가 좋다.

### 이제 표를 다뤄볼 시간

Table.

### 테이블의 구조

1. row, 행 (+ record)  
데이터 하나 하나, 데이터 자체
2. column, 열  
데이터의 타입, 데이터 구조

column, 열



id	title	description	created
1	MySQL	MySQL is ...	2018-1-1
2	Oracle	Oracle is ...	2018-1-3

← row, record, 행

table, 표

2행 4열 의 테이블

## 8. MySQL 테이블의 생성.

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 Serve	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

이 표를 만들어보자.

그 전에...

1. 내가 직접 표를 만들기 위해 sql 문을 직접 짜는 경우는 많지 않다.  
그러니까 외우려고 하고, 심란해 하지 말자.
2. 상당히 좀 코드가 복잡하다.  
근데 5 개를 위해서 이런 행동을 하는 게 아니라, 1억 개를 위해 한다고 생각하면 경제적이다.

추가로, 검색해서 알아보자.

ex) create table in mysql, mysql datatype number ... 등

## 치트 시트

mysql sql cheatsheet

자주 사용하는 SQL 명령어를 정리해봤다.

번외. sql 실수했을 때,

## 가장 간단한 방법: 명령 취소하고 다시 시작하기

MySQL 프롬프트에서 `\c` 를 입력하고 엔터를 치면, 지금까지 작성하던 `CREATE TABLE` 명령이 모두 취소됩니다.

SQL

```
mysql> CREATE TABLE topic(  
  -> id INT(11) NOT NULL,  
  -> \c  
mysql>
```

이렇게 `\c` 를 입력하면 `mysql>` 프롬프트가 새로 나타나고, 아무 일도 없었다는 듯이 다시 정확한 명령어를 입력할 수 있습니다. 이 방법이 가장 빠르고 간편합니다.

## 다른 방법: 테이블 생성 후 수정하기

물론 질문하신 것처럼 일단 테이블 생성을 마친 뒤, `ALTER TABLE` 명령어로 `AUTO_INCREMENT` 속성을 추가하는 것도 가능합니다.

### 1. 일단 테이블 생성 완료하기:

SQL

```
mysql> CREATE TABLE topic(  
  -> id INT(11) NOT NULL,  
  -> title VARCHAR(100) NOT NULL,  
  -> PRIMARY KEY(id)  
  -> );  
Query OK, 0 rows affected (0.02 sec)
```

### 2. `ALTER TABLE` 로 `id` 컬럼 수정하기:

SQL

```
mysql> ALTER TABLE topic MODIFY id INT(11) NOT NULL AUTO_INCREMENT;  
Query OK, 0 rows affected (0.03 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

결론적으로, 실수를 알아차린 즉시 `\c` 를 입력해 취소하고 다시 작성하는 것이 가장 효율적입니다.

## 진짜 표 만들기

```
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)
);
```

1. 괄호를 열고 엔터를 누르면 세미콜론 등장할 때 까지 실행이 바로 되지 않는다.
2. 가독성을 위해 엔터 후 한 칸 스페이스바 좋다.
3. INT(11) : 여기서 11 은 11자리만 기록하겠다는 의미가 아니라, 화면에 출력되는 자리수.
4. NOT NULL : NULL 허용 X, 반드시 값이 있어야 한다.
5. AUTO\_INCREMENT : 각 행의 식별자로 id column을 쓰고 싶다면, id 값은 다른 행과 절대 중복되어서는 안됨. 그러기 위해서는 id값이 자동 1씩 증가되게 만들어야 하고, 그걸 AUTO\_INCREMENT 가 해준다.
6. VARCHAR(100) : 여기서 100 은 100글자 넘어가면 자르고, 그 전까지만 저장함.
7. TEXT : 최적화 필요.

TEXT(size)	Maximum size of 65,535 characters.
MEDIUMTEXT(size)	Maximum size of 16,777,215 characters.
LONGTEXT(size)	Maximum size of 4GB or 4,294,967,295 characters.

8. NULL : NULL 허용
9. DATETIME :

Data Type	"Zero" Value
<u>DATE</u>	'0000-00-00'
<u>TIME</u>	'00:00:00'
<u>DATETIME</u>	'0000-00-00 00:00:00'
<u>TIMESTAMP</u>	'0000-00-00 00:00:00'
<u>YEAR</u>	0000

10. PRIMARY KEY : 아래 참조.

## PRIMARY KEY

는 테이블 생성 시 필수로 입력해야 한다.

PRIMARY KEY의 장점

1. 아래 같이 column 내용이 겹치면 DB가 거절한다. 즉, 중복을 방지한다.

4 PostgreSQL
5 MongoDB
4 DB2

why? primary key는 각 행을 식별할 때 사용하는 가장 중요한 column

## 번외. warning

```
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 (0.409 sec)  
  
mysql> SHOW WARNINGS;  
+-----+-----+-----+  
| Level | Code | Message  
+-----+-----+-----+  
| Warning | 1681 | Integer display width is deprecated and will be removed in a future release.  
+-----+-----+-----+  
1 row in set (0.005 sec)
```

최신 MySQL 버전(MySQL 8.0.17 이후)부터 정수 타입(INT, BIGINT 등) 뒤에 붙는 숫자(예: (11))는 더 이상 사용되지 않는(deprecated) 기능이 되었습니다.

과거에 이 숫자는 '표시 너비(display width)'를 의미했지만, 실제 저장할 수 있는 값의 범위에는 영향을 주지 않았고, ZEROFILL 옵션과 함께 쓰이지 않으면 거의 의미가 없어서 혼란을 주곤 했습니다.

따라서 최신 MySQL은 "앞으로 이 기능은 없어질 예정이니, 그냥 INT 라고만 쓰세요"라는 의미로 경고 메시지를 보여주는 것입니다.

## 테이블 지우기

```
DROP TABLE topic;
```

이게 얼마나 위험한 명령인가요, 데이터가 1억 개 있으면...

## 테이블 보기

```
SHOW TABLES;
```

```
mysql> SHOW TABLES;  
+-----+  
| Tables_in_tutorial |  
+-----+  
| topic              |  
+-----+  
1 row in set (0.129 sec)
```

## 비밀번호 바꾸기

```
SET PASSWORD = PASSWORD('12341234')
```

코드보다 더 중요한 것은,

데이터베이스가 여러 규제 정책을 가지고 있고,  
그것 덕분에 데이터를 깔끔하게, 우리가 원하는 형식으로 유지하는데 큰 도움을 준다는 것이다.

## 9.1. MySQL CRUD

### CRUD?

Create, Read, Update, Delete

### 가장 중요한 것

CREATE, READ

### 없을 수도 있는 것

UPDATE, DELETE

ex) 역사, 회계

## 9.2. SQL 의 INSERT 구문.

### CREATE!

테이블에서 한 행(데이터)을 추가하는 것.

### 이 테이블이 궁금할 때,

```
DESC topic;
```

```
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 (0.031 sec)
```

### DESC를 참고해서 이제 행을 추가해보자

```
INSERT INTO topic (column1...) VALUES (value1...)
```

```
mysql> INSERT INTO topic (title,description,created,author,profile)
VALUES ('MySQL','MySQL is ...',NOW(),'egoing','developer');
Query OK, 1 row affected (0.153 sec)
```

NOW() : 현재 시간 불러오는 함수.

## 추가된 행을 읽어보자

```
SELECT * FROM topic;
```

```
mysql> SELECT * FROM topic;
+----+-----+-----+-----+-----+-----+
| id | title | description | created | author | profile |
+----+-----+-----+-----+-----+-----+
| 1 | MySQL | MySQL is ... | 2025-07-25 19:03:37 | egoing | developer |
+----+-----+-----+-----+-----+-----+
1 row in set (0.007 sec)
```

이걸 반복해서 결과를 확인하면,

```
mysql> INSERT INTO topic (title,description,created,author,profile) VALUES ('ORACLE','ORACLE is ...',NOW(),'egoing','developer');
Query OK, 1 row affected (0.056 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 (0.039 sec)

mysql> INSERT INTO topic (title,description,created,author,profile) VALUES ('PostgreSQL','PostgreSQL is ...',NOW(),'taeho','data scientist, developer');
Query OK, 1 row affected (0.048 sec)

mysql> INSERT INTO topic (title,description,created,author,profile) VALUES ('MongoDB','MongoDB is ...',NOW(),'egoing','developer');
Query OK, 1 row affected (0.133 sec)

mysql> SELECT * FROM topic;
+----+-----+-----+-----+-----+-----+
| id | title | description | created | author | profile |
+----+-----+-----+-----+-----+-----+
| 1 | MySQL | MySQL is ... | 2025-07-25 19:03:37 | egoing | developer |
| 2 | ORACLE | ORACLE is ... | 2025-07-25 19:08:01 | egoing | developer |
| 3 | SQL Server | SQL Server is ... | 2025-07-25 19:08:52 | duru | data administrator |
| 4 | PostgreSQL | PostgreSQL is ... | 2025-07-25 19:10:15 | taeho | data scientist, developer |
| 5 | MongoDB | MongoDB is ... | 2025-07-25 19:10:47 | egoing | developer |
+----+-----+-----+-----+-----+-----+
```

**INSERT, 특히 SELECT는 정말 많이 사용한다.**

이 정도는 알아두자.

## 9.3. SQL 의 SELECT 구문.

### READ!

Create, Update, Delete는 명령이 심플하다.  
그런데 읽기의 경우 굉장히 복잡해질 수 있다.

### 검색해서 알아보기



## mysql select syntax

```
SELECT
  [ALL | DISTINCT | DISTINCTROW ]
  [HIGH_PRIORITY]
  [STRAIGHT_JOIN]
  [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
  [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
  select_expr [, select_expr ...]
  [FROM table_references
  [PARTITION partition_list]
  [WHERE where_condition]
  [GROUP BY {col_name | expr | position}
  [ASC | DESC], ... [WITH ROLLUP]]
  [HAVING where_condition]
  [ORDER BY {col_name | expr | position}
```

아래 생략...

여기서 []는 생략 가능하다는 뜻

select\_expr: 표현되어야 할 column들. column의 목록.

FROM 후 WHERE가 나옴. WHERE 후 ORDER BY가 나옴. 등등 위치 중요.

## 모든 데이터 읽기

```
SELECT * FROM topic;
```

## 원하는 column 읽기

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

## 그냥 SELECT (FROM 생략)

```
SELECT "kim", 1+1;
```

```
mysql> SELECT "kim", 1+1;
+-----+-----+
| kim | 1+1 |
+-----+-----+
| kim |    2 |
+-----+-----+
1 row in set (0.007 sec)
```

이런 column 만든 적 없는데 그냥 나옴.

## WHERE를 통해 column에서 특정 값만 읽기

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

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

id	title	created	author
1	MySQL	2025-07-25 19:03:37	egoing
2	ORACLE	2025-07-25 19:08:01	egoing
5	MongoDB	2025-07-25 19:10:47	egoing

```
3 rows in set (0.008 sec)
```

## ORDER BY로 정렬하기

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

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

id	title	created	author
5	MongoDB	2025-07-25 19:10:47	egoing
2	ORACLE	2025-07-25 19:08:01	egoing
1	MySQL	2025-07-25 19:03:37	egoing

```
3 rows in set (0.007 sec)
```

## 1억 개의 데이터를 불러오면 큰일 남

컴퓨터가 감당 못함. 제약을 걸어줘야 함.

## LIMIT로 제약 걸어주기

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

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

id	title	created	author
5	MongoDB	2025-07-25 19:10:47	egoing
2	ORACLE	2025-07-25 19:08:01	egoing

```
2 rows in set (0.005 sec)
```

## DB를 잘한다? SELECT 문을 필요에 따라 잘 사용하는 것이다. 수련이 필요.

검색을 통해 알아가며 수련하기

ex) 저자가 egoing 또는 duru 인 것을 읽으려면?

id로 정렬하고, 그 안에서 또 title이나 created로 정렬하려면? 등등 ...

ex) LIMIT를 이용하여 게시판의 page 기능 구현 등등 ...

## 9.4. SQL 의 UPDATE 구문

### UPDATE

테이블 수정은 어떻게 할까?

### 검색

```
UPDATE [LOW_PRIORITY] [IGNORE] table_reference
  SET assignment_list
  [WHERE where_condition]
  [ORDER BY ...]
  [LIMIT row_count]

value:
  {expr | DEFAULT}

assignment:
  col_name = value

assignment_list:
  assignment [, assignment] ...
```

### 코드

```
UPDATE topic SET description='Oracle is ...', title='Oracle';
```

만약에 여기까지 하면? 큰일남.

모든 행들이 이렇게 바뀜...

WHERE 문을 빠트리면 재앙이 온다.

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

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

id	title	description	created	author	profile
1	MySQL	MySQL is ...	2025-07-25 19:03:37	egoing	developer
2	ORACLE	ORACLE is ...	2025-07-25 19:08:01	egoing	developer
3	SQL Server	SQL Server is ...	2025-07-25 19:08:52	duru	data administrator
4	PostgreSQL	PostgreSQL is ...	2025-07-25 19:10:15	taeho	data scientist, developer
5	MongoDB	MongoDB is ...	2025-07-25 19:10:47	egoing	developer

```
5 rows in set (0.008 sec)
```

```
mysql> UPDATE topic SET description='Oracle is ...', title='Oracle' WHERE id=2;
Query OK, 1 row affected (0.048 sec)
Rows matched: 1 Changed: 1 Warnings: 0
```

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

id	title	description	created	author	profile
1	MySQL	MySQL is ...	2025-07-25 19:03:37	egoing	developer
2	Oracle	Oracle is ...	2025-07-25 19:08:01	egoing	developer
3	SQL Server	SQL Server is ...	2025-07-25 19:08:52	duru	data administrator
4	PostgreSQL	PostgreSQL is ...	2025-07-25 19:10:15	taeho	data scientist, developer
5	MongoDB	MongoDB is ...	2025-07-25 19:10:47	egoing	developer

```
5 rows in set (0.004 sec)
```

## 9.5. SQL 의 DELETE 구문

### DELETE

테이블에서 삭제는 어떻게 할까?

### 검색

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM tbl_name
    [PARTITION (partition_name [, partition_name] ...)]
    [WHERE where_condition]
    [ORDER BY ...]
    [LIMIT row_count]
```

여기서도 WHERE을 빠트리면 재앙이 온다.

### 행 삭제하기

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

```
mysql> DELETE FROM topic WHERE id=5;
Query OK, 1 row affected (0.137 sec)
```

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

id	title	description	created	author	profile
1	MySQL	MySQL is ...	2025-07-25 19:03:37	egoing	developer
2	Oracle	Oracle is ...	2025-07-25 19:08:01	egoing	developer
3	SQL Server	SQL Server is ...	2025-07-25 19:08:52	duru	data administrator
4	PostgreSQL	PostgreSQL is ...	2025-07-25 19:10:15	taeho	data scientist, developer

```
4 rows in set (0.005 sec)
```