IT Department

Introduction to MySQL, Data Types and Basic Operations

Soma Soleiman Zadeh

Database Systems II (IT 226)

Spring 2025 - 2026

Week 1

February 2, 2026

Outline

- What is **MySQL**?
- **MySQL** Key Benefits
- Data Types in **MySQL**
- Basic Database Operations in **MySQL**
  - **Create Database** and Activate the Database
  - **Create Table**
  - Alter Table (Add a Column, Drop a Column, Modify a Column)
  - Modify Column

What is a MySQL?

- **MySQL** is the world's most popular open-source database management system.
- **MySQL** is an SQL-based relational database designed to store and manage structured data.
- **MySQL Workbench** is a graphical tool for working with **MySQL** servers and databases.

MySQL Key Benefits

- **Ease of Use –** Installing MySQL and managing a database in MySQL is easy.
- **Reliability –** MySQL has been tested and used in many well-known companies. Many organizations depend on MySQL because of its reliability.
- **Scalability –** MySQL's native replication architecture enables organizations, including Facebook, Netflix, and Uber, to scale applications to support tens of millions of users or more.

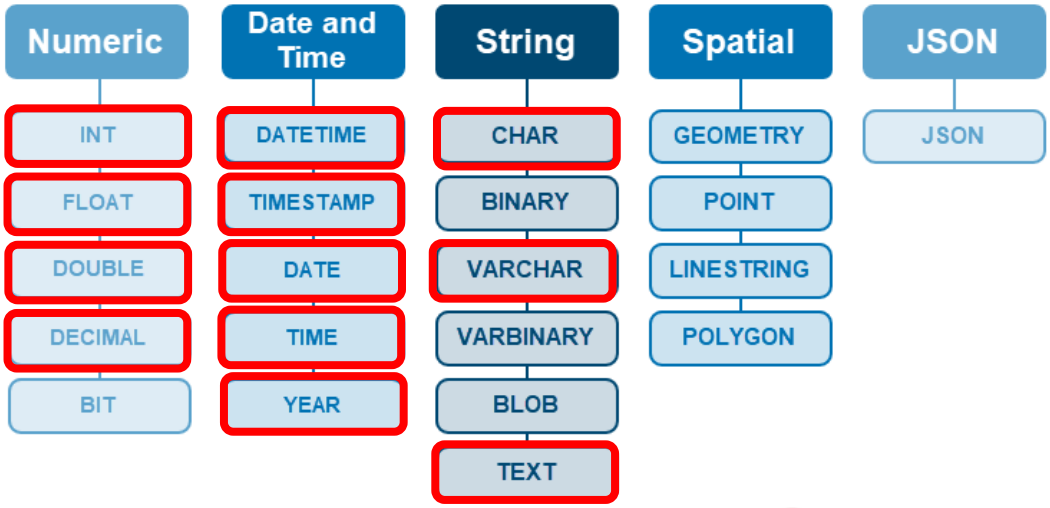
MySQL Key Benefits

- **Performance –** MySQL is a proven high-performance DBMS.
- **High Availability –** MySQL delivers a complete set of replication technologies for high availability and disaster recovery.
- **Security –** Data security entails both data protection and compliance with industry and government regulations.
- **Flexibility –** Flexibility refers to the ability to adapt to changing business needs, data and requirements.

Data Types in MySQL

- Each column in a database table must have:
  - **Column name**
  - **Data type**
- When you create a table, you need to specify the **name of each column** and decide on the **type of data stored in that column**.

Basic Data Types in MySQL



Numeric Data Types

Data Type	Description	Storage (Bytes)
INT	A standard integer	4 Bytes
DECIMAL	A fixed-point number	4 Bytes
NUMERIC	A fixed-point number	4 Bytes
FLOAT	A floating-point number	4 Bytes
DOUBLE	A floating-point number that stores larger values than FLOAT	8 bytes

String Data Types

Data Type	Description	Number of Characters
CHAR (M)	A **fixed-length** string	M → 0-255
VARCHAR (M)	A **variable-length** string	M → 0-255
TEXT (M)	A long string	M → 0-65,535

(Date and Time) Data Types

Data Type	Description	Storage (Bytes)
DATE	A date	3 Bytes
DATETIME	A date and time combination	8 Bytes
TIME	A time	3 Bytes
YEAR	A year in four-digit format	1 Byte
TIMESTAMP	A timestamp	4 Bytes

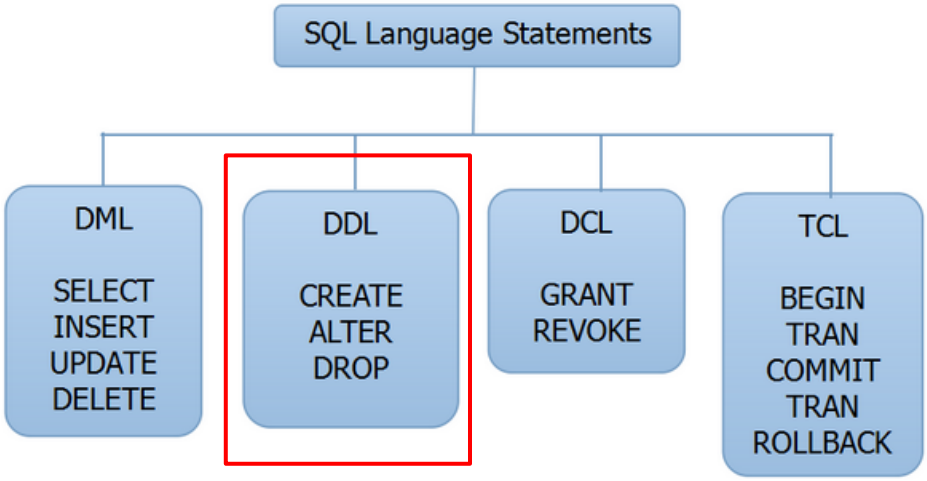
(Date and Time) Data Types

- **DATETIME** data type → is used to store both date and time information. **Example →** *2025-02-10 11:20:43*
  - **DATETIME** is appropriate for storing information about a specific date and time, such as the date and time an event occurs or a record is created.
- **TIMESTAMP** data type → also stores both date and time information.
  - **TIMESTAMP** automatically updates the value to the current date and time when a new record is inserted or an existing record is updated, **making it useful for tracking changes in your data**.

SQL Operations

- SQL commands are categorized into four categories:
  - **DDL** – **D**ata **D**efinition **L**anguage
  - **DML** – **D**ata **M**anipulation **L**anguage
  - **DCL** – **D**ata **C**ontrol **L**anguage
  - **TCL** – **T**ransaction **C**ontrol **L**anguage

SQL Operations



SQL DDL Operations

- SQL **DDL** Commands → Defining the database

CREATE DATABASE	CREATE TABLE
DROP DATABASE	ALTER TABLE
	DROP TABLE

- These statements are used to **create**, **change**, or **destroy** the structures that make up the logical model.

Create Database

- The **CREATE DATABASE** statement is used to create a new database.

Syntax:

CREATE DATABASE <database_name>;

Example:

CREATE DATABASE University;

Activating a Database

- The **USE** statement is used to activate one of the databases on the current instance of MySQL Server.
- When a database is activated, all next SQL commands are applied on that database.

Syntax:

USE <*database_name*>;

Example:

USE University;

Drop Database

- The **DROP DATABASE** statement is used to remove an existing database.

Syntax:

```
DROP DATABASE <database_name>;
```

Example:

```
DROP DATABASE University;
```

Create Table

- The **CREATE TABLE** statement is used to create a new table in a database.
- **Syntax:**

```
CREATE TABLE table_name
(Column_name1  datatype1,
 Column_name2  datatype2,
 ...,
 Column_namen  datatypen,
 (Integrity_Constraint1),
 ...,
 (Integrity_Constraintn) );
```

Integrity Constraints

- Integrity Constraints are rules to ensure that the data in the database is accurate, consistent and reliable.
- Two important integrity constraints:
  - **primary key** (col_1, …, col_n)
  - **foreign key** (col_m, …, col_n) **references** table_name (primary_col)
- SQL prevents any update to the database that violates an integrity constraint.

Create Table

```
Create Table Student
(    stu_ID      int,
     stu_name    varchar(30),
     Email       varchar(40)
);
```

- A table named **Student** is created. The table has three columns. Column names and their datatypes are specified.
- According to this syntax, this table doesn't have a primary key.

stu_ID	stu_Name	Email

Create Table

```
Create Table Student
(    stu_ID      int,
     stu_name    varchar(30),
     Email       varchar(40),
     primary key (stu_ID)
);
```

- A table named **Student** is created. The table has three columns. Column names and their datatypes are specified.
- According to this syntax, this table has a primary key, which is the student ID.

stu_ID	stu_Name	Email

Create Table

```
Create Table  Grocery_Product
(    Product_ID      int ,
     Product_name    varchar(60) ,
     Expiry_Date     date ,
     Price           numeric(4 , 1) ,
     primary key (Product_ID)
);
```

- The **Expiry_Date** column shows the date that a product expires, so the datatype is **date**.
- The **Price** values are fixed-point numbers, so both **decimal** and **numeric** can be used as its datatype.
- decimal(4,1) means a fixed-point number that has 4 digits and 1 of them is after the decimal point.

Product_ID	Product_Name	Expiry_Date	Price
1	Cheese	2026-05-01	3.4

What is the Difference?

```
Create Table Student
(    stu_ID      int,
     stu_name    varchar(30),
     Email       varchar(40),
     primary key (stu_ID)
);
```

When we insert data into this table, we have to enter values for the stu_ID column as it is the Primary key and cannot be null.

```
Create Table Student
(    stu_ID      int  auto_increment,
     stu_name    varchar(30),
     Email       varchar(40),
     primary key (stu_ID)
);
```

When we insert data into this table, there is no need to enter values for the stu_ID column, because we made it auto-increment. It means that for each inserted row into the table, stu_ID is generated automatically starting from 1 and increased by 1 for the next inserted row.

What is the Difference?

```
Create Table Student
(    stu_ID      int,
     stu_name    varchar(30),
     Email       varchar(40),
     primary key (stu_ID)
);
```

When we insert data into this table, we can leave **stu_name** and **Email** values empty (null). The **stu_ID** is the primary key of the table and can not be null.

```
Create Table Student
(    stu_ID      int,
     stu_name    varchar(30),
     Email       varchar(40)  not null,
     primary key (stu_ID)
);
```

When we insert data into this table, values of only **stu_name** can be null. The **Email** value can not be null, as we explicitly forced this column to be not null, and **stu_ID** can not be null as it is the primary key of the table.

Create Table (Example of Foreign Key)

```
Create Table Department
(    deptName      varchar(70),
     budget        int,
     primary key (deptName)
);
```

```
Create Table Student
(    stu_ID        int,
     stu_name      varchar(30),
     Email         varchar(40),
     major         varchar(70),
     primary key (stu_ID),
     foreign key (major) references Department(deptName)
);
```

deptName	budget
IT	7000
Civil Eng.	8000
Dentistry	12000

stu_ID	stu_name	Email	major
1	Leonardo	leo@gmail.com	Dentistry
2	Charles	charles@gmail.com	IT
3	Sandy	sandy@gmail.com	Dentistry

Drop Table

- The **DROP TABLE** statement is used to remove an existing table from a database.

Syntax:

```
DROP TABLE <table_name>;
```

Example:

```
DROP TABLE Student;
```

Alter Table

- The **ALTER TABLE** statement is used to:
  - **Add columns** to an existing table.
  - **DROP columns** from an existing table.
  - **Modify columns** of an existing table.
  - **Rename columns/tables**.
- The **ALTER TABLE** statement is also used to:
  - **Add** various constraints in an existing table.
  - **Drop** various constraints from an existing table.

Alter Table (Adding New Column)

- **ADD** statement is used to add column(s) to an existing table.

Syntax:

```
ALTER TABLE <table_name>
ADD column_name datatype;
```

Example:

```
ALTER TABLE Student
ADD DateOfBirth year;
```

Stu_ID	Stu_name	Email

Stu_ID	Stu_name	Email	DateOfBirth

Alter Table (Dropping an Existing Column)

- **DROP COLUMN** statement is used to delete column(s) from an existing table.

Syntax:

ALTER TABLE <table_name>
DROP COLUMN column_name;

Example:

ALTER TABLE Student
DROP COLUMN Email;

Stu_ID	Stu_name	Email	DateOfBirth

Stu_ID	Stu_name	DateOfBirth

Alter Table (Changing Datatype of a Column)

- **MODIFY COLUMN** statement is used to modify column(s) of an existing table.

Syntax:

ALTER TABLE <table_name>
MODIFY COLUMN column_name datatype;

Example:

ALTER TABLE Student
MODIFY COLUMN DateofBirth **date**;

Stu_ID	Stu_name	Email	DateOfBirth

The current datatype of **DateOfBirth** is ***year***. By executing this command, its datatype is changed to ***date***.