# Information Management (MySQL) [1]

## A Study Guide for Students of Sorsogon State University - Bulan Campus[2]

Jarrian Vince G. Gojar[3]

January 24, 2025

[1] A course in the Bachelor of Science in Computer Science.

[2] This book is a study guide for students of Sorsogon State University - Bulan Campus taking up the course Information Management.

[3] https://github.com/godkingjay

Sorsogon State University - Bulan Campus

# Contents

# List of Figures

# List of Tables

# List of Codes

# Preface

> *"You can have data without information, but you cannot have information without data."*
>
> – Daniel Keys Moran

Jarrian Vince G. Gojar
https://github.com/godkingjay

# 1

# Introduction to MySQL

## 1.1 What is MySQL?

**MySQL** is an open-source relational database management system (RDBMS). An RDBMS is a type of database management system (DBMS) that stores data in a structured format, using rows and columns. It is a software that is used to manage the creation, modification, and maintenance of a database. MySQL is a popular choice for database management in web applications and is used by many high-profile websites, including Facebook, Twitter, and YouTube.

## 1.2 MySQL Elements

### 1.2.1 Tables

A **table** is a collection of related data stored in rows and columns. Each row in a table represents a record, and each column represents a field. Tables are the basic building blocks of a database and are used to store and organize data in a structured format.

| id | name | age |
|---|---|---|
| 1 | Alice | 25 |
| 2 | Bob | 30 |
| 3 | Charlie | 35 |

Table 1.1: Example of a table

Table **??** shows an example of a table with three columns: **id**, **name**, and **age**. Each row in the table represents a record with a unique **id** value, and each column represents a field in the record.

### 1.2.2 Columns

A **column** is a vertical arrangement of data in a table. Each column represents a field in the table and contains data of a specific type. Columns are used to store different types of data, such as numbers, text, dates, and more.

In the example table above, the columns are **id**, **name**, and **age**. The **id** column stores unique identifiers for each record, the **name** column stores the names of the individuals, and the **age** column stores the ages of the individuals.

### 1.2.3 Rows

A **row** is a horizontal arrangement of data in a table. Each row represents a record in the table and contains data for each column in the table. Rows are used to store individual records in the table.

In the example table above, each row represents a record with data for the **id**, **name**, and **age** columns. The first row contains data for Alice, the second row contains data for Bob, and the third row contains data for Charlie.

### 1.2.4 Data Types

A **data type** is a classification of data based on the type of values it can hold. MySQL supports a wide range of data types, including numeric, string, date, and time data types. Data types are used to specify the type of data that can be stored in a column in a table.

#### 1.2.4.1 Numeric Data Types

Numeric data types are used to store numeric values, such as integers, decimals, and floating-point numbers. MySQL supports a variety of numeric data types, including **INT**, **DECIMAL**, **FLOAT**, **DOUBLE**, **TINYINT**, **SMALLINT**, **MEDIUMINT**, and **BIGINT**.

#### 1.2.4.2 Character and String Data Types

Character and string data types are used to sto andre text values, such as names, addresses, and descriptions. MySQL supports a variety of character and string data types, including **CHAR**, **VARCHAR**, and **TEXT**.

#### 1.2.4.3 Date and Time Data Types

Date and time data types are used to store date and time values, such as birthdates, appointment times, and event dates. MySQL supports a variety of date and time data types, including **DATE**, **TIME**, **DATETIME**, and **TIMESTAMP**.

#### 1.2.4.4 Binary Data Types

Binary data types are used to store binary data, such as images, audio files, and video files. MySQL supports a variety of binary data types, including **BLOB**, **MEDIUMBLOB**, and **LONGBLOB**.

#### 1.2.4.5 Boolean Data Type

The **BOOLEAN** data type is used to store boolean values, such as true or false. In MySQL, boolean values are represented as 1 for true and 0 for false.

### 1.2.5 NULL Values

A **NULL** value is a special value that represents the absence of a value. NULL values are used to indicate that a column does not contain any data. In MySQL, columns can be defined to allow NULL values, which means that the column can contain NULL values in addition to other values.

### 1.2.6 Comments

Comments are used to add explanatory notes to the SQL code. Comments are ignored by the MySQL server and are used to document the code for reference. Comments can be added to SQL code using the – or /* */ syntax.

```
-- This is a single-line comment

/*
  This is a multi-line comment
  that spans multiple lines
*/

SELECT * FROM users; -- This is a comment at the end of a line
```

Code 1.1: Example of a comment in SQL code

Code 1.1 shows an example of comments in SQL code. Comments can be added to SQL code using the – syntax for single-line comments and the /* */ syntax for multi-line comments.

# 2

# Managing Databases

## 2.1 Creating a Database

To create a new database in MySQL, you can use the **CREATE DATABASE** statement followed by the name of the database. The following example creates a new database named **database_name**:

```
CREATE DATABASE database_name;
```

Code 2.1: Creating a new database in MySQL

Code 2.1 shows an example of creating a new database in MySQL using the **CREATE DATABASE** statement. The statement creates a new database named **database_name**. Once the database is created, you can use it to store tables and data.

## 2.2 Showing Databases

To display a list of all databases in MySQL, you can use the **SHOW DATABASES** statement. The following example shows a list of all databases in MySQL:

```
SHOW DATABASES;
```

Code 2.2: Showing all databases in MySQL

Code 2.2 shows an example of displaying a list of all databases in MySQL using the **SHOW DATABASES** statement. The statement lists all databases that are currently available in the MySQL server.

## 2.3 Selecting a Database

To select a database in MySQL, you can use the **USE** statement followed by the name of the database. The following example selects the **database_name** database:

```
USE database_name;
```

Code 2.3: Selecting a database in MySQL

Code 2.3 shows an example of selecting a database in MySQL using the **USE** statement. The statement selects the **database_name** database, which allows you to perform operations on the tables and data in that database.

## 2.4 Removing a Database

To remove a database in MySQL, you can use the **DROP DATABASE** statement followed by the name of the database. The following example removes the **database_name** database:

```
1 DROP DATABASE database_name;
```

Code 2.4: Removing a database in MySQL

Code 2.4 shows an example of removing a database in MySQL using the **DROP DATABASE** statement. The statement deletes the **database_name** database and all tables and data stored in that database.

**Exercises**

```
1. Create a new database named "employees".
2. Display a list of all databases in MySQL.
3. Select the "employees" database.
4. Remove the "employees" database.
```

# 3

# Managing Tables

## 3.1 Creating a Table

To create a new table in MySQL, you can use the **CREATE TABLE** statement followed by the name of the table and a list of columns. The following example creates a new table named **table_name** with three columns: **id**, **name**, and **age**:

```
CREATE TABLE table_name (
  id INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(50),
  age INT
);
```

Code 3.1: Creating a new table in MySQL

Code 3.1 shows an example of creating a new table in MySQL using the **CREATE TABLE** statement. The statement creates a new table named **table_name** with three columns: **id**, **name**, and **age**. Each column is defined with a data type (INT or VARCHAR) and a maximum length (50 for VARCHAR).

## 3.2 Showing Tables

To display a list of all tables in a database in MySQL, you can use the **SHOW TABLES** statement. The following example shows a list of all tables in the current database:

```
SHOW TABLES;
```

Code 3.2: Showing all tables in MySQL

Code 3.2 shows an example of displaying a list of all tables in the current database in MySQL using the **SHOW TABLES** statement. The statement lists all tables that are currently available in the database.

**Exercises**

```
1. Display a list of all tables in the current database.
```

## 3.3 Describe Table

To display the structure of a table in MySQL, you can use the **DESCRIBE** statement followed by the name of the table. The following example shows the structure of the **table_name** table:

```
1 DESCRIBE table_name;
```

Code 3.3: Describing a table in MySQL

Code 3.3 shows an example of displaying the structure of a table in MySQL using the **DESCRIBE** statement. The statement shows the columns, data types, and other properties of the **table_name** table.

## 3.4 Renaming a Table

To rename a table in MySQL, you can use the **RENAME TABLE** statement followed by the current name of the table and the new name of the table. The following example renames the **old_table** table to **new_table**:

```
1 RENAME TABLE old_table TO new_table;
```

Code 3.4: Renaming a table in MySQL

Code 3.4 shows an example of renaming a table in MySQL using the **RENAME TABLE** statement. The statement renames the **old_table** table to **new_table**.

## 3.5 Dropping a Table

To drop a table in MySQL, you can use the **DROP TABLE** statement followed by the name of the table. The following example drops the **table_name** table:

```
1 DROP TABLE table_name;
```

Code 3.5: Dropping a table in MySQL

Code 3.5 shows an example of dropping a table in MySQL using the **DROP TABLE** statement. The statement deletes the **table_name** table and all data stored in that table.

**Exercises**

```
1. Create a database named "library".
2. Create a table named "books" with columns:
   - id (INT, PRIMARY KEY, AUTO_INCREMENT)
```

```
    - title (VARCHAR, 100)
    - author (VARCHAR, 50)
    - year (INT)
3. Create a table named "users" with columns:
    - id (INT, PRIMARY KEY, AUTO_INCREMENT)
    - name (VARCHAR, 50)
    - email (VARCHAR, 100)
    - is_staff (BOOLEAN)
4. Display the structure of the "books" table.
5. Rename the "users" table to "members".
6. Drop the "members" table.
```

## 3.6 Temporary Tables

Temporary tables are a special type of table that is created and destroyed automatically by the MySQL server. Temporary tables are useful for storing temporary data that is only needed for the duration of a session or a query. Temporary tables are created using the **CREATE TEMPORARY TABLE** statement and are automatically dropped when the session ends.

```sql
CREATE TEMPORARY TABLE temp_table (
  id INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(50)
);
```

Code 3.6: Creating a temporary table in MySQL

Code 3.6 shows an example of creating a temporary table in MySQL using the **CREATE TEMPORARY TABLE** statement. The statement creates a temporary table named **temp_table** with two columns: **id** and **name**. The temporary table is automatically dropped when the session ends.

**Exercises**

```
1. Create a temporary table named "temp_data" with columns:
    - id (INT, PRIMARY KEY, AUTO_INCREMENT)
    - value (VARCHAR, 50)
2. Display a list of all tables in the current database.
```

# 4

# References

**A. Books**

- 

**B. Other Sources**

- 