Content

[INTRODUCTION](#_Toc136882007)

1. [Download and install MySQL installer](#_Toc136882007)

2. [Creating a database and tables](#_Toc136882007)

2.1 Create new MySQL user and grant privileges

3. [Directory settings](#_Toc136882007)

[CONCLUSION](#_Toc136882008)

**INTRODUCTION**

This project aims to guide you through the process of setting up a C++ application that interacts with a MySQL database. By following the steps outlined below, you will learn how to download and install the MySQL installer, create a database and tables, create a new MySQL user, grant privileges, and configure directory settings in Visual Studio 2019.

**1 Download and install MySQL installer**

The first step is to download the MySQL Installer from the official MySQL website (<https://dev.mysql.com/downloads/windows/installer/8.0.html>). Select the version of MySQL Installer that is appropriate for your operating system. As of 07/09/2023, this is version "8.0.33". My operating system is Windows (Fig. 1.1 - 1.2)

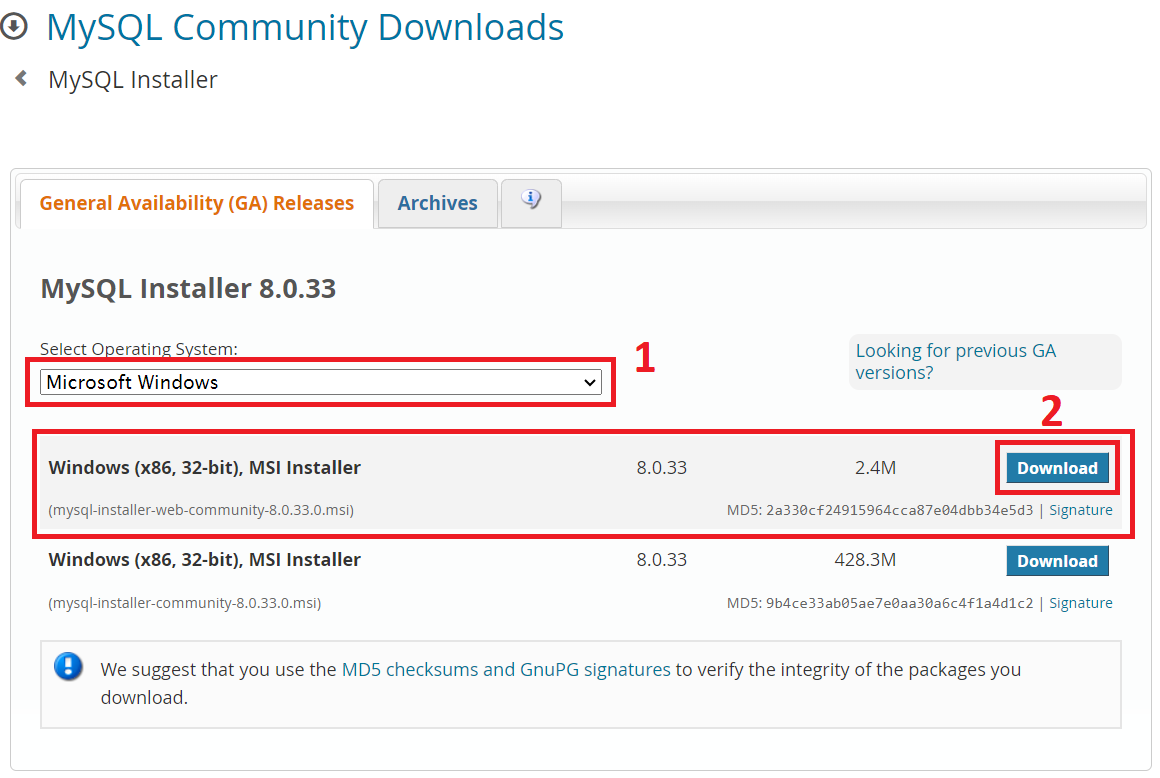


Fig. 1.1 - Download MySQL installer

Next, we install the MySQL installer (it shouldn't be difficult), so I'll show you a few installation steps. This is illustrated in (Fig. 1.2)

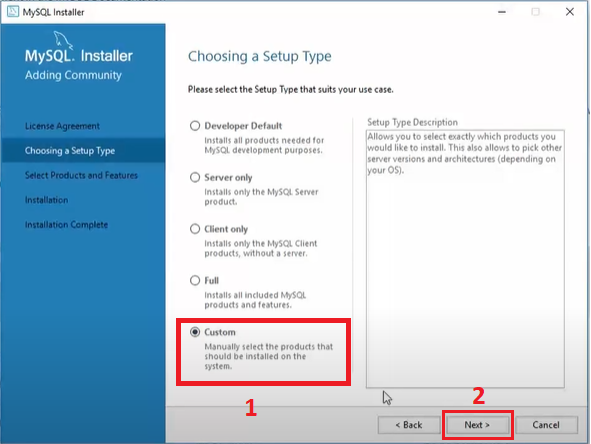


Fig. 1.2 – Installer MySQL, steps 1

Next, it is necessary to select "products and functions", as illustrated in (Fig. 1.3). Here we only need to install MySQL Server, also note that this is for 64-bit version. We also select MySQL Connector/C++.

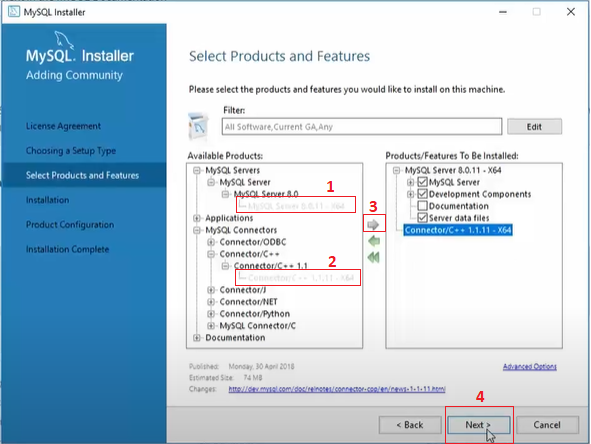


Fig. 1.3 – Installer MySQL, steps 2

All the following settings and steps are left as default (in my case)

**2 Creating a database and tables**

To execute requests to create a database and tables, we will use the MYSQL Command Line Client. Open MySQL Command Line Client and enter the password that was created during the MySQL installation. As it is illustrated in (Fig. 2.1)

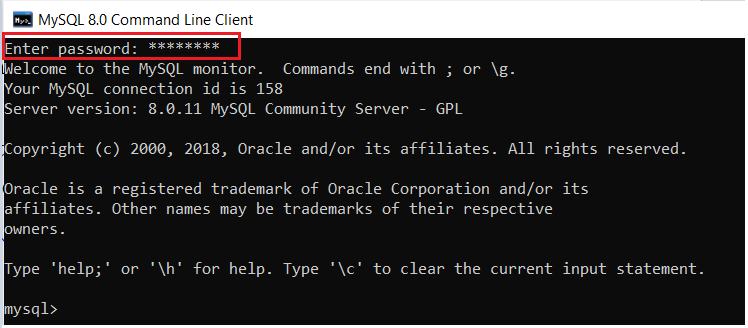


Fig. 2.1 – Login to the MySQL server

Next, we will directly execute requests to create a database and tables. ***However, it is worth noting that if you are using MySQL for the first time - you need to create a user and give him the appropriate rights (I will talk about this in section 2.1, so go to the next section and then come back here)***

The request to create a database in MySQL will look like this:

|  |
| --- |
| CREATE DATABASE database\_name; |

Instead of "database\_name" you should specify the desired name for your database. For example, if you want to create a database named "DB\_Test", the query would look like this:

|  |
| --- |
| CREATE DATABASE DB\_Test; |

Run this query on the MySQL command line and it will create a new database with the specified name. As it is illustrated in (Fig. 2.2)

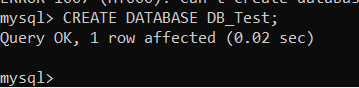


Fig. 2.2 – Creating a "DB\_Test" database

Our database will contain 2 tables, the first for users, the second for administrators. Both tables will be identical and have the following fields:

* Id (identifier)
* Login
* Password
* Email
* Reg\_Date (registration date)

To create a table, execute the following query, where «users» is the name of the table. It is also worth noting the keyword UNIQUE, which makes the field unique.

|  |
| --- |
| CREATE TABLE users (  id INT AUTO\_INCREMENT PRIMARY KEY,  Login VARCHAR(20) NULL UNIQUE,  Password VARCHAR(20) NULL,  Email VARCHAR(30) NULL UNIQUE,  Reg\_Date DATE NULL  ); |

Enter this query at the MySQL command line, and it will create a new table with the specified structure. As it is illustrated in (Fig. 2.3)

But from the first, it is necessary to connect to the newly created database, which we will create in (Fig. 2.2), using the word "Use" and database name.

|  |
| --- |
| Use DB\_Test; |

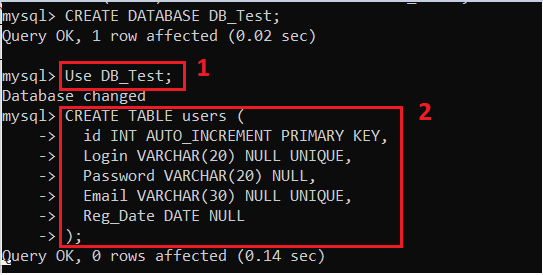


Fig. 2.3 – Creating the "users" table

Creating the next Table "Administrators" (Fig. 2.4):

|  |
| --- |
| CREATE TABLE Administrators (  id INT AUTO\_INCREMENT PRIMARY KEY,  Login VARCHAR(20) NULL UNIQUE,  Password VARCHAR(20) NULL,  Email VARCHAR(30) NULL UNIQUE,  Reg\_Date DATE NULL  ); |

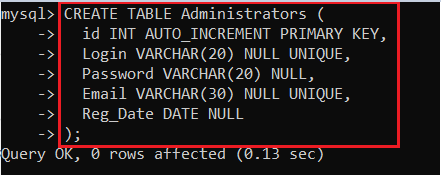


Fig. 2.4 – Creating the "Administrators" table

After creating the tables, you can check their availability using the "Show tables;" command, as illustrated in (Fig. 2.5)

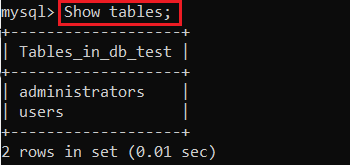


Fig. 2.5 – Viewing created tables

**2.1 Create new MySQL user and grant privileges**

For security reasons, it is generally better to create and handle data as specific users.

**How to Create New MySQL User**

1. Before you can create a new MySQL user, you need to open a terminal window (MySQL Command Line).

2. Type the password for this account and press Enter (Fig. 2.1.1). Type the password for this account and press Enter. (The password was created during MySQL installation, Root Password)

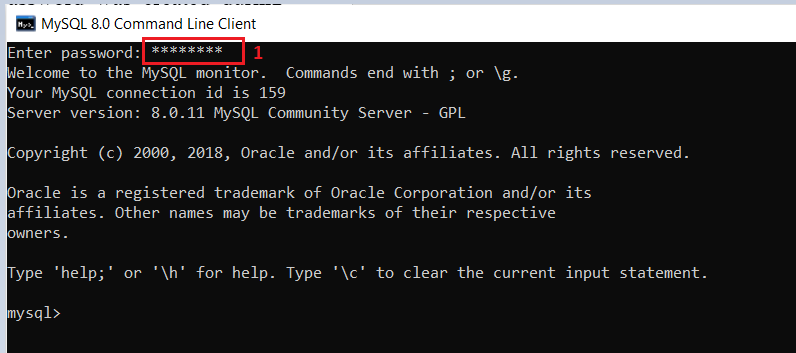


Fig. 2.1.1 – Entering a password

3. Next, create a new MySQL user with:

To create a new user in MySQL, you can use the CREATE USER statement. Here's the basic syntax:

|  |
| --- |
| CREATE USER 'username'@'localhost' IDENTIFIED BY 'password'; |

Let me break down the syntax for you:

* **'username'** represents the name of the user you want to create. You can choose any name you prefer.
* **'localhost'** specifies the host from which the user is allowed to connect. In this case, it's set to localhost, which means the user can only connect from the same machine where MySQL is running. You can change it to a specific IP address or hostname if you want to allow connections from other machines.
* **'password'** is the password you want to set for the user. Replace it with the actual password you want to use.

Here's an example to create a new user named **'myuser'** with the password **'mypassword'**:

|  |
| --- |
| CREATE USER 'myuser'@'localhost' IDENTIFIED BY 'mypassword'; |

Once you run this SQL statement, the user will be created. However, this user won't have any privileges or permissions by default. Creating a new user is illustrated in (Fig. 2.1.2)

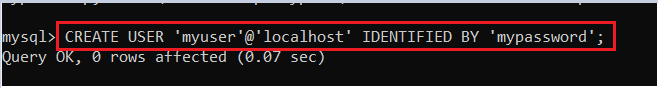


Fig. 2.1.2 – Creating a new user

**How to Grant Permissions in MySQL**

Before logging in with a new account, make sure you have set the permissions for the user.

Permissions are actions that the user is allowed to perform in the database. Depending on how much authority you want your user to have, you can grant them one, several or all of the following privileges:

* All Privileges: The user account has full access to the database
* Insert: The user can insert rows into tables
* Delete: The user can remove rows from tables
* Create: The user can create entirely new tables and databases
* Drop: The user can drop (remove) entire tables and databases
* Select: The user gets access to the select command, to read the information in the databases
* Update: The user can update table rows
* Grant Option: The user can modify other user account privileges

If you want to grant specific privileges to the user, you can use the GRANT statement. For example, to ***grant all privileges*** on a specific database to the user, you can use:

|  |
| --- |
| GRANT ALL PRIVILEGES ON your\_database.\* TO 'myuser'@'localhost'; |

Remember to replace 'your\_database' with the actual name of your database.

After running the GRANT statement, the user 'myuser' will have all privileges on the specified database.

Grant all privileges is illustrated in (Fig. 2.1.3)

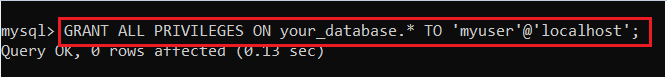


Fig. 2.1.3 – Grant all privileges

**3 Directory settings**

Right click on your project and go to «Properties». After that, first go to the «Configuration manager», and change the configuration to "Release". As shown in (Fig. 3.1)

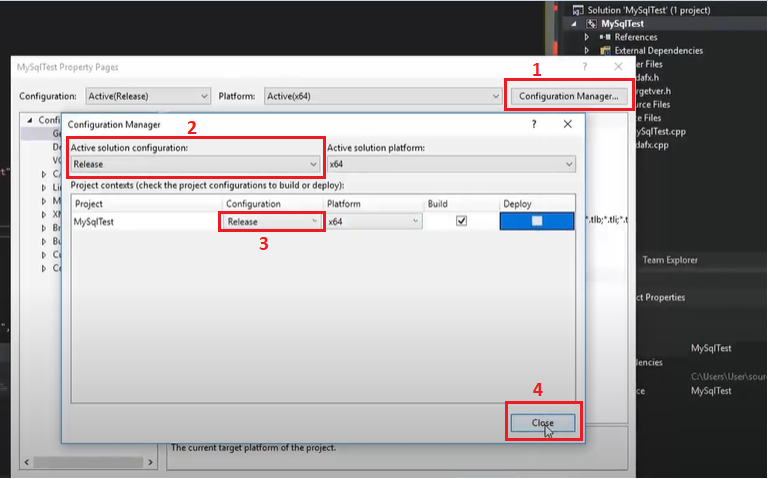


Fig. 3.1 – Change the configuration to "Release"

Inclusion of directories.

In the «Properties pages» window, under the configuration section, change to "All configurations". Go to "C/C++" -> "General" -> "Additional Include Directories" and click on "edit". As shown in (Fig. 3.2).

Here we need to include 2 directories: Connector c++ and MySQL Server (Fig. 3.3). If you installed MySQL based on the default setup. Then the default folders will have the following path:

* C:\Program Files\MySQL\Connector C++ 1.1\include
* C:\Program Files\MySQL\MySQL Server 8.0\include

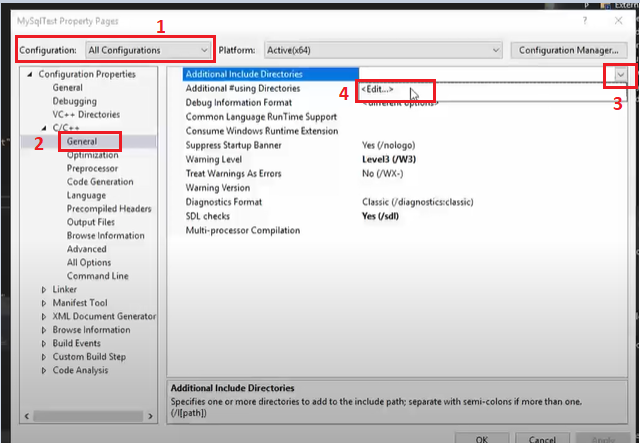


Fig. 3.2 – Additional Include Directories

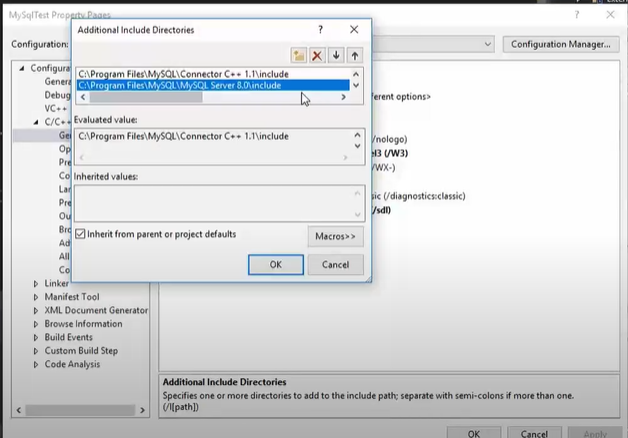


Fig. 3.3 – Including: «Connector С++» and «MySQL Server» (include)

The next step is to connect the necessary directories for "Linker". Go to the "Linker" -> "General" -> "Additional Library Directories" (Fig. 3.4).

Here we need to include 2 directories: Connector c++ (lib) and MySQL Server (lib) (Fig. 3.5). The default folders will have the following path:

* C:\Program Files\MySQL\Connector C++ 1.1\lib\opt
* C:\Program Files\MySQL\MySQL Server 8.0\lib

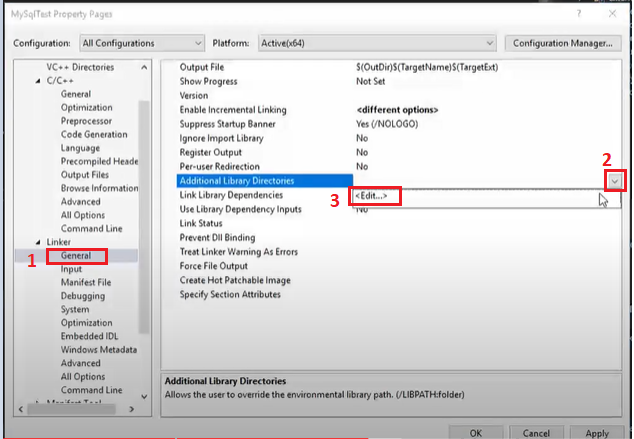


Fig. 3.4 – Additional Library Directories

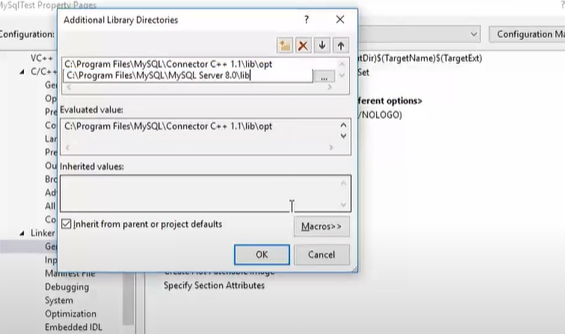


Fig. 3.5 – Including: «Connector С++» and «MySQL Server» (lib)

Next, go to the "Input" tab and select the files we need “Lib” (Fig. 3.6). Let's add two libraries: ***mysqlcppconn.lib*** and ***libmysql.lib (***Fig. 3.7).

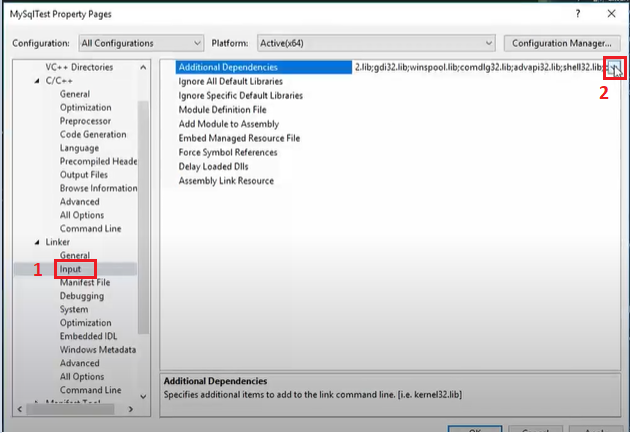


Fig. 3.6 – Additional Dependencies

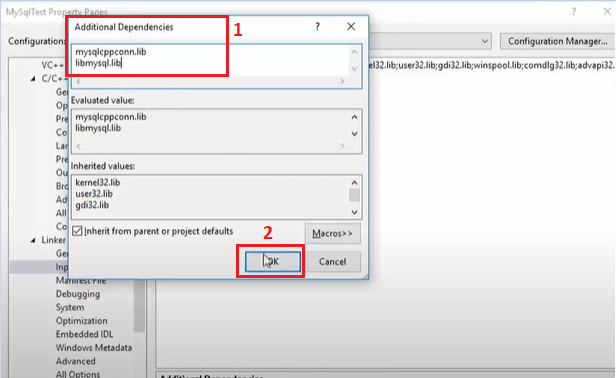


Fig. 3.7 - add two libraries: ***mysqlcppconn.lib*** and ***libmysql.lib***

Next, we will place the .dll (***libeay32.dll***, ***libmysql.dll***, ***mysqlcppconn.dll*** and ***ssleay32.dll***) according to the directory of executable files. Copy the files and paste them into the entire project directory (Fig. 3.8).

* ***libeay32.dll*** default is located : C:\ProgramFiles\MySQL\MySQL Server 8.0\bin
* ***ssleay32.dll*** default is located : C:\ProgramFiles\MySQL\MySQL Server 8.0\bin
* ***libmysql.dll***  default is located : C:\Program Files\MySQL\MySQL Server 8.0\lib
* ***mysqlcppconn.dll***  default is located : C:\Program Files\MySQL\Connector C++ 1.1\lib\opt

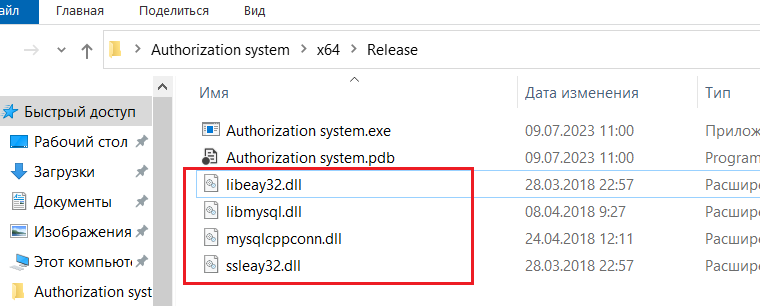


Fig. 3.8 - Place the .dll according to the directory of executable files.

**Conclusion**

After following the steps described in this report, our project is fully functional with MySQL and able to perform all necessary database queries. We have successfully configured the environment to allow our C++ application to interact with the MySQL server.

After installing MySQL and creating the database and tables, we have a proper structure to store our data. We also created a new MySQL user and gave him the necessary privileges for secure access to the database.

Setting up directories in Visual Studio 2019 ensured that our project was properly integrated with the MySQL libraries and header files.

As a result, our project fully functions with MySQL and can perform all necessary database queries. We have successfully combined the power of the C++ programming language with the capabilities of the MySQL database, which allows us to develop reliable and efficient applications for working with large amounts of data. The knowledge gained during this project will be useful for the further development and expansion of our skills in programming and working with databases.