Implementation of MySQL in Python

Anushree Raj, Rio D’Souza

1Assistant Professor, 2Professor
1M.Sc. Big Data Analytics, 2Computer Science and Engineering
1St Agnes Centre for PG Studies and Research, Mangalore, India, 2St joseph Engineering College, Vamanjoor, India

Abstract: Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. MySQL is an Open-Source database and one of the best type of RDBMS. From a Python program, we can connect to MySQL database to access the tables and manipulate the data. In the first section we show how to create database using mysql workbench. MySQL supports standard SQL. SQL is a simple language that is used to query database, and it can be used in mySQL, Oracle, Access, and many others. In the second section, a description of how to make the connection between python and database is provided. In the third section, a review of the basic SQL statements that can be implemented. In the fourth section, the main database operations are performed using python.

Keywords: CRUD, MySQL Workbench, Python, Python Database.

1. INTRODUCTION

Python is an object-oriented programming language which is easy to learn and easy to read. It comes with a large collection of build-in functionality but can easily be extended. Python is used for scripting and web development, but it is also embed it into application or can for creation graphical games for example. MySQL is a Relational Database Management System (“RDBMS”). It is used by most modern websites and web-based services as a convenient and fast-access storage and retrieval solution for large volumes of data. There are several ways you can connect to a MySQL database using Python. The Structured Query Language (SQL) is the language of databases. SQL supports a small but very powerful set of statements for manipulating, managing, and protecting data stored in a database.

2. MYSQL WORKBENCH

MySQL is an open source relational database. MySQL is cross platform which means it runs on a number of different platforms such as Windows, Linux, and Mac OS etc. MySQL is an open source relational database [1]. MySQL is cross platform which means it runs on a number of different platforms such as Windows, Linux, and Mac OS etc. MySQL supports multiple storage engines which greatly improve the server performance tuning and flexibility. MySQL Workbench functionality covers five main topics:

- **SQL Development**: Enables to create and manage connections to database servers.
- **Data Modeling (Design)**: Enables to create models of your database schema graphically, reverse and forward engineer between a schema and a live database, and edit all aspects of your database using the comprehensive Table Editor.
- **Server Administration**: Enables to administer MySQL server instances.
- **Data Migration**: Allows to migrate from Microsoft SQL Server, Microsoft Access, Sybase ASE, SQLite, SQL Anywhere, PostreSQL, and other RDBMS tables, objects and data to MySQL.
- **MySQL Enterprise Support**: Support for Enterprise products such as MySQL Enterprise Backup, MySQL Firewall, and MySQL Audit.

MySQL workbench is an integrated development environment for MySQL server. It has utilities for database modeling and designing, SQL development and server administration [2].

Fig 1: Create table using MySQL Workbench
An employee table is created using MySQL workbench with four attributes and ten rows for demonstration purpose.

![Employee Table](image)

Fig 2: Execution of SQL commands

Various queries where implemented to test different operations of SQL. The results obtained in MySQL workbench and the one through importing mysql in Python were same.

3. **BASIC MYSQL STATEMENTS**

SQL (Structured Query Language) is a standard language for relational databases that allow users to do various operations on data like, Manipulating, Creating, Dropping, etc. SQL allows you to do anything with the data. The SQL language is responsible to deal with database. The process of adding new record, retrieving a record, updating existing record and delete a record are the four main database operations [3].

Here, a short description of the four basic SQL statements is presented if we have a table called Employee, which includes four attributes:

1. The employee ID
2. The employee Name
3. The employee Gender
4. The employee Salary.

A view of the table could be as shown in Fig 3. Using the generated table, the basic operations could be performed as follows:

The database was executed to demonstrate the CRUD operations below [5]:

- **Create**- It is an SQL statement used to create a record in the table or can say it is used for creating a table.
- **Read**- It is used for fetching useful information from the database.
- **Update**- This particular SQL statement is used for updating the records in the table or updating the table.
- **Delete**- As the name itself justifies this command is used for deleting the table.

A. Add new record to the table. The SQL statement insert is used to add a new record to the table:

```sql
query = ('INSERT INTO employee (emp_id, emp_name, gender, Salaray) VALUES (111, 'Sam', 'male', 40000);')
```

The insert statement will insert data into the table employee which has the following attributes (‘emp_id’, ‘emp_name’, ‘gender’, ‘Salary’) with the following values (111, ‘Sam’, male, 40000) and the values should be ordered in the same order of attributes. After executing the statement, the table will look as shown in Fig 3.
B. Search SQL Select statement is used to retrieve a set of columns from the table as follows:
   • Select all column from table student:
     ```sql
     SELECT * from Students
     ```
   • Select set of column from student:
     ```sql
     SELECT emp_id, emp_name from employee
     ```
     Here it selects the whole rows but only for emp_id and emp_name columns
   • Select Specific row or set of rows and all or set of columns
     ```sql
     SELECT emp_id, emp_name from employee where emp_id = 105
     ```
     This statement return the emp_id and the emp_name of only one row which has emp_id = 105 because emp_id should be the primary key of the table. We can use different statement to return more than one row by searching a specific value. SELECT emp_id, emp_name from employee WHERE gender = 'female'

C. Update a value inside the table: You must first locate the row that you want to update then give it a new value as follows:
   Update employee set emp_name = ‘Sonu’ WHERE emp_id = ‘106’
   Here the name of the employee with emp_id = 106 will changed from Banu to Sonu.

D. Delete From the table Again you first locate the row or set of rows you would like to delete, then delete them:
   Delete from employee where emp_id = 107
   This will delete the row of the employee has ID =107. We can perform delete for more than one row, by specifying the condition in the where clause
   Delete from employee where gender = male
   Here, all male employee details will be deleted from the table

4. CONNECT PYTHON TO MYSQL

   The Python standard for database interfaces is the Python DB-API. Most Python database interfaces adhere to this standard. You can choose the right database for your application. Python Database API supports a wide range of database servers such as:
   − GadFly, mSQL, MySQL, PostgreSQL, Microsoft SQL Server 2000, Informix, Interbase, Oracle, Sybase [5].
   The DB API provides a minimal standard for working with databases using Python structures and syntax wherever possible. This API includes the following: Importing the API module.
   • Acquiring a connection with the database.
   • Issuing SQL statements and stored procedures.
   • Closing the connection

   In python file, we have to first establish a connection between the file and the database. Then we can add, search, delete or update the database. We can retrieve the data from the database, make any operation on it then re-add it to the database [6]. The database operations are performed using SQL statements.
5. IMPLEMENTING MYSQL IN PYTHON

MySQL is one of the most popular databases. We can download and install MySQL from the MySQL’s official website. Python needs a MySQL driver to access the MySQL database [7]. Here we will use the driver “MySQL Connector”. In this python script Mysql.connector.connect is a mysql-python library function that will create this connection. To execute this function import the library using “import mysql.connector”. We now connect to “database” MySQL database, and retrieve data from Employee table, and display it.

The connection between Python and MySQL is established as shown in the Fig 3. A connection request is sent to MySQL connector Python, gets accepted from the database and cursor is executed with result data [8].

After the connection is established, we can implement the MySql in Python by executing the commands in python program as shown in Fig 4. We use Python 3 and jupyter notebook to run the script.

Fig 3: Connection between Python and MySQL

Fig 4: Python code:

- Imports the mysql connector python library.
- Connects to a specific MySQL database using the given db name, username, password and host. This returns a connection object, which is stored in the “connection” variable.
- Using the connection object that we just created, we are creating a new cursor to retrieve and manipulate our query. The cursor object will be stored in the “eid” variable.
- sql_select_Query = (“SELECT * FROM employee”) – defines our MySQL query.
- Using the “eid” we are execute the given query.
- Fetch all the values in the rows and stores them in the given parameter.
- We print the value of the given variables, which are the column values retrieved for a particular row on the employee table.
- Finally, we close the cursor and the connections.
The following is the output of the above sample python program, which connects to a MySQL database and retrieves rows from the given table.

Fig 5: Output from Python

The queries where executed both in MySQL workbench and Python to check for the consistency of execution. We can perform any database operation, may it be a Create, Read, Update or Delete operation.

6. CONCLUSION

In this paper, we observed a very simple yet effective example of setting up a connection to a MySQL server from a Python application. We started with useful techniques to establish and close the connection with the server along with database connection pooling techniques which are very important to have in any production-grade application based on any framework and not just Python. Here we showed how to use MySQL Connector/Python to access MySQL databases and perform common database operations such as SELECT, INSERT, UPDATE and DELETE. We can extent our work to demonstrate techniques on how to call MySQL stored procedures from Python, and how to work with MySQL BLOB data.

References: