Integrating Python with MySQL Databases

The objective of this concept is to guide you through integrating Python with MySQL databases using the mysql-connector-python library. We’ll cover the installation process, explore connection establishment, and demonstrate basic database operations.

Concept Overview

Topics

* Introduction to mysql-connector-python
* Working with Cursors
* Executing SQL Queries (SELECT, INSERT, UPDATE, DELETE)
* A Complete Example

Learning Objectives

* Install and use the mysql-connector-python library
* Establish a connection to a MySQL database server
* Understand the concept of cursors in database interactions
* Execute basic SQL queries (SELECT, INSERT, UPDATE, DELETE) using Python

Introduction to mysql-connector-python

mysql-connector-python is a popular library that bridges the gap between Python and MySQL databases. It provides a comprehensive interface for interacting with MySQL servers, allowing you to execute queries, manipulate data, and manage database objects.

Installation

The recommended way to install mysql-connector-python is using pip, the Python package manager. Open your terminal or command prompt and run the following command, which download and install the library.

pip install mysql-connector-python

Connecting to MySQL Database

Once you have mysql-connector-python installed, we can start interacting with your MySQL database server. Here’s an example of how to connect:

import mysql.connector

# Replace with your connection details

mydb = mysql.connector.connect(

host="localhost",

user="yourusername",

password="yourpassword",

database="yourdatabase"

)

print(mydb.get\_server\_info())

This code snippet imports the library, establishes a connection using the connect method, and retrieves some server information using get\_server\_info. Remember to replace the connection details with your actual credentials and database name.

Working with Cursors

Cursors are objects used to execute SQL statements and fetch results from the database. You can create a cursor using the cursor method of the connection object. With the cursor in hand, you can execute various SQL queries. Once you’re finished working with the database, ensure you close the connection to release resources

Here are some examples:

import mysql.connector

# Replace with your connection details

mydb = mysql.connector.connect(

host="localhost",

user="yourusername",

password="yourpassword",

database="yourdatabase"

)

mycursor = mydb.cursor()

# Execute SQL statements using the execute() method on the cursor

# Close connection to the databasse

mycursor.close()

mydb.close()

Executing SQL Queries (SELECT, INSERT, UPDATE, DELETE)

SELECT Retrieve data from a table

mycursor.execute("SELECT \* FROM your\_table")

myresult = mycursor.fetchall()

for row in myresult:

print(row)

INSERT Insert new data into a table

sql = "INSERT INTO your\_table (name, email) VALUES (%s, %s)"

val = ("John", "john@example.com")

mycursor.execute(sql, val)

mydb.commit() # Commit the changes

UPDATE Modify existing data in a table

sql = "UPDATE your\_table SET name = %s WHERE id = %s"

val = ("Jane", 1)

mycursor.execute(sql, val)

mydb.commit() # Commit the changes

DELETE Remove data from a table

sql = "DELETE FROM your\_table WHERE id = %s"

val = (2,)

mycursor.execute(sql, val)

mydb.commit() # Commit the changes

A Complete Example

The below code first establishes a connection to your MySQL database server. Then, it creates a table named customers if it doesn’t already exist. It demonstrates inserting two customer records, followed by reading all customer data using a SELECT statement.

Next, the code updates the email address of a customer with ID 1 and retrieves the updated record. Finally, it deletes the customer with ID 2 and closes the database connections.

import mysql.connector

# Database connection details (replace with your own)

mydb = mysql.connector.connect(

host="localhost",

user="yourusername",

password="yourpassword",

database="mydb"

)

mycursor = mydb.cursor()

# Create a table named `customers` (if it doesn't exist)

mycursor.execute("""

CREATE TABLE IF NOT EXISTS customers (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

email VARCHAR(255) NOT NULL UNIQUE

)

""")

print("Table created successfully!")

# Insert some customer data

sql = "INSERT INTO customers (name, email) VALUES (%s, %s)"

val = ("John Doe", "john.doe@example.com")

mycursor.execute(sql, val)

mydb.commit()

print(mycursor.rowcount, "record(s) inserted.")

val = ("Jane Smith", "jane.smith@example.com")

mycursor.execute(sql, val)

mydb.commit()

print(mycursor.rowcount, "record(s) inserted.")

# Read all customer data

mycursor.execute("SELECT \* FROM customers")

myresult = mycursor.fetchall()

print("Customers:")

for row in myresult:

print(row)

# Update a customer's email

sql = "UPDATE customers SET email = %s WHERE id = %s"

val = ("updated.email@example.com", 1)

mycursor.execute(sql, val)

mydb.commit()

print(mycursor.rowcount, "record(s) updated.")

# Read the updated customer data

mycursor.execute("SELECT \* FROM customers WHERE id = 1")

myresult = mycursor.fetchone()

print("Updated customer:")

print(myresult)

# Delete a customer

sql = "DELETE FROM customers WHERE id = 2"

mycursor.execute(sql)

mydb.commit()

print(mycursor.rowcount, "record(s) deleted.")

# Close connections

mycursor.close()

mydb.close()

print("Database connection closed.")

Practice Exercises

**Challenge: Library Management System** Imagine you’re building a simple library management system using Python and a MySQL database. This system will allow you to store information about books (title, author, ISBN) and perform basic operations like adding new books, searching by title, and listing all books in the library.

**Tasks:** \* Setting Up: \* Install the mysql-connector-python library using pip install mysql-connector-python. \* Create a MySQL database and a table named books with columns for id (INT, Auto-Increment primary key), title (VARCHAR(255)), author (VARCHAR(255)), and ISBN (VARCHAR(255)).

* Connect to Database:
  + Write Python code to connect to your MySQL database using the provided connection details (replace placeholders with your actual credentials).
  + Adding Books:
  + Implement a function that takes book details (title, author, ISBN) as input and inserts a new record into the books table using an INSERT query.
* Searching Books:
  + Write a function that allows users to search for books by title. It should use a SELECT query with a WHERE clause to filter results based on the user’s input. Print the details of any matching books.
* Listing All Books:
  + Create a function that retrieves all book information from the books table using a SELECT query. Print the details of all books in a user-friendly format.
* (Bonus Challenge:
  + Implement functionality to delete a book from the library by its ID. Use a DELETE query with a WHERE clause to remove the specified book.

**Remember:** Make sure to commit any changes to the database using mydb.commit(). Close the connection to the database using mycursor.close() and mydb.close() when you’re finished.

**Additional Resources**

* [Python and MySQL Database: A Practical Introduction](https://savanna.alxafrica.com/rltoken/T4iz2WnPTNLrArWIjLKTHA)