Experiment No. 8

Title: CRUD operations accomplishment using python

Batch: A3 **Roll No:** 16010423099 **Experiment No.:** 8

Aim: Program on CRUD operations accomplishment using python

Resources needed: Python IDE

Theory:

What is CRUD?

CRUD stands for Create, Read, Update, and Delete, which are the four basic operations that are performed on a database. These operations allow users to manage data in databases by:

- Create: Adding new data (e.g., inserting a new record).
- Read: Retrieving and reading data (e.g., querying records).
- Update: Modifying existing data (e.g., changing a record's content).
- Delete: Removing data (e.g., deleting records from a table).

CRUD Operations Using Python

In Python, CRUD operations can be performed using various database management systems (DBMS) like SQLite, MySQL, PostgreSQL, etc. Python has libraries such as sqlite3, SQLAlchemy, and mysql-connector that help you interface with databases.

In this example, we will use SQLite with Python's built-in sqlite3 library to perform CRUD operations.

Steps to Perform CRUD Operations in Python

Step 1: Set Up Python and SQLite

Python includes an sqlite3 library that allows you to connect to SQLite databases without any additional installation.

Step 2: Import Required Library

import the sqlite3 library to interact with an SQLite database in Python.

python

```
import sqlite3
```

Step 3: Connect to the Database

to create a connection to the SQLite database. If the database doesn't exist, it will automatically be created.

```
conn = sqlite3.connect('database_name.db')
cursor = conn.cursor()
```

Step 4: Create a Table (CREATE operation)

To store data, you need to create a table inside your database.

```
cursor.execute('''
    CREATE TABLE IF NOT EXISTS users (
        id INTEGER PRIMARY KEY,
        name TEXT,
        age INTEGER
    )
''')
```

Step 5: Insert Data into the Table (CREATE operation)

```
def create_user(name, age):
    cursor.execute('INSERT INTO users (name, age) VALUES (?, ?)', (name, age))
    conn.commit()
```

To insert a new user:

```
create_user('John Doe', 30)
```

Step 6: Read Data from the Table (READ operation)

```
def read_users():
    cursor.execute('SELECT * FROM users')
    users = cursor.fetchall()
    for user in users:
        print(user)
```

To read all users:

```
read_users()
```

Step 7: Update Data in the Table (UPDATE operation)

```
def update_user(user_id, name, age):
```

```
    cursor.execute('UPDATE users SET name = ?, age = ? WHERE id = ?', (name, age, user_id))
    conn.commit()
```

To update a user:

```
update_user(1, 'Jane Smith', 25)
```

Step 8: Delete Data from the Table (DELETE operation)

```
def delete_user(user_id):
    cursor.execute('DELETE FROM users WHERE id = ?', (user_id,))
    conn.commit()
```

To delete a user:

```
delete_user(1)
```

Step 9: Close the Connection

```
conn.close()
```

Commands for CRUD Operations

Operation	SQLite SQL Command	Python Command Example
Create	INSERT INTO	cursor.execute('INSERT INTO table (col1, col2) VALUES (?, ?)', (value1, value2))
Read	SELECT	cursor.execute('SELECT * FROM table')
Update	UPDATE	cursor.execute('UPDATE table SET col1 = ? WHERE col2 = ?', (value1, value2))
Delete	DELETE	cursor.execute('DELETE FROM table WHERE col1 = ?', (value1,))

Activities:

Write a program to code for accomplishing CRUD (Create, Read, Update, Delete) operations using Python and SQLite as the database:

Create: Add new records to the database.

Read: Fetch data from the database.

Update: Modify existing records.

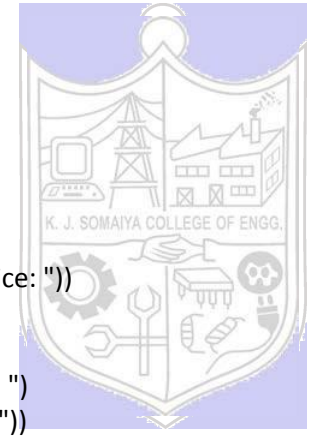
Delete: Remove records from the database.

Result and Implementation

CODE:

```
import sqlite3

def connect_to_db():
    conn = sqlite3.connect('users.db')
    cursor = conn.cursor()
    return conn, cursor

def create_table(cursor):
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS users (
            id INTEGER PRIMARY KEY,
            name TEXT NOT NULL,
            age INTEGER NOT NULL
        )
    ''')
    print("Table created successfully.")

def create_user(cursor, conn, name, age):
    cursor.execute('INSERT INTO users (name, age) VALUES (?, ?)', (name, age))
    conn.commit()
    print(f"User {name} added successfully.")

def read_users(cursor):
    cursor.execute('SELECT * FROM users')
    users = cursor.fetchall()
    if users:
        print("ID | Name | Age")
        for user in users:
            print(user)
    else:
        print("No users found.")

def update_user(cursor, conn, user_id, name, age):
```

```
    cursor.execute('UPDATE users SET name = ?, age = ? WHERE id = ?', (name, age, user_id))
    conn.commit()
    print(f"User with ID {user_id} updated successfully.")

def delete_user(cursor, conn, user_id):
    cursor.execute('DELETE FROM users WHERE id = ?', (user_id,))
    conn.commit()
    print(f"User with ID {user_id} deleted successfully.")

def close_connection(conn):
    conn.close()
    print("Connection closed.")

def crud_menu():
    conn, cursor = connect_to_db()
    create_table(cursor)

    while True:
        print("\nCRUD Menu:")
        print("1. Create User")
        print("2. Read Users")
        print("3. Update User")
        print("4. Delete User")
        print("5. Exit")

        choice = int(input("Enter your choice: "))

        if choice == 1:
            name = input("Enter user name: ")
            age = int(input("Enter user age: "))
            create_user(cursor, conn, name, age)

        elif choice == 2:
            read_users(cursor)

        elif choice == 3:
            user_id = int(input("Enter user ID to update: "))
            name = input("Enter new user name: ")
            age = int(input("Enter new user age: "))
            update_user(cursor, conn, user_id, name, age)

        elif choice == 4:
            user_id = int(input("Enter user ID to delete: "))
            delete_user(cursor, conn, user_id)

        elif choice == 5:
            close_connection(conn)
```



```
        break

    else:
        print("Invalid choice, please try again.")

crud_menu()
```

OUTPUT:

```
Table created successfully.

CRUD Menu:
1. Create User
2. Read Users
3. Update User
4. Delete User
5. Exit
Enter your choice: 1
Enter user name: qw
Enter user age: 12
User qw added successfully.

CRUD Menu:
1. Create User
2. Read Users
3. Update User
4. Delete User
5. Exit
Enter your choice: 2
ID | Name | Age
(1, 'Dhruti', 19)
(2, 'qw', 12)

CRUD Menu:
1. Create User
2. Read Users
3. Update User
4. Delete User
5. Exit
Enter your choice: 3
Enter user ID to update: 2
Enter new user name: gregor
Enter new user age: 100
User with ID 2 updated successfully.
```

```
CRUD Menu:
1. Create User
2. Read Users
3. Update User
4. Delete User
5. Exit
Enter your choice: 2
ID | Name | Age
(1, 'Dhruti', 19)
(2, 'gregor', 100)

CRUD Menu:
1. Create User
2. Read Users
3. Update User
4. Delete User
5. Exit
Enter your choice: 4
Enter user ID to delete: 1
User with ID 1 deleted successfully.

CRUD Menu:
1. Create User
2. Read Users
3. Update User
4. Delete User
5. Exit
Enter your choice: 2
ID | Name | Age
(2, 'gregor', 100)

CRUD Menu:
1. Create User
2. Read Users
3. Update User
4. Delete User
5. Exit
Enter your choice: 5
Connection closed.
```

Outcomes:

CO4: Demonstrate handling database with python and to understand network programming with Python scapy.

Questions:

- What are the key steps involved in performing CRUD operations using Python with SQLite, and how can you establish a connection to an SQLite database?

ANS:

The key steps for performing CRUD operations with Python and SQLite are:

1. **Connect to the database** using `sqlite3.connect('database_name.db')`.
2. **Create a cursor** to execute SQL commands with `conn.cursor()`.
3. **Perform CRUD operations**:
    - **Create**: `INSERT INTO` to add records.
    - **Read**: `SELECT` to retrieve data.
    - **Update**: `UPDATE` to modify records.
    - **Delete**: `DELETE` to remove records.
4. **Commit changes** with `conn.commit()` and **close the connection** using `conn.close()`.

- How can you implement the 'Update' operation in a Python-based CRUD application, and what SQL command is used to modify existing records in a database?

ANS:

To implement the 'Update' operation in a Python-based CRUD application, you first connect to the SQLite database, create a cursor, and execute an SQL `UPDATE` command to modify existing records.

Example: Python

```
import sqlite3

conn = sqlite3.connect('users.db')
cursor = conn.cursor()

cursor.execute('UPDATE users SET name = ?, age = ? WHERE id = ?', ('Jane Doe',
25, 1))
conn.commit()

conn.close()
```

SQL Command:

```
UPDATE table_name SET column1 = value1, column2 = value2 WHERE condition;
```

Conclusion: (Conclusion to be based on the objectives and outcomes achieved)

The implementation of CRUD operations using Python and SQLite demonstrates the

fundamental operations required for managing databases—creating, reading, updating, and deleting data. This practical application enables efficient data handling and manipulation, illustrating the simplicity and effectiveness of SQLite for small to medium-sized projects. Through this experiment, we gained hands-on experience with basic database interactions in Python.

References:

1. M. Grinberg, Flask Web Development: Developing Web Applications with Python, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2018.
2. W. S. Vincent, Django for Beginners: Build Websites with Python and Django, 2nd ed. Self-Published, 2021.
3. R. Coombes, Learning SQLAlchemy, 1st ed. Birmingham, UK: Packt Publishing, 2015.
4. R. J. A. Becker, Python and MySQL Development, 1st ed. Birmingham, UK: Packt Publishing, 2018.
5. K. S. Rajasekaran, Python Programming and Data Science: A Comprehensive Guide to Python Programming for Beginners, 1st ed. Singapore: Springer, 2021.