Лекція 10. Робота з БД

Лектор – доцент, кандидат фізико-математичних наук
Кириченко Віктор Вікторович

ПЛАН ЛЕКЦІЇ

1. СУБД SQLite. Підключення до бази даних.
2. Отримання та запис даних в таблиці БД.
3. Особливості роботи з віддаленою БД.

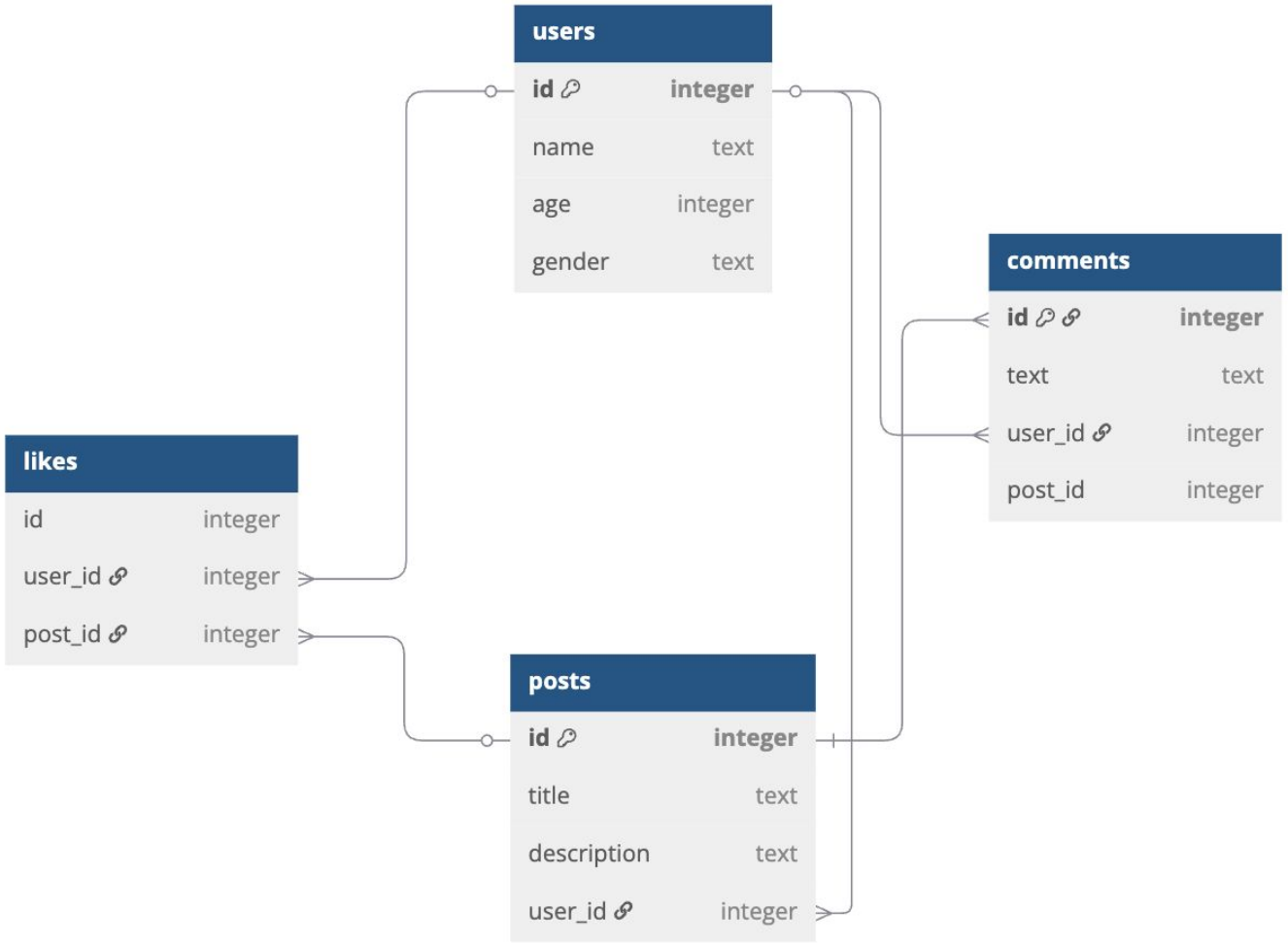
СУБД SQLite. Підключення до бази даних.

SQLite це найпростіша база даних SQL для Python, оскільки не вимагає встановлення зовнішніх SQL модулів.

За промовчанням у встановленій системі Python вже є SQL бібліотека sqlite3, яка дозволяє підключатися до бази SQLite.

Більш того, бази SQLite не потребують сервера і самодостатні, оскільки просто зчитують та записують дані у файл. На відміну від MySQL та PostgreSQL, для виконання операцій з базами даних навіть не потрібно встановлювати та запускати серверну програму.

СУБД SQLite. Підключення до бази даних.



СУБД SQLite. Підключення до бази даних.

```
import sqlite3
from sqlite3 import Error


def create_connection(path):
    connection = None
    try:
        connection = sqlite3.connect(path)
        print(«Підключення до бази даних SQLite пройшло успішно")
    except Error as e:
        print(f«Виникла помилка '{e}'»)
        return connection
```

СУБД SQLite. Підключення до бази даних.

- Метод "sqlite3.connect(path)" повертає об'єкт "(connection)". Його ж, своєю чергою, повертає і функція «create_connection()».
- Об'єкт connection можна використовувати для виконання запитів до бази SQLite. Наступний скрипт встановлює підключення до SQLite:

```
connection = create_connection(«my_bd.sqlite")
```

Коли ви запустите скрипт бази даних SQL, побачите, що у кореневому каталозі створено файл бази даних «my_bd.sqlite». Шлях до файлу можна змінити.

Отримання та запис даних в таблиці БД.

Для роботою з БД створюється курсор, за допомогою якого можна виконувати SQL код.

```
cursor = connection.cursor()
```

Створення таблиці

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

Отримання та запис даних в таблиці БД.

Додавання даних (INSERT)

```
cursor.execute("INSERT INTO users (name, age) VALUES (?, ?)", ("Марія", 30))

cursor.execute("INSERT INTO users (name, age) VALUES (?, ?)", ("Ігор", 25))

connection.commit()
```

Отримання та запис даних в таблиці БД.

Отримання даних (SELECT)

Всі записи:

```
cursor.execute("SELECT * FROM users")
rows = cursor.fetchall()
for row in rows:
    print(row)  # (1, 'Марія', 30), (2, 'Ігор', 25)
```

Один запис:

```
cursor.execute("SELECT * FROM users WHERE name = ?", ("Марія",))
user = cursor.fetchone()
    print(user)  # (1, 'Марія', 30)
```

Отримання та запис даних в таблиці БД.

Закриття з'єднання

```
cursor.close()

conn.close()
```

Особливості роботи з віддаленою БД.

Щоб підключитися до бази MySQL із Python, потрібно встановити відповідний SQL-драйвер. Один з таких - "mysql-connector-python".

```
pip install mysql-connector-python
```

Приклад:

```
import mysql.connector

conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="your_password",
    database="testdb"
)

cursor = conn.cursor()
```

Особливості роботи з віддаленою БД.

Основні елементи:

- Connection - Це об'єкт, що представляє активне з'єднання з MySQL-сервером.
- Cursor - **Об’єкт** для виконання SQL-запитів через з’єднання.
- Cxecute - Запити передаються параметризовано — не через f-рядки, щоб уникнути SQL-ін'єкцій.
- Commit - MySQL за замовчуванням працює у транзакційному режимі, тому після INSERT/UPDATE/DELETE треба викликати `.commit()`:
- rollback() - Використовується для відкату змін у разі помилки
- close() - Потрібно закривати підключення після завершення роботи, щоб уникнути витоку ресурсів.

Особливості роботи з віддаленою БД.

Створення таблиць

```
cursor.execute("""
CREATE TABLE IF NOT EXISTS users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255),
    age INT
)
""")
```

Особливості роботи з віддаленою БД.

Додавання елементів

```
cursor.execute("INSERT INTO users (name, age) VALUES (%s, %s)", ("Олена", 32))

conn.commit()
```

Отримання даних

```
cursor.execute("SELECT * FROM users")

for row in cursor.fetchall():

    print(row)
```

Особливості роботи з віддаленою БД.

Оновлення даних

```
cursor.execute("UPDATE users SET age = %s WHERE name =
%s", (33, "Олена"))

conn.commit()
```

Видалення даних

```
cursor.execute("DELETE FROM users WHERE name = %s",
("Олена",))

conn.commit()
```

Особливості роботи з віддаленою БД.

Закриття підключення до БД

```
cursor.close()

conn.close()
```

ДЯКУЮ ЗА УВАГУ