MySQL та Python

Основні поняття

- База даних (БД): Це організоване сховище інформації. Уявіть її як велику цифрову картотеку.
- Реляційна БД: Це база даних, де інформація зберігається у вигляді таблиць, що пов'язані між собою.
- СУБД (Система управління базами даних): Це програма, яка дозволяє створювати, змінювати та керувати базами даних. MySQL — одна з найпопулярніших СУБД.

Основи мови SQL

SQL (Structured Query Language) — це мова для роботи з даними в реляційних базах даних. Вона поділяється на кілька частин:

- DQL (Data Query Language): Для отримання даних (`SELECT`).
- DML (Data Manipulation Language): Для зміни даних (`INSERT`, `UPDATE`, `DELETE`).
- DDL (Data Definition Language): Для зміни структури БД (`CREATE`, `ALTER`, `DROP`).

Типи даних у MySQL

На відміну від SQLite, де є лише кілька універсальних типів, MySQL надає багато специфічних типів даних. Це допомагає зберігати інформацію ефективніше, перевіряти її коректність та оптимізувати роботу бази даних.

Категорія	Тип даних	Призначення	Приклад
Числові	`INT`	Цілі числа. Використовується для віку, кількості товарів тощо.	`age INT`

	DECIMAL(M,D)	Точні десяткові числа. Ідеально для фінансів. M — загальна кількість цифр, D — кількість цифр після коми.	price DECIMAL(10, 2)
	FLOAT, DOUBLE	Десяткові числа з плаваючою крапкою.	rating DOUBLE
Текстові	VARCHAR(N)	Рядки змінної довжини. N — максимальна кількість символів. Зберігає тільки використаний простір.	name VARCHAR(255)
	CHAR(N)	Рядки фіксованої довжини. Завжди займає N символів, навіть якщо рядок коротший.	country_code CHAR(2)
	TEXT	Великий текст, такий як опис товару або стаття.	description TEXT
Дата і час	DATE	Дата у форматі YYYY-MM-DD.	birthday DATE

	`TIME`	Час у форматі `HH:MM:SS`.	`start_time TIME`
	`DATETIME`	Дата і час разом.	`created_at DATETIME`

Порівняння MySQL та SQLite

Особливість	MySQL	SQLite
Тип	Клієнт-серверна.	Файлова, безсерверна.
Використання	Для веб-додатків, великих систем, багато користувачів.	Для локальних додатків, мобільних пристроїв, малих проєктів.
Встановлення	Потребує встановлення та налаштування сервера.	Проста: це просто один файл.
Доступ	Підтримує багато одночасних запитів.	Зазвичай один процес може записувати дані.

Покрокова інструкція

Крок 1: Встановлення MySQL Server на Windows

1. Завантажте MySQL Installer з офіційного сайту: mysql.com → Downloads → MySQL Community (GPL) Downloads.

2. Запустіть інсталятор та виберіть "Developer Default".
3. На етапі конфігурації сервера встановіть та запам'ятайте пароль для користувача `root`.

Крок 2: Робота в MySQL Workbench

1. Запустіть MySQL Workbench та підключіться до локального сервера, використовуючи пароль, який ви встановили.
2. Створіть нову базу даних `students_db`:

```
CREATE DATABASE students_db;
USE students_db;
```

3. Створіть таблицю `students` та додайте дані:

```
CREATE TABLE students (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255),
    age INT
);
```

```
INSERT INTO students (name, age) VALUES ('Іван', 18);
INSERT INTO students (name, age) VALUES ('Марія', 19);
```

4. Використовуйте SQL-запити для роботи з даними:

```
SELECT * FROM students WHERE age > 18;
SELECT * FROM students ORDER BY name ASC;
UPDATE students SET age = 20 WHERE name = 'Іван';
```

Крок 3: Робота з MySQL Shell

MySQL Shell — це потужний інструмент командного рядка для адміністрування та

роботи з базами даних MySQL.

1. Відкрийте командний рядок (CMD) або PowerShell.
2. Перейдіть до директорії, де встановлений MySQL Shell. Зазвичай це `C:\Program Files\MySQL\MySQL Shell 8.0\bin`.
3. Для підключення до вашого сервера введіть команду:

```
mysqlsh --uri root@localhost:3306
```

4. Введіть пароль, який ви встановили під час інсталяції.
5. Після підключення ви можете виконувати SQL-запити. Щоб перевірити, чи все працює, спробуйте виконати:

```
\sql
USE students_db;
SELECT * FROM students;
```

Крок 4: Взаємодія з Python

1. Встановіть необхідний конектор через командний рядок:

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

2. Напишіть Python-скрипт для підключення та виконання запиту. Зверніть увагу на відмінності!
    - Для підключення використовується `mysql.connector.connect()`, а не `sqlite3.connect()`.
    - Дуже важливо: Для передачі змінних у запит `mysql-connector-python` використовує спеціальний плейсхолдер `%s` і передає значення у вигляді кортежу, на відміну від `sqlite3`, де використовується `?`. Це допомагає запобігти SQL-ін'єкціям.

```
import mysql.connector

try:
    # Встановлення з'єднання з базою даних
    mydb = mysql.connector.connect(
      host="localhost",
      user="root",
      password="ВАШ_ПАРОЛЬ",
      database="students_db"
    )

    mycursor = mydb.cursor()

    # SQL-запит для вибору всіх даних з таблиці students
    sql = "SELECT * FROM students"

    # Виконання запиту
    mycursor.execute(sql)

    # Отримання всіх результатів
    myresult = mycursor.fetchall()

    # Виведення результатів
    print("Список студентів:")
    for student in myresult:
      print(student)

except mysql.connector.Error as err:
    print(f"Помилка: {err}")
finally:
    # Завжди закривайте з'єднання
    if 'mydb' in locals() and mydb.is_connected():
        mycursor.close()
        mydb.close()
        print("З'єднання з MySQL закрито.")
```

Завдання

Напишіть власний Python-скрипт, який:

- Запитує у вас ім'я та вік.
- Додає ці дані в таблицю `students`, використовуючи `INSERT INTO` та плейсхолдер `%s`.
- Після додавання, виводить всі записи з таблиці.