Использование SQLite в Python

Подключение и основные операции

Лазар В. И., Козлова Е. Р.

12 февраля 2025 г.

План занятия

1. Подготовка окружения
2. Подключение и базовые операции
3. Практические советы
4. Проектное задание

Установка и импорт

- **Встроенный модуль** `sqlite3` (начиная с Python 2.5).
- Никаких дополнительных библиотек не требуется.
- Импорт:

Пример

```
import sqlite3
```

- Для проверки версии SQLite используйте:

Пример

```
sqlite3.sqlite_version
```

Создание и подключение к базе данных

Пример кода на Python

```
import sqlite3

# Создаём или открываем существующий файл базы данных
conn = sqlite3.connect('example.db')

# Создаём курсор для выполнения SQL-запросов
cursor = conn.cursor()

# В конце работы не забудьте закрыть соединение
conn.close()
```

- Если файла `example.db` ещё нет, он будет создан автоматически.
- `conn` — объект соединения, `cursor` — объект для работы с SQL.

Создание таблицы

Пример SQL в Python

```
import sqlite3
conn = sqlite3.connect('example.db')
cursor = conn.cursor()
cursor.execute(''' # Создаём таблицу
    CREATE TABLE IF NOT EXISTS users (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        username TEXT NOT NULL,
        age INTEGER,
        city TEXT
    )
''')
conn.commit()  # фиксируем изменения
conn.close()
```

- `commit()` — фиксирует (сохраняет) изменения в базе данных.

Добавление данных

Пример SQL в Python

```
import sqlite3
conn = sqlite3.connect('example.db')
cursor = conn.cursor()
# Пример параметризированного запроса
cursor.execute('''
    INSERT INTO users (username, age, city)
    VALUES (?, ?, ?)
''', ('Иван', 25, 'Москва'))
conn.commit()
conn.close()
```

- Используйте **плейсхолдеры (?)** и кортеж значений для защиты от SQL-инъекций.
- Можно выполнять `execute` несколько раз для разных данных или воспользоваться `executemany`.

Выборка данных

Пример SQL в Python

```
import sqlite3
conn = sqlite3.connect('example.db')
cursor = conn.cursor()
# Выполняем SELECT-запрос
cursor.execute('SELECT id, username, age, city FROM users')
# Получаем все результаты
rows = cursor.fetchall()
for row in rows:
    print(row)
conn.close()
```

- `fetchall()` — возвращает список кортежей с результатами.
- `fetchone()` — возвращает одну строку, `fetchmany(n)` — `n` строк.

Обновление и удаление данных

Обновление

```
cursor.execute('''
    UPDATE users
    SET age = ?
    WHERE username = ?
''', (26, 'Иван'))
```

Удаление

```
cursor.execute('''
    DELETE FROM users
    WHERE username = ?
''', ('Иван',))
```

- Не забудьте `commit()` после `UPDATE` или `DELETE`.

Работа с контекстными менеджерами

Пример

```
import sqlite3

with sqlite3.connect('example.db') as conn:
    cursor = conn.cursor()
    cursor.execute('SELECT * FROM users')
    rows = cursor.fetchall()
    print(rows)
```

- С `with` не нужно явно вызывать `close()`.
- По выходу из блока `with` изменения автоматически `commit()`-ятся, если не произошло ошибки, иначе — `rollback()`.

Организация кода в проекте

- Вся логика работы с БД выносится в отдельный пакет (обычно его называют `db` или `models`)
- Для каждой сущности создаётся класс, который называется моделью этой сущности. Все операции над классом преобразуются в соответствующий SQL-код
- Обычно в каждом методе создаётся отдельный курсор для упрощения взаимодействия, но бывают случаи, когда курсор создаётся извне и передаётся в методы (dependency injection)

Задание: Мини-приложение для хранения данных

Вы создаёте приложение для библиотеки.

Требования:

1. Хранение информации о различных книгах в библиотеке (название, автор, год выпуска, издание, номер шкафа, номер полки).
2. Хранение информации о посетителях библиотеки (имя, фамилия, отчество, номер читательского билета, адрес).
3. Хранение информации о том, какой пользователь какие книги читает в данный момент.
4. Возможность внесения информации об актуальных изменениях (появление новой книги в библиотеке, регистрация нового пользователя, выдача книги пользователю, получения книги от пользователя, поиск книги по различным параметрам, и т. д.)
5. Консольный (или графический) интерфейс для взаимодействия с программой.