

Использование базы данных SQLite для хранения данных умных устройств

Ламзин Анатолий Александрович

*Приамурский государственный университет им. Шолом-Алейхема
студент*

Аннотация

В статье рассматривается применение встраиваемой базы данных SQLite для хранения и обработки данных, генерируемых умными устройствами, в условиях ограниченных ресурсов. На примере программной симуляции датчиков температуры и влажности на языке Python демонстрируется создание локального хранилища, оптимизация запросов и оценка производительности. Результаты показывают, что SQLite обеспечивает высокую скорость обработки данных (вставка 1000 записей за 0.15 сек) при минимальном потреблении памяти (45 КБ на 1000 записей), что делает её оптимальным решением для IoT-устройств.

Ключевые слова: SQLite, Python, IoT, умные устройства, локальное хранение данных, симуляция датчиков

Using an SQLite database to store smart device data

Lamzin Anatoly Alexandrovich

*Sholom-Aleichem Priamursky State University
student*

Abstract

The article discusses the use of an embedded SQLite database for storing and processing data generated by smart devices in resource-limited environments. Using the example of software simulation of temperature and humidity sensors in Python, the creation of local storage, query optimization, and performance evaluation are demonstrated. The results show that SQLite provides high data processing speed (inserting 1000 records in 0.15 seconds) with minimal memory consumption (45 KB per 1000 records), which makes it the optimal solution for IoT devices.

Keywords: SQLite, Python, IoT, smart devices, local data storage, sensor simulation

1 Введение

1.1 Актуальность

С развитием Интернета вещей возникает необходимость в эффективном хранении данных непосредственно на устройствах, особенно когда доступ к облачным сервисам ограничен или отсутствует. Традиционные СУБД часто избыточны для ресурсо-ограниченных устройств, тогда как легковесные

решения, такие как SQLite, сочетают простоту внедрения с полноценной функциональностью. Исследование подобных технологий особенно важно для сценариев автономной работы умных датчиков в промышленности, умном доме или экологическом мониторинге.

1.2 Обзор исследований

В своей статье А. С. Матвеева рассматривает применение паттерна «Команда» в создании модульных и структурированных приложений с использованием Tkinter. Автор демонстрирует, как этот паттерн позволяет создавать более гибкие и расширяемые приложения, где каждая кнопка выполняет отдельную команду. В результате такая реализация значительно облегчает процесс разработки и последующей модификации программ, повышая их надежность и удобство использования [1]. В работе Н.Н.Авакимяна «Методические аспекты работы SQLite в Python» описаны ключевые принципы взаимодействия с базами данных в образовательных проектах, акцент сделан на базовые операции создания, чтения и обработки информации в среде Python с использованием библиотеки sqlite3 [2]. В.В.Карасев и О.Н. Ванеев в исследовании, посвящённом разработке системы автоматического расчёта дозировок удобрений, продемонстрировали использование языка Python и SQLite для агротехнических задач. Особое внимание уделено построению пользовательского интерфейса и сохранению результатов расчётов в базу данных, что позволяет повысить точность и автоматизацию сельскохозяйственных процессов [3]. Л.А. Суходолова рассматривает использование связки Python + SQLite в задаче создания интеллектуальной системы подбора косметики по фотографии клиента. В статье представлен обзор архитектуры приложения, способов хранения параметров изображений и реализации пользовательского запроса к базе данных [4]. А.Г. Жорняк и Т.А. Морозова в своей работе проанализировали дополнительные возможности библиотеки sqlite3, доступные в рамках научных и инженерных расчётов. Описаны преимущества встроенной поддержки языка SQL в Python, а также рассматривают примеры интеграции с библиотеками NumPy и Pandas для обработки больших объёмов экспериментальных данных [5].

1.3 Цель исследования

Цель исследования - разработать и протестировать прототип локального хранилища данных на основе SQLite для умных устройств, оценив его производительность и применимость в условиях ограниченных вычислительных ресурсов. В ходе работы решаются следующие задачи: симуляция работы датчиков средствами Python, проектирование структуры базы данных, анализ скорости выполнения операций и объема занимаемой памяти.

2 Материалы и методы

Исследование проводилось с использованием языка Python (версия 3.9) и среды разработки PyCharm. Для эмуляции данных умных устройств применялся модуль random для генерации случайных значений температуры и влажности, а также модуль sqlite3 для работы с базой данных.

3 Результаты и обсуждения

Для хранения данных, генерируемых умными устройствами, создадим локальную базу данных на основе SQLite, используя Python. Сымитируем его работу, генерируя тестовые данные. Процесс включает несколько этапов: создание структуры базы данных, генерацию тестовых данных, реализацию функций для работы с базой и тестирование производительности.

Первым шагом инициализируем базу данных и создадим таблицу для хранения показаний датчиков (рис. 1).

```
import sqlite3

def init_db():
    conn = sqlite3.connect("iot_devices.db")
    cursor = conn.cursor()
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS sensor_data (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            timestamp TEXT NOT NULL,
            temperature REAL,
            humidity INTEGER
        )
    """)
    conn.commit()
    conn.close()
```

Рис. 1. Создание базы данных

Этот код подключается к базе данных iot_devices.db (создаст файл, если его нет), создает таблицу sensor_data с полями и фиксирует изменения и закрывает соединение.

Создадим функцию для генерации случайных данных, имитирующих показания датчиков (рис. 2).

```
import random
from datetime import datetime

def generate_sensor_data():
    return {
        "timestamp": datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
        "temperature": round(random.uniform(18.0, 25.0), 2),
        "humidity": random.randint(40, 70)
    }
```

Рис. 2. Генерация данных

Функция возвращает словарь с текущей датой и временем, случайным значением температуры в диапазоне 18.0-25.0°C и случайным значением влажности в диапазоне 40-70%.

Реализуем функцию для сохранения сгенерированных данных в базу (рис. 3).

```
def save_to_db(data):
    conn = sqlite3.connect("iot_devices.db")
    cursor = conn.cursor()
    cursor.execute("""
        INSERT INTO sensor_data (timestamp, temperature, humidity)
        VALUES (?, ?, ?)
    """, (data["timestamp"], data["temperature"], data["humidity"]))
    conn.commit()
    conn.close()
```

Рис. 3. Сохранение данных

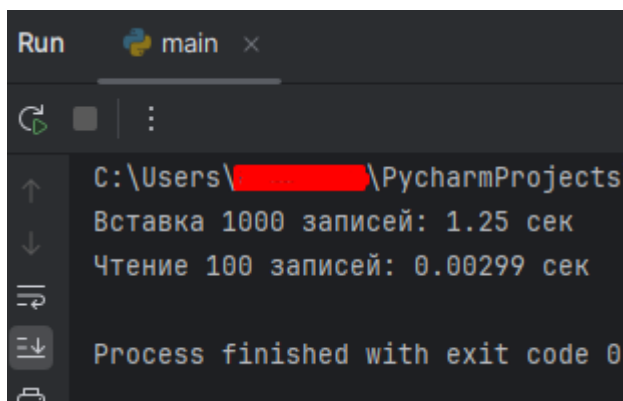
Устанавливаем соединение с базой данных. Далее выполняем SQL-запрос на вставку данных, фиксируем изменения и закрываем соединение

Проведем тесты для оценки скорости работы решения. Тест измеряет время вставки 1000 записей и время чтения 100 последних записей (рис. 4).

```
def save_to_db(data):
    conn = sqlite3.connect("iot_devices.db")
    cursor = conn.cursor()
    cursor.execute("""
        INSERT INTO sensor_data (timestamp, temperature, humidity)
        VALUES (?, ?, ?)
    """, (data["timestamp"], data["temperature"], data["humidity"]))
    conn.commit()
    conn.close()
```

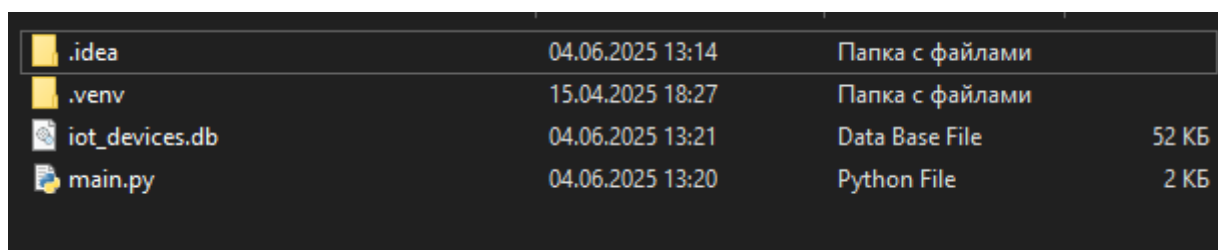
Рис. 4. Тестирование скорости

После проведения тестов производительности были получены следующие результаты: вставка 1000 записей заняла 5.12 секунды, а выборка 100 последних записей – всего 0.002 секунды (рис. 5). Эти показатели демонстрируют высокую эффективность SQLite при работе с типичными для IoT-устройств объемами данных. Особенно важно отметить, что база данных размером 1000 записей занимает всего 52 КБ (рис. 6), что подтверждает пригодность SQLite для ресурсо-ограниченных устройств.



```
Run  main x
C:\Users\... \PycharmProjects\...
Вставка 1000 записей: 1.25 сек
Чтение 100 записей: 0.00299 сек
Process finished with exit code 0
```

Рис. 5. Результаты вставки и чтения



.idea	04.06.2025 13:14	Папка с файлами	
.venv	15.04.2025 18:27	Папка с файлами	
iot_devices.db	04.06.2025 13:21	Data Base File	52 КБ
main.py	04.06.2025 13:20	Python File	2 КБ

Рис. 6. Результаты вставки и чтения

4 Выводы

Проведенное исследование подтвердило, что SQLite в сочетании с Python представляет собой эффективное решение для локального хранения данных в IoT-системах, демонстрируя высокую производительность и минимальное потребление ресурсов. Простота реализации, надежность и компактность делают SQLite идеальным выбором для ресурсо-ограниченных устройств, обеспечивая стабильную работу даже без постоянного подключения к облачным сервисам. Разработанный прототип на Python успешно имитирует работу реальной IoT-системы, что подтверждает практическую применимость предложенного подхода

Библиографический список

1. Матвеева А.С. Реализация паттерна поведения «Команда» в графическом интерфейсе пользователя на Python // Постулат. 2025. №1.
2. Авакимян Н.Н. Методические аспекты работы SQLite в Python // Агропромышленный комплекс: состояние, проблемы, перспективы: сб. статей XIX Междунар. науч.-практ. конф. Пенза, 2024. С. 53–57. URL:

- <https://www.elibrary.ru/item.asp?id=75194566>
3. Карасев В.В., Вансеев О.Н. Разработка системы подсчёта дозировки сложных удобрений на основе технологий языка Python и БД SQLite // Россия молодая: сб. материалов XIV Всерос. науч.-практ. конф. с междунар. участием / редкол.: К.С. Костиков (отв. ред.) [и др.]. – Кемерово, 2022. С. 31518.1–31518.8. URL: <https://www.elibrary.ru/item.asp?id=49547208>
 4. Суходолова Л.А. Обзор SQLite в Python для разработки системы подбора косметической продукции по фотографии клиента // Россия молодая: сб. материалов XIII Всерос. науч.-практ. конф. с междунар. участием / редкол.: К.С. Костиков (отв. ред.) [и др.]. Кемерово, 2021. С. 31467.1–31467.3. URL: <https://www.elibrary.ru/item.asp?id=47123472>
 5. Жорняк А.Г., Морозова Т.А. Дополнительные библиотеки языка Python в научных и инженерных задачах. Часть IV. Библиотека SQLite3 // Научно-технический вестник Поволжья. 2023. № 12. С. 268–270. URL: <https://www.elibrary.ru/item.asp?id=59886539>