

SQLite. Синтаксис и запросы

№ урока: 4 **Курс:** Python Advanced

Средства обучения: PyCharm

Обзор, цель и назначение урока

Изучить основы работы с библиотекой `sqlite3` и использования данной СУБД. Рассмотреть особенности данной библиотеки с практическим уклоном.

Изучив материал данного занятия, учащийся сможет:

- Позволит использовать SQLite в своих задачах.
- Создавать пользовательские агрегатные и обычные функции, расширяя стандартные возможности SQL.
- Узнаете основные запросы для работы с SQLite.

Содержание урока

1. Основные понятия и особенности СУБД SQLite.
2. Библиотека `sqlite3` в Python.

Резюме

База данных (БД) — это организованная структура, предназначенная для хранения, изменения и обработки взаимосвязанной информации, преимущественно больших объемов. Базы данных активно используются для динамических сайтов со значительными объемами данных — часто это интернет-магазины, порталы, корпоративные сайты.

Система управления базами данных (СУБД) — это комплекс программных средств, необходимых для создания структуры новой базы, ее наполнения, редактирования содержимого и отображения информации. Наиболее распространенными СУБД являются MySQL, PostgreSQL, Oracle, Microsoft SQL Server, SQLite.

В итоге, базы данных - это место хранения данных, а СУБД позволяют управлять ими. Зачастую БД и СУБД неразрывны, но не всегда. В некоторых справочниках можно увидеть употребление термина БД вместо СУБД, но это не корректная взаимозамена.

Что такое SQLite?

SQLite - это система управления реляционными базами данных (СУБД), содержащаяся в библиотеке C. В отличие от многих других систем управления базами данных, SQLite не используется механизм клиент-сервер. Библиотека SQLite выступает в роли и клиента, и сервера одновременно, а данные хранятся локально в обычном файле.

SQLite - это наиболее часто используемый механизм баз данных в мире. Он встроен во все мобильные телефоны и большинство компьютеров и входит в состав множества других приложений, которые люди используют каждый день.

Формат файла SQLite является стабильным, кроссплатформенным и обратно совместимым, и разработчики обещают сохранить его в таком виде как минимум до 2050 года. Файлы базы данных SQLite обычно используются в качестве контейнеров для передачи большого объема

данных между системами и как долгосрочный архивный формат для данных. Активно используется более 1 триллиона (1e12) баз данных SQLite.

Где SQLite применяется?

SQLite везде: на смартфонах, в браузерах, web-приложениях, серверах. Например Firefox хранит куки в sqlite, большинство мобильных приложений хранят пользовательские данные в sqlite. SQLite отлично подходит для приложений, которым надо стабильно хранить большие данные, но с небольшой нагрузкой. Настройки по умолчанию работают на надежность, а не на производительность.

Подготовка окружения для работы с SQLite

На операционной системе Linux (Ubuntu) достаточно выполнить следующую команду в терминале:

```
sudo apt install sqlite3
```

Для остановки на Windows следуйте инструкциям в разделе "Precompiled Binaries for Windows" на странице <https://www.sqlite.org/download.html>:

Precompiled Binaries for Windows

- | | |
|--|---|
| <div style="border: 1px solid red; padding: 2px; display: inline-block; border-radius: 50%; width: 20px; height: 20px; text-align: center; line-height: 20px;">1</div> | sqlite-dll-win32-x86-3330000.zip
(489.25 KiB)
<small>(sha3: d669a92271ad31bbefe4ba827c3d6621179b1964b5dd1f27c6fb9a9c4cf6a082)</small> |
| <div style="border: 1px solid red; padding: 2px; display: inline-block; border-radius: 50%; width: 20px; height: 20px; text-align: center; line-height: 20px;">2</div> | sqlite-dll-win64-x64-3330000.zip
(809.89 KiB)
<small>(sha3: 1b078c320adb1d9ff57c508eb0d2ef05bf13ca2e9cbb37c2ea72099373cdfd61)</small> |
| <div style="border: 1px solid red; padding: 2px; display: inline-block; border-radius: 50%; width: 20px; height: 20px; text-align: center; line-height: 20px;"></div> | sqlite-tools-win32-x86-3330000.zip
(1.76 MiB)
<small>(sha3: 5b13a53afe89ca0a975f0c166d5e4c33c83695ceea75ae297f5471df3adde4b0)</small> |
- A bundle of command-line tools for managing SQLite database files, including the [command-line shell](#) program, the [sqldiff.exe](#) program, and the [sqlite3_analyzer.exe](#) program.

В зависимости от разрядности системы, скачиваем или первый, или второй файл.

Чтобы проверить успешную установку, откройте терминал (как открыть терминал на Windows) и напишите следующую команду:

```
sqlite3
```

Результат должен быть примерно следующим:

```
Terminal
~ >>> sqlite3
SQLite version 3.33.0 2020-08-14 13:23:32
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite>
```

Специальные команды предназначены для формирования таблиц и других административных операций. Все они начинаются с точки.

Эти «точечные команды» обычно используются для изменения формата вывода запросов или для выполнения определенных заранее подготовленных операторов запроса. Изначально было всего несколько точечных команд, но с годами накопилось много новых функций, так что сегодня их более 60.

Пройдёмся по списку команд, которые могут пригодиться:

- .show Показывает текущие настройки заданных параметров
- .databases Показывает название баз данных и файлов
- .quit Выход из sqlite3
- .tables Показывает текущие таблицы
- .schema Отражает структуру таблицы
- .header Отобразить или скрыть шапку таблицы
- .mode Выбор режима отображения данных таблицы

- `.dump` Сделать копию базы данных в текстовом формате

Чтобы получить список доступных команд с точкой, вы можете ввести `.help` без аргументов. Или введите `.help show` для получения подробной информации о команде `.show`. Список доступных точечных команд следующий:

Больше о специальных командах можно прочитать на официальном сайте.

SQL - это стандартный язык для взаимодействия с БД. С его помощью можно сообщить БД о записи, обработке и извлечении данных.

SQL состоит из множества типов операторов, которые можно неофициально классифицировать как подязыки, обычно: язык запросов данных (DQL), язык определения данных (DDL), язык управления данными (DCL) и язык управления данными (DML). Сфера применения SQL включает запрос данных, манипулирование данными (вставка, обновление и удаление), определение данных (создание схемы и модификации), и контроль доступа к данным.

Хотя SQL по сути является декларативным языком, он также включает процедурные элементы.

Язык описания данных (DDL) состоит из команд для создания таблицы, изменения и удаления баз данных, таблиц и прочего:

- `CREATE`
- `ALTER`
- `DROP`

Язык управления данными (DML) позволяет пользователю манипулировать данными (добавлять/изменять/удалять).

- `INSERT`
- `UPDATE`
- `DELETE`

Язык запросов DQL позволяет осуществлять выборку данных.

- `SELECT`

SQLite так же поддерживает и множество других команд, список которых можно найти тут. Поскольку данный урок предназначен для начинающих, мы ограничимся перечисленным набором команд.

Типы данных SQL

Тип данных SQLite - это атрибут, определяющий тип данных любого объекта. Каждый столбец, переменная и выражение имеет связанный тип данных в SQLite.

Типы данных используются во время создания таблиц. SQLite тип данных значения связан с самим значением и сообщает БД способ работы с ним.

Каждое значение, хранящееся в базе данных SQLite, имеет один из следующих **классов (не типы)** хранения:

Классы хранения SQLite

Название	Описание
NULL	Значение - значение NULL.
INTEGER	Значение представляет собой целое число со знаком, сохраненное в 1, 2, 3, 4, 6 или 8 байтах в зависимости от величины значения.
REAL	Значение представляет собой значение с плавающей запятой, которое хранится как 8-байтовое число с плавающей точкой IEEE.
TEXT	Значение представляет собой текстовую строку, хранящуюся с

	использованием кодировки базы данных (UTF-8, UTF-16BE или UTF-16LE)
BLOB	Значение представляет собой блок данных, который хранится точно так же, как он был введен.

SQLite поддерживает концепцию **affinity** (близость) типа к столбцам. **Любой столбец может хранить данные любого типа**, но предпочтительный класс хранения для столбца называется affinity. Каждому столбцу таблицы в базе данных SQLite3 присваивается одно из следующих аффинностей типа:

Тип слияния SQLite

Название	Описание
TEXT	В этом столбце хранятся все данные с использованием классов хранения NULL, TEXT или BLOB.
NUMERIC	Этот столбец может содержать значения, используя все пять классов хранения.
INTEGER	Работает так же, как столбец с NUMERIC сродством, с исключением в выражении CAST.
REAL	Ведет себя как столбец с NUMERIC сродством, за исключением того, что он приводит целые значения в представление с плавающей запятой.
NONE	Столбец с аффинностью NONE не предпочитает один класс хранения над другим, и не предпринимаются попытки принудить данные из одного класса хранения к другому.

Наконец-то о самих типах. В следующих списках таблиц перечислены имена типов данных, которые можно использовать при создании таблиц SQLite3 с соответствующим применимым сродством.

INT, INTEGER, TINYINT, SMALLINT, MEDIUMINT, BIGINT, UNSIGNED BIG INT, INT2, INT8	INTEGER
CHARACTER(20), VARCHAR(255), VARYING CHARACTER(255), NCHAR(55), NATIVE CHARACTER(70), NVARCHAR(100), TEXT, CLOB	TEXT
BLOB, no datatype specified	NONE
REAL, DOUBLE, DOUBLE PRECISION, FLOAT	REAL
NUMERIC, DECIMAL(10,5), BOOLEAN DATE, DATETIME	NUMERIC

SQLite не имеет отдельного булевского класса хранения (*True). Вместо этого булевы значения сохраняются как целые числа 0 (ложь) и 1 (истина).

SQLite не имеет отдельного класса хранения для хранения дат и / или времени, но SQLite способен хранить даты и время как значения TEXT, REAL или INTEGER.

Более детальное описание этой темы можно прочитать [здесь](#).

Создание базы данных

Задание: создать базу данных для хранения имя (first_name) и фамилии (last_name) пользователя, названия учебного курса (course) и количества занятий (lessons).

Теперь давайте создадим базу данных. Если вы ещё находитесь в интерфейсе sqlite3, то наберите команду .quit для выхода. Теперь вводим в терминал:

sqlite3 students.db

В результате, в текущем каталоге у нас появится файл *students.db*.

Если не указать название файла, sqlite3 создаст **временную** базу данных в оперативной памяти.

Создание таблицы

Для хранения студентов нам необходимо создать таблицу. Назовём её *students_list*. Выполняем команду:

```
CREATE TABLE students_list (  
    student_id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,  
    first_name TEXT NOT NULL,  
    last_name TEXT NOT NULL,  
    course TEXT NOT NULL,  
    lessons INTEGER NOT NULL );
```

Для получения структуры таблицы наберите `.schema students_list`. `student_id` - это главный ключ каждой записи. Базам данным необходимы какие-то уникальные данные внутри стоки, чтобы эффективно сохранять и искать данные. Он имеет атрибут `AUTOINCREMENT`, что говорит о том, что база данных будет автоматически заполнять это поле. Атрибут `NOT NULL` гласит, что поля не могут быть пустыми.

Теперь можем внести данные в таблицу.

Изменение таблицы

Добавление новой колонки

Для добавления новой колонки следует использовать команду `ALTER`. К примеру введём поле `age`. Выполняем команду:

```
ALTER TABLE students_list  
ADD COLUMN age INTEGER;
```

Для проверки, что поле действительно добавилось: `.schema students_list`.

Переименование таблицы

Так же мы можем использовать команду `ALTER` для переименования таблицы `students_list` на `students`:

```
ALTER TABLE students_list  
RENAME TO students;
```

Удаление таблицы

Для удаления нашей таблицы выполните следующую команду (*пока этого делать не стоит*):

```
DROP TABLE students;
```

Внесение данных

Предположим, что нам необходим внести следующую запись:

- First name: Rachel
- Last name: Green
- Course: C#
- Lessons: 40
- Age: 26

Для вставки воспользуемся командой `INSERT`.

```
INSERT INTO students ( first_name, last_name, course, lessons, age )  
VALUES ( 'Rachel', 'Green', 'C#', 40, 26 );
```

Указывать значение для `student_id` не нужно - оно сформируется автоматически благодаря настройке `AUTOINCREMENT`.

Получение данных

Для выборки данных воспользуемся командой `SELECT`.

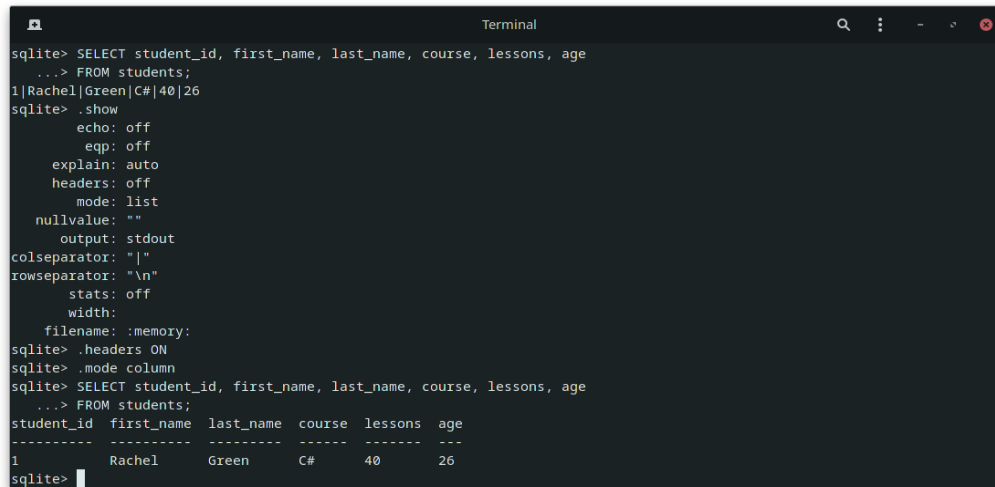
```
SELECT student_id, first_name, last_name, course, lessons, age  
FROM students;
```

Этот же запрос может выглядеть так:

```
SELECT *  
FROM students;
```

В результате из таблицы будут извлечены все строки. Результат может выглядеть без разграничения по колонкам и без заголовка. Чтобы это исправить выполняем:

```
.show  
.headers ON  
.mode column
```

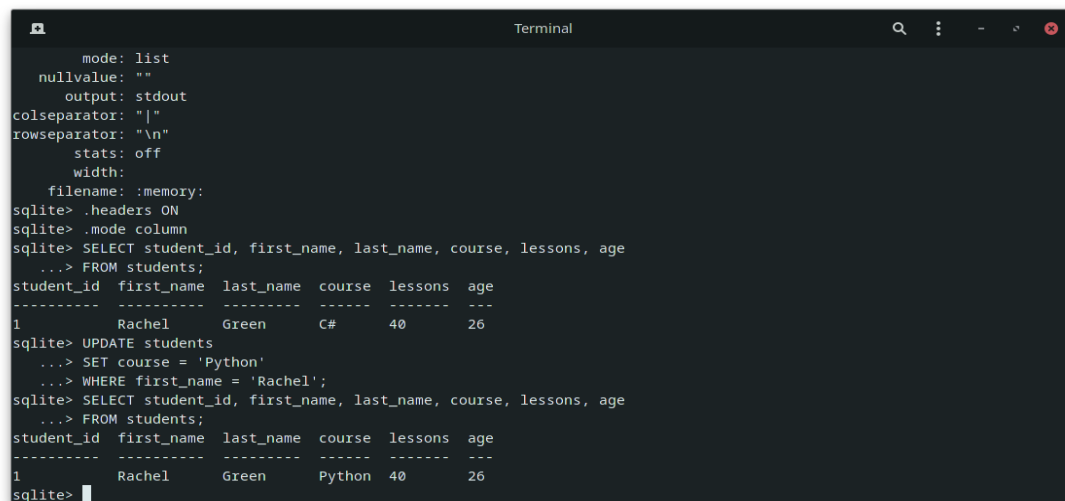


```
sqlite> SELECT student_id, first_name, last_name, course, lessons, age  
...> FROM students;  
1|Rachel|Green|C#|40|26  
sqlite> .show  
echo: off  
eqp: off  
explain: auto  
headers: off  
mode: list  
nullvalue: ""  
output: stdout  
colseparator: "|"  
rowseparator: "\n"  
stats: off  
width:  
filename: :memory:  
sqlite> .headers ON  
sqlite> .mode column  
sqlite> SELECT student_id, first_name, last_name, course, lessons, age  
...> FROM students;  
student_id first_name last_name course lessons age  
-----  
1 Rachel Green C# 40 26  
sqlite>
```

Изменение данных

Предположим, что поле course для пользователя 'Rachel' необходимо изменить на 'Python'. Выполняем следующую команду:

```
UPDATE students  
SET course = 'Python'  
WHERE first_name = 'Rachel';
```



```
mode: list  
nullvalue: ""  
output: stdout  
colseparator: "|"  
rowseparator: "\n"  
stats: off  
width:  
filename: :memory:  
sqlite> .headers ON  
sqlite> .mode column  
sqlite> SELECT student_id, first_name, last_name, course, lessons, age  
...> FROM students;  
student_id first_name last_name course lessons age  
-----  
1 Rachel Green C# 40 26  
sqlite> UPDATE students  
...> SET course = 'Python'  
...> WHERE first_name = 'Rachel';  
sqlite> SELECT student_id, first_name, last_name, course, lessons, age  
...> FROM students;  
student_id first_name last_name course lessons age  
-----  
1 Rachel Green Python 40 26  
sqlite>
```

Для удаления записи, используется команда DELETE:

```
DELETE FROM students  
WHERE first_name = 'Rachel';
```

Закрепление материала

- Что такое СУБД SQLite и что из себя представляет данное хранилище?
- Какие аффинные типы оно поддерживает и зачем нужен данный механизм-аффинирование типов?

- Как добавить собственный тип данных и что нужно создать/зарегистрировать, используя модуль `sqlite3` в Python, чтобы данные типы воспринимались в процессе выборки и вставки данных в SQLite хранилище?
- Что такое SQLite?
- Что такое БД и СУБД?
- Где SQLite хранит базу? Нужен ли отдельный сервер для этого?

Дополнительное задание

Создайте таблицу для учёта личного бюджета. Напишите несколько запросов, для добавления данных о расходах.

Самостоятельная деятельность учащегося

Задание 1

Сделать таблицу для подсчёта личных расходов со следующими полями: `id`, назначение, сумма, время.

Задание 2

Создать консольный интерфейс (CLI) на Python для добавления новых записей в базу данных

Задание 3

Создать агрегатные функции для подсчёта общего количества расходов и расходов за месяц. Обеспечить соответствующий интерфейс для пользователя.

Задание 4

Измените таблиц так, чтобы можно было добавить не только расходы, а и доходы

Задание 5

Замените назначение на `MCC` и используйте его для определения назначения платежа

Задание 6

Настройте интеграцию с API своего банка для автоматической загрузки операций по карте

Рекомендуемые ресурсы

Официальный сайт Python - SQLite

<https://docs.python.org/3.11/library/sqlite3.html>