

# SQLite. Синтаксис і запити

**№ уроку:** 4    **Курс:** Python Advanced

**Засоби навчання:** PyCharm

## Огляд, мета та призначення уроку

Вивчити основи роботи з бібліотекою `sqlite3` та використання СУБД. Розглянути особливості цієї бібліотеки з практичним нахилом.

## Вивчивши матеріал цього заняття, учень зможе:

- Використовувати SQLite у своїх завданнях.
- Створювати користувацькі агрегатні та звичайні функції, розширюючи стандартні можливості SQL.
- Вивчити основні запити для роботи з SQLite.

## Зміст уроку

- Основні поняття та особливості СУБД SQLite.
- `sqlite3`-бібліотека в Python.

## Резюме

**База даних (БД)** – це організована структура, яка призначена для зберігання, зміни та обробки взаємопов'язаної інформації, переважно великих обсягів. Бази даних активно використовуються для динамічних сайтів зі значними обсягами даних. Часто це інтернет-магазини, портали, корпоративні сайти.

**Система управління базами даних (СУБД)** – це комплекс програмних засобів, які необхідні для створення структури нової бази, її наповнення, редагування вмісту та зображення інформації. Найбільш поширеними СУБД є MySQL, PostgreSQL, Oracle, Microsoft SQL Server, SQLite.

**Бази даних** – це місця зберігання даних, а СУБД дають змогу керувати ними. Найчастіше БД і СУБД – нерозривні, але не завжди. У деяких довідниках можна побачити вживання терміна БД замість СУБД, але це некоректна взаємозаміна.

## Що таке SQLite?

**SQLite** – це система управління реляційними базами даних (СУБД), що міститься в бібліотеці C. На відміну від багатьох інших систем управління базами даних, SQLite не використовується механізм клієнт-сервер. Бібліотека SQLite одночасно є і клієнтом, і сервером, а дані зберігаються локально у звичайному файлі.

SQLite – це найчастіше використовуваний механізм баз даних у світі. Він вбудований у всі мобільні телефони та більшість комп'ютерів і входить до складу багатьох інших програм, які люди використовують щодня.

Формат файлу SQLite є стабільним, кросплатформним та обернено сумісним. Розробники обіцяють зберегти його в такому вигляді як мінімум до 2050 року. Файли бази даних SQLite зазвичай використовуються як контейнери для передання великого обсягу даних між системами та як довгостроковий архівний формат для даних. Активно використовується понад 1 трильйон (1e12) баз даних SQLite.

## Де SQLite застосовується?

На смартфонах, у браузерях, вебзастосунках, серверах. Наприклад, Firefox зберігає куки в sqlite, а більшість мобільних застосунків зберігають користувацькі дані в sqlite. SQLite чудово підходить для застосунків, яким потрібно стабільно зберігати великі дані, але з невеликим навантаженням. Параметри за замовчуванням працюють на надійність, а не на продуктивність.

## Підготовка оточення для роботи з SQLite

На операційній системі Linux (Ubuntu) достатньо виконати таку команду в терміналі:

```
sudo apt install sqlite3
```

Для зупинки на Windows дотримуйтесь інструкцій у розділі «Precompiled Binaries for Windows» на сторінці <https://www.sqlite.org/download.html>:

### Precompiled Binaries for Windows

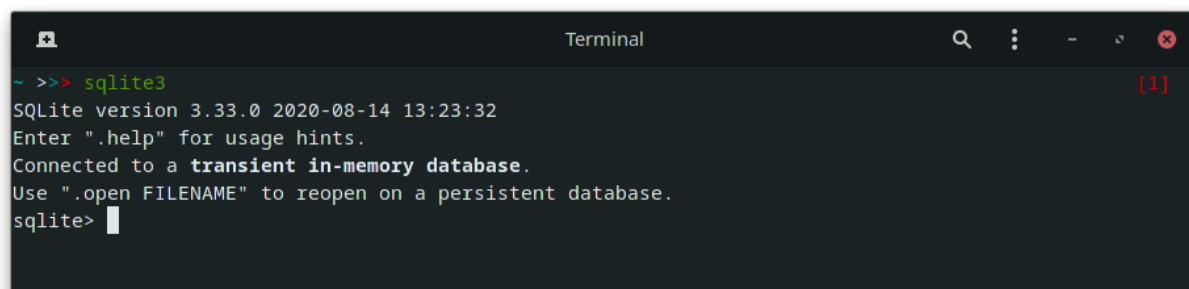
- 1 [sqlite-dll-win32-x86-3330000.zip](#) (489.25 KiB) **32-bit DLL (x86)** for SQLite version 3.33.0.  
(sha3: d669a92271ad31bbefe4ba827c3d6621179b1964b5dd1f27c6fbaa9c4cf6a082)
- 2 [sqlite-dll-win64-x86-3330000.zip](#) (809.89 KiB) **64-bit DLL (x64)** for SQLite version 3.33.0.  
(sha3: 1b078c320adb1d9ff57c508eb0d2ef05bf13ca2e9cbb37c2ea72099373cdfd61)
- [sqlite-tools-win32-x86-3330000.zip](#) (1.76 MiB) 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.  
(sha3: 5b13a53afe89ca0a975f0c166d5e4c33c83695ceea75ae297f5471df3adde4b0)

Залежно від розрядності системи, завантажуюмо або перший, або другий файл.

Щоби перевірити успішне встановлення, відкрийте термінал (як відкрити термінал на Windows) і напишіть таку команду:

```
sqlite3
```

Результат має бути приблизно таким:



```
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> [1]
```

Спеціальні команди призначені на формування таблиць та інших адміністративних операцій. Усі вони починаються із крапки.

Ці «крапкові команди» зазвичай використовуються для зміни формату виведення запитів або для виконання заздалегідь підготовлених операторів запиту. Спочатку було лише кілька крапкових команд, але з роками накопичилося багато нових функцій, тому зараз їх понад 60.

Пройдемося за списком команд, які можуть стати в пригоді:

- .show – показує поточні налаштування заданих параметрів;
- .databases – показує назву баз даних і файлів;
- .quit – вихід з sqlite3;
- .tables – показує поточні таблиці;
- .schema – зображає структуру таблиці;
- .header – зображає або приховує шапку таблиці;
- .mode – вибір режиму відображення даних таблиці;
- .dump – робить копію бази даних у текстовому форматі.

Щоб отримати список доступних команд з крапкою, можна ввести `.help` без аргументів. Або введіть `.help show` для отримання докладної інформації про команду `.show`. Більше про спеціальні команди можна прочитати на офіційному сайті.

**SQL** – це стандартна мова для взаємодії з БД. Завдяки їй можна повідомити БД про запис, обробку та вилучення даних.

SQL складається з безлічі типів операторів, які можна неофіційно класифікувати як підмови: мова запитів даних (DQL), мова визначення даних (DDL), мова керування даними (DCL) та мова керування даними (DML). Сфера застосування SQL вміщує запит даних, маніпулювання даними (вставлення, оновлення та видалення), визначення даних (створення схеми та модифікації) та контроль доступу до даних.

Хоча SQL по суті є декларативною мовою, вона також вміщує процедурні елементи.

**Мова опису даних (DDL)** складається з команд для створення таблиці, зміни та видалення баз даних, таблиць та іншого:

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

**Мова керування даними (DML)** дає користувачу змогу маніпулювати даними (додавати/змінювати/вилучати):

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

**Мова запитів DQL** дає змогу здійснювати вибірку даних:

- `SELECT`

SQLite також підтримує й безліч інших команд, список яких можна знайти тут. Оскільки цей урок призначений для початківців, ми обмежимося перерахунком набором команд.

### Типи даних SQL

Тип даних SQLite – це атрибут, який визначає тип даних будь-якого об'єкта. Кожен стовпець, змінна та вираз має зв'язаний тип даних у SQLite.

Типи даних використовуються під час створення таблиць. SQLite тип даних значення пов'язаний із самим значенням і повідомляє БД спосіб роботи з ним.

Кожне значення, що зберігається в базі даних SQLite, має один із таких класів (не типи) зберігання:

**Класи зберігання SQLite**

Назва	Опис
NULL	Значення – значення NULL.
INTEGER	Значення – ціле число зі знаком, збережене в 1, 2, 3, 4, 6 або 8 байтах залежно від величини значення.

REAL	Значення – значення з пересуваною комою, яке зберігається як 8-байтове число з пересувною крапкою IEEE.
TEXT	Значення – текстовий рядок, який зберігається за допомогою кодування бази даних (UTF-8, UTF-16BE або UTF-16LE).
BLOB	Значення – блок даних, який зберігається так само як він був введений.

SQLite підтримує концепцію **affinity (близькість)** типу до стовпців. Будь-який стовпець може зберігати дані будь-якого типу, але найкращий клас зберігання для стовпця – це affinity. Кожному стовпцю таблиці в базі даних SQLite3 присвоюється одна з таких афінностей:

#### Тип злиття SQLite

Назва	Опис
TEXT	У цьому стовпці зберігаються всі дані з використанням класів NULL, TEXT або BLOB.
NUMERIC	Цей стовпець може містити значення, використовуючи всі п'ять класів зберігання.
INTEGER	Працює так само як стовпець з NUMERIC-спорідненістю, за винятком у виразі CAST.
REAL	Поводиться як стовпець з NUMERIC-спорідненістю, за винятком того, що він наводить цілі значення в представлення з пересувною комою.
NONE	Стовпець з афінністю NONE не віддає перевагу одному класу зберігання над іншим. Не робить спроби примусити дані з одного класу зберігання до іншого.

Нарешті про самі типи. У наступних списках таблиць перераховані назви типів даних, які можна використовувати під час створення таблиць SQLite3 з відповідною застосовною спорідненістю.

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 має окремого булевого класу зберігання (True). Натомість булеві значення зберігаються як цілі числа 0 (брехня) й 1 (істина).

SQLite не має окремого класу зберігання для зберігання дат та/або часу, але SQLite спроможний зберігати дати та час як значення TEXT, REAL або INTEGER.

Докладніший опис цієї теми можна прочитати [тут](#).

### Створення бази даних

Завдання: створити базу даних для зберігання ім'я (first\_name) та прізвища (last\_name) користувача, назви навчального курсу (course) та кількості занять (lessons).

Тепер створімо базу даних. Якщо ви ще перебуваєте в інтерфейсі sqlite3, наберіть команду .quit для виходу. Тепер вводимо в термінал:

```
sqlite3 students.db
```

У результаті в поточному каталозі в нас з'явиться файл *students.db*.

Якщо не вказати назву файлу, sqlite3 створить **тимчасову** базу даних в оперативній пам'яті.

### Створення таблиці

Для зберігання студентів необхідно створити таблицю. Назвемо її *students\_list*. Виконуємо 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 );
```

Щоб отримати структуру таблиці, наберіть .schema students\_list. student\_id – це ключовий ключ кожного запису. Бази даних необхідні якісь унікальні дані всередині стоки, щоб ефективно зберігати та шукати дані. Він має атрибут AUTOINCREMENT, який говорить про те, що база даних автоматично заповнюватиме це поле. Атрибут NOT NULL зазначає: поля не можуть бути порожніми.

Тепер можемо внести дані до таблиці.

### Зміна таблиці

#### Додавання нової колонки

Для додавання нової колонки варто використовувати команду ALTER. Наприклад, введемо поле age. Виконуємо команду:

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

Для перевірки, що в поді дійсно додалося: .schema students\_list.

#### Перейменування таблиці

Також ми можемо використовувати команду ALTER для перейменування таблиці students\_list на students:

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

#### Видалення таблиці

Для видалення нашої таблиці виконайте таку команду (*поки що робити цього не варто*):

```
DROP TABLE students;
```

### Внесення даних

Припустимо, що нам необхідно внести такий запис:

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

Для вставлення скористаємося командою INSERT.

**INSERT INTO** students ( first\_name, last\_name, course, lessons, age )

**VALUES** ( 'Rachel', 'Green', 'C#', 40, 26 );

Вказувати значення student\_id не потрібно – воно сформується автоматично завдяки налаштуванню AUTOINCREMENT.

### Отримання даних

Для вибору даних скористаємося командою SELECT.

**SELECT** student\_id, first\_name, last\_name, course, lessons, age

**FROM** students;

Цей запит може виглядати так:

**SELECT** \*

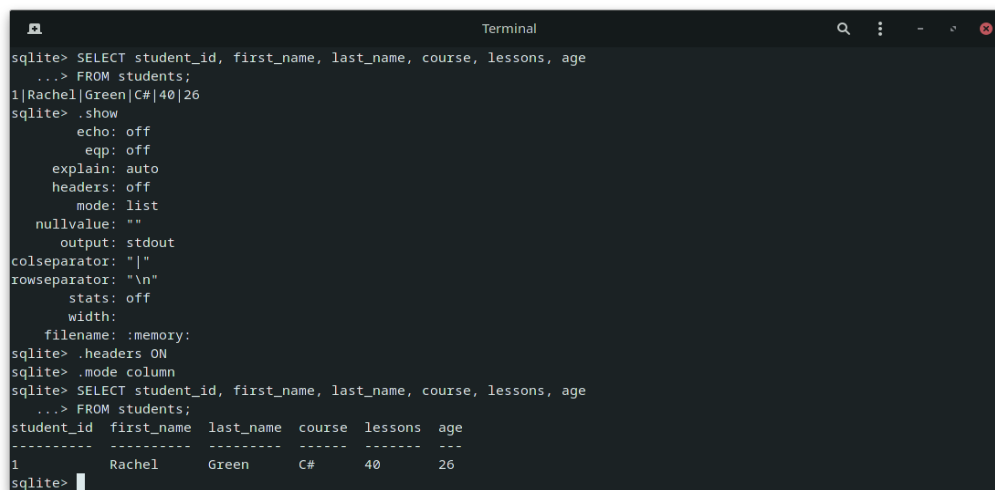
**FROM** students;

В результаті з таблиці буде вилучено всі рядки. Результат може виглядати без розмежування за колонками та без заголовка. Щоби це виправити, виконуємо:

.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>
```

### Зміна даних

Припустимо, що поле course для користувача «Rachel» необхідно змінити на «Python».

Виконуємо таку 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 видалення запису використовується команда DELETE:

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

### Закріплення матеріалу

- Що таке СУБД SQLite?
- Які аффінні типи воно підтримує та навіщо потрібен цей механізм аффінування типів?
- Як додати власний тип даних і що потрібно створити/zareєstrувати, використовуючи модуль sqlite3 у Python, щоби ці типи сприймалися в процесі вибору та вставлення даних у SQLite сховище?
- Що таке SQLite?
- Що таке БД та СУБД?
- Де SQLite зберігає базу? Чи потрібний окремий сервер для цього?

### Додаткове завдання

Створіть таблицю для врахування власного бюджету. Напишіть кілька запитів, щоби додати дані про витрати.

### Самостійна діяльність учня

Завдання 1

Зробіть таблицю для підрахунку особистих витрат із такими полями: id, призначення, сума, час.

Завдання 2

Створіть консольний інтерфейс (CLI) на Python для додавання нових записів до бази даних.

Завдання 3

Створіть агрегатні функції для підрахунку загальної кількості витрат і витрат за місяць. Забезпечте відповідний інтерфейс користувача.

Завдання 4

Змініть таблицю так, щоби можна було додати не лише витрати, а й доходи.

Завдання 5

Замініть призначення на **MCC** та використовуйте його для визначення призначення платежу.

## Завдання 6

Налаштуйте інтеграцію з API свого банку для автоматичного завантаження операцій за картою.

### Рекомендовані ресурси

Офіційний Python – SQLite

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