

## Задание №3

### Шаг 1. SQL

Основные варианты СУБД (систем управления баз данных):

- PostgreSQL
- MySQL
- SQLite

**SQLite** – самый простой вариант создания базы данных. У SQLite нет поддержки клиент/сервер, т.е. к базе данных SQLite можно подключиться только на том же сервере где установлена сама база данных, нельзя напрямую подключиться с другого сервера или ПК. Базы SQLite хранятся в текстовых файлах, что снижает скорость её работы с большими объемами данных. Для небольших баз данных SQLite – идеальный вариант.

### Шаг 2. Установка SQLite

Для начала необходимо установить пакет sqlite3. Перед установкой новых пакетов рекомендуется обновить имеющиеся пакеты. Для этого выполняем команду:

```
sudo apt update
```

```
bsk23-01-student-1@BSK23-01-1:~$ sudo apt update
Get:1 file:/etc/apt/mirrors/debian.list Mirrorlist [30 B]
Get:5 file:/etc/apt/mirrors/debian-security.list Mirrorlist [39 B]
Hit:2 https://deb.debian.org/debian bookworm InRelease
Hit:3 https://deb.debian.org/debian bookworm-updates InRelease
Hit:4 https://deb.debian.org/debian bookworm-backports InRelease
Hit:6 https://deb.debian.org/debian-security bookworm-security InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
bsk23-01-student-1@BSK23-01-1:~$
```

```
sudo apt install sqlite3
```

```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  sqlite3-doc
The following NEW packages will be installed:
  sqlite3
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 353 kB of archives.
After this operation, 546 kB of additional disk space will be used.
```

```
Get:1 file:/etc/apt/mirrors/debian.list Mirrorlist [30 B]
Get:2 https://deb.debian.org/debian bookworm/main amd64 sqlite3 amd64
3.40.1-2 [353 kB]
Fetched 353 kB in 1s (326 kB/s)
Selecting previously unselected package sqlite3.
(Reading database ... 26808 files and directories currently installed.)
Preparing to unpack .../sqlite3_3.40.1-2_amd64.deb ...
Unpacking sqlite3 (3.40.1-2) ...
Setting up sqlite3 (3.40.1-2) ...
Processing triggers for man-db (2.11.2-2) ...
bsk23-01-student-1@BSK23-01-1:~$
```

Проверим, что установили:

```
apt list | grep -E 'sqlite3.*installed'
```

В полученном списке должны увидеть:

```
sqlite3/stable,now 3.40.1-2 amd64 [installed]
```

### Шаг 3. Справка по sqlite3

Команда **man** отображает справку по использованию пакетов.

Посмотрим параметры запуска sqlite3

```
man sqlite3
```

#### SYNOPSIS

```
sqlite3 [options] [databasefile] [SQL]
```

Пункты в квадратных скобках говорят о том, что при запуске SQLite мы можем (но не обязаны) указать опции, имя базы данных и запрос SQL

```
If the database file does not exist, it will be created. If the database
file does exist, it will be opened.
```

Если при запуске sqlite укажем имя несуществующей базы данных, то она будет создана. Так и сделаем.

### Шаг 4. Создание базы данных

Смотрим в какой директории мы сейчас находимся:

```
pwd
```

Переходим в нужную директорию с помощью команды cd. Запускаем SQLite с именем базы

данных (пусть будет «pythondb»):

```
sqlite3 pythondb
```

```
SQLite version 3.40.1 2022-12-28 14:03:47  
Enter ".help" for usage hints.  
sqlite>
```

Мы находимся в программе sqlite. Короткий текст советует нам посмотреть справку. Посмотрим:

```
sqlite> .help
```

Выполним команду отображения подключенных баз данных:

```
sqlite> .databases  
main: /home/bsk23-01-student-1/python/pythondb r/w
```

### Шаг 5. Создание таблицы CREATE TABLE

Команды в SQL называют запросами. Создадим новую таблицу в нашей базе данных, написав запрос на языке SQL. Таблица будет называться «users». Поля таблицы:

**Name** с типом *String* (в этом столбце будет текст)

**Age** с типом *Int* (в этом столбце будут целые числа)

**Mac** с типом *String* (в этом поле будем указывать MAC-адрес телефона пользователя)

```
sqlite> CREATE TABLE users(name String, age Int, mac String);
```

Important: Обратите внимание, что в конце запроса должны быть кавычки

Посмотрим, какие теперь есть таблицы в нашей базе данных:

```
sqlite> .tables  
users
```

### Шаг 6. Заполнение данных таблицы INSERT

Добавим в созданную таблицу данные. Для этого используем SQL запрос «INSERT».

Указываем, что мы будем добавлять данные в поля name и age таблицы users «INTO users(name,age,mac)».

Данные задаём после «VALUES». Каждая строка в круглых скобках через запятую.

Внутри круглых скобок значения полей name (строка в кавычках) и age. «('Ivan',25,'mac1')» – одна строка. «('Dasha',23,'mac2')» - другая строка. Добавим 3 строки:

```
sqlite> INSERT INTO users(name, age, mac) VALUES ('Ivan',25, 'mac1'),  
('Dasha',23,'mac2'), ('Juliya',21,'mac3');
```

### Шаг 7. Отображение данных таблицы SELECT

Отообразим данные из таблицы users. Это делается SQL запросом «SELECT». После «SELECT» указывается список столбцов, которые нам нужно отобразить. Если написать «\*» то отобразятся все столбцы таблицы. После «FROM» пишем название таблицы, из которой нужно отобразить данные:

```
sqlite> SELECT * FROM users;
```

```
Ivan|25|mac1  
Dasha|23|mac2  
Juliya|21|mac3
```

Включим отображение заголовков столбцов:

```
sqlite> .headers on
```

Снова отобразим данные из таблицы users. Теперь с заголовками столбцов «name|age|mac»:

```
sqlite> SELECT * FROM users;
```

```
name|age|mac  
Ivan|25|mac1  
Dasha|23|mac2  
Juliya|21|mac3
```

Для того, чтобы выбрать только один столбец в списке полей вместо «\*» нужно написать название столбца:

```
sqlite> SELECT name FROM users;
```

```
name  
Ivan  
Dasha  
Juliya
```

Можно указать несколько столбцов через запятую. Столбцы будут отображаться в указанном в запросе порядке:

```
sqlite> SELECT age,name FROM users;
```

```
age|name
```

```
25|Ivan
23|Dasha
21|Juliya
```

### Шаг 8. Условия выборки WHERE

Выберем не все данные, а только данные соответствующие нужным критериям. Для этого в запрос «SELECT» добавим условия «WHERE»:

```
sqlite> SELECT age,name FROM users WHERE age<24;

age|name
23|Dasha
21|Juliya
```

Получили только 2 записи, т.к. «Ivan» не попал в нашу выборку по возрасту. Можно выполнить поиск по конкретному условию:

```
sqlite> SELECT name,age FROM users WHERE name='Juliya';

name|age
Juliya|21
```

### Шаг 9. Создание и заполнение второй таблицы

Создадим вторую таблицу. Это будет таблица MAC-адресов активных регистраций устройств WiFi-роутера с уровнями сигналов:

```
sqlite> CREATE TABLE registrations (mac String, signal String);
```

Посмотрим обновленный список таблиц в нашей базе данных:

```
sqlite> .tables

registrations users
```

Пока добавим в таблицу registrations данные вручную (в дальнейшем нам нужно будет автоматически записывать данные, полученные с роутера, в эту таблицу):

```
sqlite> INSERT INTO registrations (mac, signal) VALUES ('mac3', '-72'),
('mac1', '-85'), ('mac5', '-69');
```

Отообразим данные из этой таблицы:

```
sqlite> SELECT * FROM registrations;

mac|signal
```

```
mac3 | -72  
mac1 | -85  
mac5 | -69
```

## Шаг 10. Связь двух таблиц

Теперь наша задача - связать 2 таблицы: найти соответствия MAC-адресов в таблице «users» MAC-адресам в таблице «registrations». Для этого в разделе FROM запроса SELECT указываем обе таблицы через запятую «FROM users, registrations», а в разделе WHERE указываем условие равенства полей mac в обеих таблицах «WHERE users.mac = registrations.mac». Т.к. таблиц в запросе теперь несколько названия столбцов нужно указывать вместе с названием таблицы («users.mac», «registrations.mac»):

```
sqlite> SELECT * FROM users, registrations WHERE users.mac =  
registrations.mac;  
  
name|age|mac|mac|signal  
Ivan|25|mac1|mac1|-85  
Juliya|21|mac3|mac3|-72
```

Мы получили строки обеих таблиц, в которых поля «mac» равны. Можно выбрать не все столбцы, а только определенные. Для это вместо «\*» нужно перечислить названия столбцов через запятую (также с указанием таблиц):

```
sqlite> SELECT users.name, registrations.signal FROM users, registrations  
WHERE users.mac = registrations.mac;  
  
name|signal  
Ivan|-85  
Juliya|-72
```

Для начала нам этого хватит. Выходим из SQLite с помощью команды .quit:

```
sqlite> .quit
```

Мы вернемся в SQLite когда будем создавать таблицу пользователей WiFi роутера. Теперь нам нужно научиться выполнять SQL запросы к базе данных SQLite из Python.

## Шаг 11. SQL из Python

В Python есть готовые библиотеки почти на все случаи жизни. Есть библиотека для работы с **sqlite3**. Эта библиотека содержит функции для подключения к базе данных SQLite, для выполнения SQL запросов к базе данных, получения данных и т.п.

Для подключения библиотеки SQLite необходимо в начале скрипта написать «**import sqlite3**».

Создайте новый файл скрипта Python. Используем функции библиотеки «connect», «cursor», «execute», «fetchall» и «close»:

```
import sqlite3 #подключение библиотеки sqlite3
con = sqlite3.connect('/home/bsk23-01-student-1/python/pythondb')
#подключение к базе данных sqlite
cur = con.cursor() #создание курсора для подключения
cur.execute('SELECT * FROM users') #выполнение запроса
users = cur.fetchall() #запись результатов выполнения запроса в массив users
print(users) #вывод массива users на экран
con.close() #закрытие соединения к базе данных
```

Результат выполнения скрипта - массив данных (таблица):

```
[('Ivan', 25, 'mac1'), ('Dasha', 23, 'mac2'), ('Juliya', 21, 'mac3')]
```

### Шаг 12. Выборочное чтение данных

Для чтения одной строки массива (таблицы) нужно после названия массива «users» написать номер строки в квадратных скобках. Например «users[0]»:

```
import sqlite3
con = sqlite3.connect('/home/bsk23-01-student-1/python/pythondb')
cur = con.cursor()
cur.execute('SELECT * FROM users')
users = cur.fetchall()
print(users[0])
con.close()
```

Результат выполнения скрипта - первая (нулевая) строка массива:

```
('Ivan', 25, 'mac1')
```

Note: Попробуйте написать users[2] для отображения третьей строки.

Для чтения одной ячейки массива (таблицы) нужно написать координаты ячейки таблицы после названия массива «users» в двух квадратных скобках [x][y], где x - номер строки, y - номер столбца. Например «users[1][0]» отобразит первый столбец второй строки: Dasha

### Шаг 13. Чтение данных в цикле

Построчное чтение данных в цикле:

```
import sqlite3
con = sqlite3.connect('/home/bsk23-01-student-1/python/pythondb')
cur = con.cursor()
cur.execute('SELECT * FROM users')
users = cur.fetchall()
```

Last update: 2024/03/15  
15:08

zadanie\_3.\_znakomstvo\_s\_sql [https://sibgu-itlab-wiki.data-pool.ru/zadanie\\_3.\\_znakomstvo\\_s\\_sql?rev=1710515338](https://sibgu-itlab-wiki.data-pool.ru/zadanie_3._znakomstvo_s_sql?rev=1710515338)

```
for user in users:  
    print(user)  
con.close()
```

Сохраним этот скрипт. Вернемся к нему после того как считаем данные о регистрациях устройств в WiFi сети роутера.

From:

<https://sibgu-itlab-wiki.data-pool.ru/> - **SIBGU-ITLAB-WIKI**

Permanent link:

[https://sibgu-itlab-wiki.data-pool.ru/zadanie\\_3.\\_znakomstvo\\_s\\_sql?rev=1710515338](https://sibgu-itlab-wiki.data-pool.ru/zadanie_3._znakomstvo_s_sql?rev=1710515338)

Last update: **2024/03/15 15:08**

