

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ  
ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»  
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ**

**Отчёт о лабораторной работе №9 по дисциплине основы программной  
инженерии**

Выполнила:

Нестеренко Тамара Антоновна,  
2 курс, группа ПИЖ-б-о-20-1,

Проверил:

Доцент кафедры инфокоммуникаций,  
Воронкин Р.А.

Ставрополь, 2022 г.

# ВЫПОЛНЕНИЕ

## 1. Практическая часть

```
import sqlite3
dir(sqlite3)
['Binary', 'Connection', 'Cursor', 'DataError', 'DatabaseError', 'Date', 'DateFromTicks', 'Error', 'IntegrityError', 'InterfaceError', 'InternalError', 'NotSupportedError',
'OperationalError', 'OptimizedUnicode', 'PARSE_COLNAMES', 'PARSE_DECLTYPES', 'PrepareProtocol', 'ProgrammingError', 'Row', 'SQLite ALTER_TABLE', 'SQLite ANALYZE',
'SQLITE ATTACH', 'SQLite CREATE_INDEX', 'SQLite CREATE_TABLE', 'SQLite CREATE_TEMP_INDEX', 'SQLite CREATE_TEMP_TABLE', 'SQLite CREATE_TEMP_TRIGGER', 'SQLite CREATE_TEMP_VIEW',
'SQLITE CREATE_TRIGGER', 'SQLite CREATE_VIEW', 'SQLite CREATE_VTABLE', 'SQLite DELETE', 'SQLite DENY', 'SQLite DETACH', 'SQLite DONE', 'SQLite DROP_INDEX',
'SQLITE DROP_TABLE', 'SQLite DROP_TEMP_INDEX', 'SQLite DROP_TEMP_TABLE', 'SQLite DROP_TEMP_TRIGGER', 'SQLite DROP_TEMP_VIEW', 'SQLite DROP_TRIGGER', 'SQLite DROP_VIEW',
'SQLITE DROP_VTABLE', 'SQLite FUNCTION', 'SQLite IGNORE', 'SQLite INSERT', 'SQLite OK', 'SQLite PRAGMA', 'SQLite READ', 'SQLite RECURSIVE', 'SQLite REINDEX',
'SQLITE SAVEPOINT', 'SQLite SELECT', 'SQLite TRANSACTION', 'SQLite UPDATE', 'Time', 'TimeFromTicks', 'Timestamp', 'TimestampFromTicks', 'Warning', '__builtins__',
'__cached__', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__path__', '__spec__', 'adapt', 'adapters', 'apilevel', 'collections', 'complete_statement',
'connect', 'converters', 'datetime', 'dbapi2', 'enable_callback_tracebacks', 'enable_shared_cache', 'paramstyle', 'register_adapter', 'register_converter', 'sqlite_version',
'sqlite_version_info', 'threadsafety', 'time', 'version', 'version_info']
```

Рисунок 1 – Пример вывода классов

```
>>> db = sqlite3.connect('site.sqlite')
>>> type(db)
<class 'sqlite3.Connection'>
```

Рисунок 2 – Пример вызова функции connect()

```
# !/usr/bin/env python3
# -*- coding: utf-8 -*-

import sqlite3

con = sqlite3.connect('mydatabase.db')
```

Рисунок 3 – Пример вызова функции connect()

```
# !/usr/bin/env python3
# -*- coding: utf-8 -*-

import sqlite3

con = sqlite3.connect('mydatabase.db')

cursor_obj = con.cursor()
```

Рисунок 4 – Пример использования курсора

```
# !/usr/bin/env python3
# -*- coding: utf-8 -*-

import sqlite3
from sqlite3 import Error

def sql_connection():
    try:
        con = sqlite3.connect(':memory')
        print("Connection is established: Database is created in memory")

    except Error:
        print(Error)

    finally:
        con.close()

if __name__ == "__main__":
    sql_connection()
```

Рисунок 5 – Пример создания базы данных

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sqlite3
from sqlite3 import Error

def sql_connection():
    try:
        con = sqlite3.connect('mydatabase.db')
        return con

    except Error:
        print(Error)

    return None

def sql_table(con):
    cursor_obj = con.cursor()
    cursor_obj.execute(
        """

        CREATE TABLE employees (
            id integer PRIMARY KEY,
            name text,
            salary real,
            department text,
            position text,
            hireDate text)
        """
    )
```

Рисунок 6 – Пример создания таблиц

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sqlite3

con = sqlite3.connect('mydatabase.db')

def sql_insert(con, entities):
    cursor_obj = con.cursor()
    cursor_obj.execute(
        """
        INSERT INTO employees (id, name, salary, department, position, hireDate)
        VALUES(?, ?, ?, ?, ?, ?)
        """,
        entities
    )
    con.commit()

entities = (2, 'Andrew', 800, 'IT', 'Tech', '2018-02-06')
sql_insert(con, entities)

```

Рисунок 7 – Пример вставки данных в таблицу

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sqlite3

con = sqlite3.connect('mydatabase.db')

def sql_update(con):
    cursor_obj = con.cursor()
    cursor_obj.execute(
        "UPDATE employees SET name = 'Rogers' where id = 2"
    )
    con.commit()

sql_update(con)

```

Рисунок 8 – Пример обновления данных в таблицах

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sqlite3

con = sqlite3.connect('mydatabase.db')

def sql_fetch(con):
    cursor_obj = con.cursor()
    cursor_obj.execute("SELECT * FROM employees")

    rows = cursor_obj.fetchall()
    for row in rows:
        print(row)

sql_fetch(con)

```

Рисунок 9 – Пример выборки данных из таблицы

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sqlite3

con = sqlite3.connect('mydatabase.db')

def sql_fetch(con):
    cursor_obj = con.cursor()
    cursor_obj.execute(
        "SELECT id, name, FROM employees WHERE salary > 800.0"
    )

    rows = cursor_obj.fetchall()
    for row in rows:
        print(row)

sql_fetch(con)

```

Рисунок 10 – Пример получения списка таблиц

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sqlite3

con = sqlite3.connect('mydatabase.db')

def sql_fetch(con):
    cursor_obj = con.cursor()
    cursor_obj.execute(
        "SELECT name from sqlite_master where type='table'"
    )

    print(cursor_obj.fetchall())

sql_fetch(con)

```

Рисунок 11 – Пример проверки существования таблицы

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sqlite3

con = sqlite3.connect('mydatabase.db')

def sql_fetch(con):
    cursor_obj = con.cursor()
    cursor_obj.execute(
        "CREATE TABLE IF NOT EXISTS projects(id INTEGER, name TEXT)"
    )

    con.commit()

sql_fetch(con)

```

Рисунок 12 – Пример проверки существования таблицы

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sqlite3

con = sqlite3.connect('mydatabase.db')

cursor_obj = con.cursor()
cursor_obj.execute(
    "CREATE TABLE IF NOT EXISTS projects(id INTEGER, name TEXT)"
)

data = [
    (1, "Ridesharing"),
    (2, "Water Purifying"),
    (3, "Forensics"),
    (4, "Botany")
]

cursor_obj.executemany("INSERT INTO projects VALUES(?, ?)", data)

con.commit()

```

Рисунок 13 – Пример массовой вставки

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sqlite3
import datetime

con = sqlite3.connect('mydatabase.db')

cursor_obj = con.cursor()
cursor_obj.execute(
    """
    CREATE TABLE IF NOT EXISTS assignments(
        id INTEGER, name TEXT, date DATE
    )
    """
)

data = [
    (1, "Ridesharing", datetime.date(2017, 1, 2)),
    (2, "Water Purifying", datetime.date(2018, 3, 4))
]

cursor_obj.executemany("INSERT INTO assignments VALUES(?, ?)", data)

con.commit()

```

Рисунок 14 – Пример использования datetime



```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import argparse
import sqlite3
import typing as t
from pathlib import Path

def display_workers(staff: t.List[t.Dict[str, t.Any]]) -> None:
    if staff:
        line = '+--{}--{}--{}--{}--+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 8
        )
        print(line)
        print(
            '|{: ^4} | {: ^30} | {: ^20} | {: ^8} |'.format(
                "№",
                "Ф.И.О.",
                "Должность",
                "Год"
            )
        )
        print(line)

        for idx, worker in enumerate(staff, 1):
            print(
                '|{: >4} | {: <30} | {: <20} | {: >8} |'.format(

```

Рисунок 15 – Пример программы для решения поставленной задачи

```

PS C:\Users\тома нестеренко\PycharmProjects\1> python 2.py add --db="names.db" -s="Nesterenko" -n="Tamara" -z="Oven" -y=2022
PS C:\Users\тома нестеренко\PycharmProjects\1> python 2.py add --db="names.db" -s="Nesterenko" -n="Ivan" -z="Pyba" -y=2021
PS C:\Users\тома нестеренко\PycharmProjects\1> python 2.py display --db="names.db"
+-----+-----+-----+-----+-----+
|  №  |      Фамилия      |      Имя      |  Знак зодиака  |  Год рождения  |
+-----+-----+-----+-----+-----+
|  1  |      Nesterenko   |      Tamara   |      Oven      |      2022      |
|  2  |      Nesterenko   |      Ivan     |      Pyba      |      2021      |
+-----+-----+-----+-----+-----+

```

Рисунок 16 – Пример решения индивидуального задания

## 2. Вопросы для защиты

### 1. Каково назначение модуля sqlite3?

Непосредственно модуль `sqlite3` – это API к СУБД SQLite. Своего рода адаптер, который переводит команды, написанные на Питоне, в команды, которые понимает SQLite. Как и наоборот, доставляет ответы от SQLite в python-программу.

## **2. Как выполняется соединение с базой данных SQLite3? Что такое курсор базы данных?**

Чтобы использовать SQLite3 в Python, прежде всего, вам нужно будет импортировать модуль `sqlite3`, а затем создать объект соединения, который соединит нас с базой данных и позволит нам выполнять операторы SQL. Объект соединения создается с помощью функции `connect()`.

Курсор SQLite3 – это метод объекта соединения. Для выполнения инструкций SQLite3 сначала устанавливается соединение, а затем создается объект курсора с использованием объекта соединения следующим образом:

## **3. Как подключиться к базе данных SQLite3, находящейся в оперативной памяти компьютера?**

При создании соединения с SQLite3 автоматически создается файл базы данных, если он еще не существует. Этот файл базы данных создается на диске, мы также можем создать базу данных в оперативной памяти с помощью функции `:memory:` with the connect. Такая база данных называется базой данных в памяти.

## **4. Как корректно завершить работу с базой данных SQLite3?**

После этого вне зависимости от того возникло или нет исключение по работе с базой данных, выполняются операторы блока `finally`, в котором соединение закрывается. Заккрытие соединения необязательно, но это хорошая практика программирования, поэтому вы освобождаете память от любых неиспользуемых ресурсов.

## **5. Как осуществляется вставка данных в таблицу базы данных SQLite3?**

Чтобы вставить данные в таблицу, используется оператор `INSERT INTO`.

## **6. Как осуществляется обновление данных таблицы базы данных SQLite3?**

Чтобы обновить данные в таблице, просто создайте соединение, затем создайте объект курсора с помощью соединения и, наконец, используйте оператор `UPDATE` в методе `execute()`.

## **7. Как осуществляется выборка данных из базы данных SQLite3?**

Оператор SELECT используется для выбора данных из определенной таблицы.

## **8. Каково назначение метода rowcount?**

SQLite3 rowcount используется для возврата количества строк, которые были затронуты или выбраны последним выполненным SQL-запросом.

## **9. Как получить список всех таблиц базы данных SQLite3?**

Чтобы перечислить все таблицы в базе данных SQLite3, вы должны запросить данные из таблицы sqlite\_master, а затем использовать fetchall() для получения результатов из инструкции SELECT.

## **10. Как выполнить проверку существования таблицы как при ее добавлении, так и при её удалении?**

Чтобы проверить, не существует ли таблица уже, мы используем IF NOT EXISTS с оператором CREATE TABLE.

## **11. Как выполнить массовую вставку данных в базу данных SQLite3?**

Метод executemany можно использовать для вставки нескольких строк одновременно.

## **12. Как осуществляется работа с датой и временем при работе с базами данных SQLite3**

В базе данных Python SQLite3 мы можем легко хранить дату или время, импортируя модуль datetime.

Ссылка на репозиторий: [https://github.com/tamaranesterenko/Python\\_LR\\_9-2](https://github.com/tamaranesterenko/Python_LR_9-2)