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 №2.21 по дисциплине:

Основы программной инженерии

Выполнила:
студент группы ПИЖ-б-о-20-1
Лазарева Дарья Олеговна

Проверил:
доцент кафедры инфокоммуникаций
Романкин Р.А.

Ставрополь, 2022 г.

ВЫПОЛНЕНИЕ:

1. Модуль sqlite3 языка Python

```
>>> 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']
```

2. Создание соединения

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

3. Курсор SQLite3

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

import sqlite3

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

cursor_obj = con.cursor()
```

4. Создание базы данных

```
# !/usr/bin/env python3
# -*- cosing: 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()
```

5. Создание таблиц

```python
# !/usr/bin/env python3
# -*- cosing: 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)
        """
    )
```

6. Вставка данных в таблицу

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

import sqlite3

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


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


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

7. Обновление данных в таблицах

```python
# !/usr/bin/env python3
# -*- cosing: 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)
```

8. Выборка данных из таблицы

```python
# !/usr/bin/env python3
# -*- cosing: 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)
```

9. Получение списка таблиц

```python
# !/usr/bin/env python3
# -*- cosing: 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)
```

10. Проверка существования таблицы

```python
# !/usr/bin/env python3
# -*- cosing: 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)
```

11. Массовая вставка

```python
# !/usr/bin/env python3
# -*- cosing: 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()
```

12. SQLite3 datetime

```python
# !/usr/bin/env python3
# -*- cosing: 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()
```

13. Выполнение примера 1

```python
# !/usr/bin/env python3
# -*- cosing: 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(
```

Индивидуальное задание: Для своего варианта лабораторной работы 2.17 необходимо реализовать хранение данных в базе данных SQLite3. Информация в базе данных должна храниться не менее чем в двух таблицах.

```
PS D:\УЧЕБА\ОПИ\2.21> python idz.py add --db="idz.db" -s="Lazareva" -n="Daria" -z="123456"
-y=2002
PS D:\УЧЕБА\ОПИ\2.21> python idz.py add --db="idz.db" -s="Smirnov" -n="Sergey" -z="6543321"
 -y=2001
PS D:\УЧЕБА\ОПИ\2.21> py idz.py display --db="idz.db"
+------+----------------------------------+----------------------+----------------+---------
------------+
|  №   |             Фамилия              |         Имя          | Номер телефона |    Дата рождения     |
+------+----------------------------------+----------------------+----------------+----------------------+
|  1   |             Lazareva             |        Daria         |     123456     |         2002         |
|  2   |             Smirnov              |        Sergey        |     6543321    |         2001         |
+------+----------------------------------+----------------------+----------------+----------------------+
PS D:\УЧЕБА\ОПИ\2.21> py idz.py display --db="idz.db"
+------+----------------------------------+----------------------+----------------+--------------------+
|  №   |             Фамилия              |         Имя          | Номер телефона |   Дата рождения    |
+------+----------------------------------+----------------------+----------------+--------------------+
|  1   |             Lazareva             |        Daria         |     123456     |        2002        |
|  2   |             Smirnov              |        Sergey        |     6543321    |        2001        |
+------+----------------------------------+----------------------+----------------+--------------------+
```

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. Закрытие 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.