

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

Отчет по лабораторной работе № 2.21

**«Взаимодействие с базами данных SQLite3 с помощью
языка программирования Python»**

по дисциплине «Основы программной инженерии»

Выполнила:
Образцова Мария Дмитриевна,
2 курс, группа ПИЖ-б-о-21-1,
Проверил:
Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2023 г.


Методика и порядок выполнения работы

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.

Create a new repository

A repository contains all project files, including the revision history. Already have a repository? [Import a repository.](#)

Owner *

 obrMaria ▾

Repository name *

/ OPI_2.21

✓ OPI_2.21 is available.

Great repository names are short and memorable. Need inspiration? [How to choose a name](#)

Description (optional)



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Initialize this repository with:



Add a README file

This is where you can write a long description for your project. [Learn more about README files](#)

Add .gitignore


.gitignore template: Python ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files](#)

Choose a license

License: MIT License ▾

A license tells others what they can and can't do with your code. [Learn more about licenses](#)

This will set  main as the default branch. Change the default name in your settings



You are creating a public repository in your personal account.

Рисунок 1 – создание репозитория

```
M@DESKTOP-UVM9NOL MINGW64 ~/desktop (master)
$ git clone https://github.com/obrMaria/OPI_2.21.git
Cloning into 'OPI_2.21'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
```

Рисунок 2 – клонирование репозитория

```
M@DESKTOP-UVM9NOL MINGW64 ~/desktop/OPI_2.21 (main)
$ git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/M/desktop/OPI_2.21/.git/hooks]
```

Рисунок 3 – создание ветки develop

Проработайте примеры лабораторной работы. Создайте для них отдельные модули языка Python. Зафиксируйте изменения в репозитории.

```
>>> import sqlite3
>>> dir(sqlite3)
['Binary', 'Blob', 'Connection', 'Cursor', 'DataError', 'DatabaseError',
 'Date', 'DateFromTicks', 'Error', 'IntegrityError', 'InterfaceError',
 'InternalError', 'NotSupportedError', 'OperationalError', 'PARSE_COLNAMES',
 'PARSE_DECLTYPES', 'PrepareProtocol', 'ProgrammingError', 'Row', 'SQL
ITE_ABORT', 'SQLITE_ABORT_ROLLBACK', 'SQLITE_ALTER_TABLE', 'SQLITE_ANALY
ZE', 'SQLITE_ATTACH', 'SQLITE_AUTH', 'SQLITE_AUTH_USER', 'SQLITE_BUSY',
 'SQLITE_BUSY_RECOVERY', 'SQLITE_BUSY_SNAPSHOT', 'SQLITE_BUSY_TIMEOUT',
 'SQLITE_CANTOPEN', 'SQLITE_CANTOPEN_CONVPATH', 'SQLITE_CANTOPEN_DIRTYWAL',
 'SQLITE_CANTOPEN_FULLPATH', 'SQLITE_CANTOPEN_ISDIR', 'SQLITE_CANTOPEN_
NOTEMPDIR', 'SQLITE_CANTOPEN_SYMLINK', 'SQLITE_CONSTRAINT', 'SQLITE_CONS
TRAIT_CHECK', 'SQLITE_CONSTRAINT_COMMITHOOK', 'SQLITE_CONSTRAINT_FOREIG
NKEY', 'SQLITE_CONSTRAINT_FUNCTION', 'SQLITE_CONSTRAINT_NOTNULL', 'SQLIT
E_CONSTRAINT_PINNED', 'SQLITE_CONSTRAINT_PRIMARYKEY', 'SQLITE_CONSTRAINT
_ROWID', 'SQLITE_CONSTRAINT_TRIGGER', 'SQLITE_CONSTRAINT_UNIQUE', 'SQLIT
E_CONSTRAINT_VTAB', 'SQLITE_CORRUPT', 'SQLITE_CORRUPT_INDEX', 'SQLITE_CO
RRUPT_SEQUENCE', 'SQLITE_CORRUPT_VTAB', 'SQLITE_CREATE_INDEX', 'SQLITE_C
REATE_TABLE', 'SQLITE_CREATE_TEMP_INDEX', 'SQLITE_CREATE_TEMP_TABLE', 'S
QLITE_CREATE_TEMP_TRIGGER', 'SQLITE_CREATE_TEMP_VIEW', 'SQLITE_CREATE_TR
IGGER', 'SQLITE_CREATE_VIEW', 'SQLITE_CREATE_VTABLE', 'SQLITE_DELETE',
 'SQLITE_DENY', 'SQLITE_DETACH', 'SQLITE_DONE', 'SQLITE_DROP_INDEX', 'SQLI
TE_DROP_TABLE', 'SQLITE_DROP_TEMP_INDEX', 'SQLITE_DROP_TEMP_TABLE', 'SQL
ITE_DROP_TEMP_TRIGGER', 'SQLITE_DROP_TEMP_VIEW', 'SQLITE_DROP_TRIGGER',
```

Рисунок 4 – Пример вывода классов

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

Рисунок 5 – Пример вызова функции connect()

```
>>> con = sqlite3.connect('mydatabase.db')
>>> cursor_obj = con.cursor()
```

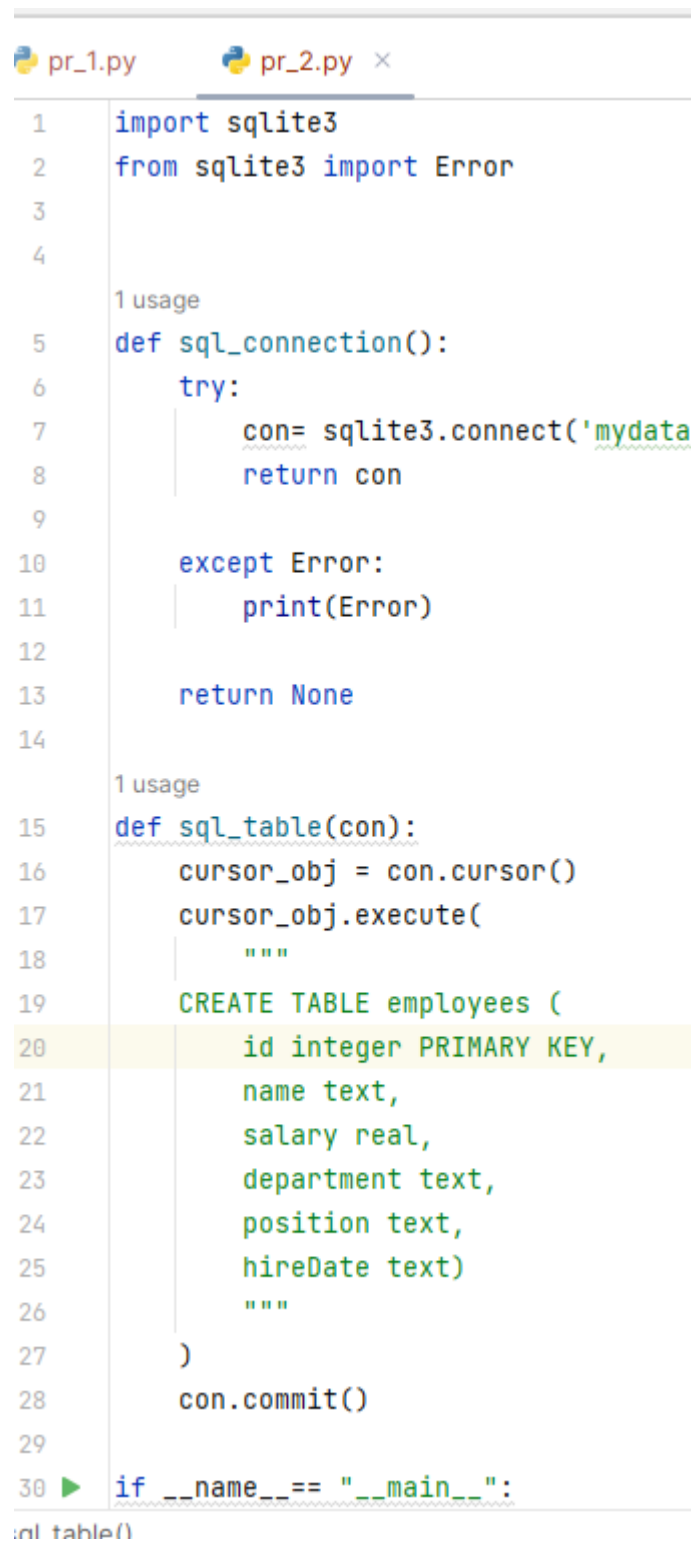
Рисунок 6 – Пример использования курсора

```
pr_1.py x
1  import sqlite3
2  from sqlite3 import Error
3
4
5  def sql_connection():
6      try:
7          con = sqlite3.connect(':memory:')
8          print("Connection is established: Database is created in memory")
9      except Error:
10         print(Error)
11     finally:
12         con.close()
13
14
15  if __name__ == "__main__":
16     sql_connection()
17
```

```
pr_1 (1) x
C:\Users\M\Desktop\OPI_2.21\PY\venv\Scripts\python.exe C:\Users\M\Desktop\OPI_2.21\PY\pr_1.py
Connection is established: Database is created in memory

Process finished with exit code 0
```

Рисунок 7 – Пример создания базы данных



The image shows a code editor with two tabs: 'pr_1.py' and 'pr_2.py'. The 'pr_2.py' tab is active, displaying Python code for connecting to a SQLite database and creating a table. The code is as follows:

```
1 import sqlite3
2 from sqlite3 import Error
3
4
5 def sql_connection():
6     try:
7         con= sqlite3.connect('mydata')
8         return con
9
10    except Error:
11        print(Error)
12
13    return None
14
15 def sql_table(con):
16     cursor_obj = con.cursor()
17     cursor_obj.execute(
18         """
19     CREATE TABLE employees (
20         id integer PRIMARY KEY,
21         name text,
22         salary real,
23         department text,
24         position text,
25         hireDate text)
26     """
27     )
28     con.commit()
29
30 if __name__ == "__main__":
31     sql_table()
```

Рисунок 8 – Пример создания таблиц

Структура БД		
<div> <div>Создать таблицу</div> <div>Создать индекс</div> <div>Модифицировать Таблицу</div> </div>		
Имя	Тип	Схема
Таблицы (1)		
employees		CREATE TABLE employees
id	integer	"id" integer
name	text	"name" text
salary	real	"salary" real
department	text	"department" text
position	text	"position" text
hireDate	text	"hireDate" text
Индексы (0)		
Представления (0)		
Триггеры (0)		

Результат 9 – создания базы данных

```

py      pr_2.py      pr_3.py x
import sqlite3

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

1 usage
def sql_insert(con, entities):
    cursor_obj = con.cursor()
    cursor_obj.execute(
        """
        INSERT INTO employees(id, name, salary, department
        VALUES(?, ?, ?, ?, ?, ?)
        """,
        entities
    )
    con.commit()
entities = (2, 'Andrew', 800, 'IT', 'Tech', '2018-02-06')
sql_insert(con,entities)

```

Рисунок 10 – Пример вставки данных в таблицу

Структура БД Данные Прагмы SQL						
Таблица: employees						
	id	name	salary	department	position	hireDate
	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр
1	2	Andrew	800.0	IT	Tech	2018-02-06

Рисунок 11– результат выполнения программы

```

1.py    pr_2.py    pr_3.py    pr_4.py x
import sqlite3

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

1 usage
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 12 – Пример обновления данных в таблицах

Таблица: employees						
	id	name	salary	department	position	hireDate
	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр
1	2	Rogers	800.0	IT	Tech	2018-02-06

Рисунок 13– результат выполнения программы


```

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 14 – Пример выборки данных из таблицы

```

import sqlite3

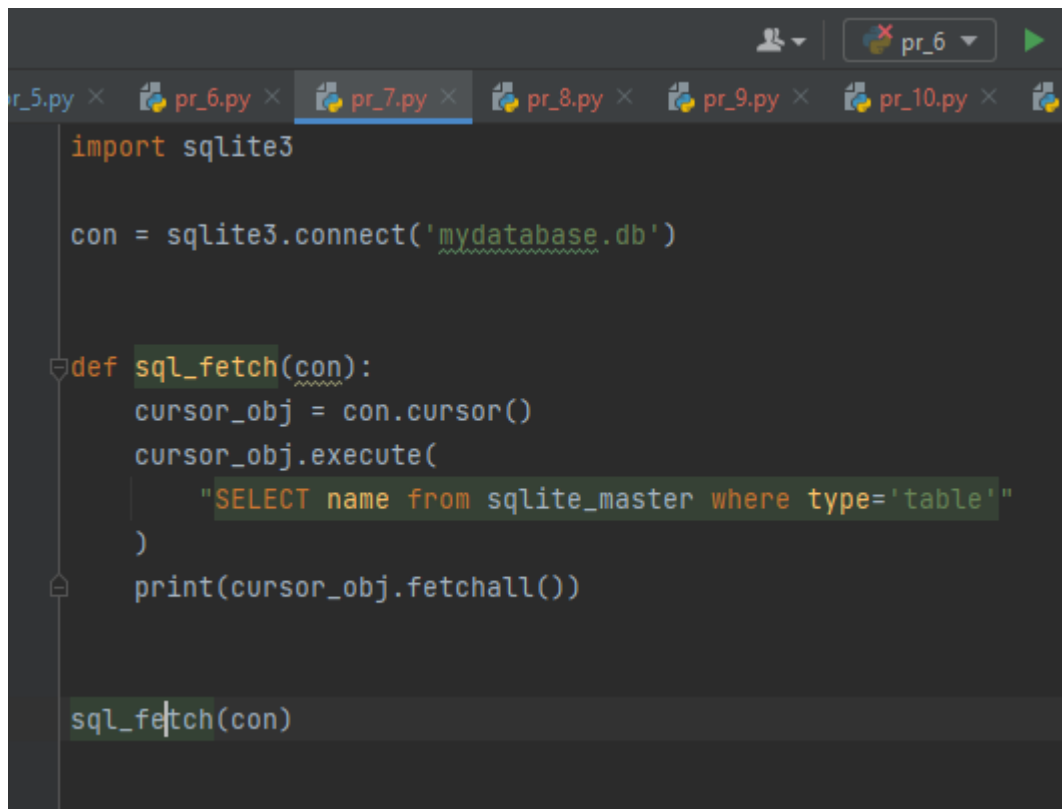
con = sqlite3.connect('mydatabase.db')

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 15 – Пример получения списка таблиц



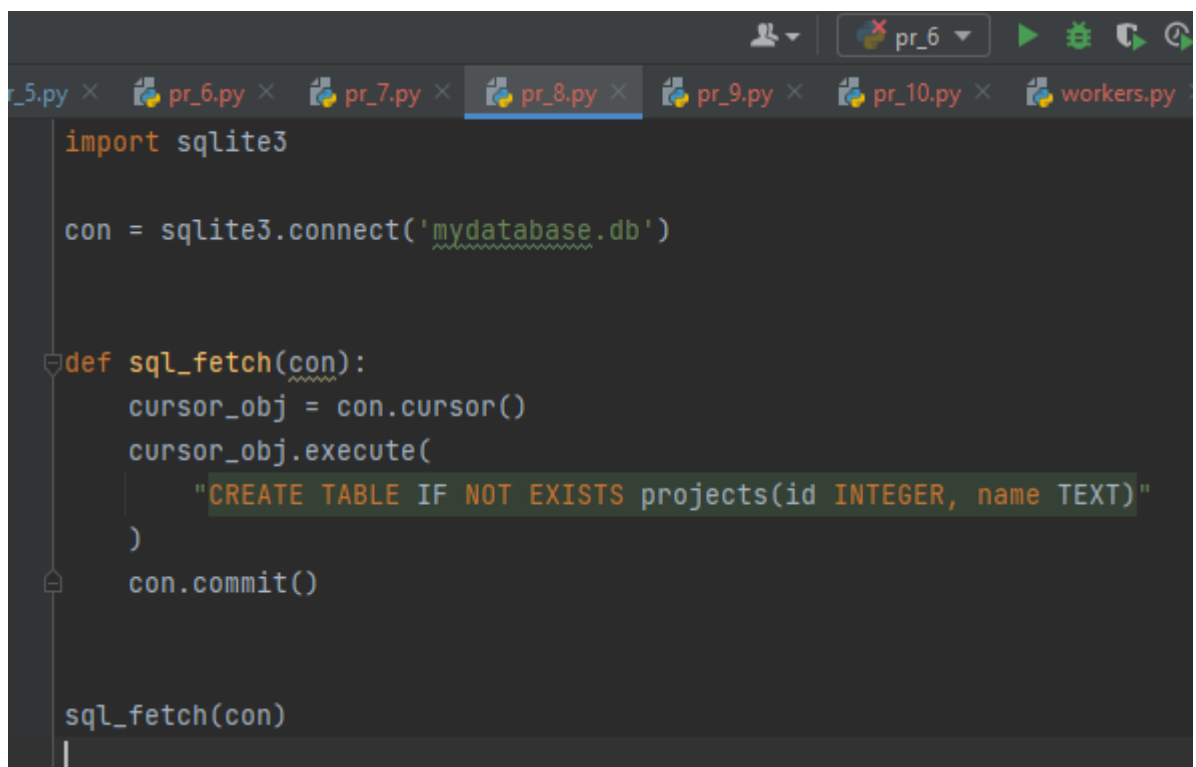
```
import sqlite3

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 16 – Пример проверки существования таблицы



```
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 17 – Пример проверки существования таблицы

```
pr_5.py × pr_6.py × pr_7.py × pr_8.py × pr_9.py × pr_10.py × workers.p
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 18 – Пример массовой вставки

```
pr_5.py × pr_6.py × pr_7.py × pr_8.py × pr_9.py × pr_10.py × workers.py ×
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 19 – Пример использования datetime

```
workers.py x
1  ▶  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  import ...
6
7
8
9
10
11  def display_workers(staff: t.List[t.Dict[str, t.Any]]) -> None:
12      """
13      Отобразить список работников.
14      """
15      # Проверить, что список работников не пуст.
16      if staff:
17          # Заголовок таблицы.
18          line = '+-{}-+-{}-+-{}-+-{}-+'.format(
19              '-' * 4,
20              '-' * 30,
21              '-' * 20,
22              '-' * 8
23          )
24          print(line)
25          print(
26              '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
27                  "No",
28                  "Ф.И.О.",
29                  "Должность",
30                  "Год"
31              )
32          )
33          print(line)
34          # Вывести данные о всех сотрудниках.
35          for idx, worker in enumerate(staff, 1):
36              print(
37                  '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
38                      idx,
39                      worker.get('name', ''),
40                      worker.get('post', ''),
41                      worker.get('year', 0)
42                  )
43              )
44          print(line)
45      else:
46          print("Список работников пуст.")
47
48
49  def create_db(database_path: Path) -> None:
50      """
51      Создать базу данных.
52      """
53      conn = sqlite3.connect(database_path)
54      cursor = conn.cursor()
55      # Создать таблицу с информацией о должностях.
56      cursor.execute(
57          """
58          CREATE TABLE IF NOT EXISTS posts (
59              ...
60          )
61      """
62      )
63      create_db()
```

Рисунок 20 – Пример

```
C:\Users\M\Desktop\OPI_2.21\PY>python workers.py add -n="sidorov" -p="ohrannik"
-y="2001"
C:\Users\M\Desktop\OPI_2.21\PY>
```

Рисунок 21 – запуск программы

Имя	Дата измене
workers	05.05.2023 19:
Ссылки	08.05.2022 23:
Сохраненные игры	08.05.2022 23:
Рабочий стол	05.05.2023 14:
Поиски	08.05.2022 23:
Объемные объекты	08.05.2022 23:

Рисунок 22 – результат выполнения примера

Имя	Тип	Схема
Таблицы (3)		
posts		CREATE TABLE posts (post_id INTEGER PRIMAR
post_id	INTEGER	"post_id" INTEGER PRIMARY KEY AUTOINCREM
post_title	TEXT	"post_title" TEXT NOT NULL
sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)
name	TEXT	"name" TEXT
seq	TEXT	"seq" TEXT
workers		CREATE TABLE workers (worker_id INTEGER PR
worker_id	INTEGER	"worker_id" INTEGER PRIMARY KEY AUTOINCR
worker_name	TEXT	"worker_name" TEXT NOT NULL
post_id	INTEGER	"post_id" INTEGER NOT NULL
worker_year	INTEGER	"worker_year" INTEGER NOT NULL
Индексы (0)		
Представления (0)		
Триггеры (0)		

Рисунок 23 – полученная база данных

Индивидуальные задания

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.

```
ind.py × ind.py ×  
  
        action="store",  
        help="The student's group"  
    )  
  
    add.add_argument(  
        "-m",  
        "--marks",  
        action="store",  
        required=True,  
        help="The student's marks"  
    )  
  
    # Создать субпарсер для отображения всех студентов.  
    _ = subparsers.add_parser(  
        "display",  
        parents=[file_parser],  
        help="Display all students"  
    )  
  
    # Создать субпарсер для поиска студентов.  
    find = subparsers.add_parser(  
        "find",  
        parents=[file_parser],  
        help="find the students"  
    )  
  
    # Выполнить разбор аргументов командной строки.  
    args = parser.parse_args(command_line)  
    # Получить путь к файлу базы данных.  
    db_path = Path(args.db)  
    create_db(db_path)  
    # Добавить студента.  
    if args.command == "add":  
        add_student(db_path, args.name, args.group, args.marks)  
    # Отобразить всех студентов.  
    display_students(db_path)  
  
main()
```

Рисунок 24 – индивидуальное задание

```
C:\Users\M\Desktop\OPI_2.21\PY>python ind.py add -n="masha"
-g="pij.21" -m="1 2 3 4 5"
```

Рисунок 25 – запуск программы

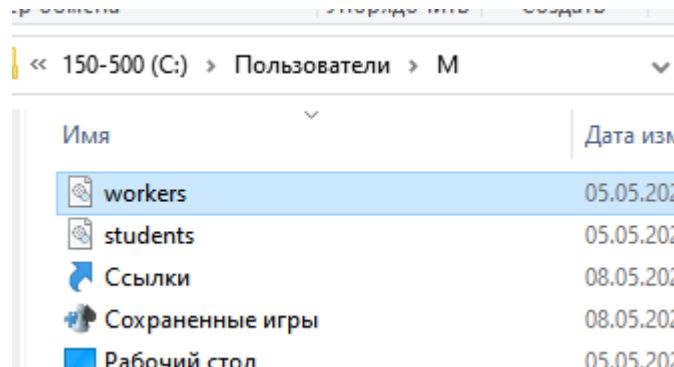


Рисунок 26 – результат выполнения

Имя	Тип	Схема
▼ Таблицы (3)		
▼ groupss		CREATE TABLE groupss (group_id INTEGER PRI
group_id	INTEGER	"group_id" INTEGER PRIMARY KEY AUTOINCRE
group_num	TEXT	"group_num" TEXT NOT NULL
▼ sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)
name	TEXT	"name" TEXT
seq	TEXT	"seq" TEXT
▼ students		CREATE TABLE students (student_id INTEGER P
student_id	INTEGER	"student_id" INTEGER PRIMARY KEY AUTOINCR
student_name	TEXT	"student_name" TEXT NOT NULL
group_id	INTEGER	"group_id" INTEGER NOT NULL
student_marks	INTEGER	"student_marks" INTEGER NOT NULL

Рисунок 27 – база данных

Задание повышенной сложности

Самостоятельно изучите работу с пакетом python-psycopg2 для работы с базами данных PostgreSQL. Для своего варианта лабораторной работы 2.17 необходимо реализовать возможность хранения данных в базе данных СУБД PostgreSQL. Информация в базе данных должна храниться не менее чем в двух таблицах.

Вопросы для защиты работы

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 работес 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.