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

Отчет о лабораторной работе №9 по дисциплине основы программной инженерии

Выполнил:

Выходцев Егор Дмитриевич, 2 курс, группа ПИЖ-б-о-20-1,

Проверил:

Доцент кафедры инфокоммуникаций, Воронкин Р.А.

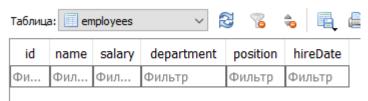
1. Примеры из методических указаний

```
C:\Users\Evil\PycharmProjects\LR#9\venv\Scripts\python.ex
Connection is established: Database is created in memory

Process finished with exit code 0
```

```
∣import sqlite3
      Ģfrom sqlite3 import Error

def sql_connection():
           try:
               con = sqlite3.connect('mydatabase.db')
               return con
           except Error:
               print(Error)
      def sql_table(con):
           cursor_obj = con.cursor()
           cursor_obj.execute(
              hireDate text)
           con.commit()
34 ▶ ||if __name__ == "__main__":
           con = sql_connection()
           sql_table(con)
```



```
import sqlite3
con = sqlite3.connect('mydatabase.db')
def sql_insert(con, entities):
    cursor_obj = con.cursor()
    cursor_obj.execute(
    cursor_obj.execute(
         entities
    con.commit()
if __name__ == "__main__":
    sql_insert(con, entities)
```

Ta6	блица:	employ	ees	~ ≈	% ♣	
	id	name	salary	department	position	hireDate
	Фи	Фильтр	Фил	Фильтр	Фильтр	Фильтр
1	1	John	700.0	HR	Manager	2017 - 01 - 04
2	2	Andrew	800.0	Π	Tech	2018-02-06

	id	name	salary	department	position	hireDate
	Фи	Фил	Фил	Фильтр	Фильтр	Фильтр
1	1	John	700.0	HR	Manager	2017 - 01 - 04
2	2	Rogers	800.0	Π	Tech	2018-02-06

```
#!/usr/bin/env python3
##:/usr/bin/env python3
##
```

```
C:\Users\Evil\PycharmProjects\LR#9\venv\Scripts\pythor
(1, 'John', 700.0, 'HR', 'Manager', '2017 - 01 - 04')
(2, 'Rogers', 800.0, 'IT', 'Tech', '2018-02-06')

Process finished with exit code 0
```

```
C:\Users\Evil\PycharmProjects\LR#9\v
(2, 'Rogers')
Process finished with exit code 0
```

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

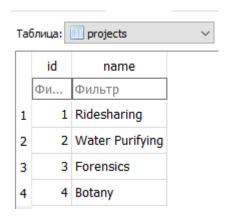
import sqlite3

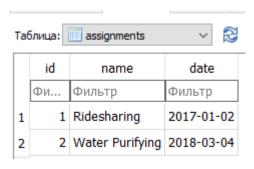
con = sqlite3.connect('mydatabase.db')

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

if __name__ == "__main__":
    sql_fetch(con)
```

```
C:\Users\Evil\PycharmProjects\LR#9
[('employees',)]
Process finished with exit code 0
```





```
🛵 ex1.py
1 ▶ \phi#!/usr/bin/env python3
     import sqlite3
       import typing as t
      Ģfrom pathlib import Path
      def display_workers(staff: t.List[t.Dict[str, t.Any]]) -> None:
           if staff:
               line = '+-{}-+-{}-+-{}-+'.format(
               print(line)
               print(
                   '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
               print(line)
               # Вывести данные о всех сотрудниках.
               for idx, worker in enumerate(staff, 1):
                   print(
                       '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
                           idx,
                           worker.get('name', ''),
        display workers() > if staff
```

```
worker.get('post', ''),
                     worker.get('year', 0)
             print(line)
    else:
|def create_db(database_path: Path) -> None:
    conn = sqlite3.connect(database_path)
    cursor = conn.cursor()
    # Создать таблицу с информацией о должностях.
     cursor.execute(
    cursor.execute(
    CREATE TABLE IF NOT EXISTS workers (
    conn.close()
 display_workers() → if staff
```

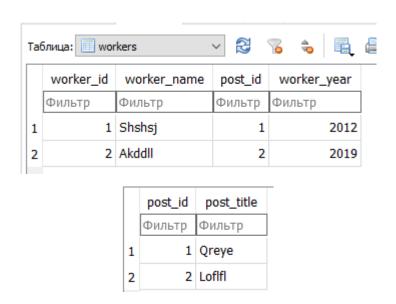
```
def add_worker(
        database_path: Path,
        name: str,
        post: str,
        year: int
   conn = sqlite3.connect(database_path)
   cursor = conn.cursor()
   cursor.execute(
        (post,)
   row = cursor.fetchone()
   if row is None:
        cursor.execute(
            (post,)
        post_id = cursor.lastrowid
   else:
        post_id = row[0]
    cursor.execute(
        (name, post_id, year)
    conn.commit()
create_db()
```

```
conn.close()
def select_all(database_path: Path) -> t.List[t.Dict[str, t.Any]]:
    conn = sqlite3.connect(database_path)
    cursor = conn.cursor()
    cursor.execute(
    rows = cursor.fetchall()
    conn.close()
             "name": row[0],
             "post": row[1],
             "year": row[2],
         for row in rows
def select_by_period(
         database_path: Path, period: int
) -> t.List[t.Dict[str, t.Any]]:
    conn = sqlite3.connect(database_path)
    cursor = conn.cursor()
    cursor.execute(
 add_worker()
```

```
FROM workers
        (period,)
    rows = cursor.fetchall()
    conn.close()
    return [
            "name": row[0],
            "post": row[1],
            "year": row[2],
        for row in rows
def main(command_line=None):
    file_parser = argparse.ArgumentParser(add_help=False)
    file_parser.add_argument(
        default=str(Path.home() / "workers.db"),
    parser = argparse.ArgumentParser("workers")
    parser.add_argument(
    subparsers = parser.add_subparsers(dest="command")
    add = subparsers.add_parser(
```

```
parents=[file_parser],
   help="Add a new worker"
add.add_argument(
add.add_argument(
add.add_argument(
_ = subparsers.add_parser(
   parents=[file_parser],
select = subparsers.add_parser(
   parents=[file_parser],
select.add_argument(
```

```
required=True,
                args = parser.parse_args(command_line)
                db_path = Path(args.db)
                create_db(db_path)
                if args.command == "add":
                      add_worker(db_path, args.name, args.post, args.year)
                elif args.command == "display":
                      display_workers(select_all(db_path))
                elif args.command == "select":
                      display_workers(select_by_period(db_path, args.period))
           if __name__ == "__main__":
                main()
PS C:\Users\Evil\PycharmProjects\LR#9\examples> python ex1.py add
PS C:\Users\Evil\PycharmProjects\LR#9\examples> python ex1.py add --db="workers.db" --name="Akddll" --post="Loflft" --year="2019
                               | Qreye
PS C:\Users\Evil\PycharmProjects\LR#9\examples> python ex1.py select --db="workers.db" --period=12
Список работников пуст.
PS C:\Users\Evil\PycharmProjects\LR#9\examples> <mark>python ex1.py select --db="workers.db"</mark>
                               | Qreye
PS C:\Users\Evil\PycharmProjects\LR#9\examples> [
```



2. Индивидуальное задание (рис. 1-12).

```
🛵 idz.py
      ⇒import argparse
       import sqlite3
       import typing as t
       ậfrom pathlib import Path
      | def display_flights(flights: t.List[t.Dict[str, t.Any]]) -> None:
           if flights:
               # Заголовок таблицы.
                line = '+-{}-+-{}-+-{}-+'.format(
                print(line)
                print(
                    '| {:^4} | {:^30} | {:^20} | {:^15} |'.format(
                print(line)
                # Вывести данные о всех сотрудниках.
                for idx, flight in enumerate(flights, 1):
                    print(
                        '| {:>4} | {:<30} | {:<20} | {:<15} |'.format(
                            idx,
                            flight.get('flight_destination', ''),
        select_all() > "flight_destination"
```

Рисунок 1 – Код программы

```
flight.get('flight_number', ''),
                     flight.get('airplane_type', 0)
             print(line)
    else:
        print("Список рейсов пуст.")
def create_db(database_path: Path) -> None:
    conn = sqlite3.connect(database_path)
    cursor = conn.cursor()
    cursor.execute(
    cursor.execute(
         FOREIGN KEY(num_id) REFERENCES flight_numbers(num_id)
    conn.close()
 select_all() > "flight_destination"
```

Рисунок 2 – Код программы, продолжение

```
def add_flight(
        database_path: Path,
        flight_destination: str,
        flight_number: str,
        airplane_type: int
   conn = sqlite3.connect(database_path)
   cursor = conn.cursor()
   cursor.execute(
       (flight_number,)
   row = cursor.fetchone()
   if row is None:
        cursor.execute(
            (flight_number,)
        num_id = cursor.lastrowid
        num_id = row[0]
    cursor.execute(
        (flight_destination, num_id, airplane_type)
   conn.commit()
   conn.close()
create db()
```

Рисунок 3 – Код программы, продолжение

```
def select_all(database_path: Path) -> t.List[t.Dict[str, t.Any]]:

     conn = sqlite3.connect(database_path)
     cursor = conn.cursor()
     cursor.execute(
     rows = cursor.fetchall()
     conn.close()
             "flight_destination": row[0],
             "flight_number": row[1],
             "airplane_type": row[2],
         for row in rows
 def select_flights(
         database_path: Path, air_type: str
; -> t.List[t.Dict[str, t.Any]]:
     conn = sqlite3.connect(database_path)
     cursor = conn.cursor()
     cursor.execute(
 add_flight() > if row is None
```

Рисунок 4 – Код программы, продолжение

```
(air_type,)
    rows = cursor.fetchall()
    conn.close()
    return [
            "flight_destination": row[0],
            "flight_number": row[1],
            "airplane_type": row[2],
        for row in rows
def main(command_line=None):
    file_parser = argparse.ArgumentParser(add_help=False)
    file_parser.add_argument(
        default=str(Path.home() / "workers.db"),
    parser = argparse.ArgumentParser("workers")
    parser.add_argument(
    subparsers = parser.add_subparsers(dest="command")
    add = subparsers.add_parser(
        parents=[file_parser],
    add.add_argument(
select_flights()
```

Рисунок 5 – Код программы, продолжение

```
add.add_argument(
            add.add_argument(
216
            _ = subparsers.add_parser(
                parents=[file_parser],
            select = subparsers.add_parser(
                parents=[file_parser],
            select.add_argument(
        main()
```

Рисунок 6 – Код программы, продолжение

```
# 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 работника.

if args.command == "add":

add_flight(db_path, args.dest, args.flight_num, args.type)

# Отобразить всех работников.

elif args.command == "display":

display_flights(select_all(db_path))

# Выбрать требуемых рааботников.

elif args.command == "select":

display_flights(select_flights(db_path, args.type))

pass

if __name__ == "__main__":

main()
```

Рисунок 7 – Код программы, продолжение

```
PS C:\Users\Evil\PycharmProjects\LR#9> python idz.py add --db="flights.db" --dest="Miami" --flight_num="M1245" --type="Passenger"
PS C:\Users\Evil\PycharmProjects\LR#9> python idz.py add --db="flights.db" --dest="Egypt" --flight_num="EG436" --type="Sanitary"
```

Рисунок 8 – Добавление рейсов в базу данных

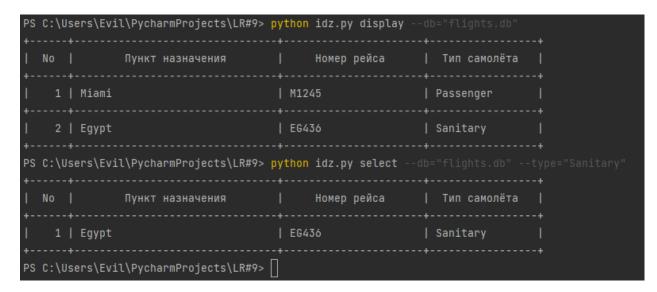


Рисунок 9 – Результат выполнения программы

Им	Я	Тип	Схема
~	Таблицы (3)		
	> flight_numbers		CREATE TABLE flight_numbers (num_id INTEGER PRIMARY KEY AUTOINCREMENT, num_title TEXT NOT NULL)
	> III flights		CREATE TABLE flights (flight_id INTEGER PRIMARY KEY AUTOINCREMENT, flight_destination TEXT NOT NULL, num_id INTEGER NOT NULL, airplane_type TEXT
	> sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)

Рисунок 10 – Структура созданной таблицы

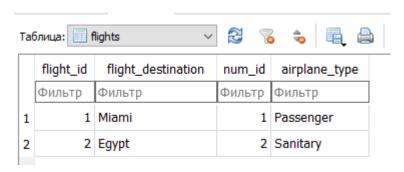


Рисунок 11 – Содержимое таблицы «flights»

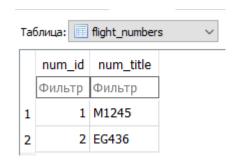


Рисунок 12 – Содержимое таблицы «flight_numbers»

- 4. Ответы на вопросы
- 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 руthоп-программу. Модуль sqlite3 содержит много классов, функций и констант. Их перечень можно посмотреть с помощью функции dir().

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(). Вызов функции connect() приводит к созданию объекта-экземпляра от класса Connection. Этот объект обеспечивает связь с файлом базы данных, представляет конкретную БД в программе.

Для взаимодействия с базой данных SQLite3 в Python необходимо создать объект cursor. Вы можете создать его с помощью метода cursor().

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?

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? con.close() для высвобождения памяти.
- 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 (). Предположим, что мы хотим обновить имя сотрудника, чей идентификатор равен 2. Для обновления мы будем использовать оператор UPDATE, а для сотрудника, чей

идентификатор равен 2. Мы будем использовать предложение WHERE в качестве условия для выбора этого сотрудника.

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

7. Как осуществляется выборка данных из базы данных SQLite3?

Оператор SELECT используется для выбора данных из определенной таблицы. Если вы хотите выбрать все столбцы данных из таблицы, вы можете использовать звездочку (*).

8. Каково назначение метода rowcount?

SQLite3 rowcount используется для возврата количества строк, которые были затронуты или выбраны последним выполненным SQL-запросом. Когда мы используем rowcount с оператором SELECT, будет возвращено значение -1, поскольку количество выбранных строк неизвестно до тех пор, пока все они не будут извлечены.

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 . sqlite_master — это главная таблица в SQLite3, которая хранит все таблицы.

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

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

CREATE TABLE IF NOT EXISTS table_name (column1, column2, ..., columnN)

DROP TABLE IF EXISTS table_name

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

Метод executemany можно использовать для вставки нескольких строк одновременно.

```
cursor_obj.execute(
      "CREATE TABLE IF NOT EXISTS projects(id INTEGER, name TEXT)"
     )
     data = [
     (1, "Ridesharing"),
     (2, "Water Purifying"),
     (3, "Forensics"),
     (4, "Botany")
     1
     cursor_obj.executemany("INSERT INTO projects VALUES (?, ?)", data)
     con.commit()
      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.
     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))
     1
     cursor_obj.executemany("INSERT INTO assignments VALUES(?, ?, ?)",
data)
     con.commit()
```