

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №7
дисциплины «анализ данных»

Выполнил:
Середа Кирилл Витальевич
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника», очная
форма обучения

(подпись)

Руководитель практики:
Воронкин Роман Александрович

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

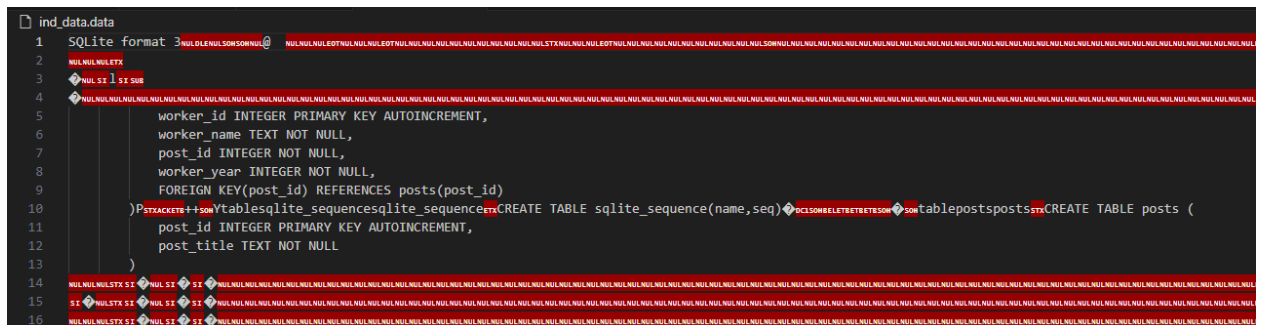
Ставрополь, 2024 г.

Тема: Взаимодействие с базами данных SQLite3 с помощью языка программирования Python

Цель работы: исследовать взаимодействие с базами данных SQLite3 с помощью языка программирования Python

Ход работы

Выполнил пример лабораторной работы в файле primer.py, в нем реализована возможность хранения данных в базе данных SQLite3. На рис. 1 расположена результат работы этого приложения.

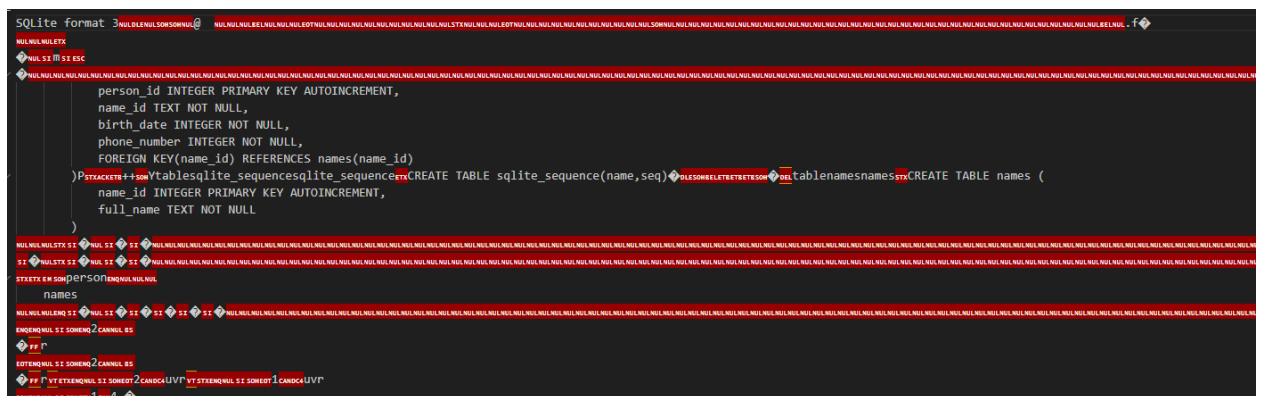


```
1 sqlite format 3
2
3
4
5 worker_id INTEGER PRIMARY KEY AUTOINCREMENT,
6 worker_name TEXT NOT NULL,
7 post_id INTEGER NOT NULL,
8 worker_year INTEGER NOT NULL,
9 FOREIGN KEY(post_id) REFERENCES posts(post_id)
10 )
11 CREATE TABLE sqlite_sequence(name,seq)
12 CREATE TABLE posts (
13   post_id INTEGER PRIMARY KEY AUTOINCREMENT,
14   post_title TEXT NOT NULL
15 )
16
```

Рисунок 1 – Результат работы primer.py

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. На рисунке 2 отображен результат выполнения команды primer.py add, на рисунке 3 отображен результат выполнения команды primer.py display, на рисунке 4 отображен результат выполнения команды primer.py select.



```
1 sqlite format 3
2
3
4
5 person_id INTEGER PRIMARY KEY AUTOINCREMENT,
6 name_id TEXT NOT NULL,
7 birth_date INTEGER NOT NULL,
8 phone_number INTEGER NOT NULL,
9 FOREIGN KEY(name_id) REFERENCES names(name_id)
10 )
11 CREATE TABLE sqlite_sequence(name,seq)
12 CREATE TABLE names (
13   name_id INTEGER PRIMARY KEY AUTOINCREMENT,
14   full_name TEXT NOT NULL
15 )
16
```

Рисунок 2 – Результат выполнения команды ind.py add

№	Имя	Дата рождения	Номер телефона
1	Оганесов	24	3432423
2	Оганесов	24	343242354
3	Оганedfgsцов	24	343242354
4	Оганedfgsцов	24	34543242354
5	Оганedfgsцов	24	34543242354

Рисунок 3 – Результат выполнения команды ind.py display

№	Имя	Дата рождения	Номер телефона
1	Оганесов	24	343242354
2	Оганedfgsцов	24	343242354

Рисунок 4 – Результат выполнения команды ind.py select

Задание повышенного уровня:

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.

	person_id [PK] integer	name_id integer	birth_date integer	phone_number integer
1	2	1	24	3243243
2	3	1	24	324324356
3	4	1	2564	32432435
4	6	2	2564	32432435

Рисунок 5 – Результат выполнения команды ind.py add таблица person

	name_id [PK] integer	full_name text
1	1	ZLJKD
2	2	ZLJKdfgD

Рисунок 6 – Результат выполнения команды ind.py add таблица name

№	Имя	Дата рождения	Номер телефона
1	ZLJKD	24	3243243
2	ZLJKD	24	324324356
3	ZLJKD	2564	32432435
4	ZLJKdfgD	2564	32432435

Рисунок 7 – Результат выполнения команды `ind.py display`

№	Имя	Дата рождения	Номер телефона
1	ZLJKD	2564	32432435
2	ZLJKdfgD	2564	32432435

Рисунок 8 – Результат выполнения команды `ind.py select`

Вывод: в ходе выполнения практической работы было исследовано взаимодействие с базами данных SQLite3 с помощью языка программирования Python.

Контрольные вопросы:

1. Назначение модуля `sqlite3`

Модуль `sqlite3` предназначен для взаимодействия с базой данных SQLite3 из Python. Он позволяет создавать, управлять и взаимодействовать с базами данных SQLite3 из Python-приложений. Модуль `sqlite3` обеспечивает доступ к функциям SQLite3 через простой и удобный интерфейс, что делает его популярным инструментом для работы с базами данных в Python.

2. Выполнение соединения с базой данных SQLite3 и курсор базы данных

Для выполнения соединения с базой данных SQLite3 в Python с использованием модуля `sqlite3`, необходимо выполнить следующие шаги:

1. Импортировать модуль `sqlite3`: `import sqlite3`.
- 1) Установить соединение с базой данных: `connection = sqlite3.connect('database.db')`.

2) Создать курсор базы данных: `cursor = connection.cursor()`. Курсор представляет собой механизм, который позволяет выполнять операции с базой данных, такие как выполнение SQL-запросов, получение результатов и управление данными.

3. Подключение к базе данных SQLite3, находящейся в оперативной памяти компьютера

Для подключения к базе данных SQLite3, находящейся в оперативной памяти компьютера, можно использовать специальное ключевое слово `":memory:"`. Пример подключения к такой базе данных:

```
connection = sqlite3.connect(':memory:')
```

4. Корректное завершение работы с базой данных SQLite3

Для корректного завершения работы с базой данных SQLite3 в Python, необходимо закрыть курсор и соединение с базой данных:

```
cursor.close()
```

```
connection.close()
```

5. Вставка данных в таблицу базы данных SQLite3

Для вставки данных в таблицу базы данных SQLite3 в Python с использованием модуля `sqlite3`, можно выполнить SQL-запрос с использованием метода `execute()`:

```
cursor.execute("INSERT INTO table_name (column1, column2) VALUES (value1, value2)")
```

```
connection.commit()
```

6. Обновление данных таблицы базы данных SQLite3

Для обновления данных в таблице базы данных SQLite3 в Python с использованием модуля `sqlite3`, можно выполнить SQL-запрос с использованием метода `execute()`:

```
cursor.execute("UPDATE table_name SET column1 = new_value WHERE condition")
```

```
connection.commit()
```

7. Выборка данных из базы данных SQLite3

Для выборки данных из базы данных SQLite3 в Python с использованием модуля `sqlite3`, можно выполнить SQL-запрос с использованием метода `execute()` и получить результаты с помощью метода `fetchall()`:

```
cursor.execute("SELECT * FROM table_name")
results = cursor.fetchall()
```

8. Назначение метода `rowcount`

Метод `rowcount` в модуле `sqlite3` возвращает количество строк, затронутых последним выполненным SQL-запросом. Этот метод может быть использован для получения информации о количестве измененных или выбранных строк после выполнения операций в базе данных SQLite3.

9. Получение списка всех таблиц базы данных SQLite3

Для получения списка всех таблиц базы данных SQLite3 в Python с использованием модуля `sqlite3`, можно выполнить SQL-запрос к системной таблице `sqlite_master`:

```
cursor.execute("SELECT name FROM sqlite_master WHERE type='table'")
tables = cursor.fetchall()
```

10. Проверка существования таблицы при ее добавлении или удалении

Для проверки существования таблицы при ее добавлении или удалении в базе данных SQLite3 в Python с использованием модуля `sqlite3`, можно выполнить запрос к системной таблице `sqlite_master` и проверить наличие соответствующей записи.

11. Проверка существования таблицы при ее добавлении или удалении

Для проверки существования таблицы при ее добавлении или удалении в базе данных SQLite3 в Python с использованием модуля `sqlite3`, можно выполнить запрос к системной таблице `sqlite_master` и проверить наличие соответствующей записи. Например, чтобы проверить существование таблицы с именем `"table_name"`, можно выполнить следующий SQL-запрос:

```
cursor.execute("SELECT name FROM sqlite_master WHERE type='table'  
AND name='table_name'")
```

Если результат запроса содержит записи, то таблица существует.

12. Массовая вставка данных в базу данных SQLite3

Для массовой вставки данных в базу данных SQLite3 в Python с использованием модуля `sqlite3`, можно воспользоваться методом `executemany()`. Например, если у нас есть список кортежей `data` с данными для вставки, то можно выполнить следующий код:

```
data = [(value1, value2), (value3, value4), (value5, value6)]  
cursor.executemany("INSERT INTO table_name (column1, column2)  
VALUES (?, ?)", data)  
connection.commit()
```

13. Работа с датой и временем при работе с базами данных SQLite3

При работе с датой и временем в базах данных SQLite3 в Python, можно использовать тип данных `DATE` для хранения даты и `TIMESTAMP` для хранения даты и времени. Также, можно воспользоваться модулем `datetime` для работы с датой и временем в Python и взаимодействия с базой данных SQLite3. Например, для вставки даты в таблицу можно использовать следующий код:

```
import datetime  
current_date = datetime.date.today()  
cursor.execute("INSERT INTO table_name (date_column) VALUES (?)",  
(current_date,))  
connection.commit()
```

Это позволит вставить текущую дату в столбец `"date_column"` таблицы `"table_name"`.