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

Институт цифрового развития
Кафедра инфокоммуникаций

«Лабораторная работа 2.20 Основы работы с SQLite3»

ОТЧЕТ
по лабораторной работе №23
дисциплины
«Основы программной инженерии»

Выполнил:
Луценко Дмитрий Андреевич
2 курс, группа ПИЖ-б-о-21-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка и сопровождение программного обеспечения», очная форма обучения

_____
(подпись)

Проверил:
_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Лабораторная работа 2.20 Основы работы с SQLite3

Цель работы: исследовать базовые возможности системы управления базами данных SQLite3.

Ход работы:

```
C:\Users\dimaf>sqlite3 your.db
SQLite version 3.41.2 2023-03-22 11:56:21
Enter ".help" for usage hints.
sqlite> CREATE TABLE pages (
(x1...> title TEXT
(x1...> url TEXT,
(x1...> theme INTEGER,
(x1...> num INTEGER);
sqlite> .tables
pages
sqlite> DROP TABLE pages
   ...> ;
sqlite> .tables
sqlite>
```

Рисунок 1 – Пример 1

```
sqlite> CREATE TABLE pages (
(x1...> _id INTEGER PRIMARY KEY AUTOINCREMENT,
(x1...> title TEXT,
(x1...> url TEXT,
(x1...> theme INTEGER,
(x1...> num INTEGER);
```

Рисунок 2 – Пример 2. Создание таблицы с полем ID, автоматически увеличивающимся на 1 при внесении новой записи

```
sqlite> CREATE TABLE sections (
(x1...> _id INTEGER PRIMARY KEY,
(x1...> name TEXT);
sqlite> INSERT INTO sections (id, name) VALUES (1, 'Information');
Parse error: table sections has no column named id
sqlite> INSERT INDO sections(_id, name) VALUES (1, 'Information');
Parse error: near "INDO": syntax error
  INSERT INDO sections(_id, name) VALUES (1, 'Information');
         ^--- error here
sqlite> INSERT INTO sections(_id, name) VALUES (1, 'Information');
sqlite> INSERT INTO sections VALUES (2, 'Digital Systems');
sqlite> INSERT INTO sections(name, _id) VALUES ('Boolean Algebra', 3);
sqlite> SELECT * FROM sections;
1|Information
2|Digital Systems
3|Boolean Algebra
```

Рисунок 3 – Пример 3

```
sqlite> create table customer(name);
sqlite> select *
   ...> from customer;
sqlite> .schema customer
CREATE TABLE customer(name);
sqlite>
```

Рисунок 4 – Задача 1

.schema возвращает список и структуру всех таблиц в базе. В данном случае она вернула CREATE TABLE customer(name);

```
sqlite> CREATE TABLE city(name);
Run Time: real 0.000 user 0.000479 sys 0.000000
sqlite> select count(*) from city;
0
Run Time: real 0.000 user 0.000148 sys 0.000000
```

Рисунок 5 – Задача 2

Вместо .something нужно написать команду .timer.

```
sqlite> .import --csv city.csv city
sqlite> select max(length(city)) from city;
25
```

Рисунок 6 – Задача 3

Запрос вернул число 25.

```
sqlite> .mode csv
sqlite> .import city.csv city
```

Рисунок 7 – Задача 4

Для импорта csv файла без опции –csv необходимо было указать .mode csv.

```
sqlite> SELECT timezone, count(city)
   ...> FROM city
   ...> WHERE federal_district == "Приволжский" OR
   ...> federal_district == "Сибирский"
   ...> GROUP BY timezone
   ...> ORDER BY timezone ASC;
┌──────────┬──────────────┐
│ timezone │ count(city)  │
├──────────┼──────────────┤
│ UTC+3    │ 101          │
│ UTC+4    │ 41           │
│ UTC+5    │ 58           │
│ UTC+6    │ 6            │
│ UTC+7    │ 86           │
│ UTC+8    │ 22           │
└──────────┴──────────────┘
```

Рисунок 8 – Задача 5

Ответ на 5 задачу: 58.

```
sqlite> .mode box
sqlite> WITH dist AS(
   ...> SELECT city,
   ...> ((53.195 - geo_lat) * (53.195 - geo_lat) + (50.107 - geo_lon) * (50.107 - geo_lon))
   ...> as distance FROM city)
   ...> SELECT city, distance FROM dist
   ...> ORDER BY distance ASC
   ...> limit 4;
┌──────────┬──────────────────────┐
│   city   │       distance       │
├──────────┼──────────────────────┤
│ Самара   │ 3.25960000032931e-09 │
│ Новокуйбышевск │ 0.0344928813314883 │
│ Чапаевск │ 0.128219944384608    │
│ Кинель   │ 0.278804683469801    │
└──────────┴──────────────────────┘
```

Рисунок 9 – Задача 6

```
sqlite> SELECT timezone, count(city)
   ...> FROM city
   ...> GROUP BY timezone
   ...> ORDER BY count(city) DESC
   ...> ;
```

timezone	count(city)
UTC+3	660
UTC+5	173
UTC+7	86
UTC+4	66
UTC+9	31
UTC+8	28
UTC+2	22
UTC+10	22
UTC+11	17
UTC+6	6
UTC+12	6

Рисунок 10 – Задача 7

```
sqlite> .mode csv
sqlite> .headers ON
sqlite> .separator |
sqlite> SELECT timezone, count(city)
   ...> FROM city
   ...> GROUP BY timezone
   ...> ORDER BY count(city) DESC
   ...> ;
timezone|count(city)
UTC+3|660
UTC+5|173
UTC+7|86
UTC+4|66
UTC+9|31
UTC+8|28
UTC+2|22
UTC+10|22
UTC+11|17
UTC+6|6
UTC+12|6
```

Рисунок 11 – Задача 7 в формате CSV

Индивидуальное задание.

Загрузите в SQLite выбранный Вами датасет в формате CSV (датасет можно найти на сайте Kaggle). Сформируйте более пяти запросов к таблицам БД. Выгрузите результат выполнения запросов в форматы CSV и JSON.

```
sqlite> .once zap1.csv
sqlite> SELECT restaurant, item, calories
   ...> FROM fastfood WHERE
   ...> calories >= 500
   ...> ;
sqlite> .mode json
sqlite> .once zap1.json
sqlite> SELECT restaurant, item, calories
   ...> FROM fastfood WHERE
   ...> calories >= 500
   ...> ;
```

Рисунок 12 – Запрос 1

```
sqlite> .once zap2.csv
sqlite> SELECT restaurant, item
   ...> FROM fastfood WHERE restaurant == 'Mcdonalds'
   ...> ;
sqlite> .mode json
sqlite> .once zap2.json
sqlite> SELECT restaurant, item
   ...> FROM fastfood WHERE
   ...> restaurant == 'Mcdonalds'
   ...> ;
```

Рисунок 13 – Запрос 2

```
sqlite> .once zap3.csv
sqlite> SELECT restaurant, count(item)
   ...> FROM fastfood
   ...> GROUP BY restaurant
   ...> ORDER BY count(item) DESC;
```

Рисунок 14 – Запрос 3 в csv файл

```
sqlite> .mode json
sqlite> .once zap3.json
sqlite> SELECT restaurant, count(item)
   ...> FROM fastfood
   ...> GROUP BY restaurant
   ...> ORDER BY count(item) DESC;
```

Рисунок 15 – Запрос 3 в json файл

```
sqlite> .once zap4.csv
sqlite> WITH perc AS(
(x1...> SELECT item, ((cal_fat * 1.00) / (calories * 1.00) * 100.00)
(x1...> as percent FROM fastfood)
   ...> SELECT item, percent from perc
   ...> ORDER BY percent DESC
   ...> ;
```

Рисунок 16 – Запрос 4 в csv файл

```
sqlite> .once zap4.json
sqlite> WITH perc AS(
(x1...> SELECT item, ((cal_fat * 1.00) / (calories * 1.00) * 100.00)
(x1...> as percent FROM fastfood)
   ...> SELECT item, percent from perc
   ...> ORDER BY percent DESC
   ...> ;
```

Рисунок 17 – Запрос 4 в json файл

```
sqlite> .mode csv
sqlite> .once zap5.csv
sqlite> SELECT item
   ...> FROM fastfood
   ...> WHERE item like '%Burger%'
   ...> ;
sqlite> .mode json
sqlite> SELECT item
   ...> FROM fastfood
   ...> WHERE item like '%Burger%'
   ...> ;
```

Рисунок 18 – Запрос 5

Вывод: в ходе лабораторной работы исследованы базовые возможности системы управления базами данных SQLite3.

Ответы на контрольные вопросы:

1. Каково назначение реляционных баз данных и СУБД? В РБД существуют механизмы установления связей между таблицами. Делается это с помощью так называемых первичных и внешних ключей. Назначение СУБД: Представим, что есть большая база данных, скажем, предприятия. Это очень большой файл, его используют множество человек сразу, одни изменяют данные, другие выполняют поиск информации. Табличный процессор не может следить за всеми операциями и правильно их обрабатывать. Кроме того, загружать в память большую БД целиком – не лучшая идея. Здесь требуется программное обеспечение с другими возможностями. ПО для работы с базами данных называют системами управления базами данных, то есть СУБД.

2. Каково назначение языка SQL? Язык SQL предназначен для создания и изменения реляционных баз данных, а также извлечения из них данных. Другими словами, SQL – это инструмент, с помощью которого человек управляет базой данных.

3. Из чего состоит язык SQL? Язык SQL состоит из операторов, инструкций и вычисляемых функций.

4. В чем отличие СУБД SQLite от клиент-серверных СУБД? SQLite – это система управления базами данных, отличительной особенностью которой является ее встраиваемость в приложения. Это значит, что большинство СУБД являются самостоятельными приложениями, взаимодействие с которыми организовано по принципу клиент-сервер. Программа-клиент посылает запрос на языке SQL, СУБД, которая в том числе может находиться на удаленном компьютере, возвращает результат запроса. В свою очередь SQLite является написанной на языке C библиотекой, которую динамически или статически подключают к программе. Для большинства

языков программирования есть свои привязки (API) для библиотеки SQLite. Так в Python СУБД SQLite импортируют командой import sqlite3. Причем модуль sqlite3 входит в стандартную библиотеку языка и не требует отдельной установки. С другой стороны, библиотеку SQLite можно скачать с сайта разработчика. Она встроена в консольную утилиту sqlite3, с помощью которой можно на чистом SQL создавать базы данных и управлять ими. Также существуют включающие SQLite приложения с графическим интерфейсом пользователя от сторонних разработчиков. Уход от клиент-серверной модели вовсе не означает, что SQLite – это учебная или урезанная СУБД. Это означает лишь специфику ее применения в роли встраиваемого компонента.

5. Как установить SQLite в Windows и Linux? В Ubuntu установить sqlite3 можно командой sudo apt install sqlite3 . В этом случае утилита вызывается командой sqlite3 . Также можно скачать с сайта https://sqlite.org архив с последней версией библиотеки, распаковать и вызвать в терминале утилиту. Для операционной системы Windows скачивают свой архив (sqlite tools-win32-*.zip) и распаковывают. Далее настраивают путь к каталогу, добавляя адрес каталога к переменной PATH (подобное можно сделать и в Linux). Возможно как и в Linux работает вызов утилиты по ее адресу. Android же имеет уже встроенную библиотеку SQLite.

6. Как создать базу данных SQLite? При вызове утилиты sqlite3 в качестве аргумента можно указать имя базы данных. Если БД существует, она будет открыта. Если ее нет, она будет создана и открыта. $ sqlite3 your.db

7. Как выяснить в SQLite какая база данных является текущей? Выяснить, какая база данных является текущей, можно с помощью команды .databases утилиты sqlite3. Если вы работаете с одной БД, а потом открываете другую, то текущей становится вторая БД.

8. Как создать и удалить таблицу в SQLite? Таблицы базы данных создаются с помощью директивы CREATE TABLE языка SQL. После CREATE TABLE идет имя таблицы, после которого в скобках перечисляются имена столбцов и их тип

9. Что является первичным ключом в таблице? Для реляционных баз данных важно, чтобы каждую запись-строку таблицы можно было однозначно идентифицировать. То есть в таблицах не должно быть полностью совпадающих строк. Записи должны отличаться хотя бы по одному полю. С этой целью принято создавать дополнительное поле, которое часто называют ID или подобно. Чтобы исключить возможность ввода одинаковых идентификаторов, столбец ID назначают первичным ключом. PRIMARY KEY – ограничитель, который заставляет СУБД проверять уникальность значения данного поля у каждой добавляемой записи.

10. Как сделать первичный ключ таблицы автоинкрементным? Если нам не важно, какие конкретно идентификаторы будут записываться в поле _id, а важна только уникальность поля, следует назначить полю еще один ограничитель – автоинкремент – AUTOINCREMENT.

11. Каково назначение инструкций NOT NULL и DEFAULT при создании таблиц? Ограничитель NOT NULL используют, чтобы запретить оставление поля пустым. По умолчанию, если поле не является первичным ключом, в него можно не помещать данные. В этом случае полю будет присвоено значение NULL. В случае NOT NULL вы не сможете добавить запись, не указав значения соответствующего поля.

Однако, добавив ограничитель DEFAULT, вы сможете не указывать значение. DEFAULT задает значение по умолчанию. В результате, когда данные в поле не передаются при добавлении записи, поле заполняется тем, что было указано по умолчанию.

12. Каково назначение внешних ключей в таблице? Как создать внешний ключ в таблице? С помощью внешнего ключа устанавливается связь между записями разных таблиц. Внешний ключ в одной таблице для другой является первичным. Внешние ключи не обязаны быть уникальными. В одной таблице может быть несколько внешних ключей, при этом каждый будет устанавливать связь со своей таблицей, где он является первичным.

FOREIGN KEY (theme) REFERENCES sections(_id) FOREIGN KEY является ограничителем, так как не дает нам записать в поле столбца theme какое-либо иное значение, которое не встречается в качестве первичного ключа в таблице sections.

13. Как выполнить вставку строки в таблицу базы данных SQLite? С помощью оператора INSERT языка SQL выполняется вставка данных в таблицу.

Синтаксис команды: INSERT INTO (, , ...) VALUES (, , ...);

После INSERT INTO указывается имя таблицы, после в скобках перечисляются столбцы. После слова VALUES перечисляются данные, вставляемые в поля столбцов.

14. Как выбрать данные из таблицы SQLite? С помощью оператора SELECT осуществляется выборочный просмотр данных из таблицы. В простейшем случае оператор имеет следующий синтаксис, где вместо указывается имя таблицы: SELECT * FROM ; Такая команда отображает значения всех столбцов и строк заданной таблицы. На выборку всех столбцов указывает звездочка после слова SELECT. А все строки будут выбраны потому, что после имени таблицы нет оператора WHERE языка SQL. WHERE позволяет задавать условие, согласно которому отображаются только удовлетворяющие ему строки.

15. Как ограничить выборку данных с помощью условия WHERE? Условие WHERE используется не только с оператором SELECT, также с UPDATE и DELETE. С помощью WHERE определяются строки, которые будут выбраны, обновлены или удалены. По сути это фильтр. После ключевого слова WHERE записывается логическое выражение, которое может быть как простым (содержащим операторы = или ==, >, =, <=, !=, BETWEEN), так и сложным (AND, OR, NOT, IN, NOT IN).

16. Как упорядочить выбранные данные?

При выводе данных их можно не только фильтровать с помощью WHERE, но и сортировать по возрастанию или убыванию с помощью

оператора ORDER BY. ASC – сортировка от меньшего значения к большему. DESC – сортировка от большего значения к меньшему.

17. Как выполнить обновление записей в таблице SQLite?

UPDATE ... SET – обновление полей записи UPDATE имя_таблицы SET имя_столбца = новое_значение WHERE условие; Чаще всего условием является ID конкретной записи, в результате чего обновляется только она: sqlite> UPDATE pages SET num = 10 ...> WHERE _id = 3;

18. Как удалить записи из таблицы SQLite? DELETE FROM – удаление записей таблицы DELETE FROM имя_таблицы WHERE условие; Без WHERE будут удалены все строки, однако сама таблица останется. Она будет пустой. Для удаления самой таблицы из базы данных используется команда DROP TABLE имя_таблицы; . Примеры: sqlite> DELETE FROM pages WHERE _id = 6; sqlite> DELETE FROM pages WHERE theme = 2;

19. Как сгруппировать данные из выборки из таблицы SQLite?

В SQL кроме функций агрегирования есть оператор GROUP BY, который выполняет группировку записей по вариациям заданного поля. То есть GROUP BY группирует все записи, в которых встречается одно и то же значение в указанном столбце, в одну строку. Так следующая команда выведет не количество тем, а их номера: sqlite> SELECT theme FROM pages ...> GROUP BY theme;

20. Как получить значение агрегатной функции (например: минимум, максимум, количество записей и т. д.) в выборке из таблицы SQLite? Вывод количества столбцов таблицы: sqlite> SELECT count() FROM pages; Поиск максимального ID: sqlite> SELECT max(_id) FROM pages

21. Как выполнить объединение нескольких таблиц в операторе SELECT? JOIN – соединение таблиц В SQL для соединения данных из разных таблиц используется оператор JOIN. В случае с нашим примером запрос будет выглядеть так: sqlite> SELECT pages.title, ...> sections.name AS theme ...> FROM pages JOIN sections ...> ON pages.theme == sections._id;

22. Каково назначение подзапросов и шаблонов при работе с таблицами SQLite? Подзапрос позволяет объединять два запроса в один. Шаблон позволяет искать записи, если неизвестно полное имя поля.

23. Каково назначение представлений VIEW в SQLite? Бывает удобно сохранить результат выборки для дальнейшего использования. Для этих целей в языке SQL используется оператор CREATE VIEW, который создает представление – виртуальную таблицу. В эту виртуальную таблицу как бы сохраняется результат запроса. Таблица виртуальная потому, что на самом деле ее нет в базе данных. В такую таблицу не получится вставить данные, обновить их или удалить. Можно только посмотреть хранящиеся в ней данные, сделать из нее выборку. С другой стороны, если вы вносите изменения в реальные таблицы, они будут отражены и в виртуальных, потому что СУБД каждый раз, когда запрашивается представление, использует SQL выражение представления для обновления данных.

24. Какие существуют средства для импорта данных в SQLite? Команда .import

25. Каково назначение команды .schema? Она показывает схему данных всей таблицы.

26. Как выполняется группировка и сортировка данных в запросах SQLite?

Группировка и сортировка

Сколько городов в каждом из федеральных округов?

```
select federal_district as district, count(*)
as city_count
from city
group by 1
order by 2 desc;
```

27. Каково назначение "табличных выражений" в SQLite? Это обычный селект, к которому можно для краткости обращаться по имени, как к таблице.

28. Как осуществляется экспорт данных из SQLite в форматы CSV и JSON? Кроме .once есть команда .output , которая тоже направляет вывод в указанный файл. Вот в чем разница: .once samara.csv действует только для следующей команды (select from city в нашем примере). Если выполнить еще один селект — его результаты уже пойдут не в файл, а на экран. .output samara.csv действует до тех пор, пока не будет явно отменена. Сколько бы селектов вы не выполнили, их результаты SQLite запишет в samara.csv . Отменить можно, выполнив еще один .output без параметров

29. Какие еще форматы для экспорта данных Вам известны? Markdown, HTML. Также Экспорт таблицы может осуществляться в формат текстовых файлов (*.txt, *.csv), файлов SQL-запросов (*.sql), баз данных SQLite (*.sqlite, *sqlitedb), баз данных Microsoft Access (*.mdb, *.accdb), баз данных Microsoft SQL Server (*.mdf), таблиц Paradox (*.db) и таблиц dBase (*.dbf).