

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

Институт цифрового развития
Кафедра инфокоммуникаций

«Основы работы с SQLite3»

ОТЧЕТ
по лабораторной работе №20
дисциплины
«Основы программной инженерии»

Выполнила:

Кувшин Ирина Анатольевна

2 курс, группа ПИЖ-б-о-21-1,

011.03.04 «Программная инженерия»,

направленность (профиль) «Разработка

и сопровождение программного

обеспечения», очная форма обучения

(подпись)

Проверил:

(подпись)

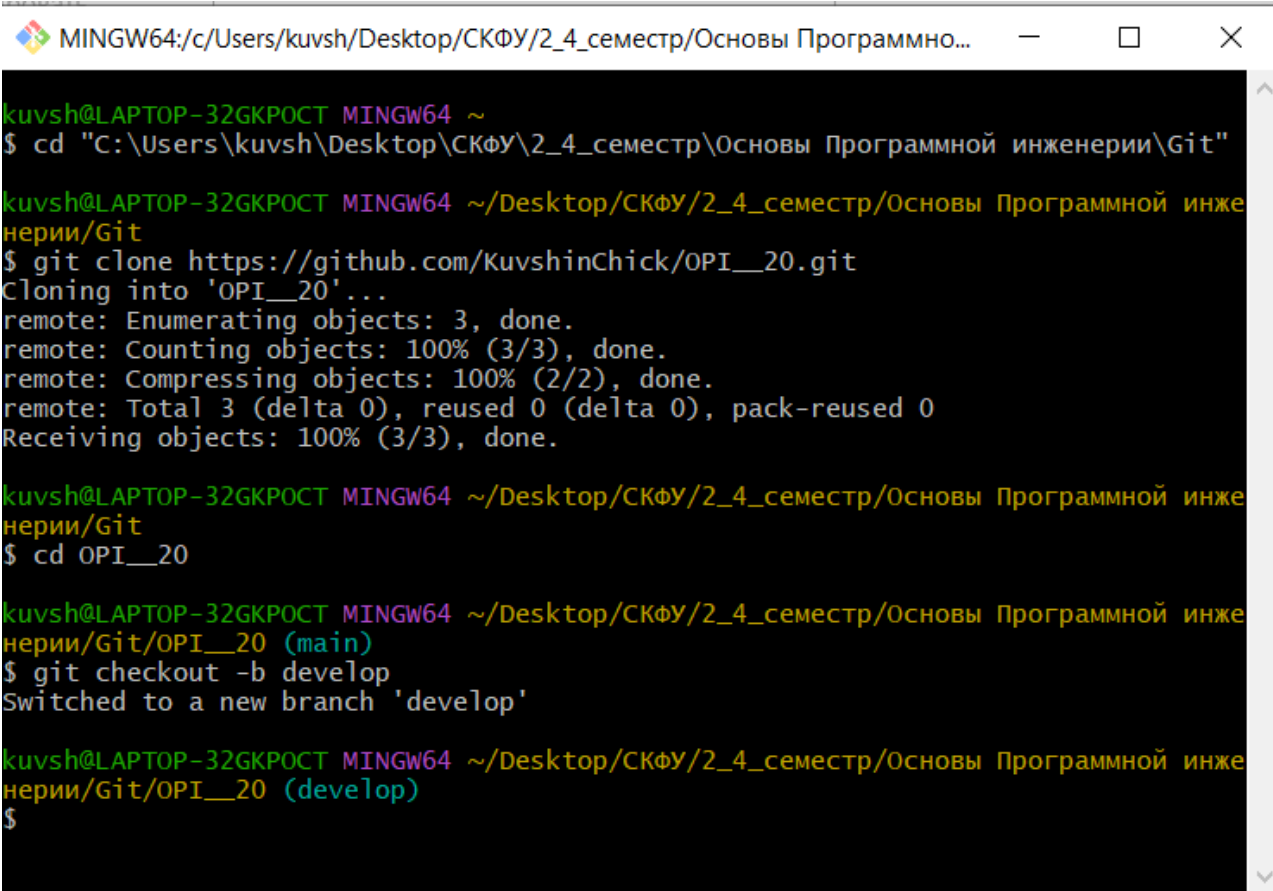
Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

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

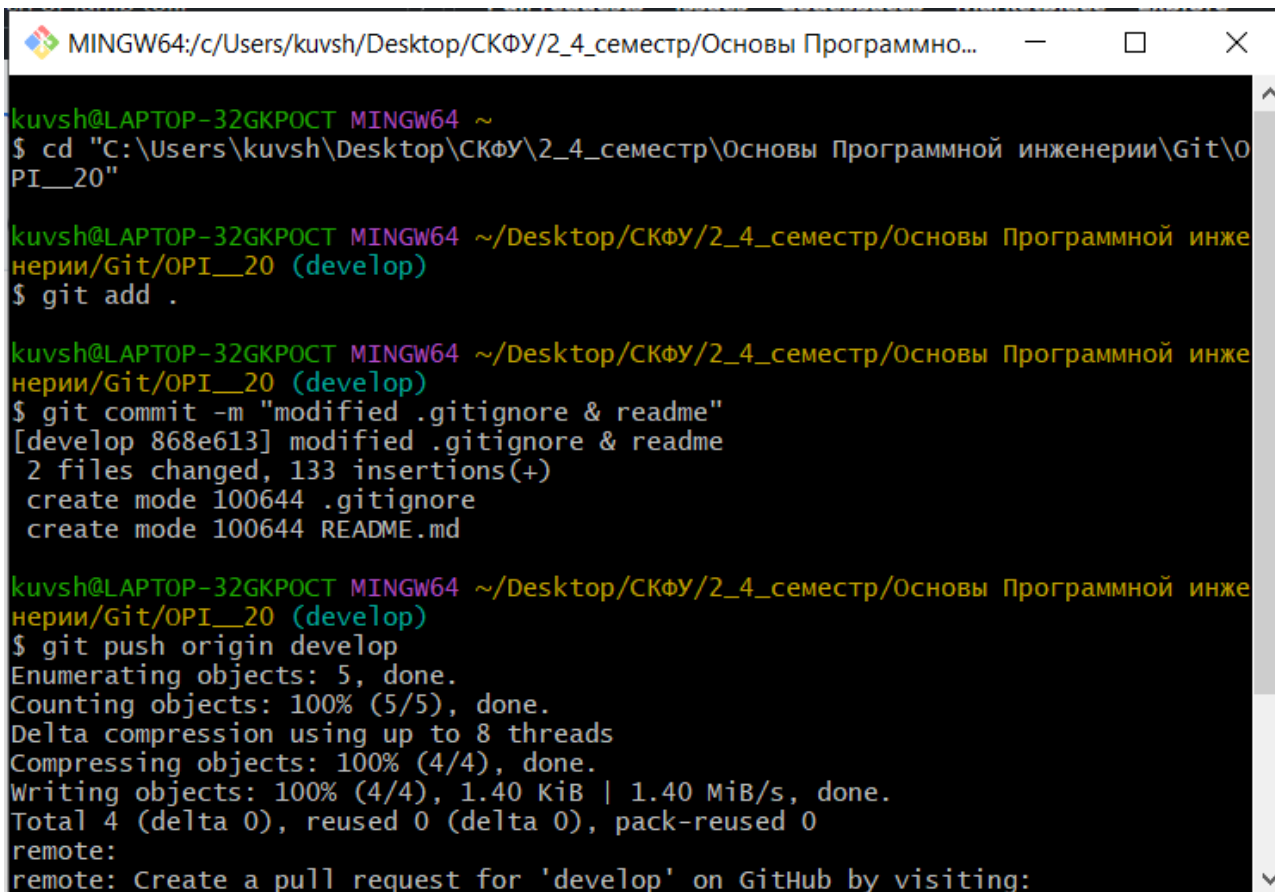
Ход работы:

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.
3. Выполните клонирование созданного репозитория.
4. Дополните файл .gitignore необходимыми и для выбранного языка программирования и интегрированной среды разработки.
5. Добавьте в файл README.md информацию о группе и ФИО студента, выполняющего лабораторную работу.
6. Добавьте файл README и зафиксируйте сделанные изменения.



```
MINGW64:/c:/Users/kuvsh/Desktop/СКФУ/2_4_семестр/Основы Программно...
kuvsh@LAPTOP-32GKPOCT MINGW64 ~
$ cd "C:\Users\kuvsh\Desktop\СКФУ\2_4_семестр\Основы Программной инженерии\Git"
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Основы Программной инженерии/Git
$ git clone https://github.com/KuvshinChick/OPI__20.git
Cloning into 'OPI__20'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Основы Программной инженерии/Git
$ cd OPI__20
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Основы Программной инженерии/Git/OPI__20 (main)
$ git checkout -b develop
Switched to a new branch 'develop'
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Основы Программной инженерии/Git/OPI__20 (develop)
$
```

Рисунок 20.1 – Клонирование репозитория и создание ветки develop



```
MINGW64:/c:/Users/kuvsh/Desktop/СКФУ/2_4_семестр/Основы Программно...
kuvsh@LAPTOP-32GKPOCT MINGW64 ~
$ cd "C:\Users\kuvsh\Desktop\СКФУ\2_4_семестр\Основы Программной инженерии\Git\OPI__20"

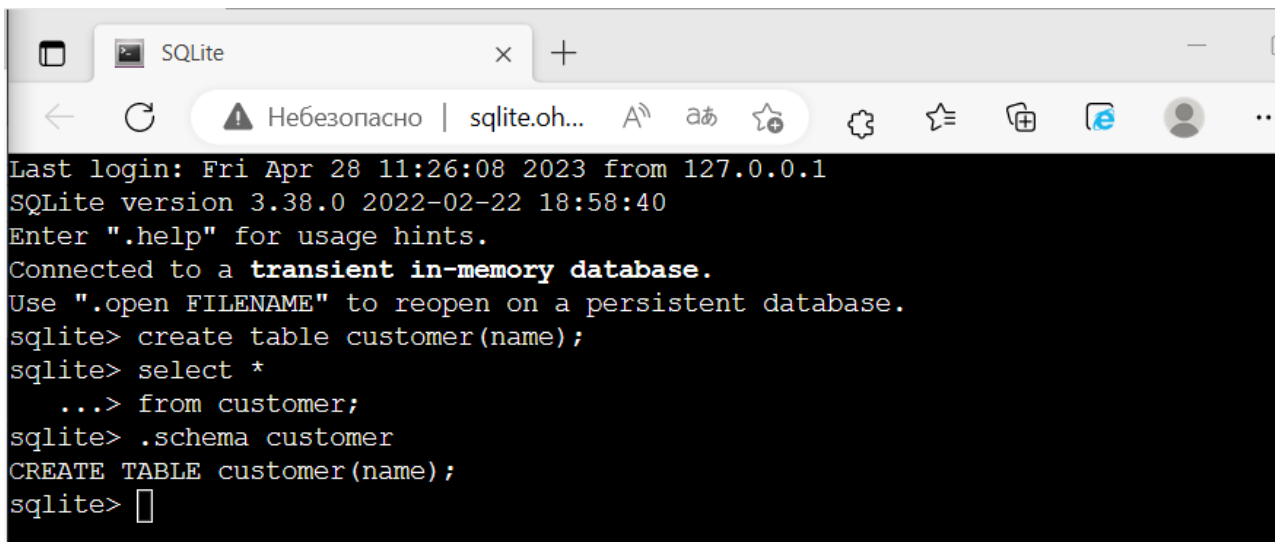
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Основы Программной инженерии/Git/OPI__20 (develop)
$ git add .

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Основы Программной инженерии/Git/OPI__20 (develop)
$ git commit -m "modified .gitignore & readme"
[develop 868e613] modified .gitignore & readme
2 files changed, 133 insertions(+)
create mode 100644 .gitignore
create mode 100644 README.md

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Основы Программной инженерии/Git/OPI__20 (develop)
$ git push origin develop
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 1.40 KiB | 1.40 MiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
```

Рисунок 20.2 – Обновление .gitignore и readme

7. Решите задачу: выполните в песочнице команды:



```
SQLite
Last login: Fri Apr 28 11:26:08 2023 from 127.0.0.1
SQLite version 3.38.0 2022-02-22 18:58:40
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> create table customer(name);
sqlite> select *
...> from customer;
sqlite> .schema customer
CREATE TABLE customer(name);
sqlite> 
```

Рисунок 20.3 – Проработка и результат запроса

Вот что здесь происходит: Первая команда (create table) создает таблицу customer с единственным столбцом name . Вторая команда (select) показывает содержимое таблицы customer (она пустая). Третья команда (.schema) показывает список и структуру всех таблиц в базе. create и select — это SQL-

запросы, часть стандарта SQL. Запрос может занимать несколько строк, а в конце всегда ставится точка с запятой. `.schema` — это специальная команда SQLite, не часть стандарта SQL. Специальные команды всегда начинаются с точки, занимают ровно одну строку, а точку запятой в конце ставить не надо.

8. Решите задачу: с помощью команды `.help` найдите в песочнице команду, которая отвечает за вывод времени выполнения запроса. Если ее включить, в результатах запроса добавится строчка:

```
sqlite> .help
.archive ...           Manage SQL archives
.auth ON|OFF           Show authorizer callbacks
.backup ?DB? FILE      Backup DB (default "main") to FILE
.bail on|off           Stop after hitting an error.  Default OFF
.binary on|off         Turn binary output on or off.  Default OFF
.cd DIRECTORY          Change the working directory to DIRECTORY
.changes on|off        Show number of rows changed by SQL
.check GLOB            Fail if output since .testcase does not match
.clone NEWDB           Clone data into NEWDB from the existing database
.connection [close] [#] Open or close an auxiliary database connection
.databases             List names and files of attached databases
.dbconfig ?op? ?val?   List or change sqlite3_db_config() options
.dbinfo ?DB?          Show status information about the database
.dump ?OBJECTS?        Render database content as SQL
.echo on|off           Turn command echo on or off
.eqp on|off|full|...   Enable or disable automatic EXPLAIN QUERY PLAN
.excel                Display the output of next command in spreadsheet
.exit ?CODE?           Exit this program with return-code CODE
.expert              EXPERIMENTAL. Suggest indexes for queries
.explain ?on|off|auto? Change the EXPLAIN formatting mode.  Default: auto
.filectrl CMD ...      Run various sqlite3_file_control() operations
.fullschema ?--indent? Show schema and the content of sqlite_stat tables
.headers on|off        Turn display of headers on or off
.help ?-all? ?PATTERN? Show help text for PATTERN
.import FILE TABLE    Import data from FILE into TABLE
.imposter INDEX TABLE Create imposter table TABLE on index INDEX
.indexes ?TABLE?       Show names of indexes
.limit ?LIMIT? ?VAL?   Display or change the value of an SQLITE_LIMIT
.lint OPTIONS          Report potential schema issues.
.load FILE ?ENTRY?     Load an extension library
.log FILE|off          Turn logging on or off.  FILE can be stderr/stdout
.mode MODE ?OPTIONS?   Set output mode
.nononce STRING        Suspend safe mode for one command if nonce matches
.nullvalue STRING      Use STRING in place of NULL values
.once ?OPTIONS? ?FILE? Output for the next SQL command only to FILE
.open ?OPTIONS? ?FILE? Close existing database and reopen FILE
.output ?FILE?         Send output to FILE or stdout if FILE is omitted
```

Рисунок 20.4 – Проработка и результат запроса

```
sqlite> .timer on
sqlite> select count(*) from customer;
0
Run Time: real 0.000 user 0.000000 sys 0.000169
sqlite> █
```

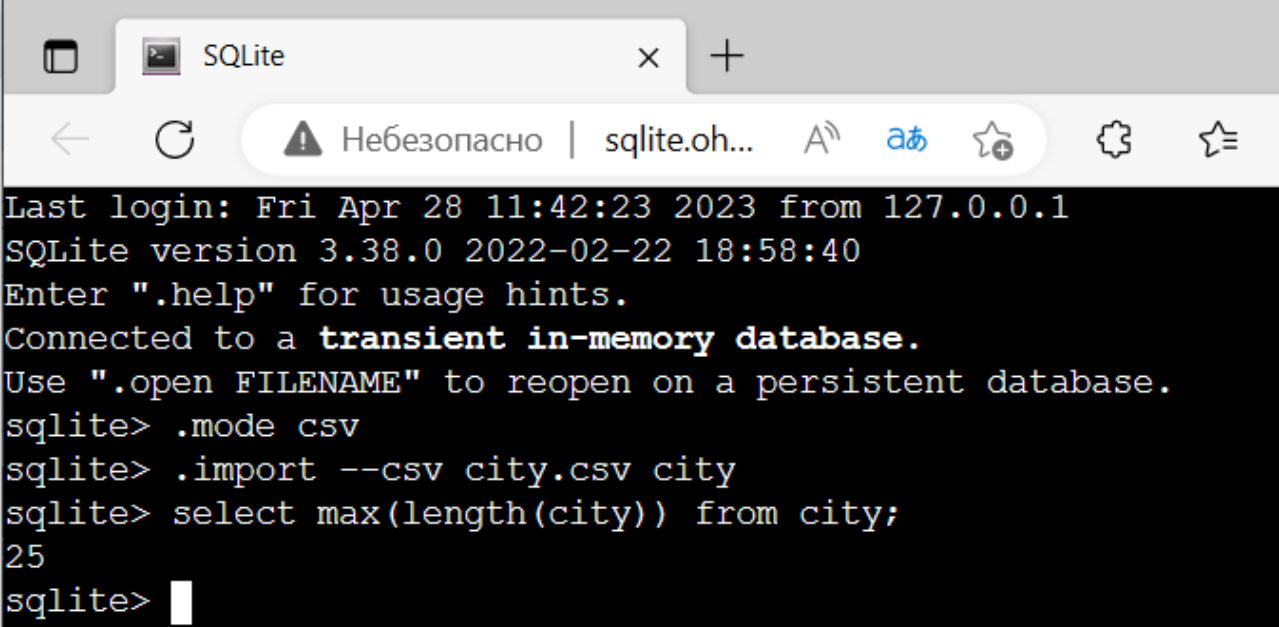
Рисунок 20.5 – Проработка и результат запроса

9. Решите задачу: загрузите файл city.csv в песочнице:

```
sqlite> .import --csv city.csv city
sqlite> select max(length(city)) from city;
25
Run Time: real 0.001 user 0.001065 sys 0.000000
sqlite> █
```

Рисунок 20.6 – Проработка и результат запроса

10. Решите задачу: загрузите файл city.csv в песочнице с помощью команды `.import`, но без использования опции `--csv`.



```
Last login: Fri Apr 28 11:42:23 2023 from 127.0.0.1
SQLite version 3.38.0 2022-02-22 18:58:40
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .mode csv
sqlite> .import --csv city.csv city
sqlite> select max(length(city)) from city;
25
sqlite> █
```

Рисунок 20.7 – Проработка и результат запроса

11. Решите задачу: напишите в песочнице запрос, который посчитает количество городов для каждого часового пояса в Сибирском и Приволжском

федеральных округах. Выведите столбцы timezone и city_count , отсортируйте по значению часового пояса:

```
sqlite> Select timezone, count(*) as city_count
...> from city
...> group by 1
...> order by 1 asc;
UTC+10,22
UTC+11,17
UTC+12,6
UTC+2,22
UTC+3,660
UTC+4,66
UTC+5,173
UTC+6,6
UTC+7,86
UTC+8,28
UTC+9,31
sqlite> □
```

Рисунок 20.8 – Проработка и результат запроса

12. Решите задачу: напишите в песочнице запрос, который найдет три ближайших к Самаре города, не считая саму Самару.

```
sqlite> select city, geo_lat, geo_lon from city where
...> city == "Самара";
```

city	geo_lat	geo_lon
Самара	53.1950306	50.1069518

```
sqlite> with dist as(
...> select address,
...> (POW(53.1950306 - geo_lat,2) + POW( 50.1069518 - geo_lon,2)) as distance
...> from city)
...> select address, distance from dist
...> order by distance asc
...> limit 5;
```

address	distance
г Самара	0.0
Самарская обл, г Новокуйбышевск	0.0344833790157681
Самарская обл, г Чапаевск	0.128213124744169
Самарская обл, г Кинель	0.278853932906283
Самарская обл, г Жигулевск	0.417580687385884

Рисунок 20.9 – Проработка и результат запроса

13. Решите задачу: напишите в песочнице запрос, который посчитает количество городов в каждом часовом поясе. Отсортируйте по количеству городов по убыванию.

А теперь выполните этот же запрос, но так, чтобы результат был в формате CSV, с заголовками, с разделителем «pipe» |

```
sqlite> SELECT timezone, count(*) as counter
...> from city
...> group by 1
...> order by 2 desc;
```

timezone	counter
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

```
sqlite> .mode csv
sqlite> .separator |
sqlite> .headers on
sqlite> SELECT timezone, count(*) as counter
...> from city
...> group by 1
...> order by 2 desc;
timezone|counter
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 20.10 – Проработка и результат запроса

14. Выполните индивидуальное задание. Каждый запрос к базе данных сохраните в файл с расширением sql. Зафиксируйте изменения.

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.

[Video Game Sales | Kaggle](#)

```
sqlite> .import --csv vgsales.csv sale
sqlite> SELECT Platform, count(*) as counter
...> from sale
...> group by 1
...> order by 2 asc;
GG|1
PCFX|1
TG16|2
3D0|3
SCD|6
WS|6
NG|12
GEN|27
DC|52
GB|98
NES|98
2600|133
```

```
sqlite> .mode csv
sqlite> .once req_1.csv
sqlite> SELECT Platform, count(*) as counter
...> from sale
...> group by 1
...> order by 2 asc;
sqlite> .once req_1.json
sqlite> SELECT Platform, count(*) as counter
...> from sale
...> group by 1
...> order by 2 asc;
sqlite>
```

Рисунок 20.11 – Результат проработки запроса #1


```

sqlite> .once req_2.csv
sqlite> SELECT Name, Year
...> from sale Where
...> Year BETWEEN 2015 AND 2023;
sqlite> .once req_2.json
sqlite> SELECT Name, Year
...> from sale Where
...> Year BETWEEN 2015 AND 2023;
sqlite>

```

```

1  "Call of Duty: Black Ops 3",2015
2  "FIFA 16",2015
3  "Star Wars Battlefront (2015)",2015
4  "Call of Duty: Black Ops 3",2015
5  "Fallout 4",2015
6  "FIFA 17",2016
7  Splatoon,2015
8  "Uncharted: The Nathan Drake Collection",2015
9  Halo 5: Guardians,2015
10 "Uncharted 4: A Thief's End",2016
11 "Fallout 4",2015
12 "NBA 2K16",2015
13 "Batman: Arkham Knight",2015
14 "The Witcher 3: Wild Hunt",2015
15 "Tom Clancy's The Division",2016
16 "Star Wars Battlefront (2015)",2015
17 "Metal Gear Solid V: The Phantom Pain",2015
18 "Assassin's Creed Syndicate",2015
19 "Monster Hunter X",2015
20 "FIFA 16",2015

```

JSON file length: 32 398 lines: 963 Ln: 9 Col: 25 Pos: 242 Windows (CR LF) UTF-8 INS

Рисунок 20.12 – Результат проработки запроса #2

```

sqlite> .once req_3.csv
sqlite> SELECT Name, Platform
...> from sale Where
...> Platform == "PS4";
sqlite> .once req_3.json
sqlite> SELECT Name, Platform
...> from sale Where
...> Platform == "PS4";
sqlite>

```

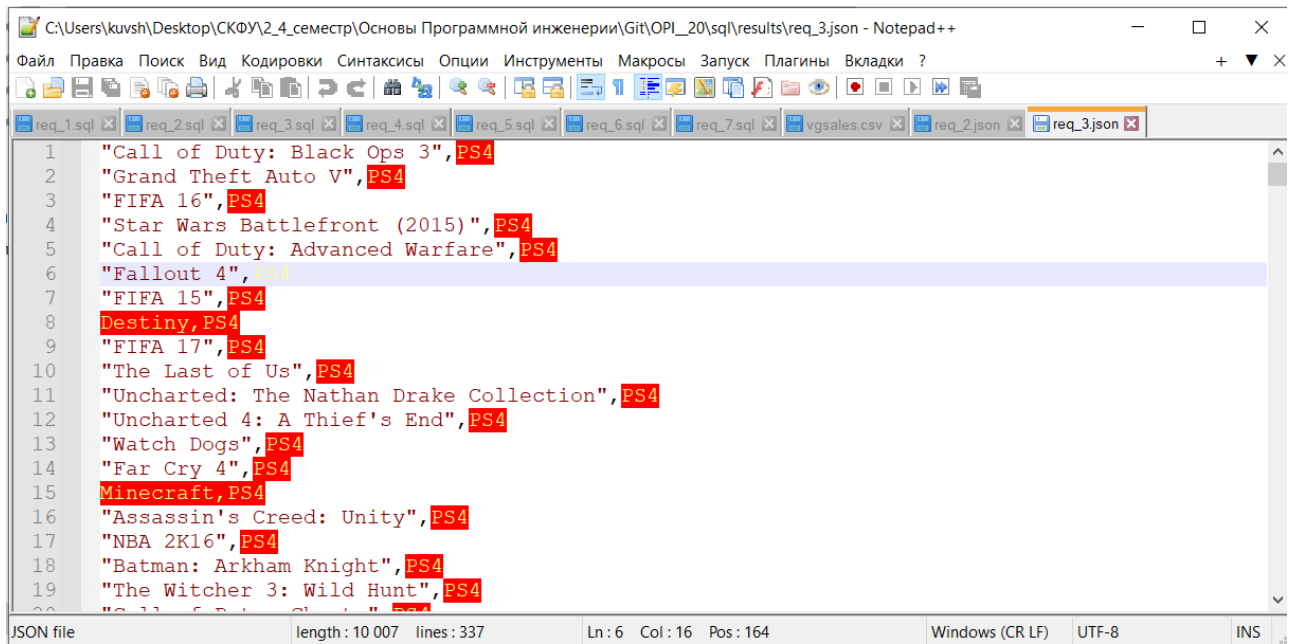


Рисунок 20.13 – Результат проработки запроса #3

```
sqlite> .once req_4.csv
sqlite> SELECT Name, Genre
...> from sale Where
...> Genre == "Racing";
sqlite> .once req_4.json
sqlite> SELECT Name, Genre
...> from sale Where
...> Genre == "Racing";
sqlite>
```



Рисунок 20.14 – Результат проработки запроса #4

```

sqlite> .once req_5.csv
sqlite> SELECT max(EU_Sales) FROM sale;
sqlite> .once req_5.json
sqlite> SELECT max(EU_Sales) FROM sale;

```

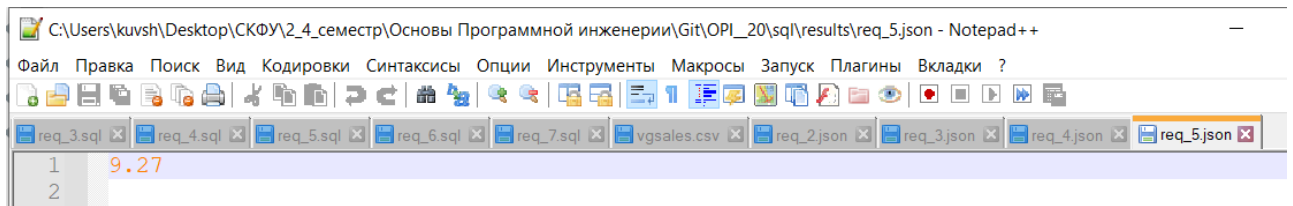


Рисунок 20.15 – Результат проработки запроса #5

```

sqlite> .import --csv vgsales.csv sale
sqlite> .mode csv
sqlite> .once req_6.csv
sqlite> SELECT Name, Genre
...> from sale Where
...> Name Like 'Mortal%';
sqlite> .once req_6.json
sqlite> SELECT Name, Genre
...> from sale Where
...> Name Like 'Mortal%';
sqlite> _

```

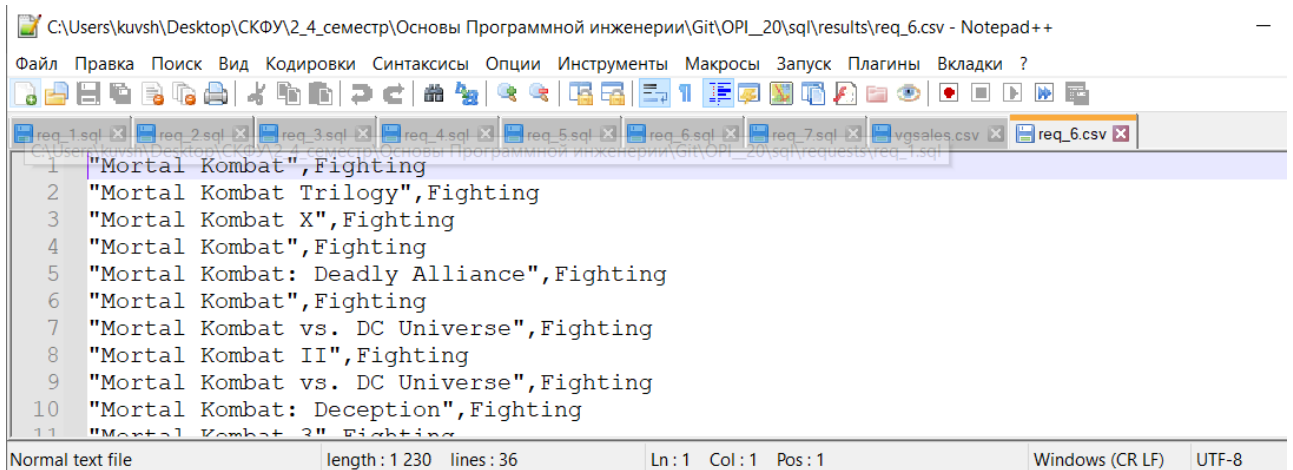


Рисунок 20.16 – Результат проработки запроса #6

```

sqlite> .once req_7.csv
sqlite> select Publisher, count()
...> from sale
...> group by Publisher
...> order by count(*) desc;
sqlite> .once req_7.json
sqlite> select Publisher, count()
...> from sale
...> group by Publisher
...> order by count(*) desc;
sqlite>

```

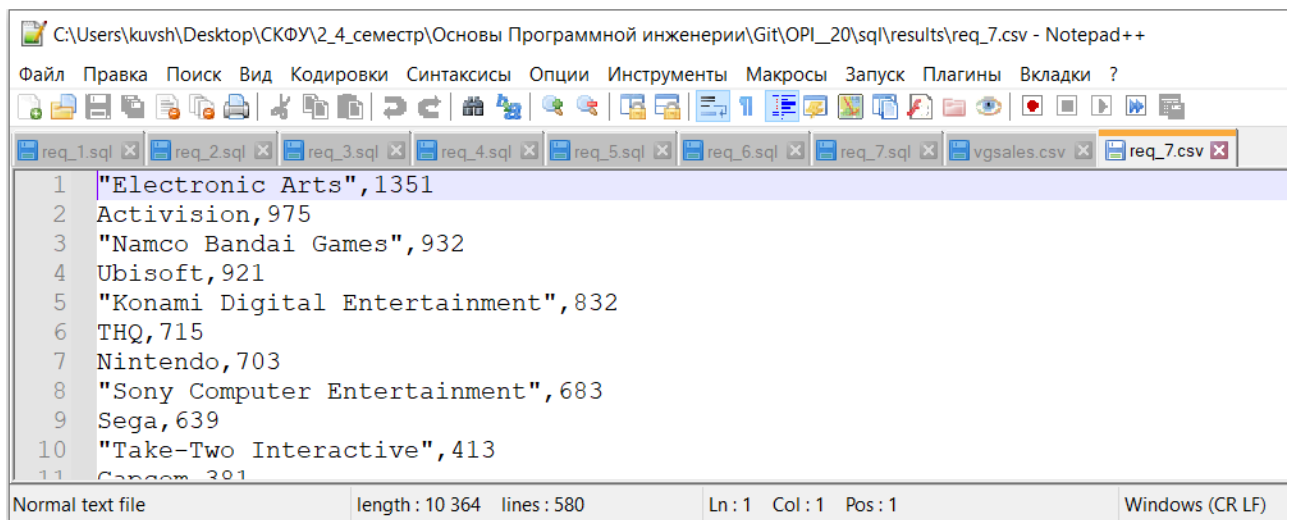


Рисунок 20.17 – Результат проработки запроса #7

15. Добавьте отчет по лабораторной работе в формате PDF в папку doc репозитория. Зафиксируйте изменения.

16. Отправьте изменения в локальном репозитории в удаленный репозиторий GitHub.

17. Проконтролируйте изменения, произошедшие в репозитории GitHub.

18. Отправьте адрес репозитория GitHub на электронный адрес преподавателя.

Контрольные вопросы

1. Каково назначение реляционных баз данных и 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.

Теперь вернемся к вопросу о том, что такое реляционная базы данных (РБД). Слово "реляция" происходит от "relation", то есть "отношение". Это означает, что в РБД существуют механизмы установления связей между таблицами. Делается это с помощью так называемых первичных и внешних ключей.

2. Каково назначение языка SQL?

SQL – это язык программирования декларативного типа. 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?

Сам язык SQL состоит из операторов, инструкций и вычисляемых функций. Зарезервированные слова, которыми обычно выступают операторы, принято писать заглавными буквами. Однако написание их не прописными, а строчными буквами к ошибке не приводит.

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.

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?

С помощью `sqlite3` создать или открыть существующую базу данных можно двумя способами. Во-первых, 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
```

Во вторых, работая в самой программе, можно выполнить команду

```
.open 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:

```
sqlite> CREATE TABLE pages (  
...> title TEXT,  
...> url TEXT,  
...> theme INTEGER,  
...> num INTEGER);
```

Для удаления целой таблицы из базы данных используется директива DROP TABLE, после которой идет имя удаляемой таблицы.

9. Что является первичным ключом в таблице?

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.

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

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.

Однако в SQLite поддержка внешнего ключа по умолчанию отключена. Поэтому, даже назначив столбец внешним ключом, вы сможете записывать в его поля любые значения. Чтобы включить поддержку внешних ключей в sqlite3, надо выполнить команду PRAGMA foreign_keys = ON; . После этого добавить в таблицу запись, в которой внешний ключ не совпадает ни с одним первичным из другой таблицы, не получится.

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 <table_name>
(<column_name1>, <column_name2>, ...)
VALUES
(<value1>, <value2>, ...);
```

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

```
sqlite> INSERT INTO sections
...> (_id, name) VALUES
...> (1, 'information');
```

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 <table_name> указывается имя таблицы:

```
SELECT * FROM <table_name>;
```

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). Примеры:

```
sqlite> SELECT * FROM pages
...> WHERE _id == 3;

sqlite> SELECT * FROM pages WHERE
...> theme == 2 AND num == 100;

sqlite> SELECT * FROM pages WHERE
...> theme <= 2;
```

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.

```
sqlite> SELECT url, title, theme
...> FROM pages
...> ORDER BY url ASC;
amount-information|Amount of Information|1
binary|Binary System|2
information|What is Information|1
logic-low|Lows of Logic Algebra|3
octal|Octal System|2

sqlite> SELECT url, title FROM pages
...> WHERE theme == 1
...> ORDER BY url DESC;
information|What is Information
amount-information|Amount of Information
```

17. Как выполнить обновление записей в таблице SQLite?

Операторы UPDATE и DELETE надо использовать с осторожностью. Если с помощью WHERE не заданы обновляемые или удаляемые строки, будут обновлены или удалены все записи таблицы. Поэтому данные команды почти всегда используются совместно с WHERE.

UPDATE ... SET – обновление полей записи

Синтаксис команды:

```
UPDATE имя_таблицы
SET имя_столбца = новое_значение
WHERE условие;
```

18. Как удалить записи из таблицы SQLite?

DELETE FROM – удаление записей таблицы

Синтаксис команды удаления из таблицы одной или нескольких записей:

```
DELETE FROM имя_таблицы WHERE условие;
```

19. Как сгруппировать данные из выборке 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;
1
2
3
```

Таким образом мы можем узнать, на какие темы имеются страницы в базе данных.

20. Как получить значение агрегатной функции (например: минимум, максимум, количество записей и т. д.) в выборке из таблицы SQLite?

Как быть, если надо посчитать общее количество строк таблицы или найти запись, содержащую максимальное значение, или посчитать сумму значений столбца? Для этих целей в языке SQL предусмотрены различные функции агрегирования данных. Наиболее используемые – count(), sum(), avg(), min(), max(). Используют их совместно с оператором SELECT.

Вывод количества столбцов таблицы:

```
sqlite> SELECT count() FROM pages;
```

Поиск максимального ID:

```
sqlite> SELECT max(_id) FROM pages;
```

Количество различных вариантов значения столбца:

```
sqlite> SELECT count(DISTINCT theme)
...> FROM pages;
```

21. Как выполнить объединение нескольких таблиц в операторе SELECT?

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.

24. Какие существуют средства для импорта данных в SQLite?

Чтобы не добавлять города вручную, возьмем готовый набор данных — [city.csv](#) ([1 скачать](#)). Скачаем файл и загрузим данные ([песочница](#)):

```
$ sqlite3 city-1.db  
SQLite version 3.34.0 2020-12-01 16:14:00  
Enter ".help" for usage hints.  
sqlite> .mode box  
sqlite> .import --csv city.csv city  
sqlite> select count(*) from city;
```

count(*)
1117

Команда `.import` автоматически создала таблицу `city` со всеми столбцами из `city.csv` и загрузила данные из файла. Неплохо!

25. Каково назначение команды .schema?

С помощью команд `.schema` и `PRAGMA TABLE_INFO()` можно посмотреть схему 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
;
```

district	city_count
центральный	304
приволжский	200
северо-западный	148
уральский	115
сибирский	114
южный	96
дальневосточный	82
северо-кавказский	58

27. Каково назначение "табличных выражений" в SQLite?

Выражение `with history as (...)` создает именованный запрос. Название — `history`, а содержание — селекты в скобках (век основания для каждого города). К `history` можно обращаться по имени в остальном запросе, что мы и делаем.

Строго говоря, селекты в блоке `with` называют «табличным выражением» (common table expression, CTE). Так что если встретите в документации — не удивляйтесь. Запомните главное: это обычный селекты, к которому можно для краткости обращаться по имени, как к таблице.

28. Как осуществляется экспорт данных из SQLite в форматы CSV и JSON?

Вывод по умолчанию

Без заголовков, разделитель — запятая.

```
.mode csv
select kladr_id, city
from city
where region = 'Самарская'
limit 3;
6300000200000,"жигулевск"
6300001000000,"кинель"
6301700100000,"нефтегорск"
```

CSV — проверенный универсальный формат. Но что, если удобнее выгрузить в другом? SQLite это умеет.

JSON

```
.mode json
select kladr_id, city
from city
where region = 'Самарская'
limit 3;
[{"kladr_id":"6300000200000","city":"жигулевск"},
{"kladr_id":"6300001000000","city":"кинель"},
{"kladr_id":"6301700100000","city":"нефтегорск"}]
```

29. Какие еще форматы для экспорта данных Вам известны?

Markdown, HTML.

Формат текстовых файлов (*.txt, *.csv), файлов SQL-запросов (*.sql), баз данных SQLite (*.sqlite, *.sqlitedb), баз данных Microsoft Access (*.mdb, *.accdb).