Институт цифрового развития
Кафедра инфокоммуникаций

«Основы работы с SQLite»

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

Выполнил:
Мизин Глеб Егорович
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.

7. Решите задачу: выполните в песочнице команды:

```
create table customer(name);

select *
from customer;

.schema customer
```

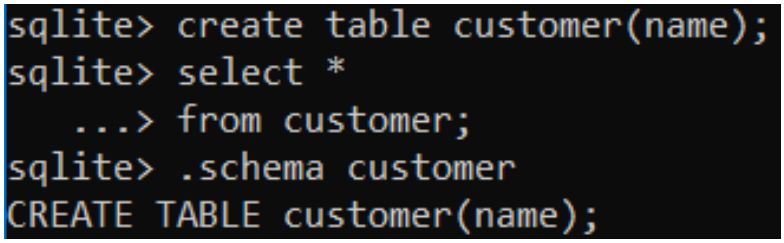


Рисунок 1 – Решение задания №1

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:

```
Run Time: real XXX user XXX sys XXX
```

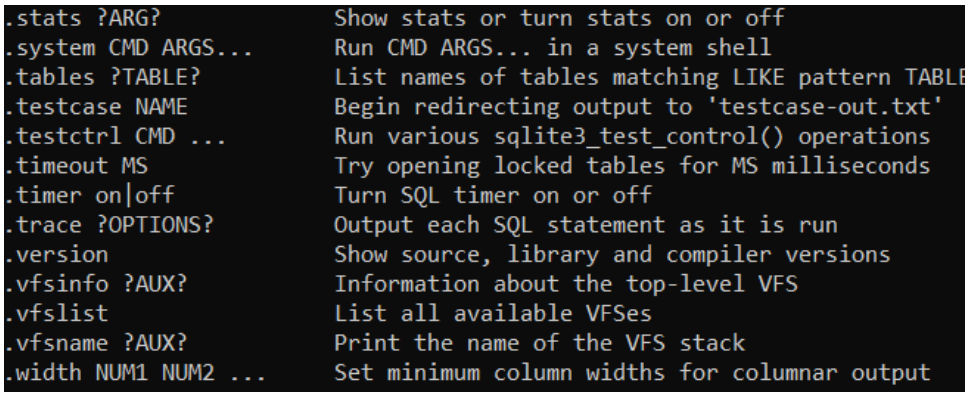


Рисунок 2 – Вывод команды .help

```
sqlite> .timer on
sqlite> select count(*) from city;
┌──────────┐
│ count(*) │
├──────────┤
│ 3        │
└──────────┘
Run Time: real 0.015 user 0.000000 sys 0.000000
sqlite>
```

Рисунок 3 – Выполнение необходимого запроса

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

```
.import --csv city.csv city
```

Затем выполните такой запрос:

```
select max(length(city)) from city;
```

Какое число он вернул?

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

Рисунок 4 – Выполнение заданного запроса

10. Решите задачу: загрузите файл city.csv в песочнице с помощью команды `.import`, но без использования опции `--csv`. Эта опция появилась только в недавней версии SQLite (3.32, май 2020), так что полезно знать способ, подходящий для старых версий.

    Вам поможет команда `.help import`. Всего должно получиться две команды:

```
do_something
.import city.csv city
```

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

Рисунок 5 – Выполнение необходимого 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:

```
┌──────────┬─────────────┐
| timezone | city_count  |
├──────────┼─────────────┤
| UTC+3    | xxx         |
| UTC+4    | xx          |
| UTC+5    | xx          |
| UTC+6    | x           |
| UTC+7    | xx          |
| UTC+8    | xx          |
└──────────┴─────────────┘
```

Укажите в ответе значение `city_count` для `timezone = UTC+5`.

```
sqlite> Select timezone, count(*) as city_count
   ...> from city
   ...> group by 1
   ...> order by 1 asc;
UTC+10,44
UTC+11,34
UTC+12,12
UTC+2,44
UTC+3,1320
UTC+4,132
UTC+5,346
UTC+6,12
UTC+7,172
UTC+8,56
UTC+9,62
timezone,1
```

Рисунок 6 – Результат выполнения запроса

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

    Укажите в ответе названия этих трех городов через запятую в порядке удаления от Самары. Например:

    ```
    Нижний Новгород, Москва, Владивосток
    ```

    > Чтобы посчитать расстояние между двумя городами, используйте формулу из школьного курса геометрии:
    >
    > $$distance^2 = (lat_1 - lat_2)^2 + (lon_1 - lon_2)^2 \quad (1)$$
    >
    > Где (lat_1, lon_1) — координаты первого города, а (lat_2, lon_2) — координаты второго.

Рисунок 7 – результат работы

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

    ```
    ┌────────────┬──────────────┐
    │ timezone   │ city_count   │
    ├────────────┼──────────────┤
    │ UTC+3      │ xxx          │
    │ UTC+5      │ xxx          │
    │ UTC+7      │ xxx          │
    │ UTC+4      │ xxx          │
    │ ...        │              │
    └────────────┴──────────────┘
    ```

    А теперь выполните этот же запрос, но так, чтобы результат был

    - в формате CSV,
    - с заголовками,
    - с разделителем «pipe» |

    Как выглядит четвертая строка результата?

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

timezone	counter
UTC+3	1980
UTC+5	519
UTC+7	258
UTC+4	198
UTC+9	93
UTC+8	84
UTC+2	66
UTC+10	66
UTC+11	51
UTC+6	18
UTC+12	18
timezone	2

Рисунок 8 – Первый запрос для задания

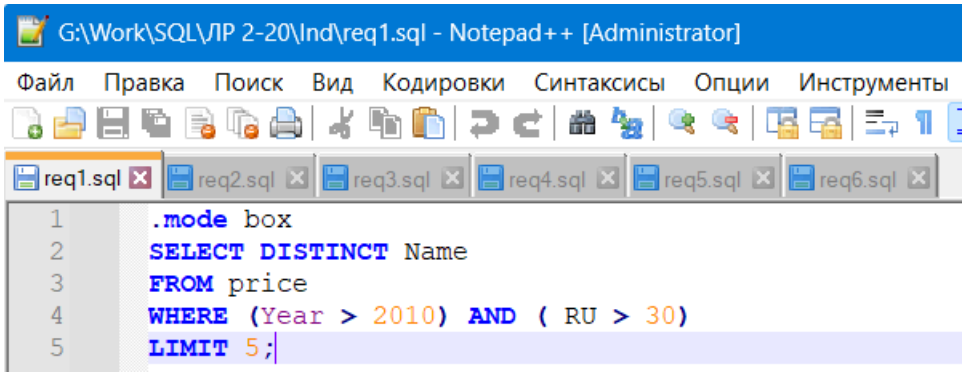
```
sqlite> .mode csv
sqlite> .separator |
sqlite> .headers on
sqlite> select timezone, count(*) as counter
   ...> from city
   ...> group by 1
   ...> order by 2 desc;
timezone|counter
UTC+3|1980
UTC+5|519
UTC+7|258
UTC+4|198
UTC+9|93
UTC+8|84
UTC+2|66
UTC+10|66
UTC+11|51
UTC+6|18
UTC+12|18
timezone|2
```

Рисунок 9 – Второй запрос

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.

Запросы:




```
.mode box
SELECT DISTINCT Name
FROM price
WHERE (Pubr = 'Nintendo') AND ( RU > JP)
LIMIT 10;
```

```
.mode box
SELECT Genre, count(*) as counter
FROM price
group by 1
order by 2 desc;
```

```
.mode box
SELECT Name, Genre
FROM price
WHERE RU = (SELECT MAX(RU) FROM price AS g2 WHERE g2.Genre = price.Genre)
GROUP by Genre;
```

```
.mode box
SELECT Genre, count(*)
FROM price
WHERE Pubr = 'Nintendo' AND (Global > NA)
GROUP BY Genre;
```

```
.mode box
SELECT Name, LENGTH(Name)
FROM price
WHERE LENGTH(Name) = (SELECT MAX(LENGTH(Name)) FROM price);
```

Контрольные вопросы

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(), avr(), 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 (⬇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 — селект в скобках (век основания для каждого города). К `history` можно обращаться по имени в остальном запросе, что мы и делаем.

> Строго говоря, селект в блоке `with` называют «табличным выражением» (common table expression, CTE). Так что если встретите в документации — не удивляйтесь. Запомните главное: 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?

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).