

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

**Отчет по лабораторной работе № 2.20**

**«Основы работы с SQLite3»**

**по дисциплине «Основы программной инженерии»**

Выполнила:  
Образцова Мария Дмитриевна,  
2 курс, группа ПИЖ-б-о-21-1,  
Проверил:  
Доцент кафедры  
инфокоммуникаций,  
Воронкин Р.А.

Ставрополь, 2023 г.

## Методика и порядок выполнения работы

- Изучить теоретический материал работы.
- Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.

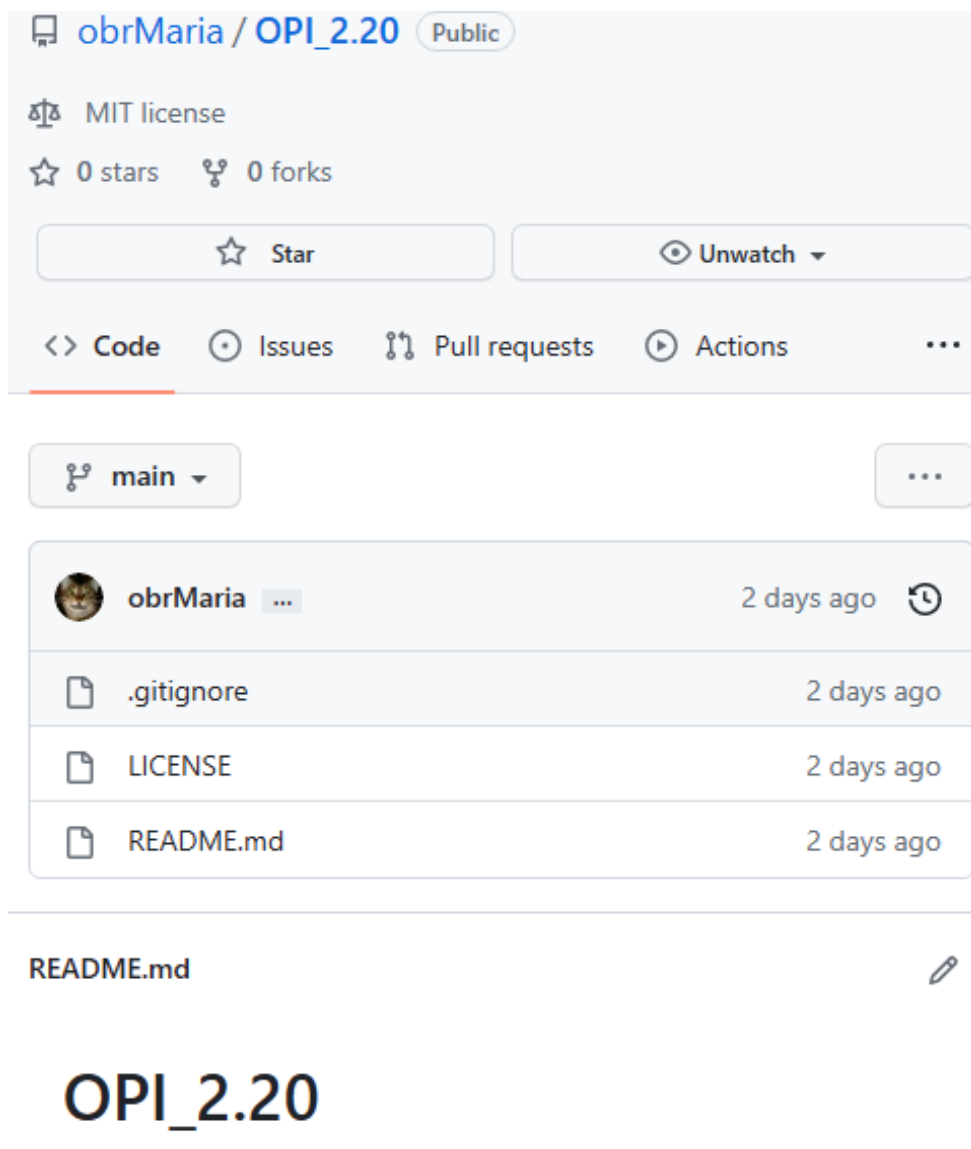


Рисунок – создание репозитория

```
C:\Users\A\Desktop>git clone https://github.com/obrMaria/OPI_2.20.git
Cloning into 'OPI_2.20'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
```

Рисунок – клонирование репозитория

```

A@DESKTOP-GV2A7LK MINGW64 ~/desktop/OPI_2.20 (main)
$ git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/A/desktop/OPI_2.20/.git/hooks]

A@DESKTOP-GV2A7LK MINGW64 ~/desktop/OPI_2.20 (develop)

```

Рисунок – создание ветки develop

## ПРОРАБОТКА ПРИМЕРОВ

```

Last login: Sun Apr 30 13:03:18 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 box
sqlite> create table city (id integer primary key, name text);
sqlite> insert into city (name) values ('Москва'), ('Санкт-Петербург'),
...> ('Новосибирск');
sqlite> select * from city;

```

id	name
1	Москва
2	Санкт-Петербург
3	Новосибирск

```

sqlite>

```

Рисунок – Пример работы в песочнице

```

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

```

Рисунок – Создание и удаление таблицы

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

Рисунок – Первичный ключ

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

Рисунок – Автоинкремент

```
sqlite> CREATE TABLE pages (
...> _id INTEGER PRIMARY KEY AUTOINCREMENT,
...> title TEXT,
...> url TEXT NOT NULL,
...>
...> theme INTEGER NOT NULL,
...> num INTEGER NOT NULL DEFAULT 0);
sqlite> .schema pages
CREATE TABLE pages (
_id INTEGER PRIMARY KEY AUTOINCREMENT,
title TEXT,
url TEXT NOT NULL,

theme INTEGER NOT NULL,
num INTEGER NOT NULL DEFAULT 0);
sqlite> PRAGMA TABLE_INFO(pages);
```

cid	name	type	notnull	dflt_value	pk
0	_id	INTEGER	0		1
1	title	TEXT	0		0
2	url	TEXT	1		0
3	theme	INTEGER	1		0
4	num	INTEGER	1	0	0

```
sqlite>
```

Рисунок – Ограничитель NOT NULL

```
sqlite> CREATE TABLE sections (
...> _id INTEGER PRIMARY KEY,
...> name TEXT);
```

```

sqlite> CREATE TABLE pages (
...> _id INTEGER PRIMARY KEY AUTOINCREMENT,
...> title TEXT,
...> url TEXT NOT NULL,
...> theme INTEGER NOT NULL,
...> num INTEGER NOT NULL DEFAULT 100,
...> FOREIGN KEY (theme) REFERENCES sections(_id)
...> );

sqlite> PRAGMA foreign_keys;

foreign_keys
0

sqlite> PRAGMA foreign_keys = ON;
sqlite> PRAGMA foreign_keys;

foreign_keys
1

sqlite>

```

Рисунок – Внешний ключ

```

sqlite> INSERT INTO sections
...> (_id, name) VALUES
...> (1, 'information');
sqlite> INSERT INTO sections
...> (name, _id)
...> VALUES
...> ('BOOLEAN algebra', 3);
sqlite> INSERT INTO sections
...> VALUES (2, 'digital systems');
sqlite> SELECT * FROM sections;
1|information
2|digital systems
3|BOOLEAN algebra
sqlite>

sqlite> INSERT INTO pages VALUES
...> (1, 'what is information',
...> 'information',1,1);
sqlite> INSERT INTO pages
...> (title,url,theme,num)
...> VALUES
...> ('amount of information',
...> 'amount-informations', 1, 2);
sqlite> SELECT * FROM pages;
1|what is information|information|1|1
2|amount of information|amount-informations|1|2

```

Рисунок – Оператор INSERT

```

sqlite> .mode csv
sqlite> SELECT * FROM pages;
1,"what is information",information,1,1
2,"amount of information",amount-informations,1,2
sqlite> .mode html
sqlite> SELECT * FROM pages;
<TR><TD>1</TD>
<TD>what is information</TD>
<TD>information</TD>
<TD>1</TD>
<TD>1</TD>
</TR>
<TR><TD>2</TD>
<TD>amount of information</TD>
<TD>amount-informations</TD>
<TD>1</TD>
<TD>2</TD>
</TR>
sqlite> SELECT * FROM pages;
_id title url theme num
-----
1 what is information information 1 1
2 amount of information amount-informations 1 2
sqlite> .mode box
sqlite> SELECT * FROM pages;

```

_id	title	url	theme	num
1	what is information	information	1	1
2	amount of information	amount-informations	1	2

```

sqlite>

```

Рисунок – Оператор SELECT

```

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

```

_id	title	url	theme	num
1	What is Information	information	1	1
2	Amount of Information	amount-information	1	2

Рисунок – Оператор WHERE

```

sqlite> SELECT url, title, theme
...> FROM pages
...> ORDER BY url ASC;

```

url	title	theme
amount-information	Amount of Information	1
information	What is Information	1

```

sqlite> SELECT url, title, theme
...> FROM pages
...> ORDER BY url DESC;

```

url	title	theme
information	What is Information	1
amount-information	Amount of Information	1

Рисунок – ORDER BY сортировка

```

sqlite> UPDATE pages SET num = 10
...> WHERE _id = 3;
sqlite> UPDATE pages SET num = 10
...> WHERE _id = 1;
sqlite> SELECT * FROM pages;

```

_id	title	url	theme	num
1	What is Information	information	1	10
2	Amount of Information	amount-information	1	2

```

sqlite> DELETE FROM pages WHERE _id = 2;
sqlite> SELECT * FROM pages;

```

_id	title	url	theme	num
1	What is Information	information	1	10

Рисунок – UPDATE и DELETE – обновление и удаление данных

```

sqlite> SELECT count() FROM pages;

```

count()
4

```

sqlite> SELECT max(_id) FROM pages;

```

max(_id)
5

```

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

```

count(DISTINCT theme)
3

```

sqlite> SELECT DISTINCT theme FROM pages;

```

theme
1
2
3

```

sqlite> SELECT theme, count()
...> FROM pages
...> GROUP BY theme;

```

theme	count()
1	2
2	1
3	1

```

sqlite> SELECT theme, max(num)
...> FROM pages GROUP BY theme;

```

theme	max(num)
1	10
2	100
3	100

```

sqlite>

```

Рисунок – Агрегирование и группировка



```

sqlite> SELECT pages.title,
...> sections.name AS theme
...> FROM pages JOIN sections
...> ON pages.theme == sections._id;
title                theme
-----
What is Information  information
Amount of Information information
BINARY SUSTEM       Digital Systems
LOW of LOGIC         Boolean Algebra

sqlite> SELECT pages.title, sections.name
...> FROM pages JOIN sections
...> ON pages.theme == sections._id
...> WHERE pages.theme == 2
...> OR pages.theme == 3;
title                name
-----
BINARY SUSTEM       Digital Systems
LOW of LOGIC         Boolean Algebra

sqlite> SELECT sections.name AS theme,
...> count() AS qty_articles
...> FROM pages JOIN sections
...> ON pages.theme == sections._id
...> GROUP BY sections.name
...> ORDER BY sections._id;
theme                qty_articles
-----
information          2
Digital Systems      1
Boolean Algebra      1
sqlite>

```

Рисунок – JOIN – соединение таблиц

```

PS C:\Users\A> cd ~\Desktop
>> .\sqlite3.exe city-1.db
SQLite version 3.38.0 2022-02-22 18:58:40
Enter ".help" for usage hints.
sqlite> .mode box
sqlite> .import --csv city.csv city
sqlite> select count(*) FROM city;

```

count(*)
1117

```

sqlite> .schema city
CREATE TABLE IF NOT EXISTS "city"(
"address" TEXT, "postal_code" TEXT, "country" TEXT, "federal_district" TEXT,
"region_type" TEXT, "region" TEXT, "area_type" TEXT, "area" TEXT,
"city_type" TEXT, "city" TEXT, "settlement_type" TEXT, "settlement" TEXT,
"kladr_id" TEXT, "fias_id" TEXT, "fias_level" TEXT, "capital_marker" TEXT,
"okato" TEXT, "oktmo" TEXT, "tax_office" TEXT, "timezone" TEXT,
"geo_lat" TEXT, "geo_lon" TEXT, "population" TEXT, "foundation_year" TEXT);
sqlite> select federal_district, city, population
...> from city limit 10;

```

federal_district	city	population
Южный	Адыгейск	12689
Южный	Майкоп	144055
Сибирский	Горно-Алтайск	62861
Сибирский	Алейск	28528
Сибирский	Барнаул	635585
Сибирский	Белокуриха	15072
Сибирский	Бийск	203826
Сибирский	Горняк	13040
Сибирский	Заринск	47035
Сибирский	Змеиногорск	10569

```

sqlite>

```

Рисунок – Загрузка данных из файлов

```

sqlite> 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

Рисунок – Группировка и сортировка

```

...> WHERE city LIKE '%Красный%';
sqlite> SELECT address
...> FROM city
...> WHERE city LIKE '%Красный%';

```

address
Ростовская обл, г Красный Сулин
Саратовская обл, г Красный Кут
Тверская обл, г Красный Холм

Рисунок – Фильтрация

```

sqlite> SELECT region, city, foundation_year
...> FROM city
...> WHERE foundation_year BETWEEN 1990 AND 2020;

```

region	city	foundation_year
Ингушетия	Магас	1995
Татарстан	Иннополис	2012

Рисунок – Пример городов, которые появились за последние 30 лет

```

sqlite> SELECT count(*)
...> FROM city
...> WHERE federal_district in ('Приволжский', 'Уральский');

```

count(*)
315

Рисунок – Пример количества городов в Приволжском и Уральском округах

```

sqlite> with history as (
...> SELECT
...> city,
...> (foundation_year/100)+1 as century
...> from city
...> )
...> SELECT
...> century || '-й век' as dates,
...> count(*) as city_cout
...> from history
...> GROUP BY century
...> order by century desc
...> ;

```

dates	city_cout
21-й век	1
20-й век	263
19-й век	189
18-й век	191
17-й век	137
16-й век	79
15-й век	39
14-й век	38
13-й век	27
12-й век	44
11-й век	8
10-й век	6
9-й век	4
5-й век	2
3-й век	1
1-й век	88

```
sqlite>
```

Рисунок – Пример количества городов основанных в каждом веке

```

lite> .mode csv
lite> .once samara.csv
lite> select kladr_id, city from city where region = 'Самарская';
lite> .exit

```

Рисунок – Пример экспорта файла csv

```

6301700100000,нефтегорск
sqlite> .mode json
sqlite> 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"}]
sqlite> from city

```

Рисунок – Пример экспорта файла json

```

sqlite> .mode insert cities
sqlite> select kladr_id, city
...> from city
...> where region = 'Самарская'
...> limit 3;
INSERT INTO cities(kladr_id,city) VALUES('6300000200000','Жигулевск');
INSERT INTO cities(kladr_id,city) VALUES('6300001000000','Кинель');
INSERT INTO cities(kladr_id,city) VALUES('6301700100000','Нефтегорск');

```

Рисунок – Пример экспорта с помощью команды insert

```

sqlite> .mode markdown
sqlite> select kladr_id, city
...> from city
...> where region = 'Самарская'
...> limit 3;
| kladr_id | city |
|-----|-----|
| 6300000200000 | Жигулевск |
| 6300001000000 | Кинель |
| 6301700100000 | Нефтегорск |
sqlite> .mode html
sqlite> select kladr_id, city
...> from city
...> where region = 'Самарская'
...> limit 3;
<TR><TH>kladr_id</TH>
<TH>city</TH>
</TR>
<TR><TD>6300000200000</TD>
<TD>Жигулевск</TD>
</TR>
<TR><TD>6300001000000</TD>
<TD>Кинель</TD>
</TR>
<TR><TD>6301700100000</TD>
<TD>Нефтегорск</TD>
</TR>
sqlite>

```

Рисунок – Пример экспорта файла markdown и html

```

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

```

Рисунок – задача 1

С помощью команды .schema можно посмотреть схему таблицы.  
Данная команда показала какие столбцы есть в таблице.

```

sqlite> .timer ON
sqlite> select count(*) from city;
count(*)
1117
Run Time: real 0.015 user 0.000000 sys 0.015625

```

Рисунок – задача 2

вместо .something нужно написать .timer

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

max(length(city))
25

Рисунок – задача 3

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

Рисунок – задача 4

Какая команда должна быть вместо do\_something?

Ответ: .mode

```
sqlite> select
...> timezone, count(city)
...> FROM city
...> WHERE federal_district = 'Сибирский' OR federal_district == 'Приволжский'
...> 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 – 5 задача

Укажите в ответе значение city\_count для timezone = UTC+5 .

Ответ: 58

```
sqlite> .import --csv city.csv city
sqlite> WITH geo_las AS (select geo_lat AS geo_las FROM city WHERE city = 'Самара'),
...> geo_los AS (select geo_lon AS geo_los FROM city WHERE city = 'Самара'),
...> geo_lam AS (SELECT geo_lat AS geo_lam, city FROM city),
...> geo_lou as (SELECT geo_lon AS geo_lou FROM city)
...> SELECT sqrt((POWER((geo_las - geo_lam),2) + POWER((geo_los - geo_lou),2)))
...> AS distance, city
...> FROM (geo_las ,geo_los ,geo_lam, geo_lou )
...> WHERE city != 'Самара'
...> ORDER BY distance ASC
...> LIMIT 3;
```

distance	city
0.00105299999999886	Заречный
0.0094843000000004	Каменка
0.01199310000000051	Елизovo

Рисунок – 6 задача

```
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

```
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
sqlite>
```

Рисунок – 7 задача

### Индивидуальное задание

Каждый запрос к базе данных сохранить в файл с расширением sql. 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 z_1.csv
sqlite> SELECT City, Product_line, Branch
...> FROM sales
...> WHERE Gender == 'Female'
...> LIMIT 15;
```

```
sqlite> .once z_1.json
sqlite> SELECT City, Product_line, Branch
...> FROM sales
...> WHERE Gender == 'Female'
...> LIMIT 15;
```

Рисунок – Первый запрос в csv, json файлы

```

sqlite> .once z_2.csv
sqlite> SELECT
...> Product_line AS product,
...> count(*) AS count_product
...> FROM sales
...> GROUP BY product
...> ORDER BY count_product desc
...> limit 10;
sqlite>

sqlite> .once z_2.json
sqlite> SELECT
...> Product_line AS product,
...> count(*) AS count_product
...> FROM sales
...> GROUP BY product
...> ORDER BY count_product DESC
...> LIMIT 10;
sqlite>

```

Рисунок – Второй запрос в csv, json файлы

```

sqlite> .once z_3.csv
sqlite> SELECT
...> Gender, Total, Payment
...> FROM sales
...> WHERE Payment LIKE 'C%'
...> limit 10;
sqlite> .mode json
sqlite> .once z_3.json
sqlite> SELECT
...> Gender, Total, Payment
...> FROM sales
...> WHERE Payment LIKE 'C%'
...> limit 10;
sqlite>

```

Рисунок – Третий запрос в csv, json файлы

```

sqlite> .once z_4.csv
sqlite> SELECT
...> Date,Time, Product_line
...> FROM sales
...> WHERE Date LIKE '1/21%';
sqlite>

sqlite> .once z_4.json
sqlite> SELECT
...> Date,Time, Product_line
...> FROM sales
...> WHERE Date LIKE '1/21%';
sqlite>

```

Рисунок – Четвертый запрос в csv, json файлы

```

sqlite> .once z_5.csv
sqlite> WITH in_total AS (
...> SELECT ID,Product_line, (Quantity*Unit_price)
...> AS spent
...> FROM sales)
...> SELECT ID,Product_line, spent
...> FROM in_total
...> LIMIT 10;
sqlite> .mode json
sqlite> .once z_5.json
sqlite> WITH in_total AS (
...> SELECT ID,Product_line, (Quantity*Unit_price)
...> AS spent
...> FROM sales)
...> SELECT ID,Product_line, spent
...> FROM in_total
...> LIMIT 10;
sqlite>

```

Рисунок – Пятый запрос в csv, json файлы



Контрольные вопросы:

1. Каково назначение реляционных баз данных и 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. В отличие от привычных нам процедурных языков, в которых есть условия, циклы и функции, в декларативных языках подобных алгоритмических конструкций почти нет. Декларативные выражения представляют собой скорее запросы, описание того, что хочет получить человек.

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. При этом ключевыми операциями являются создание таблиц, добавление записей в таблицы, изменение и удаление записей, выборка записей из таблиц, изменение структуры таблиц.

3. Из чего состоит язык SQL?

Сам язык SQL состоит из операторов, инструкций и вычисляемых функций. Резервированные 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. 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.

#### 5. Как установить SQLite в Windows и Linux?

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. 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.

#### 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.

13. Как выполнить вставку строки в таблицу базы данных SQLite?

С помощью оператора INSERT языка SQL выполняется вставка данных в таблицу.

14. Как выбрать данные из таблицы SQLite?

С помощью оператора SELECT.

15. Как ограничить выборку данных с помощью условия WHERE?

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.

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.

17. Как выполнить обновление записей в таблице SQLite?

UPDATE ... SET – обновление полей записи

18. Как удалить записи из таблицы SQLite?

DELETE FROM – удаление записей таблицы

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.

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

Для этих целей в языке SQL предусмотрены различные функции агрегирования данных. Наиболее используемые – count(), sum(), avr(), min(), max().

21. Как выполнить объединение нескольких таблиц в операторе SELECT?

После FROM указываются обе сводимые таблицы через JOIN. В данном случае неважно, какую указывать до JOIN, какую после. После ключевого слова ON записывается условие сведения. Условие сообщает, как соединять строки разных таблиц.

22. Каково назначение подзапросов и шаблонов при работе с таблицами SQLite?

Шаблоны реализуют поиск по таблице, если неизвестно полное название данных в строке. Подзапросы помогают уменьшить работу путём создания дополнительного запроса внутри основного

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?

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

25. Каково назначение команды .schema?

Показывает какие столбцы есть в таблице, тип их данных и прочие свойства.

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

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

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.

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

```
.mode csv
```

29. Какие еще форматы для экспорта данных Вам известны?

```
.mode list , .mode json
```