**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 № 8
# «Основы работы с SQLite3»

**по дисциплине «Основы программной инженерии»**

Выполнила:
Первых Дарья Александровна,
2 курс, группа ПИЖ-б-о-20-1

Проверил:
Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2022 г

ВЫПОЛНЕНИЕ

```
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     │
└────┴─────────────────┘
```

Рисунок 1 – Пример работы в песочнице

```
sqlite> CREATE TABLE pages (
   ...> title TEXT,
   ...> url TEXT,
   ...> theme INTEGER,
   ...> num INTEGER);
sqlite> .tables
city   pages
sqlite> DROP TABLE pages
   ...> .tables
   ...> ;
Parse error: no such table: pages.tables
sqlite> DROP TABLE pages;
sqlite> .tables
city
```

Рисунок 2 – Создание и удаление таблицы

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

Рисунок 3 – Первичный ключ

```
sqlite> CREATE TABLE pages (
   ...> _id INTEGER PRIMARY KEY AUTOINCREMENT,
   ...> title TEXT,
   ...> url TEXT,
   ...> theme INTEGER,
   ...> num INTEGER);
Parse error: table pages already exists
```

Рисунок 4 – 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 |

Рисунок 5 – 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            |
+--------------+
```

Рисунок 6 – 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;
+-----+-----------------+
| _id |      name       |
+-----+-----------------+
| 1   | information     |
| 2   | Digital Systems |
| 3   | Boolean Algebra |
+-----+-----------------+
```

```
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-information', 1, 2);
```

```
sqlite> SELECT * FROM pages;
```

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

Рисунок 7 – Оператор INSERT

```
sqlite> .mode csv
sqlite> SELECT * FROM pages;
Parse error: no such table: pages
sqlite> .mode box
sqlite> .import --csv city.csv city
sqlite> select count(*) from city;
```

| count(*) |
|---|
| 1117 |

Рисунок 8 – Пример подсчёта строк

```
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);
```

Рисунок 9 – Пример просмотра названий столбцов таблицы

```
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 |

Рисунок 10 – Пример просмотра содержимого таблицы

```
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 11 – Пример группировки

```
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 12 – Пример фильтрации

```
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 13 – Пример городов, которые появились за последние 30 лет

```
sqlite> select count(*)
   ...> from city
   ...> where
   ...> federal_district in ('Приволжский','Уральский')
   ...> ;
```

| count(*) |
|---|
| 315 |

Рисунок 14 – 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_count
   ...> from history
   ...> group by century
   ...> order by century desc
   ...> ;
```

| dates | city_count |
|---|---|
| 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 |

Рисунок 15 – Пример количества городов основанных в каждом веке

```
sqlite> select kladr_id, city from city where region = 'Самарская';
6300000200000,"Жигулевск"
6300001000000,"Кинель"
6301700100000,"Нефтегорск"
6300000300000,"Новокуйбышевск"
6300000400000,"Октябрьск"
6300000500000,"Отрадный"
6300000900000,"Похвистнево"
6300000100000,"Самара"
6300000800000,"Сызрань"
6300000700000,"Тольятти"
6300000600000,"Чапаевск"
```

Рисунок 16 – Пример выгрузки данных

```
sqlite> .mode csv
sqlite> select kladr_id, city
   ...> from city
   ...> where region = 'Самарская'
   ...> limit 3;
6300000200000,"Жигулевск"
6300001000000,"Кинель"
6301700100000,"Нефтегорск"
```

Рисунок 17 – Пример вывода по умолчанию

```
sqlite> .mode csv
sqlite> .headers on
sqlite> select kladr_id, city
   ...> from city
   ...> where region = 'Самарская'
   ...> limit 3;
kladr_id,city
6300000200000,"Жигулевск"
6300001000000,"Кинель"
6301700100000,"Нефтегорск"
```

Рисунок 18 – Пример вывода с заголовками

```
sqlite> .mode list
sqlite> .headers on
sqlite> .separator ,
sqlite> select kladr_id, city
   ...> from city
   ...> where region = 'Самарская'
   ...> limit 3;
kladr_id,city
6300000200000,Жигулевск
6300001000000,Кинель
6301700100000,Нефтегорск
```

Рисунок 19 – Пример вывода без значений в кавычках и другими разделителями

```
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"}]
```

Рисунок 20 – 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 21 – 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>
```

Рисунок 22 – Пример экспорта файла markdown и html

```
sqlite> .import --csv samara.csv samara
sqlite> .mode box
sqlite> select * from samara limit 5;
┌──────────────────────────────────┐
│      6300000200000|"Жигул        │
├──────────────────────────────────┤
│ 6300001000000|"Кинель"           │
│ 6301700100000|"Нефтегорск"       │
│ 6300000300000|"Новокуйбышевск"   │
│ 6300000400000|"Октябрьск"        │
│ 6300000500000|"Отрадный"         │
└──────────────────────────────────┘
```

Рисунок 23 – Пример экспорта файла csv

```
sqlite> drop table if exists samara;
sqlite> create table samara (kladr_id, name);
sqlite> .mode csv
sqlite> .headers on
sqlite> .separator |
sqlite> .import samara.csv samara
sqlite> .mode box
sqlite> select * from samara limit 5;
┌─────────────┬──────────────┐
│  kladr_id   │     name     │
├─────────────┼──────────────┤
│ 6300000200000 │ Жигулевск  │
│ 6300001000000 │ Кинель     │
│ 6301700100000 │ Нефтегорск │
│ 6300000300000 │ Новокуйбышевск │
│ 6300000400000 │ Октябрьск  │
└─────────────┴──────────────┘
```

Рисунок 24 – Пример настраивания разделителей, заголовка, кавычек

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

Рисунок 25 – Задание №7

```
sqlite> .timer on
sqlite> select count(*) from city;
┌──────────┐
│ count(*) │
├──────────┤
│ 1117     │
└──────────┘
Run Time: real 0.001 user 0.000210 sys 0.000120
```

Рисунок 26 – Задание №8

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

Рисунок 27 – Задание №9

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

Рисунок 28 – Задание №10

```
sqlite> with zone as(
   ...> select
   ...> city,
   ...> timezone
   ...> from city
   ...> where federal_district in ('Сибирский', 'Приволжский')
   ...> )
   ...> select
   ...> timezone,
   ...> count(*) as city_count
   ...> from zone
   ...> group by timezone
   ...> order by timezone asc
   ...> ;
┌──────────┬────────────┐
│ timezone │ city_count │
├──────────┼────────────┤
│ UTC+3    │ 101        │
│ UTC+4    │ 41         │
│ UTC+5    │ 58         │
│ UTC+6    │ 6          │
│ UTC+7    │ 86         │
│ UTC+8    │ 22         │
└──────────┴────────────┘
```

Рисунок 29 – Задание №11

```
sqlite> .mode box
sqlite> with dist as(
   ...> select address,
   ...> ((53.195 - geo_lat) * (53.195 - geo_lat) + (50.107 - geo_lon) * (50.107 - geo_
lon)) as distance
   ...> from city
   ...> )
   ...> select
   ...> address,
   ...> distance from dist
   ...> order by distance asc
   ...> limit 5;
┌──────────────────────────────┬──────────────────────┐
│           address            │       distance       │
├──────────────────────────────┼──────────────────────┤
│ г Самара                     │ 3.25960000032931e-09 │
│ Самарская обл, г Новокуйбышевск │ 0.0344928813314883   │
│ Самарская обл, г Чапаевск    │ 0.128219944384608    │
│ Самарская обл, г Кинель      │ 0.278804683469801    │
│ Самарская обл, г Жигулевск   │ 0.417652346753368    │
└──────────────────────────────┴──────────────────────┘
```

Рисунок 30 – Задание №12

```
sqlite> with zone as(
   ...> select
   ...> city,
   ...> timezone
   ...> from city
   ...> )
   ...> select
   ...> timezone,
   ...> count(*) as city_count
   ...> from zone
   ...> group by timezone
   ...> order by city_count desc
   ...> ;
```

| timezone | city_count |
|---|---|
| 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 31 – Задание №13

**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?**

Описание баз данных и выполнение к ним запросов. 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.

Однако в процессе развития языка SQL в нем появились новые средства. Стало возможно описывать и хранить такие объекты как индексы, представления, триггеры и процедуры. То есть в современных диалектах SQL есть элементы процедурных языков.

**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. 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:

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

**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.

В этом случае SQLite будет сам записывать в поле уникальное целочисленное значение по нарастающей от записи к записи. Поскольку это поле заполняется автоматически, то при добавлении записи в таблицу его игнорируют.

**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.

**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>;

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). 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;

Примеры с BETWEEN и IN:

sqlite> SELECT _id, title

...> FROM pages WHERE

...> _id BETWEEN 2 AND 8;

3|Amount of Information 4|Binary System

5|Octal System

6|Lows of Logic Algebra sqlite> SELECT _id, title

...> FROM pages WHERE

...> _id IN (1,2);

**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.

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.

Вывод по умолчанию:

.mode csv

select kladr_id, city from city

where region = 'Самарская' limit 3; 6300000200000,"Жигулевск" 6300001000000,"Кинель"

6301700100000,"Нефтегорск"

.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. 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).