**Практикум. Часть 2**

В прошлый раз мы закрыли программу sqlite3.exe. Запустим её снова, как в прошлый раз, набрав команду **sqlite3 catbase.db**. Так как на прошлом уроке мы создали базу данных, которая хранится в файле catbase.db, то запуская команду, мы открываем эту базу. Чтобы убедиться в этом, наберите команду:

sqlite>.tables

Должно отобразиться название таблицы **schema test**.

Теперь рассмотрим способы извлечения различной информации о базе данных, таблицах и других вещах.

В конце последнего урока мы создали индекс для таблицы. Чтобы получить информацию о нём, наберите команду:

sqlite> .indices test

На экране увидим **text\_idx**, как и было введено в прошлый раз.

Чтобы увидеть строение таблицы, нужно ввести команду:

sqlite> .schema test

В результате получим:

CREATE TABLE test (id integer primary key, value text);

CREATE INDEX text\_idx on test (value);

Знакомые строчки, не так ли? Именно так мы создавали таблицу и индекс. Можно не указывать имя таблицы, тогда будет возрващена информация обо всех базах данных, таблицах и других элементах базы данных.

Существуют другие различные команды для получения информаци:

* type - тип объекта (таблица, индекс, вид, триггер)
* name - имя объекта
* tbl\_name - связанная с объектом таблица
* rootpage - корневая страница индекса в базе данных
* sql - определение SQL (DDL)

Введем следующую команду:

sqlite> .mode column

sqlite> .headers on

sqlite> select type, name, tbl\_name, sql from sqlite\_master order by type;

Получим результат:

type name tbl\_name sql

---------- ---------- ---------- -------------------------------------

index text\_idx test CREATE INDEX text\_idx on test (value)

table test test CREATE TABLE test (id integer primary

view schema schema CREATE VIEW schema as select \* from s

Таким образом мы видим все информацию как на ладони. Это полезно для тех случаев, когда мы смотрим не свою базу, а чужую, и мы не знаем, как она устроена.

Небольшая подсказка: наши команды сохраняются в истории. Поэтому во время ввода вы можете нажать на клавишу Стрелка Вверх на клавиатуре, чтобы быстро получить доступ к длинной строке. Также можно нажать клавишу F7 - в этом случае появится диалоговое окно со списком предыдущих команд.

**Экспорт данных**

Можно экспортировать объекты базы данных в SQL-формате при помощи команды **.dump**. Без аргументов будет экспортирована вся база. Чтобы экспорт шел в файл, а не на экран (по умолчанию), то используйте команду **.output [filename]**. А чтобы восстановить вывод данных на экран, используйте команду **.output stdout**:

sqlite> .output file.sql

sqlite> .dump

sqlite> .output stdout

Когда вы выполните эти команды, то у вас на диске появится файл file.sql (если такой файл уже был, то он будет перезаписан).

**Импорт данных**

Импортировать данные можно двумя способами. Если данные содержатся в SQL-формате, то можно воспользоваться командой **.read**. Если файл содержит данные в формате CSV, то используется команда **.import [file][table]**. Данные обычно разделяются вертикальной чертой |. Но можно использовать и другие разделители. Разделитель можно задать командой **.separator**, а увидить используемый разделитель можно командой **.show**.

sqlite> .show

echo: off

explain: off

headers: on

mode: column

nullvalue: ""

output: stdout

separator: "|"

stats: off

width:

В строчке **separator** вы можете видеть используемый символ.

Мы использовали для экспорта команду **.dump**, поэтому, чтобы импортировать данные из созданного файла **file.sql**, нам подойдет команда **.read**. Давайте удалим все данные из таблицы test и вида schema и восстановим данные из файла file.sql. Чтобы быть уверенным, что данные восстановились из бекапа, можно чуть отредактировать сохраненный файл, например, заменить имена котов с английского на русский.

sqlite> drop table test;

sqlite> drop view schema;

sqlite> .read file.sql

sqlite> select \* from test;

Скорее всего в командной строке вместо русских символов отобразятся кракозябры из-за проблем с кодировкой. Снова верните английские имена котов в файле file.sql и повторите четыре предыдущих команды. Теперь имена отобразятся нормально, а мы заодно убедились, что теперь можем восстанавливать данные.

**Настройка**

Мы можем поменять вид приглашения, который по умолчанию имеет вид **sqlite>** через команду **.prompt**. Давайте поменяем на более знакомое и правильное:

sqlite> .prompt 'cat>'

cat>

Теперь вместо глупого приглашения используется любимое нами слово.

Возвращаемые данные можно форматировать различными способами при помощи команды **.mode** с параметрами csv, column, html, insert, line, list, tabs, tcl. По умолчанию используется .list. Предположим, вы хотите сделать дамп базы данных в CVS-формате:

sqlite3> .output file.csv

sqlite3> .separator ,

sqlite3> select \* from test;

sqlite3> .output stdout

Если открыть файл текстовым редактором, то увидим:

1,barsik

2,murzik

3,vaska

4,ryzhik

**Экспорт и импорт части данных**

Не всегда нужно сохранять всю базу, иногда нужно сохранить только её часть. Предположим, мы решили сохранить только те ряды таблицы, в которых имена котов заканчиваются на **ik**:

sqlite> .output ik.csv

sqlite> .separator ,

sqlite> select \* from test where value like '%ik';

sqlite> .output stdout

Теперь, если мы захотим импортировать сохраненные данные в такую же таблицу с схожей структурой, то делаем следующее:

sqlite> create table test2(id integer primary key, value text);

sqlite> .import ik.csv test2

На этом второй урок завершён.