SQLite – это автономный, работающий без сервера транзакционный механизм базы данных SQL

SQLITE

Библиотека для работы с БД

SQLite входит в состав стандартной библиотеки Python

SQLITE

- SQLite библиотека для работы с базой данных.
- SQLite не требует наличия серверной программы для работы.
- SQLite — это облегченный язык запросов к базе данных SQL – для работы с SQLite нет необходимости устанавливать сервер, ожидающий запросы по какому-нибудь порту, т.е. SQLite работает с файлом базы данных напрямую.

ХАРАКТЕРИСТИКА. ПЛЮСЫ

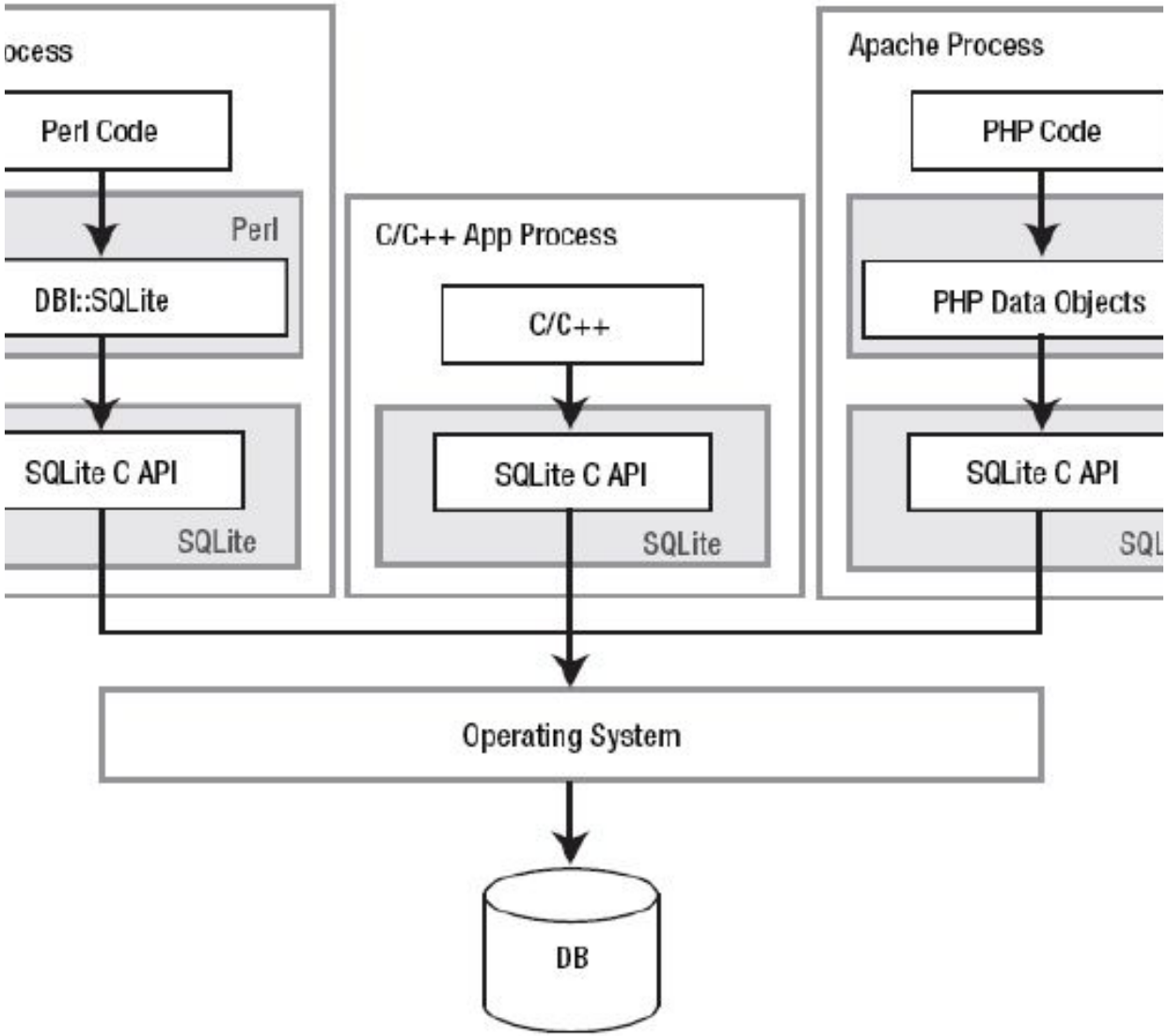
- не требуется сервер, все данные хранятся в одном файле;
- поддерживает полный набор SQL92, включая триггеры, индексы, автоинкрементные столбцы и др.;
- производительная, переносимая и надежная база данных;
- подойдет для маленьких не нагруженных сайтов, визиток, блогов, лендингов.

ХАРАКТЕРИСТИКА. МИНУСЫ:

- отсутствие авторизации при подключении. То есть, если кто-то знает где лежит ваша база и она при этом открыта для полного доступа, то он может считать все данные с этой базы.

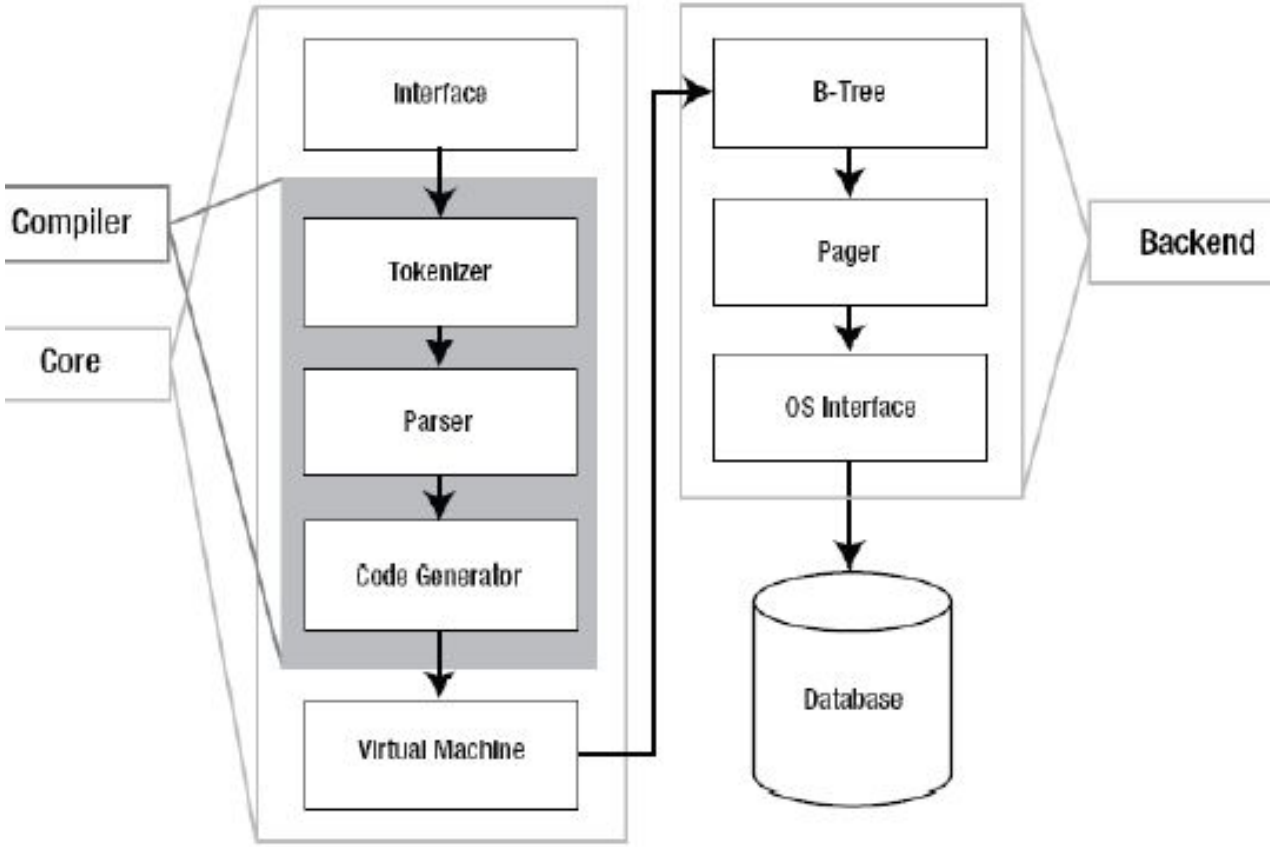
ВСТРАИВАЕМАЯ БАЗА ДАННЫХ

- Встраиваемость означает, что она существует не как процесс, отдельно от обслуживаемого процесса, а является частью некоторого прикладного приложения.
- Не требуются сетевые настройки, файерволы, пользователи и т.д., так как и клиент и сервер работают в одном пространстве, что избавляет от проблем конфигурирования.
- На текущем рынке встроенных баз данных существует много продуктов разных производителей, SQLite справляется с открытыми исходниками и не требует лицензионных сборов и спроектирована как встраиваемая БД.


ocess
Perl Code
Perl
DBI::SQLite
SQLite C API
SQLite
C/C++ App Process
C/C++
SQLite C API
SQLite
Apache Process
PHP Code
PHP Data Objects
SQLite C API
SQL
Operating System
DB

АРХИТЕКТУРА

- SQLite имеет модульную архитектуру, отображающую уникальные подходы к управлению реляционными базами данных.
  - 8 отдельных модулей разделяют обработку запроса на отдельные задачи.



ЯЗЫК SQL В SQLITE

- SQLite использует язык SQL, который является единственным и универсальным средством, позволяющим использовать реляционную базу данных.
- SQL спроектирован для структурирование, чтения, записи, сортирования, фильтрации, зашиты, генерации , группирование и управления информацией.

ЯЗЫК SQL

- **SQL** — «Структурированный язык запросов» — универсальный компьютерный язык, применяемый для создания, модификации и управления данными в реляционных базах данных.
- SQL основывается на исчислении кортежей.

ОПЕРАТОРЫ SQL

- Любой сервер SQL поддерживает четыре так называемых оператора манипулирования данными, и в целом эти операторы лежат в основе подавляющего большинства операций, выполняемых с реляционной базой данных.
- Этими четырьмя основными операторами базы данных являются SELECT, INSERT, UPDATE и DELETE. Операторы SQL, называемые также командами — очень удобны и позволяют выполнять практически все необходимые действия в базе данных.

ОСНОВНЫЕ ВОЗМОЖНОСТИ ЯЗЫКА SQL

- Выборка данных (извлечение из базы данных содержащейся в ней информации)
- Организация данных (определение структуры базы данных и установление отношений между её элементами)
- Обработка данных (добавление/изменение/удаление)
- Управление доступом (ограничение возможностей ряда пользователей на доступ к некоторым категориям данных, защита данных от несанкционированного доступа)
- Обеспечение целостности данных (защита данных от разрушения)
- Управление состоянием СУБД

Основные особенности этого языка SQL

- Язык SQL не чувствителен к регистру.
- Если пишется несколько операторов подряд, то в конце каждого ставится ";".
- Комментарии заключаются в "{ }" и "/* … */".
- Имена файлов, таблиц, полей, переменных нельзя писать по-русски и с пробелом

ТИПЫ ДАННЫХ

Объект	Тип
STRING	Строка и символ
BINARY	Бинарный объект
NUMBER	Число
DATETIME	Дата и время
ROWID	Идентификатор записи
None	NULL-значение (отсутствующее значение)

СОЗДАНИЕ БАЗЫ ДАННЫХ

- SQLite создает базу данных только после того, как пользователь создаст в базе данных таблицу.
- Для создания первой таблицы у пользователя есть возможность задать постоянные параметры для базы данных (например, таблица кодирования UTF-8).

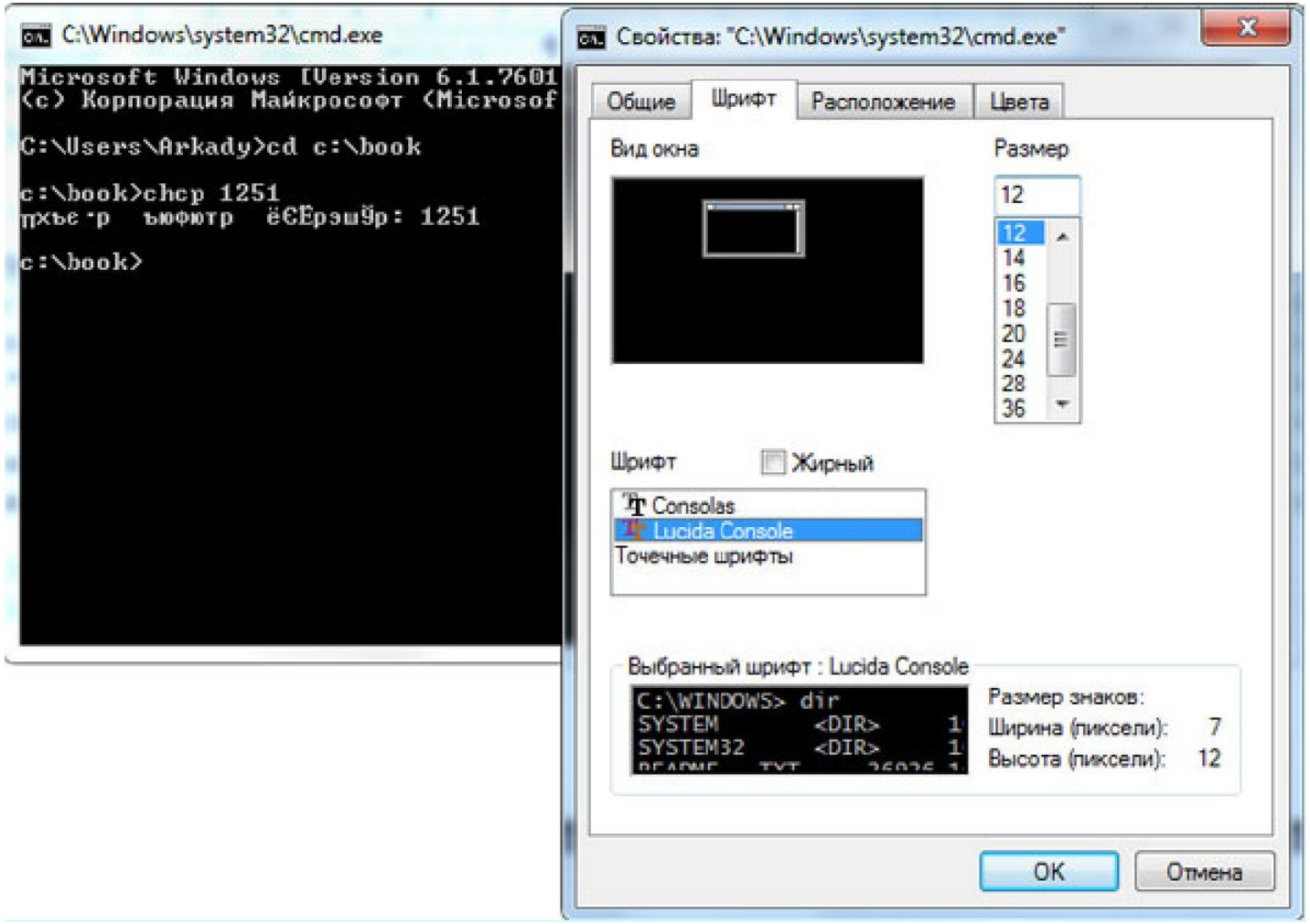
СОЗДАНИЕ БАЗЫ ДАННЫХ

```
cd C:\book
```

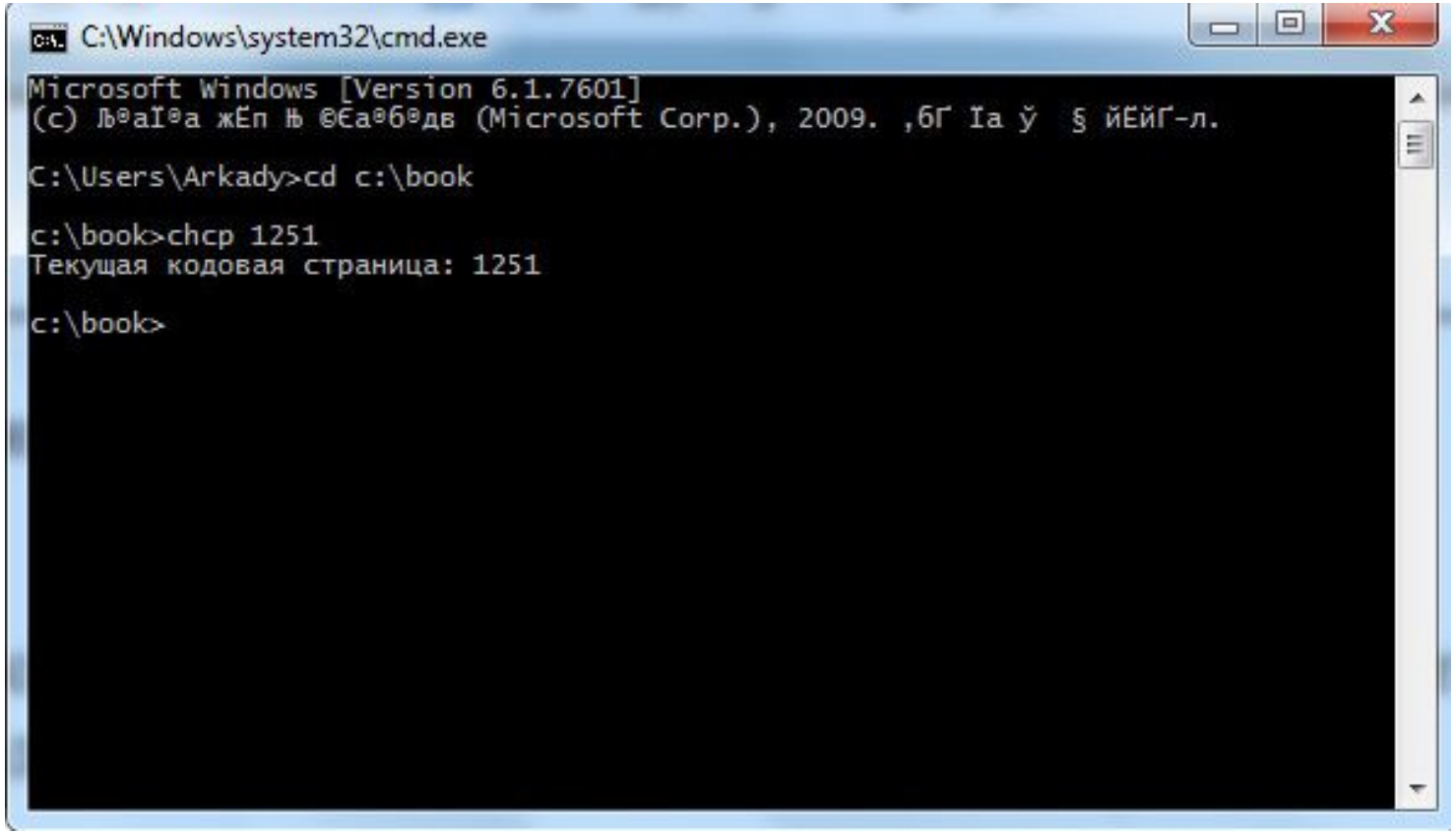
СОЗДАНИЕ БАЗЫ ДАННЫХ

- По умолчанию в консоли используется кодировка ср866. Чтобы сменить кодировку на ср1251, в командной строке вводим команду:
- chcp1251

СОЗДАНИЕ БАЗЫ ДАННЫХ

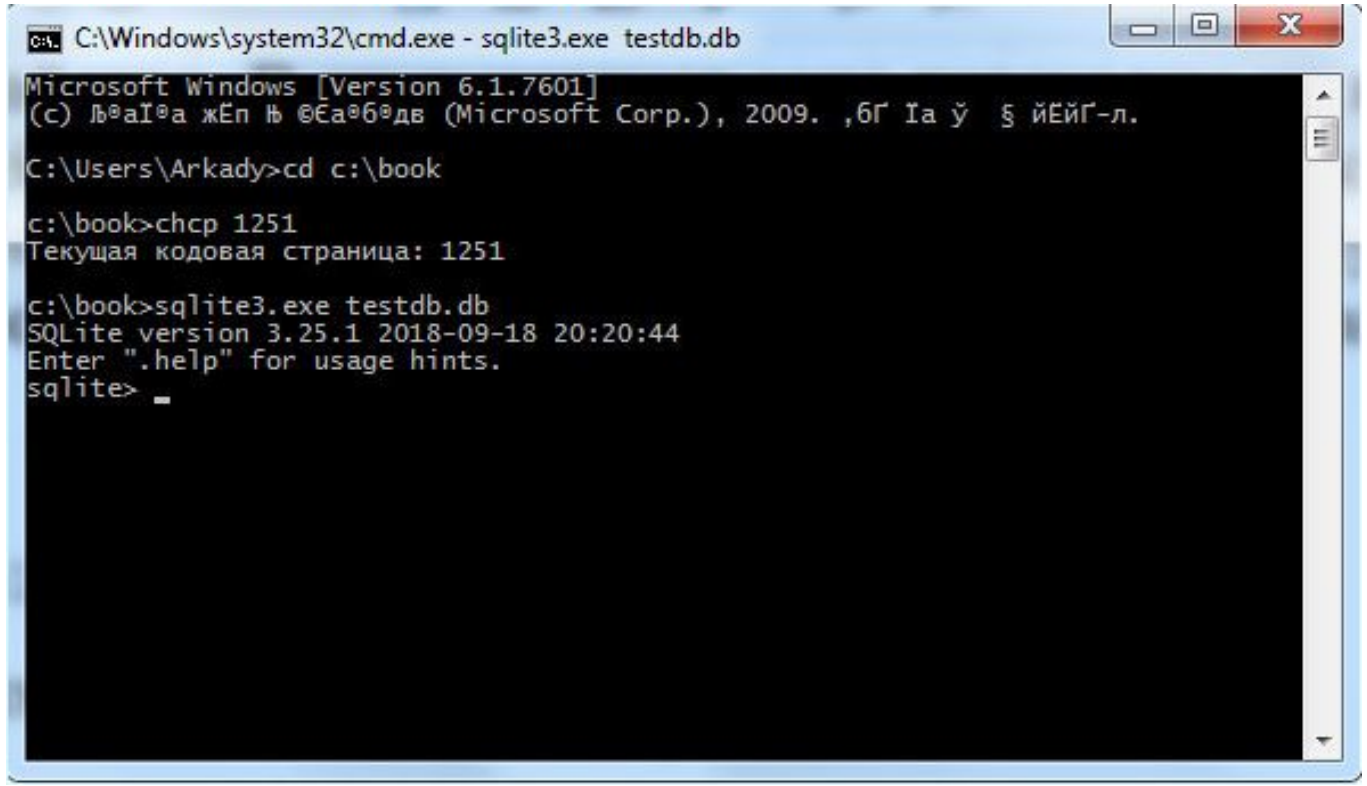


СОЗДАНИЕ БАЗЫ ДАННЫХ



СОЗДАНИЕ БАЗЫ ДАННЫХ

```
C:\book>sqlite3.exe testdb.db
```



SQL-СКРИПТ ДЛЯ СОЗДАНИЯ ТАБЛИЦЫ ОТДЕЛ

- **create table otdel**
- **(kod_otdela int not null auto_increment,**
- **primary key(kod_otdela),**
- **otdel varchar(50),**
- **nach varchar(50));**

CREATE TABLE <имя таблицы>
(<список вида <имя поля> <тип> [<размер>]>);

```
create [temp|temporary] table … ;
```

СОЗДАНИЕ ТАБЛИЦЫ БАЗЫ ДАННЫХ

```
CREATE TABLE 'user'
       ('id' INTEGER PRIMARY KEY  AUTOINCREMENT
       NOT NULL  UNIQUE ,
       'name' VARCHAR(20),
       'age' INT(3),
       'city' VARCHAR(50))")
```

- Данной инструкцией мы создаем таблицу user(id, name, age, city)
- Типы данных: integer, real, text, blob, NULL.
- Для редактирования таблицы используется инструкция:

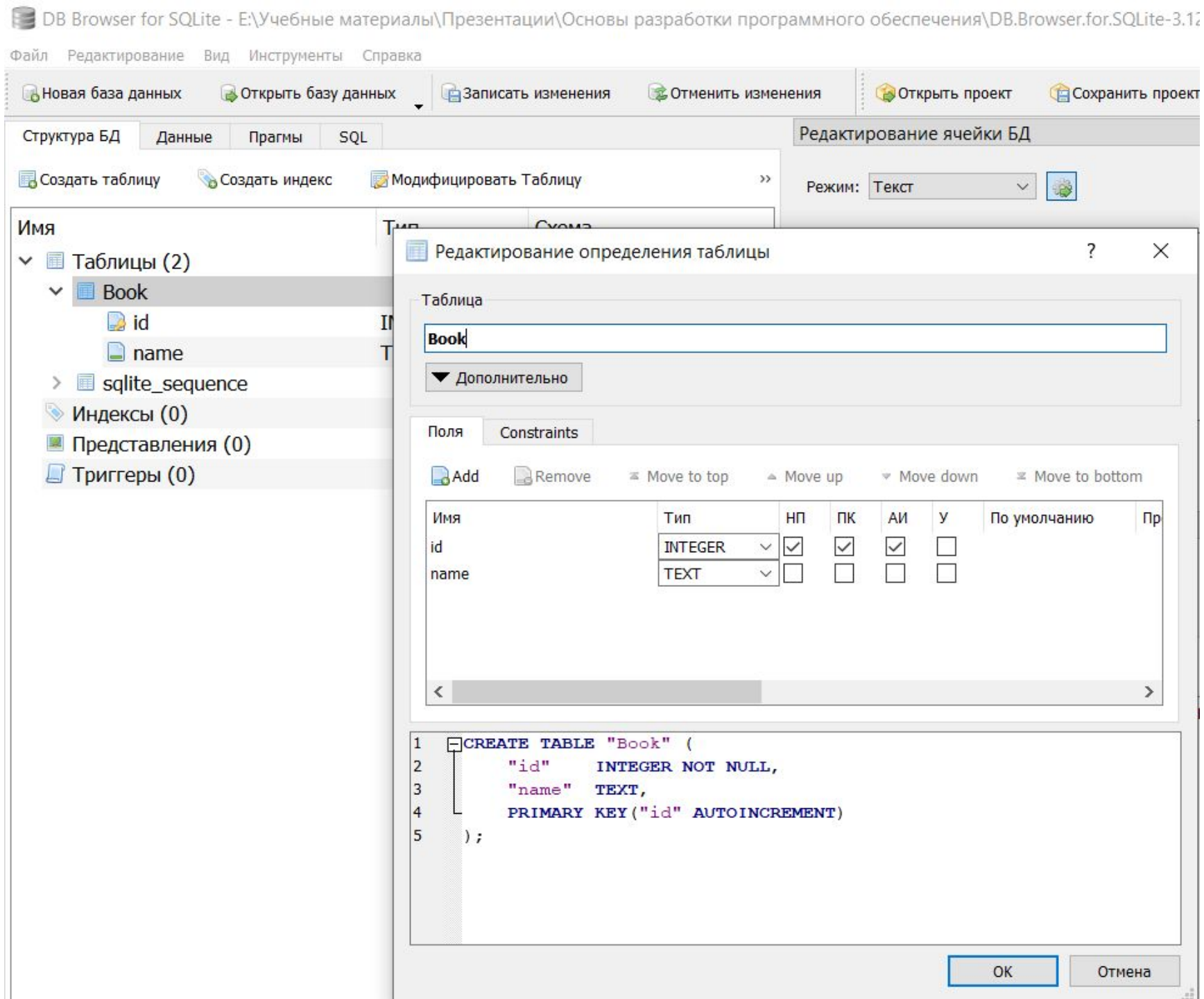
```
alter table table { rename to name | add column column_def }
```

ИЗМЕНЕНИЕ СТРУКТУРЫ ТАБЛИЦЫ

- ALTER TABLE <имя таблицы> <действие> <имя поля> <тип>
  - Пример:
- ALTER TABLE Pers Drop God, add year_b date

УДАЛЕНИЕ ТАБЛИЦЫ

- Drop <имя таблицы>


DB Browser for SQLite - E:\Учебные материалы\Презентации\Основы разработки программного обеспечения\DB.Browser.for.SQLite-3.12
Файл
Редактирование
Вид
Инструменты
Справка
Новая база данных
Открыть базу данных
Записать изменения
Отменить изменения
Открыть проект
Сохранить проект
Структура БД
Данные
Прагмы
SQL
Создать таблицу
Создать индекс
Модифицировать Таблицу
Имя
Тип
Схема
Таблицы (2)
Book
id
name
sqlite_sequence
Индексы (0)
Представления (0)
Триггеры (0)
Редактирование ячейки БД
Режим:
Текст
Редактирование определения таблицы
Таблица
Book
Дополнительно
Поля
Constraints
Add
Remove
Move to top
Move up
Move down
Move to bottom
Имя
Тип
НП
ПК
АИ
У
По умолчанию
id
INTEGER
name
TEXT
1 CREATE TABLE "Book" (
2 "id" INTEGER NOT NULL,
3 "name" TEXT,
4 PRIMARY KEY("id" AUTOINCREMENT)
5);
OK
Отмена

ЗАДАНИЕ

- Создайте базу данных db_**00**, где 00, это ваш номер
- Создайте таблицу Группа gr, sp, year_n
- Создайте таблицу Student с полями id, fam, name, year_b, gr
- Измените структуру, добавьте поле шифр специальности, удалите специальность, добавьте таблицу Специальность, подумайте какие там поля
- Добавьте связи между таблицами

ДОБАВЛЕНИЕ ДАННЫХ

Процесс ввода в таблицу базы данных новой информации обычно называется загрузкой данных.

Оператор INSERT заполняет таблицу данными.

INSERT into table_name (column1, column2, ...)
values (value1, value2...);

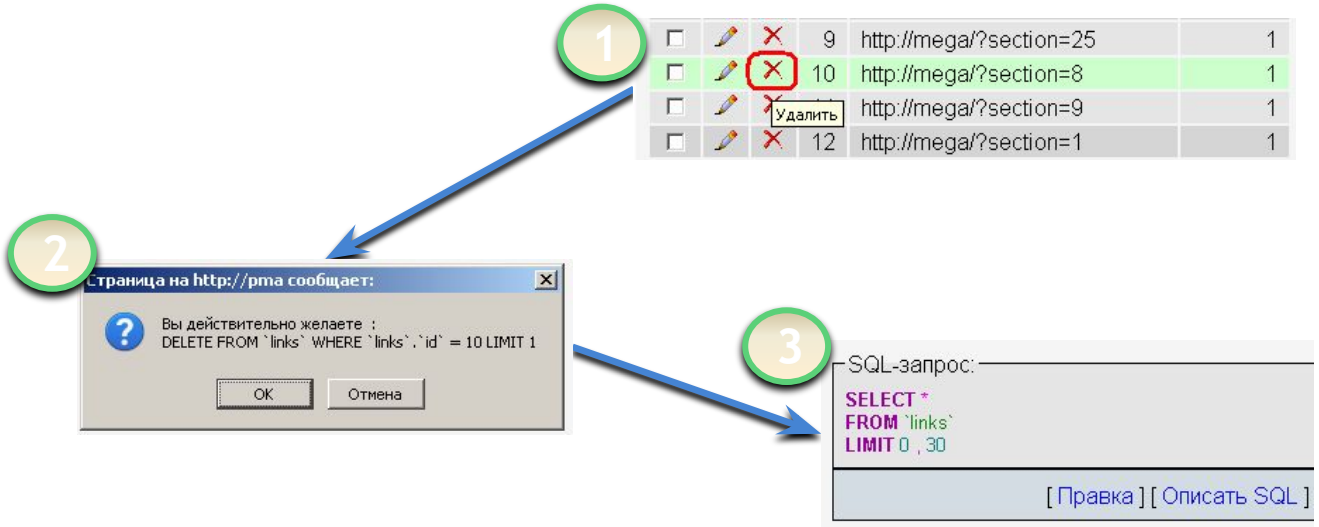
- **insert into otdel (otdel,nach)**
- **values ('ЦИТ','Иванов Иван Иванович');**

ВВОД ДАННЫХ

- Ввод данных в таблицу осуществляется командой:

```
INSERT INTO user VALUES (null,'Svetlana',18,'Novgorod’);
```

УДАЛЕНИЕ ДАННЫХ:



DELETE FROM `links` WHERE `links`.`id`=10 LIMIT 1;

ВЫБОРКА ДАННЫХ

- SELECT *имена полей (через ,)* | *| *DISTINCT*| *ALL*
- FROM *имя таблицы*

- *Обязательным* является только одно предложение — FROM без которого оператор SELECT не может работать.

СИНТАКСИС SELECT

- SELECT *поля*
  FROM *таблицы*
  [WHERE *условие выборки*]
  [GROUP BY *поле группировки*]
  [HAVING *условие для группы*]
  [ORDER BY *поле сортировки*]
  [LIMIT *количество записей*]

ЗАДАНИЕ УСЛОВИЙ ПРИ ВЫБОРКЕ ДАННЫХ

1. операторы сравнения (=, <>, >=, <=,>,<);
2. логические операторы (Is null, BETWEEN.. .AND, IN, LIKE, EXISTS, UNIQUE, ALL, ANY);
3. операторы объединения (AND, OR);
4. операторы отрицания (IS NOT NULL NOT BETWEEN NOT IN NOT LIKE NOT EXISTS NOT UNIQUE).

ОБНОВЛЕНИЕ ЗАПИСЕЙ

UPDATE *имя_таблицы*
SET
{ имя поля=значение} } [,...]
[WHERE условие_обновления]

Пример: Update Tovar Set Cena=Cena*1.1

ВЫВОД РЕЗУЛЬТИРУЮЩИХ ДАННЫХ

```
select * from food_types order by id limit 1 offset 1;
```

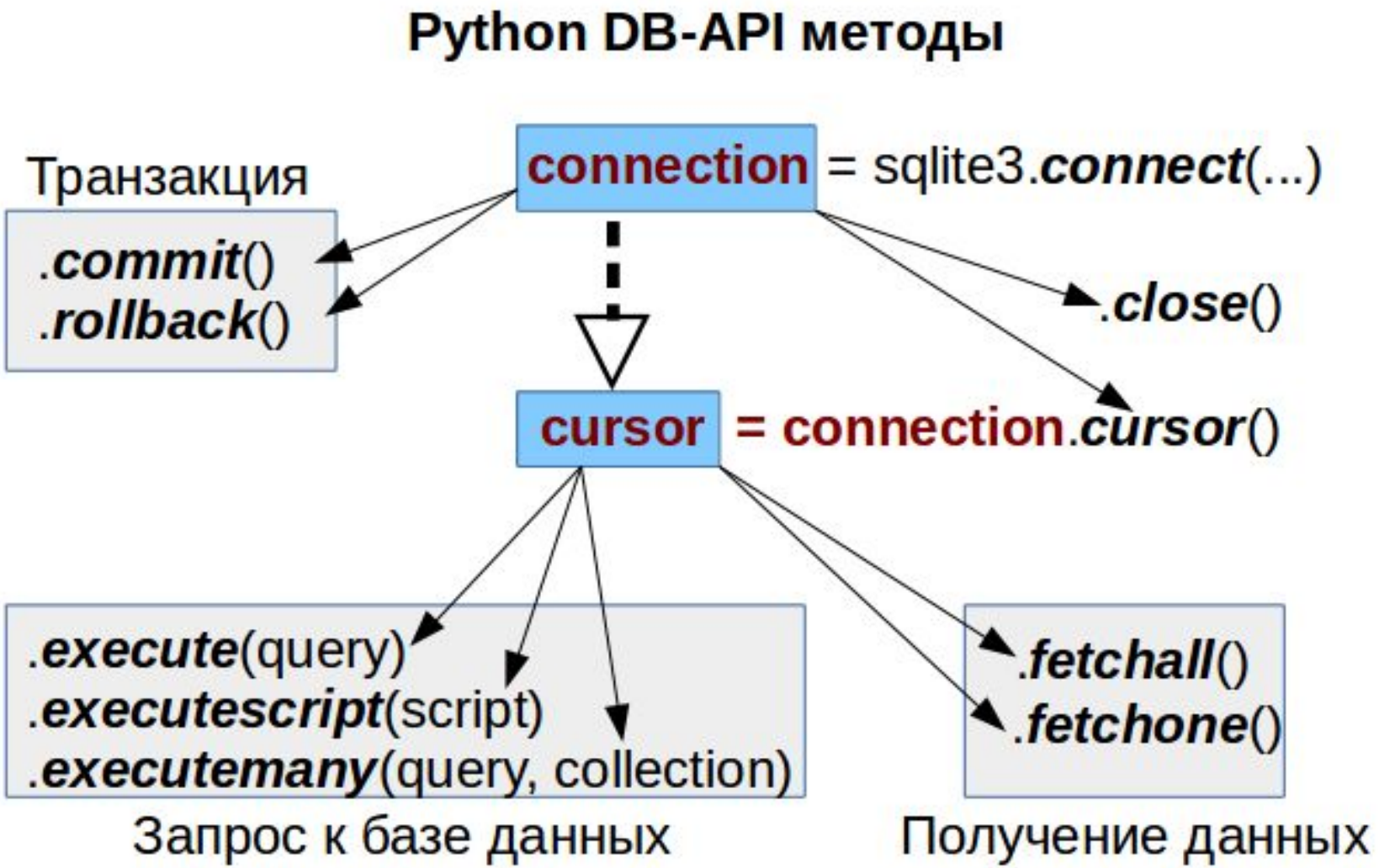
ЗАДАНИЕ

- Заполните таблицы записями
- Напишите запрос на изменение данных в таблице Студент, измените год рождения студента с определённым id
- Напишите запрос на удаление данных о студенте с определённым id
- Напишите запрос на выборку студентов с группы 1994
- Напишите запрос на нахождение студентов групп с шифром специальности, начинающимся на 09

Доступ к базе данных SQLite из Python

PYTHON DB-API

- **Python DB-API** – это не конкретная библиотека, а набор правил, которым подчиняются отдельные модули, реализующие работу с конкретными базами данных
- Отдельные нюансы реализации для разных баз могут отличаться, но общие принципы позволяют использовать один и тот же подход при работе с разными базами данных


Python DB-API методы
connection = sqlite3.connect(...)
Транзакция
.commit()
.rollback()
.close()
cursor = connection.cursor()
.execute(query)
.executescript(script)
.executemany(query, collection)
.fetchall()
.fetchone()
Запрос к базе данных
Получение данных

ИНТЕРФЕЙС МОДУЛЯ

- Доступ к базе данных осуществляется с помощью объекта-соединения (connection object). DB-API-совместимый модуль должен предоставлять функциюконструктор connect() для класса объектов-соединений.
- Конструктор должен иметь следующие именованные параметры:
  - dsn Название источника данных в виде строки
  - user Имя пользователя
  - password Пароль
  - host Адрес хоста, на котором работает СУБД
  - database Имя базы данных

ОБЪЕКТ-СОЕДИНЕНИЕ

- Объект-соединение, получаемый в результате успешного вызова функции connect(), должен иметь следующие методы:
  - close() Закрывает соединение с базой данных.
  - commit() Завершает транзакцию.
  - rollback() Откатывает начатую транзакцию (восстанавливает исходное состояние). Закрытие соединения при незавершенной транзакции автоматически производит откат транзакции.
  - cursor() Возвращает объект-курсор, использующий данное соединение.
- Если база данных не поддерживает курсоры, модуль сопряжения должен их имитировать

ТРАНЗАКЦИЯ

- Транзакция – это группа из одной или нескольких операций, которые изменяют базу данных
- Транзакция соответствует логически неделимой операции над базой данных, а частичное выполнение транзакции приводит к нарушению целостности БД
  - Например, при переводе денег с одного счета на другой операции по уменьшению первого счета и увеличению второго являются транзакцией.
- Методы commit() и rollback() обозначают начало и конец транзакции в явном виде
- Не все базы данных поддерживают механизм транзакций

ОБЪЕКТ-КУРСОР

- **Курсор** (от англ. cursor - CURrrent Set Of Records, текущий набор записей) служит для работы с результатом запроса. Результатом запроса обычно является одна или несколько прямоугольных таблиц со столбцами-полями и строками-записями
- Приложение может читать и обрабатывать полученные таблицы и записи в таблице по одной, поэтому в курсоре хранится информация о текущей таблице и записи
- Конкретный курсор в любой момент времени связан с выполнением одной SQL-инструкции

АТРИБУТЫ ОБЪЕКТА-КУРСОРА

- **arraysize** Атрибут, равный количеству записей, возвращаемых методом **fetchmany().** По умолчанию равен 1
- **callproc**(procname[, params]) Вызывает хранимую процедуру procname с параметрами из изменчивой последовательности params. Хранимая процедура может изменить значения некоторых параметров последовательности. Метод может возвратить результат, доступ к которому осуществляется через fetch - методы
- **close**() Закрывает объект-курсор
- **description** Этот доступный только для чтения атрибут является

последовательностью из семи элементных последовательностей (name, type_code, display_size, internal_size, precision, scale, null_ok)

АТРИБУТЫ ОБЪЕКТА-КУРСОРА

- **execute**(operation[, parameters]) Исполняет запрос к базе данных или команду СУБД.
- **executemany**(operation, seq_of_parameters) Выполняет серию запросов или команд, подставляя параметры в заданный шаблон.
- **fetchall**() Возвращает все (или все оставшиеся) записи результата запроса.

РАБОТА С БАЗОЙ ДАННЫХ

1. Подключение к базе данных (вызов connect() с получением объекта соединения)
2. Создание одного или нескольких курсоров (вызов метода объекта соединения cursor() с получением объекта-курсора)
3. Исполнение команды или запроса (вызов метода execute() или его вариантов)
4. Получение результатов запроса (вызов метода fetchone() или его вариантов)
5. Завершение транзакции или ее откат (вызов метода объекта-соединения commit() или rollback())
6. Когда все необходимые транзакции произведены, подключение закрывается вызовом метода close() объекта-соединения

IMPORT SQLITE3

- Python имеет встроенную поддержку SQLite базы данных, для этого не требуется ничего дополнительно устанавливать, достаточно в скрипте указать импорт стандартной библиотек

PYTHON DB-API МОДУЛИ В ЗАВИСИМОСТИ ОТ БАЗЫ ДАННЫХ

База данных	DB-API модуль
SQLite	sqlite3
PostgreSQL	psycopg2
MySQL	mysql.connector
ODBC	pyodbc

СОЕДИНЕНИЕ С БАЗОЙ, ПОЛУЧЕНИЕ КУРСОРА

```
# Импортируем библиотеку, соответствующую типу нашей базы данных
import sqlite3

# Создаем соединение с нашей базой данных
conn = sqlite3.connect('Chinook_Sqlite.sqlite')

# Создаем курсор - это специальный объект который делает запросы и получает их результаты
cursor = conn.cursor()

…

# Закрываем соединение с базой данных
conn.close()
```

КАК СОЗДАВАТЬ БАЗУ ДАННЫХ

```
import sqlite3

conn = sqlite3.connect("mydatabase.db")
cursor = conn.cursor()

# Создание таблицы
cursor.execute("""CREATE TABLE albums
                  (title text, artist text, release_date text,
                   publisher text, media_type text)
               """)
```

```
 RESTART: F:/Учебные материалы/Задания/ПОКС и Web-серверов/18-19/sqlite/create.p
y
Запрос успешно выполнен
```

```python
import sqlite3
con=sqlite3.connect("books.db")
cur=con.cursor()
sql="""\
create table user(
id_user integer primary key autoincrement,
email text,
pass text);
"""
try:
    cur.executescript(sql)
except sqlite3.DatabaseError as err:
    print("Ошибка:",err)
else:
    print("Запрос успешно выполнен")
cur.close()
con.close()
input()
```

ЗАПИСЬ В БАЗУ

```
# Делаем INSERT запрос к базе данных, используя обычный SQL-синтаксис
cursor.execute("insert into Artist values (Null, 'A Aagrh!') ")

# Если мы не просто читаем, но и вносим изменения в базу данных - необходимо сохранить транзакцию
conn.commit()

# Проверяем результат
cursor.execute("SELECT Name FROM Artist ORDER BY Name LIMIT 3")
results = cursor.fetchall()
print(results)
```

```
import sqlite3
con=sqlite3.connect("books.db")
cur=con.cursor()
sql="""\
insert into user(email,pass)
values ('loric23@yandex.ru','P@ssw0rd');
"""
try:
    cur.executescript(sql)
except sqlite3.DatabaseError as err:
    print("Ошибка:",err)
else:
    print("Запрос успешно выполнен")
cur.close()
con.close()
input()
```

РАЗБИВАЕМ ЗАПРОС НА НЕСКОЛЬКО СТРОК В ТРОЙНЫХ КАВЫЧКАХ

- Длинные запросы можно разбивать на несколько строк в произвольном порядке, если они заключены в тройные кавычки — одинарные (‘’’…’’’) или двойные (“””…”””)

```
cursor.execute("""
  SELECT name
  FROM Artist
  ORDER BY Name LIMIT 3
""")
```

ОБЪЕДИНЯЕМ ЗАПРОСЫ К БАЗЕ ДАННЫХ В ОДИН ВЫЗОВ МЕТОДА

- Метод курсора .execute() позволяет делать только один запрос за раз, при попытке сделать несколько через точку с запятой будет ошибка.

```
cursor.executescript("""
 insert into Artist values (Null, 'A Aagrh!');
 insert into Artist values (Null, 'A Aagrh-2!');
""")
```

ЧТЕНИЕ ИЗ БАЗЫ

```
# запрос к базе данных
cursor.execute("SELECT Name FROM Artist ORDER BY Name LIMIT 3")

# Получаем результат запроса
results = cursor.fetchall()
results2 =  cursor.fetchall()

print(results)
print(results2)
```

```
>>> con=sqlite3.connect("books.db")
>>> cur=con.cursor()
>>> cur.execute("select * from user")
<sqlite3.Cursor object at 0x03875C20>
>>> cur.fetchone()
(1, 'loric23@yandex.ru', 'P@ssw0rd')
```

ПОДСТАНОВКА ЗНАЧЕНИЯ В ЗАПРОС

- С подставновкой по порядку на места знаков вопросов:

```
cursor.execute("SELECT Name FROM Artist ORDER BY Name
LIMIT ?", ('2'))
```

- И с использованием именнованных замен:

```
cursor.execute("SELECT Name from Artist ORDER BY Name LIMIT
:limit", {"limit": 3})
```

ЗАДАНИЕ

- Напишите программу для подключения к базе данных и получение данных из таблиц

ЗАДАНИЕ

- Добавим логин и пароль в таблицу Student
- Напишем программу для регистрации/авторизации пользователя с проверкой, что он есть в базе данных (можно подумать о том, что пароль может быть зашифрован). Выполните задачу с графическим интерфейсом

ВЫВОДЫ

- Для Python разработан стандарт, называемый DB-API, которого должны придерживаться все разработчики модулей сопряжения с реляционными базами данных.
- Благодаря этому API код прикладной программы становится менее зависимым от марки используемой базы данных, его могут понять разработчики, использующие другие базы данных.
- DB-API описывает имена функций и классов, которые должен содержать модуль сопряжения с базой данных, и их семантику. Модуль сопряжения должен содержать класс объектов-соединений с базой данных и класс для курсоров - специальных объектов, через которые происходит коммуникация с СУБД на прикладном уровне.