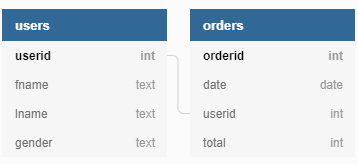
**Лекция 3**

**Загрузка данных из БД SQLite**

SQL и Python — обязательные инструменты для любого специалиста в сфере анализа данных. SQLite — это часть стандартного пакета Python 3, поэтому ничего дополнительно устанавливать не придется.

Создадим базу данных в SQLite с помощью Python, несколько таблиц и настроим отношения:



**Типы данных SQLite в Python**

SQLite для Python предлагает меньше типов данных, чем есть в других реализациях SQL. С одной стороны, это накладывает ограничения, но, с другой стороны, в SQLite многое сделано проще. Основные типы:

NULL — значение NULL

INTEGER — целое число

REAL — число с плавающей точкой

TEXT — текст

BLOB — бинарное представление крупных объектов, хранящееся в точности с тем, как его ввели

Для загрузки библиотеки нужно использовать следующую команду:

import sqlite3

Создание базы данных SQLite в Python

Есть несколько способов создания базы данных в Python с помощью SQLite. Для этого используется объект Connection, который и представляет собой базу. Он создается с помощью функции connect().

Создадим файл .db, поскольку это стандартный способ управления базой SQLite. Файл будет называться orders.db. За соединение будет отвечать переменная conn.

conn = sqlite3.connect('orders.db')

Эта строка создает объект connection, а также новый файл orders.db в рабочей директории. Если нужно использовать другую, ее нужно обозначить явно:

conn = sqlite3.connect(r'ПУТЬ-К-ПАПКИ/orders.db')

Если файл уже существует, то функция connect осуществит подключение к нему.

Символ «r» дает понять Python, что речь идет о «сырой» строке, где символы «/» не отвечают за экранирование.

Функция connect создает соединение с базой данных SQLite и возвращает объект, представляющий ее.

**Резидентная база данных**

Еще один способ создания баз данных с помощью SQLite в Python — создание их в **памяти**. Это отличный вариант для тестирования, ведь такие базы существуют только в оперативной памяти.

conn = sqlite3.connect(:memory:)

**Создание объекта cursor**

После создания объекта соединения с базой данных нужно создать объект cursor. Он позволяет делать SQL-запросы к базе. Используем переменную cur для хранения объекта:

cur = conn.cursor()

Теперь выполнять запросы можно следующим образом:

cur.execute("ВАШ-SQL-ЗАПРОС-ЗДЕСЬ;")

Обратите внимание на то, что сами запросы должны быть помещены в кавычки — это важно. Это могут быть одинарные, двойные или тройные кавычки. Последние используются в случае особенно длинных запросов, которые часто пишутся на нескольких строках.

**Создание таблиц в SQLite в Python**

Пришло время создать первую таблицу в базе данных. С объектами соединения (conn) и cursor (cur) это можно сделать. Будем следовать этой схеме.

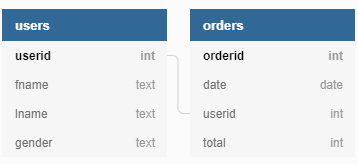


Таблица users.

cur.execute("""CREATE TABLE IF NOT EXISTS users(

userid INT PRIMARY KEY,

fname TEXT,

lname TEXT,

gender TEXT);

""")

conn.commit()

В коде выше выполняются следующие операции:

Функция execute отвечает за SQL-запрос

1. SQL генерирует таблицу users
2. IF NOT EXISTS поможет при попытке повторного подключения к базе данных. Запрос проверит, существует ли таблица. Если да — проверит, ничего ли не поменялось.
3. Создаем первые четыре колонки: userid, fname, lname и gender. Userid — это основной ключ.
4. Сохраняем изменения с помощью функции commit для объекта соединения.

Для создания второй таблицы просто повторим последовательность действий, используя следующие команды:

cur.execute("""CREATE TABLE IF NOT EXISTS orders(

orderid INT PRIMARY KEY,

date TEXT,

userid TEXT,

total TEXT);

""")

conn.commit()

После исполнения этих двух скриптов база данных будет включать две таблицы. Теперь можно добавлять данные.

**Добавление данных с SQLite в Python**

По аналогии с запросом для создания таблиц для добавления данных также нужно использовать объект cursor.

cur.execute("""INSERT INTO users(userid, fname, lname, gender)

VALUES('00001', 'Alex', 'Smith', 'male');""")

conn.commit()

В Python часто приходится иметь дело с переменными, в которых хранятся значения. Например, это может быть кортеж с информацией о пользователе.

user = ('00002', 'Lois', 'Lane', 'Female')

Если его нужно загрузить в базу данных, тогда подойдет следующий формат:

cur.execute("INSERT INTO users VALUES(?, ?, ?, ?);", user)

conn.commit()

В данном случае все значения заменены на знаки вопроса и добавлен параметр, содержащий значения, которые нужно добавить.

SQLite ожидает получить значения в формате кортежа. Однако в переменной может быть и список с набором кортежей. Таким образом можно добавить несколько пользователей:

more\_users = [('00003', 'Peter', 'Parker', 'Male'), ('00004', 'Bruce', 'Wayne', 'male')]

Но нужно использовать функцию executemany вместо обычной execute:

cur.executemany("INSERT INTO users VALUES(?, ?, ?, ?);", more\_users)

conn.commit()

Если применить execute, то функция подумает, то пользователь хочет передать в таблицу два объекта (два кортежа), а не два кортежа, каждый из которых содержит по 4 значения для каждого пользователя. Хотя в первую очередь вообще должна была возникнуть ошибка.

**Скрипты для загрузки данных**

Следующие скрипты можно скопировать и вставить для добавления данных в обе таблицы:

customers = [

('00005', 'Stephanie', 'Stewart', 'female'), ('00006', 'Sincere', 'Sherman', 'female'), ('00007', 'Sidney', 'Horn', 'male'),

('00008', 'Litzy', 'Yates', 'female'), ('00009', 'Jaxon', 'Mills', 'male'), ('00010', 'Paul', 'Richard', 'male'),

('00011', 'Kamari', 'Holden', 'female'), ('00012', 'Gaige', 'Summers', 'female'), ('00013', 'Andrea', 'Snow', 'female'),

('00014', 'Angelica', 'Barnes', 'female'), ('00015', 'Leah', 'Pitts', 'female'), ('00016', 'Dillan', 'Olsen', 'male'),

('00017', 'Joe', 'Walsh', 'male'), ('00018', 'Reagan', 'Cooper', 'male'), ('00019', 'Aubree', 'Hogan', 'female'),

('00020', 'Avery', 'Floyd', 'male'), ('00021', 'Elianna', 'Simmons', 'female'), ('00022', 'Rodney', 'Stout', 'male'),

('00023', 'Elaine', 'Mcintosh', 'female'), ('00024', 'Myla', 'Mckenzie', 'female'), ('00025', 'Alijah', 'Horn', 'female'),

('00026', 'Rohan', 'Peterson', 'male'), ('00027', 'Irene', 'Walters', 'female'), ('00028', 'Lilia', 'Sellers', 'female'),

('00029', 'Perla', 'Jefferson', 'female'), ('00030', 'Ashley', 'Klein', 'female')

]

orders = [

('00001', '2020-01-01', '00025', '178'), ('00002', '2020-01-03', '00025', '39'), ('00003', '2020-01-07', '00016', '153'),

('00004', '2020-01-10', '00015', '110'), ('00005', '2020-01-11', '00024', '219'), ('00006', '2020-01-12', '00029', '37'),

('00007', '2020-01-14', '00028', '227'), ('00008', '2020-01-18', '00010', '232'), ('00009', '2020-01-22', '00016', '236'),

('00010', '2020-01-26', '00017', '116'), ('00011', '2020-01-28', '00028', '221'), ('00012', '2020-01-31', '00021', '238'),

('00013', '2020-02-02', '00015', '177'), ('00014', '2020-02-05', '00025', '76'), ('00015', '2020-02-08', '00022', '245'),

('00016', '2020-02-12', '00008', '180'), ('00017', '2020-02-14', '00020', '190'), ('00018', '2020-02-18', '00030', '166'),

('00019', '2020-02-22', '00002', '168'), ('00020', '2020-02-26', '00021', '174'), ('00021', '2020-02-29', '00017', '126'),

('00022', '2020-03-02', '00019', '211'), ('00023', '2020-03-05', '00030', '144'), ('00024', '2020-03-09', '00012', '112'),

('00025', '2020-03-10', '00006', '45'), ('00026', '2020-03-11', '00004', '200'), ('00027', '2020-03-14', '00015', '226'),

('00028', '2020-03-17', '00030', '189'), ('00029', '2020-03-20', '00004', '152'), ('00030', '2020-03-22', '00026', '239'),

('00031', '2020-03-23', '00012', '135'), ('00032', '2020-03-24', '00013', '211'), ('00033', '2020-03-27', '00030', '226'),

('00034', '2020-03-28', '00007', '173'), ('00035', '2020-03-30', '00010', '144'), ('00036', '2020-04-01', '00017', '185'),

('00037', '2020-04-03', '00009', '95'), ('00038', '2020-04-06', '00009', '138'), ('00039', '2020-04-10', '00025', '223'),

('00040', '2020-04-12', '00019', '118'), ('00041', '2020-04-15', '00024', '132'), ('00042', '2020-04-18', '00008', '238'),

('00043', '2020-04-21', '00003', '50'), ('00044', '2020-04-25', '00019', '98'), ('00045', '2020-04-26', '00017', '167'),

('00046', '2020-04-28', '00009', '215'), ('00047', '2020-05-01', '00014', '142'), ('00048', '2020-05-05', '00022', '173'),

('00049', '2020-05-06', '00015', '80'), ('00050', '2020-05-07', '00017', '37'), ('00051', '2020-05-08', '00002', '36'),

('00052', '2020-05-10', '00022', '65'), ('00053', '2020-05-14', '00019', '110'), ('00054', '2020-05-18', '00017', '36'),

('00055', '2020-05-21', '00008', '163'), ('00056', '2020-05-24', '00024', '91'), ('00057', '2020-05-26', '00028', '154'),

('00058', '2020-05-30', '00022', '130'), ('00059', '2020-05-31', '00017', '119'), ('00060', '2020-06-01', '00024', '137'),

('00061', '2020-06-03', '00017', '206'), ('00062', '2020-06-04', '00013', '100'), ('00063', '2020-06-05', '00021', '187'),

('00064', '2020-06-09', '00025', '170'), ('00065', '2020-06-11', '00011', '149'), ('00066', '2020-06-12', '00007', '195'),

('00067', '2020-06-14', '00015', '30'), ('00068', '2020-06-16', '00002', '246'), ('00069', '2020-06-20', '00028', '163'),

('00070', '2020-06-22', '00005', '184'), ('00071', '2020-06-23', '00022', '68'), ('00072', '2020-06-27', '00013', '92'),

('00073', '2020-06-30', '00022', '149'), ('00074', '2020-07-04', '00002', '65'), ('00075', '2020-07-05', '00017', '88'),

('00076', '2020-07-09', '00007', '156'), ('00077', '2020-07-13', '00010', '26'), ('00078', '2020-07-16', '00008', '55'),

('00079', '2020-07-20', '00019', '81'), ('00080', '2020-07-22', '00011', '78'), ('00081', '2020-07-23', '00026', '166'),

('00082', '2020-07-27', '00014', '65'), ('00083', '2020-07-30', '00021', '205'), ('00084', '2020-08-01', '00026', '140'),

('00085', '2020-08-05', '00006', '236'), ('00086', '2020-08-06', '00021', '208'), ('00087', '2020-08-07', '00021', '169'),

('00088', '2020-08-08', '00004', '157'), ('00089', '2020-08-11', '00017', '71'), ('00090', '2020-08-13', '00025', '89'),

('00091', '2020-08-16', '00014', '249'), ('00092', '2020-08-18', '00012', '59'), ('00093', '2020-08-19', '00013', '121'),

('00094', '2020-08-20', '00025', '179'), ('00095', '2020-08-22', '00017', '208'), ('00096', '2020-08-26', '00024', '217'),

('00097', '2020-08-28', '00004', '206'), ('00098', '2020-08-30', '00017', '114'), ('00099', '2020-08-31', '00017', '169'),

('00100', '2020-09-02', '00022', '226')

]

Используйте следующие запросы:

cur.executemany("INSERT INTO users VALUES(?, ?, ?, ?);", customers)

cur.executemany("INSERT INTO orders VALUES(?, ?, ?, ?);", orders)

conn.commit()

**Получение данных с SQLite в Python**

**Использование fetchone() в SQLite в Python**

Начнем с использования функции fetchone(). Создадим переменную one\_result для получения только одного результата:

cur.execute("SELECT \* FROM users;")

one\_result = cur.fetchone()

print(one\_result)

Она вернет следующее:

[(1, 'Alex', 'Smith', 'male')]

**Использование fetchmany() в SQLite в Python**

Если же нужно получить много данных, то используется функция fetchmany(). Выполним другой скрипт для генерации 3 результатов:

cur.execute("SELECT \* FROM users;")

three\_results = cur.fetchmany(3)

print(three\_results)

Он вернет следующее:

[(1, 'Alex', 'Smith', 'male'), (2, 'Lois', 'Lane', 'Female'), (3, 'Peter', 'Parker', 'Male')]

**Использование fetchall() в SQLite в Python**

Функцию fetchall() можно использовать для получения всех результатов. Вот что будет, если запустить скрипт:

cur.execute("SELECT \* FROM users;")

all\_results = cur.fetchall()

print(all\_results)

**Удаление данных в SQLite в Python**

Теперь рассмотрим процесс удаления данных с SQLite в Python. Здесь та же структура. Предположим, нужно удалить любого пользователя с фамилией «Parker». Напишем следующее:

cur.execute("DELETE FROM users WHERE lname='Parker';")

conn.commit()

Если затем сделать следующей запрос:

cur.execute("select \* from users where lname='Parker'")

print(cur.fetchall())

Будет выведен пустой список, подтверждающий, что запись удалена.

**Объединение таблиц в SQLite в Python**

Наконец, посмотрим, как использовать объединение данных для более сложных запросов. Предположим, нужно сгенерировать запрос, включающий имя и фамилию каждого покупателя заказа.

Для этого напишем следующее:

cur.execute("""SELECT \*, users.fname, users.lname FROM orders

LEFT JOIN users ON users.userid=orders.userid;""")

print(cur.fetchall())