Программирование и разработка веб-приложений

Использование Python для работы с базой данных

SQLite

Несколько таблиц

Новая БД

```
import sqlite3
import os
import datetime

db_path='e:/sqlite/'
db_file='db11.db'
full_path=os.path.join(db_path,db_file)

print(sqlite3.apilevel)
print(full_path)
con=sqlite3.connect(full_path)
con.close()
```

Создание таблиц

```
sql='''\
        CREATE TABLE IF NOT EXISTS author(
                id_author INTEGER PRIMARY KEY,
                name_author TEXT,
                descr_author TEXT
        );
        CREATE TABLE publication(
                id_publication INTEGER PRIMARY KEY,
                name_publication TEXT
        );
        CREATE TABLE books(
                id_book INTEGER PRIMARY KEY,
                id_author INTEGER,
                id_publication INTEGER,
                title_book TEXT,
                descr_book TEXT,
                number_book INTEGER
        );
        '''
```

```
con=sqlite3.connect(full_path)
cur=con.cursor()
cur.executescript(sql)
cur.close()
con.close()
```

Заполнение автора

```
con=sqlite3.connect(full_path)
cur=con.cursor()
sql='''\
        INSERT INTO author (name_author, descr_author)
        VALUES ("Чуковский", "Автор множества книг для детей")
        '''
cur.execute(sql)
con.commit()        # Завершаем транзакцию
cur.close()         # Закрываем объект-курсор
con.close()         # Закрываем соединение
```

Заполнение типов публикаций с использованием кортежей и словаря

```
con=sqlite3.connect(full_path)
cur=con.cursor()

var1=("Роман",)
var2=(2,"Рассказ")
var3={"id":3, "name":"Стихотворение"}

sql1='''INSERT INTO publication (name_publication) VALUES(?)'''
sql2='''INSERT INTO publication VALUES (?,?)'''
sql3='''INSERT INTO publication VALUES (:id, :name)'''
```

```
cur.execute(sql1,var1)
cur.execute(sql2,var2)
cur.execute(sql3,var3)
con.commit()
cur.close()
con.close()
```

Создание и использование списка из кортежей для заполнения

```
var_list=[
                (1,1,"Айболит","Добрый доктор",100),
                (1,2, "Бармалей", "Злой разбойник",200),
                (1,3, "Тяни-толкай","Непонятоное существо",200)
]

sql='''\
        INSERT INTO books(id_author, id_publication, title_book, descr_book, number_book)
        VALUES (?,?,?,?,?)
        '''
cur.executemany(sql,var_list)
```

Исполнение и просмотр числа изменений

```
con.execute('''
            UPDATE publication SET name_publication='роман в стихах'
            WHERE id_publication = 3
            ''')
cur.execute ('''
            INSERT INTO publication (name_publication)
            VALUES ('поэма для детей')
            ''')
print(con.total_changes)
```

Вывод элементов по одному

```
print(cur.execute('''
    SELECT * FROM books
    '''))
print(cur.fetchone())
print(cur.fetchone())
print(cur.fetchone())
print(cur.fetchone())
print(cur.fetchone())
```

```
C:\Users\shtennikov>python G:\SQLite\sqlite17.py
<sqlite3.Cursor object at 0x005461E0>
(1, 1, 1, 'Айболит', 'Добрый доктор', 100)
(2, 1, 2, 'Бармалей', 'Злой разбойник', 200)
(3, 1, 3, 'Тяни-толкай', 'Непонятоное существо', 200)
None
None
Ok
0
```

Использование метода итератора __next__()

```
print(cur.execute('''SELECT * FROM books'''))
print(cur.__next__())
print(cur.__next__())
print(cur.__next__())
print(cur.__next__())
print(cur.__next__())
```

```
C:\Users\shtennikov>python G:\SQLite\sqlite18.py
<sqlite3.Cursor object at 0x01E861E0>
(1, 1, 1, 'Айболит', 'Добрый доктор', 100)
(2, 1, 2, 'Бармалей', 'Злой разбойник', 200)
(3, 1, 3, 'Тяни-толкай', 'Непонятоное существо', 200)
Traceback (most recent call last):
  File "G:\SQLite\sqlite18.py", line 24, in <module>
    print(cur.__next__())
StopIteration
```

Вывод через for

```
print(cur.execute('''SELECT * FROM books'''))
for i in cur:
    print("{o}".format(i))
```

```
C:\Users\shtennikov>python G:\SQLite\sqlite19.py
<sqlite3.Cursor object at 0x003861A0>
(1, 1, 1, 'Айболит', 'Добрый доктор', 100)
(2, 1, 2, 'Бармалей', 'Злой разбойник', 200)
(3, 1, 3, 'Тяни-толкай', 'Непонятоное существо', 200)
Ok
0
```

Использование fetchmany

```
print(cur.execute('''SELECT * FROM books'''))
print(cur.arraysize)
print(cur.fetchmany())
print(cur.fetchmany())
print(cur.fetchmany())
print(cur.fetchmany())
```

```
C:\Users\shtennikov>python G:\SQLite\sqlite20.py
<sqlite3.Cursor object at 0x006D61E0>
1
[(1, 1, 1, 'Айболит', 'Добрый доктор', 100)]
[(2, 1, 2, 'Бармалей', 'Злой разбойник', 200)]
[(3, 1, 3, 'Тяни-толкай', 'Непонятоное существо', 200)]
[]
Ok
0
```

Сравнение fetch

```
cursor.execute("SELECT id, name FROM `table`")
for i in range(cursor.rowcount):
   id, name = cursor.fetchone()
   print id, name
```

```
cursor.execute("SELECT id, name FROM `table`")
result = cursor.fetchmany()
while result:
   for id, name in result:
      print (id, name)
   result = cursor.fetchmany()
```

```
cursor.execute("SELECT id, name FROM `table`")
for id, name in cursor.fetchall():
   print (id, name)
```

Размер массива объектов

print(cur.execute('''SELECT * FROM books'''))

print(cur.**arraysize**)

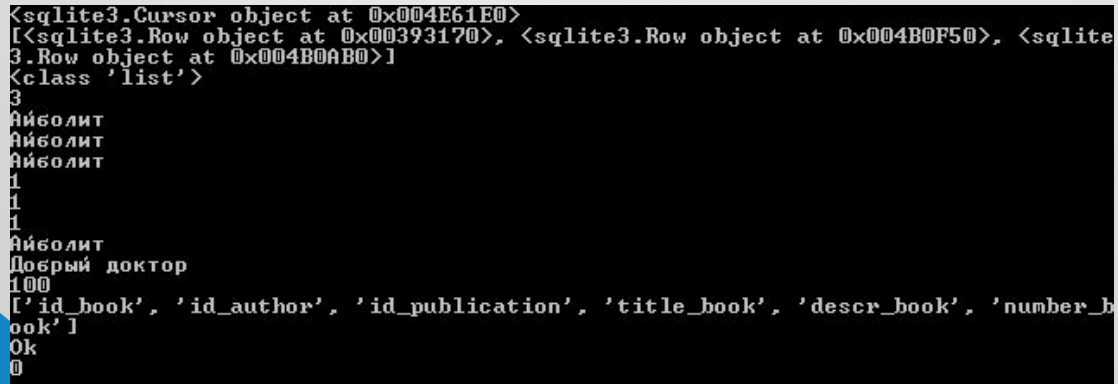
print(cur.fetchall())

print(cur.**fetchall**())# второй fetchall выводит пустой список

```
C:\Users\shtennikov>python G:\SQLite\sqlite21.py
<sqlite3.Cursor object at 0x021761E0>
1
[(1, 1, 1, 'Айболит', 'Добрый доктор', 100), (2, 1, 2, 'Бармалей', 'Злой разбойн
ик', 200), (3, 1, 3, 'Тяни-толкай', 'Непонятоное существо', 200)]
[]
Ok
0
```

row_factory с Row
и обращение по индексам и ключам

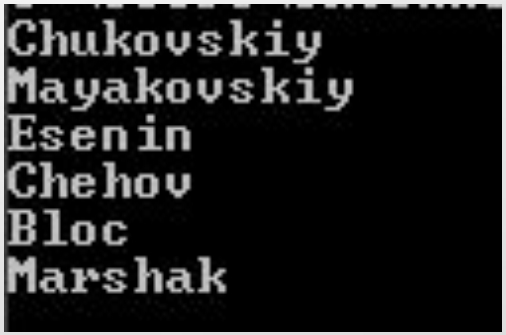
```
con.row_factory=sqlite3.Row
a_list=cur.fetchall()
print(a_list)
print(type(a_list))
print(len(a_list))
print(a_list[0][3])
print(a_list[0]['title_book'])
print(a_list[0]['TITLE_BOOK'])
for i in a_list[0]:
        print(i)
        print(a_list[0].keys())
```



row_factory

Существует возможность изменить это свойство на имя функции, которая принимает курсор (cur) и исходную строку как кортеж и возвращает измененную строку результата. Это позволяет, например, получить доступ к столбцам по имени

```
def dict_factory(cursor, row):
    d = {}
    for i, col_name in enumerate(cursor.description):
        d[col_name[0]] = row[i]
    return d
con.row_factory = dict_factory
cur = con.cursor()
cur.execute(''' SELECT * FROM author''')
for i in cur.fetchall():
    print(i["author_name"])
cur.close()
con.close()
```



```
def dict_factory(cursor, row):
    d = {}
    for i, col_name in enumerate(cursor.description):
        d[col_name[0]] = row[i]
        d[i]=row[i]
    return d

con.row_factory = dict_factory
cur = con.cursor()
cur.execute('"SELECT * FROM author"')
for i in cur.fetchall():
    print(i[0], i["author_name"])
```

con.row_factory = sqlite3.Row

Поддерживает отображающийся доступ по имени столбца, индексу, итерации, представлению,. Если два объекта Row имеют точно такие же столбцы и их элементы равны, то они сравниваются как равные.

```
con.row_factory = sqlite3.Row
cur = con.cursor()
cur.execute(''' SELECT * FROM author''')
a_list=cur.fetchall()
print(type(a_list[0]))
print(len(a_list[0]))
print(a_list[0])
print(a_list[0]['author_name'])
print(a_list[0].keys())
```

```
<class 'sqlite3.Row'>
3
<sqlite3.Row object at 0x001E3170>
Chukovskiy
['id_author', 'author_name', 'author_descr']
```

text_factory
работа с кодировками и обработка текста

con.text_factory=bytes # представление в байтовом виде

print(cur.execute('''SELECT * FROM books '''))

print(cur.fetchone())

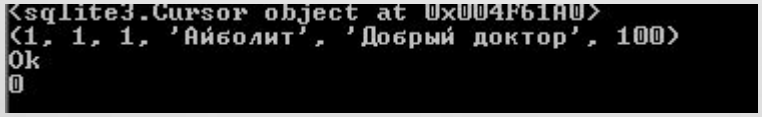
```
<sqlite3.Cursor object at 0x022361A0>
(1, 1, 1, b'\xd0\x90\xd0\xb9\xd0\xb1\xd0\xbe\xd0\xbb\xd0\xb8\xd1\x82', b'\xd0\x9
4\xd0\xbe\xd0\xb1\xd1\x80\xd1\x8b\xd0\xb9 \xd0\xb4\xd0\xbe\xd0\xba\xd1\x82\xd0\x
be\xd1\x80', 100)
Ok
0
```

Пользовательская text_factory

con.**text_factory**=lambda s: str(s, "utf-8")

print(cur.execute("SELECT * FROM books"))

print(cur.fetchone())



```
def myo1(input):
    return input.decode('cp1251')
con.text_factory = myo1
```

```
Pisatel Poet Poet Pisatel Poet Poet
```

text_factory

Свойство используется для управления возвращаемыми текстовыми значениями

Удаление записей

```
>>> cur.execute("INSERT INTO author VALUES
(NULL,'Маршак','Автор стихов')")
<sqlite3.Cursor object at 0x1007ebf80>

>>> cur.execute("DELETE FROM author WHERE
name_author='Маршак'")
<sqlite3.Cursor object at 0x1007ebf80>
>>> cur.fetchall()
[]

>>> cur.execute("SELECT * FROM author")
<sqlite3.Cursor object at 0x1007ebf80>
>>> cur.fetchall()
[(1, 'Чуковский', 'Автор множества книг для детей')]
```

Отмена действий rollback()

```
var1=("Маршак",)
sql1="'INSERT INTO author (name_author) VALUES(?) '"
cur.execute(sql1,var1)
cur.execute("SELECT * FROM author")
print(cur.fetchall())
con.rollback()
cur.execute("SELECT * FROM author")
print(cur.fetchall())
```

Получение состояния транзакции

con.in_transaction – True – есть активная транзакция,
False - нет такой