Программирование и разработка веб-приложений

Использование Python для работы с базой данных

SQLite

Одна таблица

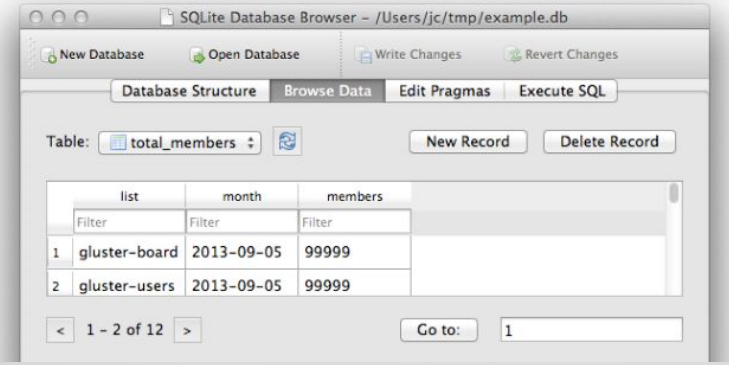
sqlitebrowser.org

News

2016-12-17 - The v3.9.1 binary for OSX has been rebuilt using Qt 5.7.1, to fix an important colour display problem on macOS Sierra.
2016-12-15 - An initial DBHub.io server is online, running our latest development code. Testing and feedback is encouraged.
Note - The data on this server will probably be wiped/reset every few days.

2016-10-20 - PortableApp version of DB4S 3.9.1 now available. Thanks John! :)

Screenshot



Download 32-bit Windows .exe

Download 64-bit Windows .exe

Download PortableApp

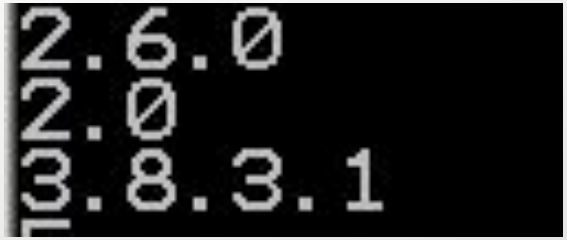
Download Mac .dmg

Download source .zip

Download .tar.gz file

Работа с SQLite

```
import sqlite3
print(sqlite3.version)
print(sqlite3.apilevel)
print(sqlite3.sqlite_version)
```



Открытие базы (создание) и закрытие

```
import sqlite3
import os
db_path='e://sqlite//'
db_file='db01.db'
full_path=os.path.join(db_path,db_file)
con=sqlite3.connect(full_path)
#Соединение с базой данных. connect() возвращает объект соединения.
con.close()
#Закрытие соединения
```

Объект Курсор (Cursor, cur)

- cur = con.**cursor**()

Создание и удаление таблицы

cur.execute('**DROP TABLE IF EXISTS** books') #удаление в случае если таблица существует

cur.execute('**CREATE TABLE** books (id **INTEGER PRIMARY KEY**,title **VARCHAR**(30), author VARCHAR(30), pers **TEXT**)')

cur.execute('**CREATE TABLE IF NOT EXISTS** books (id INTEGER PRIMARY KEY,title VARCHAR(30), author VARCHAR(30), pers TEXT)')

#создание таблицы, если она не была создана ранее

Небольшое пояснение

execute - выполнение запроса

INTEGER PRIMARY KEY прибавляется автоматически.

INTEGER PRIMARY KEY AUTOINCREMENT используют разные алгоритмы для создания ID

Заполнение данных в таблице

cur.**execute**('**INSERT INTO** books (id, author,pers) **VALUES** (NULL, "Aibolit", "Doctor Aibolit")')

con.**commit**()

author='Barmaley'

pers='Zloy razboinik'

cur.execute('INSERT INTO books VALUES **(NULL,?,?)',(author,pers)**)

con.**commit**()

Вывод номера последней записи и вывод всех записей

```
print (cur.lastrowid)
#2
cur.execute('SELECT * FROM books')
<sqlite3.Cursor object at 0x02E262A0>
print(cur.fetchall())
```

```
2
[(1, 'Aibolit', 'Doctor Aibolit'), (2, 'Barmaley', 'Zloy razboinik')]
```

Просмотр по элементам

```
cur.execute('SELECT * FROM books')
for i in cur:
    print('*'*10)
    print('id: ',i[0])
    print('author: ',i[1])
    print('pers: ',i[2])
```

Или сразу через cur.**fetchall**()

```
cur=con.cursor()
cur.execute('''
    SELECT * FROM books
    ''')
for i in cur.fetchall():
    print(i)
```

Просмотр по элементам через дополнительный объект

```
rows= cur.fetchall()
for i in rows:
    print('*'*10)
    print('id: ',i[0])
    print('author: ',i[1])
    print('pers: ',i[2])
```

Выбор по id из словаря

```
cur.execute('SELECT author,pers FROM books WHERE id=:id',{'id':id})

rows= cur.fetchall()

for row in rows:

    print (row)
```

Выполнение запросов – SQL прописан отдельно

```
import sqlite3
con=sqlite3.connect('catalog.db')
cur=con.cursor()

sql='''
CREATE TABLE user (
        id_user INTEGER PRIMARY KEY AUTOINCREMENT,
        user_name TEXT,
        user_dict TEXT
);
'''
cur.executescript(sql)

cur.close()
con.close()
```

```
con=sqlite3.connect(full_path)
cur=con.cursor()
#CREATE TABLE
#CREATE TABLE IF NOT EXISTS
cur.execute('''CREATE TABLE IF NOT EXISTS books(

                    id_books INTEGER PRIMARY KEY,
                    author_book TEXT,
                    title_book TEXT,
                    publish_book DATE,
                    age_book DATE,
                    keyword_book TEXT,
                    value_book REAL
                    )''')
con.commit()
cur.close()
con.close()
```

SQL в cur.execute

```
con=sqlite3.connect(full_path)
con.execute('''CREATE TABLE IF NOT EXISTS books(

                id_books INTEGER PRIMARY KEY,
                author_book TEXT,
                title_book TEXT,
                publish_book DATE,
                age_book DATE,
                keyword_book TEXT,
                value_book REAL
                )''')
```

SQL в con.execute

```
con.commit()
#cur.close()
con.close()
```

Обновление значений полей

id=6

cur.execute('**UPDATE** books SET pers=? WHERE id=?', ('Dochka melnika',id))

print(cur.rowcount)

Использование row_factory

```
con.row_factory=sqlite3.Row

cur=con.cursor()
cur.execute('''
        SELECT * FROM books
        ''')
#for i in cur.fetchall():
#       print(i)

#for i in cur.fetchall():
#       print(i['title_book'],i['author_book'])

for i in cur.fetchall():
        id_books, author_book, title_book, publish_book, age_book, keyword_book, value_book =i
        print(title_book,author_book)
```

вывод последнего id

```
print(cur.lastrowid)
```

create_function

```
import sqlite3
import md5
def md5sum(t):
        return md5.md5(t).hexdigest()
con = sqlite3.connect("my.db")
con.create_function("md5", 1, md5sum)
cur = con.cursor()
cur.execute("select md5(?)", ("foo",))
print (cur.fetchone()[0])
```

cbd18db4cc2f85cedef654fccc4a4d8

create_aggregate

```
class MySum (object):
        def __init__(self):
                self.count = 0
        def step(self, value):
                self.count += value
        def finalize(self):
                return self.count
con = sqlite3.connect("my.db")
con.create_aggregate("mysum", 1, MySum)
cur = con.cursor()
cur.execute("create table test(i)")
cur.execute("insert into test(i) values (1)")
cur.execute("insert into test(i) values (10)")
cur.execute("insert into test(i) values (15)")
cur.execute("select mysum(i) from test")
print (cur.fetchall())
```

Создает пользовательскую совокупную (агрегатную) функцию.

[(26,)]

Методы connection (con)

- close() - закрывает соединение с БД
- open() – устанавливает соединение с БД
- commit() - завершает текущую транзакцию
- cursor() - возвращает объект Cursor (cur) для выполнения запросов
- rollback() - откатывает изменения в текущей транзакции

Методы cursor (cur)

- close() - закрывает курсор
- execute(*sql*[, <*значения*>]) - выполняет один запрос
- executemany(*sql*, *args*) - выполняет запрос для нескольких значений
- exequtescript(*sql*) – выполняет запросы по созданному ранее скрипту
- fetchall() - возвращает список кортежей всех записей запроса
- fetchmany() - возвращает список кортежей записей запроса
- fetchone() - возвращает одну запись из результата запроса в виде кортежа

Создание БД в ОП

```
>>> con=sqlite3.connect(':memory:')
>>> cur=con.cursor()
>>> sql='''
... CREATE TABLE IF NOT EXISTS author (
... id_author INTEGER PRIMARY KEY AUTOINCREMENT,
... author_name TEXT,
... author_descr TEXT
... );
... '''
```

```
>>> cur.executescript(sql)
<sqlite3.Cursor object at 0x1005eaf80>
>>> sql='''
... INSERT INTO author (author_name,author_descr)
... VALUES ('Chukovskiy','Pisatel')
... '''
>>> cur.executescript(sql)
<sqlite3.Cursor object at 0x1005eaf80>
>>> cur.execute('''
... SELECT * FROM author
... ''')
<sqlite3.Cursor object at 0x1005eaf80>
>>> cur.fetchall()
[(1, 'Chukovskiy', 'Pisatel')]
```