استفاده پیشرفته از SQLite در پایتون

به دنبال سری SQLite پایتون، این مقاله درباره استفاده های پیشرفته از ماژول SQLite3 است. اگر مقاله قبلی‌را ندارید، می توانید آن‌را از سایت بگیرید.

استفاده از نوع های date و datetime در SQLite

بعضی اوقات نیاز داریم نوع های date و datetimeرا در پایگاه داده SQLite3 خود وارد کنیم. وقتی query اظافه کردن‌را با یک شیء date و datetime اجرا می کنید، ماژول sqlite3 آداپتور پیش فرض‌را صدا زده و آنهارا به فرمت ISO تبدیل می کند. وقتی یک query را برای گرفتن آن مقدار ها اجرا می کنید، ماژول sqlite3 یک شیء رشته ای بر می گرداند:

```
import sqlite3
from datetime import date, datetime

db = sqlite3.connect(':memory:')
c = db.cursor()
c.execute(' ' 'CREATE TABLE example(id INTEGER PRIMARY KEY, created_at DATE)' ' ')

# Insert a date object into the database
today = date.today()
c.execute(' ' 'INSERT INTO example(created_at) VALUES(?)' ' ', (today,))
db.commit()

# Retrieve the inserted object
c.execute(' ' 'SELECT created_at FROM example' ' ')
row = c.fetchone()
print('The date is {0} and the datatype is {1}'.format(row[0], type(row[0])))
# The date is 2013-04-14 and the datatype is <class 'str'>
db.close()
```

مشکل اینه که اگر شیء dateرا داخل پایگاه داده وارد می کردیم، بیشتر اوقات شما یک شیء date انتظار دارید وقتی که بازگردانده می شود، نه یک شیء رشته ای. این مشکل میتواند با رد کردن PARSE_DECLTYPES و PARSE_COLNAMES در متد connect حل شود:

```
import sqlite3
from datetime import date, datetime
```

```
db = sqlite3.connect(':memory:')
detect_types=sqlite.PARSE_DECLTYPES|sqlite3.PARSE_COLNAMES)
c = db.cursor()
c.execute(' ' 'CREATE TABLE example(id INTEGER PRIMARY KEY, created_at DATE)' ' ')
# Insert a date object into the database
today = date.today()
c.execute(' ' 'INSERT INTO example(created_at) VALUES(?)' ' ', (today,))
db.commit()

# Retrieve the inserted object
c.execute(' ' 'SELECT created_at FROM example' ' ')
row = c.fetchone()
print('The date is {0} and the datatype is {1}'.format(row[0], type(row[0])))
# The date is 2013-04-14 and the datatype is <class 'datetime.date>
db.close()
```

عوض کردن متد connect، پایگاه داده حال یک شیء dateرا بر می گرداند. ماژول sqlite3 از نوع ستون برای برگنداندن شیء صحیح استفاده می کند. پس، اگر نیاز داشته باشیم که با یک شیء datetime کار کنیم، باید ستون‌را داخل جدول به عنوان نوع timestamp معین کنیم:

```
c.execute(' ' 'CREATE TABLE example(id INTEGER PRIMARY KEY, created_at timestamp)' ' ')
# Insert a datetime object
row = datetime.row()
c.execute(' ' 'INSERT INTO example(created_at) VALUES(?)' ' ', (row,))
db.commit()

# Retrieve the inserted object
c.execute(' ' 'SELECT created_at FROM example' ' ')
row = c.fetchone()
print('The date is {0} and the datatype is {1}'.format(row[0], type(row[0])))
# The date is 2013-04-14 16:29:11.666274 and the datatype is <class 'datetime.datetime'>
```

در صورتی که نوع ستون‌را به عنوان DATE معین کردید، اما باید با یک شیء datetime کار کنید، لازم است که query خودرا تغییر داده تا شیء به درستی تجزیه(parse) شود:

```
c.execute(' ' 'CREATE TABLE example(id INTEGER PRIMARY KEY, created_at DATE)' ' ')
# We are going to insert a datetime object into a DATE column
now = datetime.now()
c.execute(' ' 'INSERT INTO example(created_at) VALUES(?)' ' ', (now,))
db.commit()

# Retrieve the inserted object
c.execute(' ' 'SELECT created_at as “created_at [timestamp]” FROM example' ' ')
```

استفاده از "created_at [timestamp]" در کوئری SQL آداپتوررا وادار به تجزیه شیء به صورت صحیح می کند.

وارد کردن چندین rows با executemany در SQLite

بعضی اوقات نیاز داریم مجموعه ای از شیء هارا وارد پایگاه داده کنیم، ماژول sqlite3 متد executemany را برای اجرای کوئری SQL در برابر یک مجموعه فراهم می کند.

```
# Import the sqlite3 module
import sqlite3
db = sqlite.connect(':memory:')
c = db.cursor()
c.execute(' ' 'CREATE TABLE users(id INTEGER PRIMARY KEY, name TEXT, phone TEXT)' ' ')
users = [
        ('John',        '5557241'),
        ('Adam'         '5547874'),
        ('Jack'         '5484522'),
        ('Monthy'       '6656565')
]

c.executemany(' ' 'INSERT INTO users(name, phone) VALUES(?,?)' ' ', users)
db.commit()

# print the users
c.execute(' ' 'SELECT * FROM users' ' ')
```

```
for row in c:
        print(row)

db.close()
```

لطفا توجه داشته باشید که هر یک از عناصر مجموعه باید یک تاپل باشند.

اجرا فایل SQL با executescript در SQLite

متد اجرا(execute) فقط به شما اجازه اجرای یک جمله SQLرا می دهد. اگر می خواهید چندین جمله متفاوت SQLرا اجرا کنید باید از متد executescript استفاده کنید:

```
# Import the sqlite3 module
import sqlite3
db = db.cursor()
script = ' ' '(CREATE TABLE users(id INTEGER PRIMARY KEY, name TEXT, phone TEXT);
            CREATE TABLE accounts(id INTEGER PRIMARY KEY, description TEXT);

             INSERT INTO users(name, phone) VALUES ('John', '5557241'),
             ('Adam', '5547874'), ('Jack', '5484522');' ' '
c.execute(script)

# Print the results
c.execute(' ' 'SELECT * FROM users' ' ')
for row in c:
        print(row)

db.close()
```

اگر میخواهید اسکریپت‌را از فایل بخوانید:

```
fd = open('myscript.sql', 'r')
script = fd.read()
c.executescript(script)
fd.close()
```

لطفا توجه داشته باشید برای گرفتن استثناء ها بهتر است کد خودرا با شرط های try/except/else احاطه کنید. برای یادگیری بیشتر درباره کلمات کلیدی try/except/else، به مقاله گرفتن استثناء های پایتون – کلمات کلیدی try/except/else مراجعه نمایید.

معین کردن توابع SQL در SQLite

بعضی اوقات باید از توابع خودمان در جملات استفاده کنیم، مخصوصا مواقعی که داده‌را برای انجام دادن کار خاصی وارد می کنیم. یک مثال خوب می تواند موقعی باشد که بایدرمز هارا در پایگاه داده ذخیره کنیم و بخواهیم آن‌رمز هارا کد(encrypt) کنیم:

```
import sqlite3 # Import the SQLite3 module
import hashlib

def encrypt_password(password):
        # Do not use this agorithm in  real environment
        encrypted-pass = hashlib.sha1(password.encode('utf-8')).hexdigest()
        return encrypted_pass

db = sqlite3.connect(':memory:')
# Register the function
db.create_function('encrypt', 1, encrypt_password)
c = db.cursor()
c.execute(' ' 'CREATE TABLE users(id INTEGER PRIMARY KEY, email TEXT, password TEXT)'
' ')
user = ('Johndoe@example.com', '12345678')
c.execute(' ' 'INSERT INTO users(email, password) VALUES(?,encrypt(?))' ' ', user
```

تابع create_function سه پارامتر می گیرد: name(نامی که برای صدا زدن تابع داخل جمله به کار می رود)، تعداد پارامتر هایی که تابع توقع دارد(در اینجا 1 پارامتر) و یک شیء دارای قابلیت صدازنی (خود تابع). برای استفاده از تابع ثبت شدیمان، با استفاده از ()encrypt در جمله آن‌را صدا زدیم.

و در آخر، لطفا یک الگوریتم کدگذاری خوب برای ذخیره‌رمزهایتان استفاده نمایید!

مترجم: علی مرادی

تماس با من: adeadmarshal@gmail.com

لینک مطلب اصلی: http://www.pythoncentral.io/advanced-sqlite-usage-in-python/