مقدمه پایگاه داده در پایتون

SQLite3 یک موتور ساده پایگاه داده است.این پایگاه داده بدون سرور، بدون نیاز به تنظیمات و تعاملی است. این پایگاه خیلی سریع و سبک است، و کل پایگاه می تواند روی یک فایل دیسک ذخیره شود. در خیلی از برنامه ها به عنوان ذخیره داخلی داده استفاده می شود. کتابخانه استاندارد پایتون دارای یک ماژول به نام "sqlite3" برای کار با این پایگاه داده است. ماژول یک محیط SQL است با معین کننده های DB-API 2.0.

استفاده از ماژول SQLite پایتون

برای استفاده از ماژول SQLite3 ما باید یک جمله import به اسکریپت پایتون خود اضافه کنیم:

```
import sqlite3
```

برقراری ارتباط SQLite با پایگاه داده

ما از تابع sqlite3.connect برای ارتباط با پایگاه داده استفاده می کنیم. می توانیم از آرگومان ":memory:" برای ساخت پایگاه موقت در رم استفاده کنیم و نام یک فایل را برای باز کردن یا ساختن آن رد کنیم.

```
# Create a database in RAM
db = sqlite3.connect(':memory:')
# Creates or opens a file called mydb with a SQLite3 DB
db = sqlite3.connect('data/mydb')
```

وقتی کارمان با پایگاه تمام شد باید ارتباط را قطع کنیم:

```
db.close()
```

ساخت (CREATE) و حذف (DROP) جدول ها

برای انجام هر عملی با پایگاه داده باید یک شیء مکان نما (cursor) بگیریم و جمله های SQL را برای اجرا در شیء مکان نما رد کنیم. در آخر مهم است که تغییرات را تثبیت (commit) کنیم. ما می رویم که جدول کاربران را با ستون های نام، تلفن، ایمیل، و پسورد بسازیم.

```
# Get a cursor object
cursor = db.cursor()
cursor.execute(''' 
        CREATE TABLE users(id INTEGER PRIMARY KEY, name TEXT,
```

```
                    phone TEXT, email TEXT unique, password TEXT)
' ' ')
db.commit()
```

برای حذف یک جدول:

```
# Get a cursor object
cursor = db.cursor()
cursor.execute(' ' 'DROP TABLE users' ' ')
db.commit()
```

لطفا توجه کنید که تابع commit روی شیء db اجرا می شود، نا روی شیء cursor. اگر تایپ کنیم cursor.commit نتیجه زیررا می گیریم:

AttributeError: 'sqlite3.Cursor' object has no attribute 'commit'

اظافه کردن (INSERT) داده داخل پایگاه داده

برای اظافه کردن داده از cursor برای اجرای query استفاده می کنیم. اگر ارزش هارا از متغیر های پایتون می خواهید توسیه می شود از placeholder "?" استفاده کنید. هیچوقت از عمل های رشته یا الحاق کردن (concatenation) برای ساخت query ها استفاده نکنید زیرا ناامن هستند. در این مثال می خواهیم دو کاربر به پایگاه داده اظافه کنیم، اطلاعات کاربران در متغیر های پایتون ذخیره می شوند.

```
Cursor = db.cursor()
name1 = 'Andres'
phone1 = '3366858'
email1 = 'user@example.com'
# A very secure password
password1 = '12345'

name2 = 'John'
phone2 = '5557241'
email2 = 'john@example.com
password2 = 'abcdef'

# Insert user 1
```

```
cursor.execute(' ' 'INSERT INTO users(name, phone, email, password)
                    VALUES(?,?,?,?)' ' ', (name1,phone1,email1,password1))
print('First user inserted')

# Insert user 2
cursor.execute(' ' 'INSERT INTO users(name, phone, email, password)
                    VALUES(?,?,?,?)' ' ', (name2, phone2, email2, password2))
print('Second user inserted')

db.commit()
```

ارزش های متغیر های پایتون داخل یک تاپل (tuple) رد خواهند شد. روش دیگر انجام اینکار رد کردن یک دیکشنری با استفاده از placeholder روبرو است: "keyname:"

```
cursor.execute(' ' 'INSERT INTO users(name, phone, email, password)
                    VALUES(:name, :phone, :email, :password)' ' ',
                    {'name' :name1, 'phone' :phone1, 'email' :email1, 'password' :password1})
```

اگر می خواهید چندین کاربر اظافه کنید از executemany و یک لیست با تاپل استفاده کنید:

```
users = [(name1,phone1, email1, password1),
         (name2,phone2. email2, password2),
         (name3,phone3, email3, password3)]
cursor.executemany(' ' ' INSERT INTO users(name, phone, email, password)
VALUES(?,?,?,?)' ' ', users
db.commit()
```

اگر می خواهید id آخرین سطری (row) که همین الان اظافه کردیدرا بگیرید از lastrowid استفاده کنید:

```
id = cursor.lastrowid
print('Last row id: %d' % id)
```

باز یافتن داده (SELECT) با SQLite

برای بازگرفتن داده، queryرا در مقابل شیء cursor اجرا کنید و بعد از fetchone() برای بازگرفتن یک سطح تنها یا

fetchall() برای بازگرفتن کل سطرها استفاده کنید.

```
cursor.execute(' ' 'SELECT name, email, phone FROM users' ' ')
user1 = cursor.fetchone() #retrieve the first row
print(user1[0])          # Print the first column retrieved(user's name)
all_rows = cursor.fetchall()
for row in all_rows:
        # row[0] returns the first column in the query (name), row[1] returns email column.
        Print('{0} : {1}, {2}'.format(row[0], row[1], row[2]))
```

شیء cursor مثل یک تکرار کننده عمل می کند، fetchall()را اتوماتیک اجرا می کند:

```
cursor.execute(' ' 'SELECT name, email, phone FROM users' ' ')
for row in cursor:
        # row[0] returns the first column in the query (name), row[1] returns email column.
        Print('{0} : {1}, {2}'.format(row[0], row[1], row[2]))
```

برای بازگرفتن داده با شرط، از placeholder علامت "?" دوباره استفاده کنید:

```
user_id = 3
cursor.execute(' ' 'SELECT name, email, phone FROM users WHERE id=?' ' ', (user_id,))
user = cursor.fetchone()
```

به روز رسانی (UPDATE) و حذف (DELETE) داده

مراحل به روز رسانی و حذف داده دقیقا مثل اظافه کردن داده است:

```
# Update user with id 1
newphone = '3113093164'
userid = 1
cursor.execute(' ' 'UPDATE users SET phone = ? WHERE id = ?' ' ',
(newphone, userid))

# Delete user with id 2
delete_userid = 2
cursor.execute(' ' 'DELETE FROM users WHERE id = ? ' ' ', (delete_userid,))
```

```
db.commit()
```

استفاده از تعامل های SQLite

تعامل ها(transactions) خصوصیت کاربردی سیستم های پایگاه داد هستند. آنها طرفیت اتمی (atomicity) پایگاه داده‌را اطمینان می بخشند. از commit برای ذخیره تغییرات استفاده کنید:

```
cursor.execute(' ' 'UPDATE users SET phone = ? WHERE id = ? ' ' ',
(newphone, userid))
db.commit() #Commit the change
```

یا rollback که تمام تغییرات دیتابیس‌را تا commit قبل به حالت اولیه باز می گرداند:

```
cursor.execute(' ' 'UPDATE users SET phone = ? WHERE id = ? ' ' ',
(newphone, userid))
# The user's phone is not updated
db.rollback()
```

لطفا به یاد داشته باشید تا همیشه commitرا برای ذخیره تغییرات صدا بزنید. اگر ارتباط‌را با استفاده از close قطع کنید یا ارتباط با فایل قطع شد(شاید برنامه ناگهانی بسته شود)، تغییراتی که commit نشده باشند از دست خواهند رفت.

استثناء های پایگاه داده SQLite

برای بهتر شدن کد همیشه عملکرد های پایگاه داده‌را با یک شرط try یا یک مدیریت محتوا (content manager) احاطه کنید:

```
import sqlite3 #Import the SQLite3 module
try:
	# Creates or opens a file called mydb with a SQLite3 DB
	db = sqlite3.connect('data/mydb')
	# Get a cursor
	cursor.execute(' ' 'CREATE TABLE IF NOT EXISTS
				users(id INTEGER PRIMARY KEY, name TEXT, phone TEXT, email
TEXT unique, passport TEXT)' ' ')
	# Commit the change
	db.commit()
```

```
# Catch the exception
except Exception as e:
        # Roll back any change if something goes wrong
        db.rollback()
        raise e
finally:
        # Close the db connection
        db.close()
```

در این مثال از یک شرط try/except/finally برای گرفتن هر استثناء در کد استفاده کردیم. کلمه کلیدی finally خیلی مهم است زیرا همیشه ارتباط پایگاه داده‌را به درستی قطع می کند. لطفا برای مطالب بیشتر درباره استثناء ها به این مقاله مراجعه کنید. لطفا نگاهی به زیر بیاندازید:

```
# Catch the exeption
except Exception as e:
        raise e
```

این یک شرط catch-all نام دارد، این در اینجا فقط به عنوان یک مثال مطرح شده است، در یک برنامه واقعی باید یک استثناء معین‌را بگیرید مثل IntegrityError یا DatabaseError، برای اطلاعات بیشتر لطفا به استثناء DB-API 2.0 مراجعه نمایید.

می توانیم از شیء connection به عنوان مدیریت محتوا برای اعمال اتوماتیک commit و rollback استفاده کنید:

```
name1 = 'Andres'
phone1 = '3366858'
email1 = 'user@example.com
# A very secure password
password1 = '12345'

try:
        with db:
                db.execute(' ' 'INSERT INTO users(name, phone, email, password)
                        VALUES(?,?,?,?)' ' ', (name1,phone1,email1,password1))
except sqlite3.IntegrityError:
        print('Record already exists')
finally:
        db.close()
```

در مثال بالا اگر جمله insert یک استثناء را نشان دهد، تعامل به عقب باز گردانده می شود و پیغام چاپ خواهد شد؛ مگر نه تعامل تثبیت(commit) می شود. لطفا توجه کنید که executeرا بر شیء db صدا می‌زنیم، نه بر شیء cursor.

کارخانه سطر و انواع داده SQLite

جدول زیر رابطه بین انواع داده SQLite و انواع داده پایتون را نشان می دهد:

- نوع None به NULL تبدیل شده است
- نوع int به INTEGER تبدیل شده است
- نوع float به REAL تبدیل شده است
- نوع str به TEXT تبدیل شده است
- نوع bytes به BLOB تبدیل شده است

کلاس sqlite3.Row برای دسترسی به سطون های query توسط نام به جای شاخص(index) استفاده می شود:

```
db = sqlite3.connect('data/mydb')
db.row_factory = sqlite3.Row
cursor = db.cursor()
cursor.execute(' ' 'SELECT name, email, phone FROM users' ' ')
for row in cursor:
        # row['name'] returns the name column in the query, row['email'] returns email column.
        print('{0} : {1}, {2}'.format(row['name'], row['email'], row['phone']))
db.close()
```

مترجم: علی مرادی

تماس با من: adeadmarshal@gmail.com

سایت: http://tarjomeebook.ir

لینک مطلب اصلی: http://www.pythoncentral.io/introduction-to-sqlite-in-python/