SQLite – Perl

SQLite - Python

安装

SQLite3 可使用 sqlite3 模块与 Python 进行集成。sqlite3 模块是由 Gerhard Haring 编写的。它提供了一个与 PEP 249 描述的 DB-API 2.0 规范兼容的 SQL 接口。您不需要单独安装该模块，因为 Python 2.5.x 以上版本默认自带了该模块。

为了使用 sqlite3 模块，您首先必须创建一个表示数据库的连接对象，然后您可以有选择地创建光标对象，这将帮助您执行所有的 SQL 语句。

Python sqlite3 模块 API

以下是重要的 sqlite3 模块程序，可以满足您在 Python 程序中使用 SQLite 数据库的需求。如果您需要了解更多细节，请查看 Python sqlite3 模块的官方文档。

序号	API & 描述
1	**sqlite3.connect(database [,timeout ,other optional arguments])** 该 API 打开一个到 SQLite 数据库文件 database 的链接。您可以使用 ":memory:" 来在 RAM 中打开一个到 database 的数据库连接，而不是在磁盘上打开。如果数据库成功打开，则返回一个连接对象。 当一个数据库被多个连接访问，且其中一个修改了数据库，此时 SQLite 数据库被锁定，直到事务提交。timeout 参数表示连接等待锁定的持续时间，直到发生异常断开连接。timeout 参数默认是 5.0（5 秒）。 如果给定的数据库名称 filename 不存在，则该调用将创建一个数据库。如果您不想在当前目录中创建数据库，那么您可以指定带有路径的文件名，这样您就能在任意地方创建数据库。
2	**connection.cursor([cursorClass])** 该例程创建一个 **cursor**，将在 Python 数据库编程中用到。该方法接受一个单一的可选的参数 cursorClass。如果提供了该参数，则它必须是一个扩展自 sqlite3.Cursor 的自定义的 cursor 类。
3	**cursor.execute(sql [, optional parameters])** 该例程执行一个 SQL 语句。该 SQL 语句可以被参数化（即使用占位符代替 SQL 文本）。sqlite3 模块支持两种类型的占位符：问号和命名占位符（命名样式）。 例如：cursor.execute("insert into people values (?, ?)", (who, age))
4	**connection.execute(sql [, optional parameters])** 该例程是上面执行的由光标（cursor）对象提供的方法的快捷方式，它通过调用光标（cursor）方法创建了一个中间的光标对象，然后通过给定的参数调用光标的 execute 方法。
5	**cursor.executemany(sql, seq_of_parameters)** 该例程对 seq_of_parameters 中的所有参数或映射执行一个 SQL 命令。
6	**connection.executemany(sql[, parameters])**

	该例程是一个由调用光标（cursor）方法创建的中间的光标对象的快捷方式，然后通过给定的参数调用光标的 execute many 方法。
7	**cursor.executescript(sql_script)** 该例程一旦接收到脚本，会执行多个 SQL 语句。它首先执行 COMMIT 语句，然后执行作为参数传入的 SQL 脚本。所有的 SQL 语句应该用分号（;）分隔。
8	**connection.executescript(sql_script)** 该例程是一个由调用光标（cursor）方法创建的中间的光标对象的快捷方式，然后通过给定的参数调用光标的 execute script 方法。
9	**connection.total_changes()** 该例程返回自数据库连接打开以来被修改、插入或删除的数据库总行数。
10	**connection.commit()** 该方法提交当前的事务。如果您未调用该方法，那么自您上一次调用 commit() 以来所做的任何动作对其他数据库连接来说是不可见的。
11	**connection.rollback()** 该方法回滚自上一次调用 commit() 以来对数据库所做的更改。
12	**connection.close()** 该方法关闭数据库连接。请注意，这不会自动调用 commit()。如果您之前未调用 commit() 方法，就直接关闭数据库连接，您所做的所有更改将全部丢失！
13	**cursor.fetchone()** 该方法获取查询结果集中的下一行，返回一个单一的序列，当没有更多可用的数据时，则返回 None。
14	**cursor.fetchmany([size=cursor.arraysize])** 该方法获取查询结果集中的下一行组，返回一个列表。当没有更多的可用的行时，则返回一个空的列表。该方法尝试获取由 size 参数指定的尽可能多的行。
15	**cursor.fetchall()** 该例程获取查询结果集中所有（剩余）的行，返回一个列表。当没有可用的行时，则返回一个空的列表。

连接数据库

下面的 Python 代码显示了如何连接到一个现有的数据库。如果数据库不存在，那么它就会被创建，最后将返回一个数据库对象。

```
#!/usr/bin/python

import sqlite3
```

```
conn = sqlite3.connect('test.db')

print "Opened database successfully";
```

在这里，您也可以把数据库名称复制为特定的名称 **:memory:**，这样就会在 RAM 中创建一个数据库。现在，让我们来运行上面的程序，在当前目录中创建我们的数据库 **test.db**。您可以根据需要改变路径。保存上面代码到 sqlite.py 文件中，并按如下所示执行。如果数据库成功创建，那么会显示下面所示的消息：

```
$chmod +x sqlite.py
$./sqlite.py
Open database successfully
```

创建表

下面的 Python 代码段将用于在先前创建的数据库中创建一个表：

```
#!/usr/bin/python

import sqlite3

conn = sqlite3.connect('test.db')
print "Opened database successfully";
c = conn.cursor()
c.execute('''CREATE TABLE COMPANY
       (ID INT PRIMARY KEY     NOT NULL,
       NAME           TEXT    NOT NULL,
       AGE            INT     NOT NULL,
       ADDRESS        CHAR(50),
       SALARY         REAL);''')
print "Table created successfully";
conn.commit()
conn.close()
```

上述程序执行时，它会在 **test.db** 中创建 COMPANY 表，并显示下面所示的消息：

```
Opened database successfully
Table created successfully
```

INSERT 操作

下面的 Python 程序显示了如何在上面创建的 COMPANY 表中创建记录：

```
#!/usr/bin/python

import sqlite3

conn = sqlite3.connect('test.db')
c = conn.cursor()
print "Opened database successfully";

c.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
      VALUES (1, 'Paul', 32, 'California', 20000.00 )");

c.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
      VALUES (2, 'Allen', 25, 'Texas', 15000.00 )");

c.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
      VALUES (3, 'Teddy', 23, 'Norway', 20000.00 )");

c.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
      VALUES (4, 'Mark', 25, 'Rich-Mond ', 65000.00 )");

conn.commit()
print "Records created successfully";
conn.close()
```

上述程序执行时，它会在 COMPANY 表中创建给定记录，并会显示以下两行：

```
Opened database successfully
Records created successfully
```

SELECT 操作

下面的 Python 程序显示了如何从前面创建的 COMPANY 表中获取并显示记录：

```
#!/usr/bin/python

import sqlite3

conn = sqlite3.connect('test.db')
c = conn.cursor()
print "Opened database successfully";

cursor = c.execute("SELECT id, name, address, salary  from COMPANY")
for row in cursor:
   print "ID = ", row[0]
   print "NAME = ", row[1]
   print "ADDRESS = ", row[2]
```

```
   print "SALARY = ", row[3], "\n"

print "Operation done successfully";
conn.close()
```

上述程序执行时，它会产生以下结果：

```
Opened database successfully
ID =  1
NAME =  Paul
ADDRESS =  California
SALARY =  20000.0

ID =  2
NAME =  Allen
ADDRESS =  Texas
SALARY =  15000.0

ID =  3
NAME =  Teddy
ADDRESS =  Norway
SALARY =  20000.0

ID =  4
NAME =  Mark
ADDRESS =  Rich-Mond
SALARY =  65000.0

Operation done successfully
```

UPDATE 操作

下面的 Python 代码显示了如何使用 UPDATE 语句来更新任何记录，然后从 COMPANY 表中获取并显示更新的记录：

```
#!/usr/bin/python

import sqlite3

conn = sqlite3.connect('test.db')
c = conn.cursor()
print "Opened database successfully";

c.execute("UPDATE COMPANY set SALARY = 25000.00 where ID=1")
conn.commit()
print "Total number of rows updated :", conn.total_changes

cursor = conn.execute("SELECT id, name, address, salary  from COMPANY")
```

```
for row in cursor:
   print "ID = ", row[0]
   print "NAME = ", row[1]
   print "ADDRESS = ", row[2]
   print "SALARY = ", row[3], "\n"

print "Operation done successfully";
conn.close()
```

上述程序执行时，它会产生以下结果：

```
Opened database successfully
Total number of rows updated : 1
ID =  1
NAME =  Paul
ADDRESS =  California
SALARY =  25000.0

ID =  2
NAME =  Allen
ADDRESS =  Texas
SALARY =  15000.0

ID =  3
NAME =  Teddy
ADDRESS =  Norway
SALARY =  20000.0

ID =  4
NAME =  Mark
ADDRESS =  Rich-Mond
SALARY =  65000.0

Operation done successfully
```

DELETE 操作

下面的 Python 代码显示了如何使用 DELETE 语句删除任何记录，然后从 COMPANY 表中获取并显示剩余的记录：

```
#!/usr/bin/python

import sqlite3

conn = sqlite3.connect('test.db')
c = conn.cursor()
print "Opened database successfully";
```

```
c.execute("DELETE from COMPANY where ID=2;")
conn.commit()
print "Total number of rows deleted :", conn.total_changes

cursor = conn.execute("SELECT id, name, address, salary  from COMPANY")
for row in cursor:
   print "ID = ", row[0]
   print "NAME = ", row[1]
   print "ADDRESS = ", row[2]
   print "SALARY = ", row[3], "\n"

print "Operation done successfully";
conn.close()
```

上述程序执行时，它会产生以下结果：

```
Opened database successfully
Total number of rows deleted : 1
ID =  1
NAME =  Paul
ADDRESS =  California
SALARY =  20000.0

ID =  3
NAME =  Teddy
ADDRESS =  Norway
SALARY =  20000.0

ID =  4
NAME =  Mark
ADDRESS =  Rich-Mond
SALARY =  65000.0

Operation done successfully
```

SQLite – Perl

点我分享笔记