Python Database Programming

For Introduction to Programming Using Python

By Y. Daniel Liang

To write database programs using SQL in Python, you need to know RDBM and SQL. An introduction to RDBMS and SQL can be found in **http://www.cs.armstrong.edu/liang/intro9e/databasesupplement.html**.

Python provides an API for accessing SQL database. You can write Python programs to access a relational database system such as MySQL, Oracle, DB2, Sybase, or SQLite. Since SQLite comes with Python, we will use SQLite to demonstrate database programming in Python.

To open a SQLite database, use

```
import sqlite3

db = sqlite3.connect(filename)
```

The **sqlite3.connect(filename)** function returns a database object for the database file. If the file does not exist, the function creates the database file.

After a db object is created, you can use the following methods on a db object:

db.close(): closes the database.

db.commit(): commits any pending changed to the database.

db.rollback(): rolls back any pending transactions to the state that existed before the transaction began.

db.cursor(): Returns a database cursor object through which a SQL statement can be executed.

To execute a SQL statement, first obtain a cursor as follows:

```
cursor = db.cursor()
```

Now you can execute a SQL statement using cursor's execute method. For example, the following code executes a CREATE TABLE statement.

```
cursor.execute("create table Course ( " +
               "courseId char(5), subjectId char(4) not null, " +
               "courseNumber integer, title varchar(50) not null, " +
               "numOfCredits integer, primary key (courseId))")
```

The following code executes an INSERT statement.

```
cursor.execute("insert into Course (courseId, subjectId, " +
               " courseNumber, title, numOfCredits) " +
               "values ('11113', 'CSCI', '3720', 'Database Systems', 3)")

db.commit()
```

The following code executes a SELECT statement.

```
cursor.execute("select * from Course")
```

To obtain a row from the query result, invoke **fetchone()** as follows:

```
row = cursor.fetchon()
```

row is a tuple that consists of the elements for the fields.

You can display all the elements using

```
if row != None:
    for element in row:
        print(element)
```

To obtain all rows in the query, invoke **fetchall()** as follows:

```
rows = cursor.fetchall()
```

rows is a list consisting of the tuples.

Listing 1 gives a complete program that creates a table, inserts rows, queries database, and displays the query result.

Listing 1 dbdemo.py

```
import sqlite3

db = sqlite3.connect("db")

cursor = db.cursor()

cursor.execute("create table Course ( " +
               "courseId char(5), subjectId char(4) not null, " +
               "courseNumber integer, title varchar(50) not null, " +
               "numOfCredits integer, primary key (courseId))")

cursor.execute("insert into Course (courseId, subjectId, " +
               " courseNumber, title, numOfCredits) " +
               "values ('11113', 'CSCI', '3720', 'Database Systems', 3)")

cursor.execute("insert into Course (courseId, subjectId, " +
               " courseNumber, title, numOfCredits) " +
               "values ('11111', 'CSCI', '1301', 'Introduction to Programming', 3)")
db.commit()

cursor.execute("select * from Course")

rows = cursor.fetchall()
```

```
print(rows)
db.close()
```