《Python统计计算》

（2021年秋季学期）

翟祥
北京林业大学

E-mail: zhaixbh@126.com

编程基础

第10章 数据库编程

主要内容

- SQLite简介
- 通过sqlitestudio访问SQLite数据库
- 在Python中访问SQLite数据库

（一）SQLite简介

- 已内嵌在Python中，使用时需要导入sqlite3
- 使用c语言开发，支持大多数SQL91标准
- 不支持外键限制
- 支持原子的、一致的、独立和持久的事务
- 通过数据库级上的独占性和共享锁定来实现独立事务，当多个线程和进程同一时间访问同一数据库时，只有一个可以写入数据

- 支持2TB的数据库，每个数据库完全存储在单个磁盘文件中，以B+数据结构的形式存储，一个数据库就是一个文件，通过复制即可实现备份
- 如果读取记录显示乱码，尝试修改程序以使用utf-8编码格式
- www.sqlabs.com==>SQLiteManager
- SQLite Database Browser

（二）通过sqlitestudio访问SQLite数据库

- 可以通过sqlitestudio工具创建、操作数据库
- 可以在运行SQLite数据库的同时通过参数创建SQLite数据库，具体方法如下：

  sqlite3 数据库文件名

- SQLite 数据库文件的扩展名为.db。如果指定的数据库文件存在，则打开该数据库；否则创建该数据库。
- 创建SQLite 数据库test.db。执行下面的命令：

  sqlite3 test.db

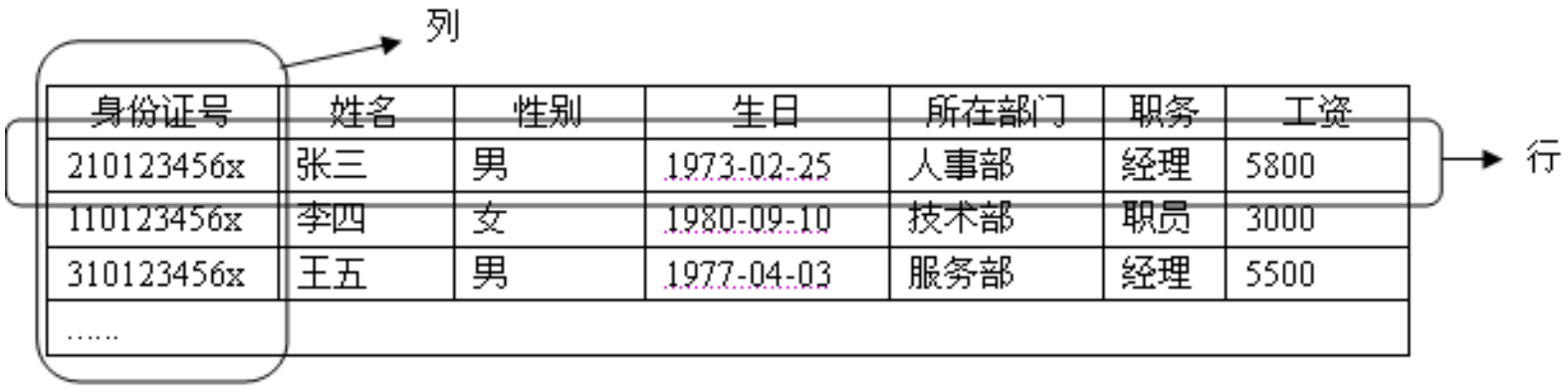
- 进入SQLite数据库的命令行窗口后，可以执行下面的命令保存test.db数据库。

  .save test.db

数据类型

数据类型	说明
NULL	空值
integer	带符号的整型
real	实数
text	字符串文本
blob	二进制对象
smallint	16 位整数
decimal(p,s)	小数。p指数字的位数，s指小数点后有几位数。
float	32位浮点数字
double	64位实数
char(n)	固定长度的字符串，n为字符串的长度，不能超过 254
varchar(n)	不固定长度的字符串，n为字符串的最大长度，不能超过 4000
graphic(n)	和char(n)一样，不过其单位是两个字节double-bytes， n不能超过127。graphic支持两个字节长度的字体，例如汉字
vargraphic(n)	长度不固定且最大长度为 n 的双字节字符串，n不能超过 4000
date	日期，包含年、月、日
time	时间包含小时、分钟、秒
timestamp	时间戳，包含年、月、日、时、分、秒、千分之一秒
datetime	日期和时间

创建表



身份证号	姓名	性别	生日	所在部门	职务	工资
210123456x	张三	男	1973-02-25	人事部	经理	5800
110123456x	李四	女	1980-09-10	技术部	职员	3000
310123456x	王五	男	1977-04-03	服务部	经理	5500
……						

使用CREATE TABLE语句创建表

```
CREATE TABLE 表名 (
    列名1  数据类型和长度1  列属性1,
    列名2  数据类型和长度2  列属性2,
    ……
    列名n  数据类型和长度n  列属性n
)
```

【例1】

列名	数据类型	具体说明
Emp_id	integer	员工编号
Emp_name	varchar(50)	员工姓名
Sex	char(2)	员工性别
Title	varchar(50)	员工职务
Wage	float	工资
IdCard	varchar(20)	身份证号
Dep_id	integer	所在部门编号

CREATE TABLE语句创建表Employees的方法

```
CREATE TABLE Employees (
    Emp_id   integer,
    Emp_name      varchar(50)   NOT NULL,
    Sex         char(2)       DEFAULT('男'),
    Title       varchar(50)   NOT NULL,
    Wage      float   DEFAULT(0),
    IdCard    varchar(20)   NOT NULL,
    Dep_id    integer       NOT NULL
)
```

CREATE TABLE语句中常用的关键字

- PRIMARY KEY，定义此列为主键列。定义为主键的列可以唯一标识表中的每一行记录。
- AutoIncrement，定义此列为自增列，即由系统自动生成此列的值。
- NOT NULL，指定此列不允许为空。NULL表示允许空，但因为它是默认设置，不需要指定。
- DEFAULT，指定此列的默认值。例如，指定Sex列的默认值为“男”，可以使用DEFAULT('男')。这样，在向表中插入数据时，如果不指定此列的值，则此列采用默认值。

查看当前数据库里的表

```
SELECT name FROM sqlite_master
    WHERE type='table'
    ORDER BY name;
```

如果在sqlite命令行下，可以直接使用 .tables 或 .schema 命令来得到完整的数据库集（包括表s和索引s ）。 这两个命令支持通配符。

如果在其它宿主程序中（例如 C/C++等），你可以从一个特殊的表 “SQLITE_MASTER” 得到类似的信息。

- 运行结果如下：

Employees

查看表Employees的结构

```
select * from sqlite_master where type="table" and
name="Employees";
```

运行结果如下：

```
table|Employees|Employees|2|CREATE TABLE Employees (
    Emp_id      int,
    Emp_name    varchar(50)     NOT NULL,
    Sex char(2) DEFAULT('男'),
    Title       varchar(50)     NOT NULL,
    Wage        float   DEFAULT(0),
    IdCard      varchar(20)     NOT NULL,
    Dep_id      int     NOT NULL
)
```

向表中添加列

- 使用ALTER TABLE语句向表中添加列的基本语法如下：

ALTER TABLE 表名 ADD COLUMN 列名 数据类型和长度 列属性

【例2】

```
ALTER TABLE Employees ADD Tele VARCHAR(50) NULL
ALTER TABLE Employees MODIFY Tele CHAR(50) NULL
```

执行下面的语句可以查看表Employees的结构。

```
.schema Employees
```

运行结果如下：

```
table|Employees|Employees|2|CREATE TABLE Employees
 (
    Emp_id      int,
    Emp_name    varchar(50)     NOT NULL,
    Sex char(2) DEFAULT('男'),
    Title       varchar(50)     NOT NULL,
    Wage        float   DEFAULT(0),
    IdCard      varchar(20)     NOT NULL,
    Dep_id      int     NOT NULL,
    Tele VARCHAR(50) NULL)
```

向表中插入数据

- 可以使用INSERT语句向表中插入数据。基本使用方法如下：

```
INSERT INTO 表名 (列名1, 列名2, …, 列名n)
VALUES (值1, 值2, …, 值n);
```

【例3】

字段Emp_name的值	字段IdCard的值	字段Title的值	字段DepId的值
张三	1101234567890	部门经理	1
李四	2101234567890	职员	1
王五	3101234567890	职员	1
赵六	4101234567890	部门经理	2
高七	5101234567890	职员	2
马八	6101234567890	职员	2
钱九	7101234567890	部门经理	3
孙十	8101234567890	职员	3

INSERT语句如下

```
INSERT INTO Employees (Emp_name, IdCard, Dep_id, Title) VALUES('张三', '1101234567890', 1, '部门经理');
INSERT INTO Employees (Emp_name, IdCard, Dep_id, Title)  VALUES('李四', '2101234567890', 1, '职员');
INSERT INTO Employees (Emp_name, IdCard, Dep_id, Title)  VALUES('王五', '3101234567890', 1, '职员');
INSERT INTO Employees (Emp_name, IdCard, Dep_id, Title)  VALUES('赵六', '4101234567890', 2, '部门经理');
INSERT INTO Employees (Emp_name, IdCard, Dep_id, Title)  VALUES('高七', '5101234567890', 2, '职员');
INSERT INTO Employees (Emp_name, IdCard, Dep_id, Title)  VALUES('马八', '6101234567890', 2, '职员');
INSERT INTO Employees (Emp_name, IdCard, Dep_id, Title)  VALUES('钱九', '7101234567890', 3, '部门经理');
INSERT INTO Employees (Emp_name, IdCard, Dep_id, Title)  VALUES('孙十', '8101234567890', 3, '职员');
```

修改表中的数据

- 可以使用UPDATE语句修改表中的数据。
  UPDATE语句的基本使用方法如下所示：

UPDATE 表名 SET 列名1 = 值1, 列名2 = 值2, …, 列名n = 值n

WHERE 更新条件表达式

SQLite的关系运算符和比较函数

关系运算符	功能描述
==	等于，例如a==1
=	与==相同
!=	不等于，例如a!=1
<>	与!=相同，例如a<>1
<=	小于或等于，例如a<=1
<	小于，例如a<1
>=	大于或等于，例如a>=1
>	大于，例如a>1
!<	检查左操作数的值是否不小于右操作数的值，如果是则条件为真
!>	检查左操作数的值是否不大于右操作数的值，如果是则条件为真

删除数据

可以使用DELETE语句删除表中的数据，基本使用方法如下所示：

```
DELETE FROM 表名WHERE 删除条件表达式
```

当执行DELETE语句时，指定表中所有满足WHERE子句条件的行都将被删除。

【例5】 删除表Employees中列Emp_name等于“李明”的数据，可以使用以下SQL语句：

```
DELETE FROM Employees WHERE
Emp_name = '李明';
```

查询数据

可以使用SELECT语句查询表中的数据，基本使用方法如下所示：

SELECT * FROM 表名WHERE 删除条件表达式

*表示查询表中所有的字段，当执行SELECT语句时，指定表中所有满足WHERE子句条件的行都将被返回。

【例6】 查询表Employees中列Title等于“部门经理”的数据，可以使用以下SQL语句：

```
SELECT * FROM Employees WHERE Title= '部门经理';
```

执行结果如下：

```
|赵六|男|部门经理|0.0|410123456789O|2|
|钱九|男|部门经理|0.0|710123456789O|3|
```

（三）在Python中访问SQLite数据库

- Python中内置了sqlite3模块，可以很方便地访问SQLite数据库。首先需要使用下面的语句导入sqlite3模块：

```
import sqlite3
```

1. 创建和打开数据库

- 使用connect()方法可以创建和打开数据库，具体方法如下：

数据库连接对象 = sqlite3.connect(数据库名)

- 【例5】 创建数据库d:\test\test.db，如果已经存在，则将其打开，代码如下；

```
import sqlite3
cx = sqlite3.connect("d:/test/test.db")
```

2. 执行SQL语句

- 使用execute ()方法可以执行SQL语句，具体方法如下：

  数据库连接对象.execute(SQL语句)

- 【例5】 在数据库d:\test\test.db中使用execute()方法创建表Employees。具体代码如下：

```
#ex_5.py
import sqlite3
cx = sqlite3.connect("d:/test/test.db")
sql = "CREATE TABLE Employees (Emp_id  integer,
Emp_name    varchar(50)   NOT NULL, Sex        char(2),
Title    varchar(50)   NOT NULL, Wage      float
        DEFAULT(0), IdCard  varchar(20)   NOT NULL,
Dep_idintegerNOT NULL)"
cx. execute(sql)
```

【例6】

```
#ex_6.py
import sqlite3
cx = sqlite3.connect("d:/test/test.db")
sql = "INSERT INTO Employees
(Emp_name, IdCard, Dep_id, Title)
VALUES('Johney', '1101234567890', 1, '')";
cx. execute(sql)
cx.commit(); # 提交
cx.close();
```

3．使用游标查询数据

身份证号	姓名	性别	生日	所在部门	职务	工资
210123456x	张三	男	1973-02-25	人事部	经理	5800
110123456x	李四	女	1980-09-10	技术部	职员	3000
310123456x	王五	男	1977-04-03	服务部	经理	5500
……						

- Python可以使用下面的方法创建一个游标对象：

  游标对象 =数据库连接对象.cursor()

- 可以使用游标对象的execute()方法执行SELECT语句将查询结果保存在游标中，方法如下：

  游标对象.execute(SELECT语句)

- 可以使用游标对象的fetchall()方法获取游标中所有的数据到一个元组中，方法如下：

  结果集元组 =游标对象.fetchall()

4. SQLite主要方法

- connect(database[, timeout, isolation_level, detect_types, factory])

  Opens a connection to the SQLite database file *database*. You can use":memory:" to open a database connection to a database that resides in RAM instead of on disk.

- sqlite3.Connection.execute()

  Executes a SQL statement.

- sqlite3.Connection.cursor()

  Return a cursor for the connection.

- sqlite3.Connection.commit()

  Commit the current transaction.
- sqlite3.Connection.rollback()

  Roll back the current transaction.
- sqlite3.Connection.close()

  Closes the connection.

5. SQLite游标

close(...)： Closes the cursor.
execute(...)：Executes a SQL statement.
executemany(...)：Repeatedly executes a SQL statement.
executescript(...)：Executes a multiple SQL statements at once. Non-standard.
fetchall(...)：Fetches all rows from the resultset.
fetchmany(...)： Fetches several rows from the resultset.
fetchone(...)：Fetches one row from the resultset.
next(...)：x.next() -> the next value, or raise StopIteration
setinputsizes(...)：Required by DB-API. Does nothing in pysqlite.
setoutputsize(...)：Required by DB-API. Does nothing in pysqlite.

Data descriptors defined here:
arraysize
connection
description
lastrowid
row_factory
rowcount

6. SQLite编程—创建表

```
#ex_0.py
import sqlite3
conn=sqlite3.connect("D:/test/addressBook.db")

sql = "CREATE TABLE addressList (name
varchar(50),sex char(2),phone varchar(50),QQ
char(12), address varchar(50))"
conn.execute(sql)

conn.close()
```

7. SQLite编程—插入数据

```
#ex_1.py
import sqlite3
conn=sqlite3.connect("D:/test/addressBook.db")
cur=conn.cursor()
cur.execute('''insert into addressList(name , sex , phone , QQ ,
    address)
values('王小丫' , '女' , '13888997011' , '66735' , '北京市' )''')
cur.execute('''insert into addressList(name, sex, phone, QQ, address)
values('李莉', '女', '15808066055', '675797', '天津市')''')
cur.execute('''insert into addressList(name, sex, phone, QQ, address)
values('李星草', '男', '15912108090', '3232099', '昆明市')''')
conn.commit()
conn.close()
```

8. SQLite编程—查询数据

```
#ex_2.py
import sqlite3
conn=sqlite3.connect('D:/test/addressBook.db')
cur=conn.cursor()
cur.execute('select * from addressList')
li=cur.fetchall()

for line in li:
    for item in line:
        s=item
        print(s+'\t',end=" ")

    print('\n')

conn.close()
```

9. 访问其他类型数据库

- 除了SQLite数据库以外，Python还可以操作ACCESS、SQLServer、MYSQL等多种类型的数据库。

python™

谢谢Q/A