Python SQLite 操作

Python /SQLite

SQLite

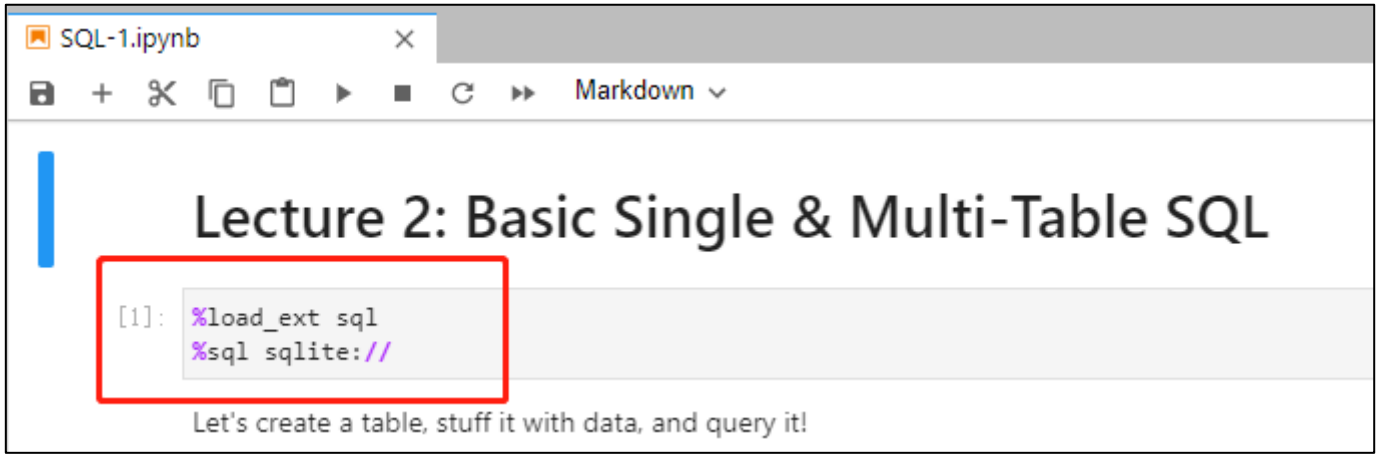
- SQLite is free and can be downloaded from www.sqlite.org
- To download select the "`Precomplied Binaries`" for your operating system (Mac OS, Windows or Linux)
- To use SQLite you need to load the "DB Browser for SQLite" from https://sqlitebrowser.org

Data types for fields

- `Integer`: the value is an integer value
- `Real`: the value is a floating-pint value
- `Text`: the value is a string text
- `Blob`: the value is stored exactly as it was input.
- You can also specify if the field cannot be left blank by adding `NOT NULL` to the end of the field when you create it

使用SQLite

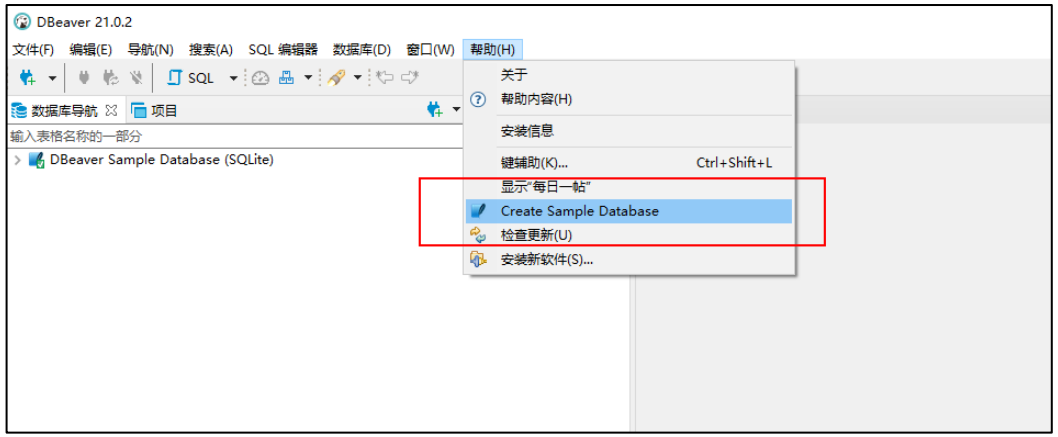
- 由于SQLite确实非常“lite”，各种常用数据库的地方都有集成
- 我们已经在Jupyter Notebook中使用了python自带的SQLite数据库：
  - sqlite:// or sqlite://:memory:
  - sqlite:///relative path to file
  - sqlite:////absolute path to file



- 除此以外，DBeaver等客户端程序也集成了本地的SQLite程序

DBeaver Sample Database

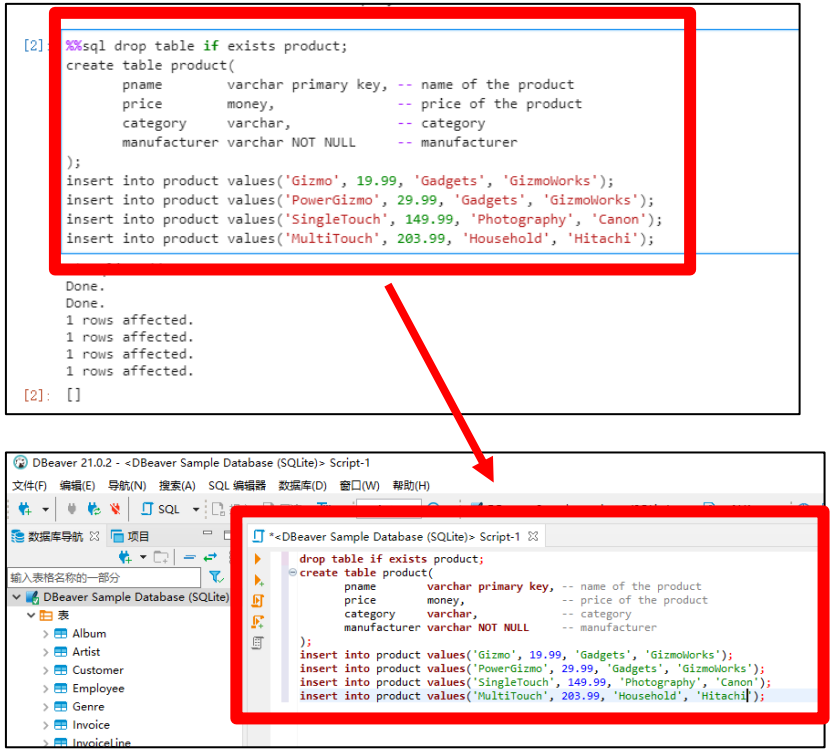
- DBeaver中集成的Sample Database就是使用的本地SQLite

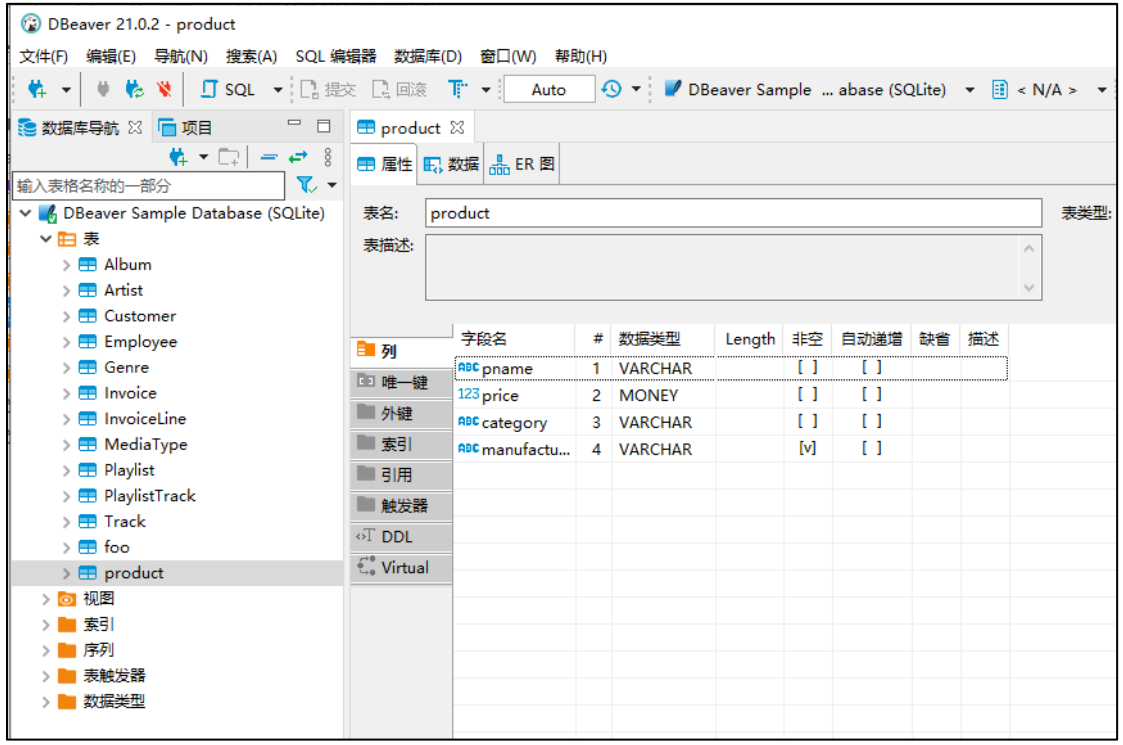


- 之前课程Notebook中的SQLite代码均可以直接在DBeaver中使用

DBeaver Sample Database

- 从DBeaver中可视化的查看创建的数据表：





在Python中使用SQLite

- 在上述方法中，我们都是直接使用SQL语言操作数据库

- 一般的数据分析流程：
  - ◆读取数据：SQL
  - ◆分析数据：Python
  - ◆将分析的中间/最终结果进行保存：SQL

- 更便捷的方法：将整个流程都集中在python代码中

Example code

我们来看一个Python操作SQLite数据库的例子：

- 创建一个名为“PhoneBook.db”的SQL database
- 建立一个名为“Names”的table，并插入数据

```
In [146]: runfile('C:/Users/mireilla/.s
(1, 'Simon', 'Pierre', '0141647 1367')
(2, 'Rita', 'McVey', '0141887 2354')
(3, 'Marc', 'Blondel', '0123456 7987')
```

```
import sqlite3
# Connect to the database called PhoneBook or create one if there is none
with sqlite3.connect("PhoneBook.db") as db:
    cursor = db.cursor()

#    Create a table called Names with foru fields
cursor.execute(""" CREATE TABLE IF NOT EXISTS Names(
id integer PRIMARY KEY,
firstname text,
surname text,
phonenumber text); """)

# Insert data into the table
cursor.execute(""" INSERT INTO Names(id,firstname,surname,phonenumber)
VALUES ("1", "Simon","Pierre","0141647 1367")""")
db.commit() # Saves the changes

# Insert data into the table Names
cursor.execute(""" INSERT INTO Names(id, firstname,surname,phonenumber)
VALUES ("2", "Rita","McVey","0141887 2354")""")
db.commit() # saves the chnages

# Insert data into a table called Names
cursor.execute(""" INSERT INTO Names(id,firstname,surname,phonenumber)
VALUES ("3", "Marc", "Blondel", "0123456 7987")""")
db.commit() # saves the changes

# Select everything from the table called Names and prints one row per line.
cursor.execute(" SELECT * FROM Names")
for x in cursor.fetchall():
    print(x)

db.close() # close the database
```

Python SQLite3 - Example code

```
#允许 Python使用 SQLite3库
import sqlite3

#连接 university 数据库. 如果没有该数据库，则创建. 对应的db文件存放在相同目录下.
db = sqlite3.connect("university.db")
cursor=db.cursor()

# 创建students 表，具有4个域 (id, name, class and grade). 指定了每个域的数据类型与primary key，以及哪些子域必须为非空.
#三个双引号（triple speech marks）允许分行显示更便于阅读，而不是显示在一行内.
cursor.execute("""CREATE TABLE IF NOT EXIST students(
        id integer PRIMARY KEY,
        name text NOT NULL,
        class text NOT NULL,
        grade integer);""")

# db.commit命令将execute的操作提交到数据库
db.commit()
```

重复执行会导致数据库锁死

想一想，练一练

- 创建 university 数据库
- 创建students表
- 至少包括 ID，姓名，性别，年龄等必要字段，并自己为这些字段选择合适的字段类型（非空）。

Students 学生信息表
(id, name, gender, age, major)

- （请大家投稿）

- 提示：
- db = sqlite3.connect(“university.db”)
- cursor=db.cursor()

参考答案1

```
# 数据库文件是university.db，不存在，则自动创建
with sqlite3.connect('university.db') as db:
# 创建一个cursor:
    cursor = db.cursor()

# 执行一条SQL语句：创建students表
cursor.execute('create table if not exists students( \
    id integer primary key, \
    name text not null, \
    gender text not null, \
    age intger not null, \
    major text not null)')
db.commit()
```

插入数据

- 插入固定数据

```
cursor.execute("""INSERT INTO students(id,name,class,grade)
  VALUES(1,"Mary","Python",67)""")
db.commit()
```

- 插入可变数据

```
newID = input ("Enter ID number: ")
newName = input("Enter name: ")
newClass = input("Enter class: ")
newGrade = input("Enter grade: ")
cursor.execute("""INSERT INTO students(id, name, class, grade)
  VALUES(?, ?, ?, ?)""", (newID, newName, newClass, newGrade))
db.commit()
```

如何将变量中的数据插入SQL

读取数据

- 读取数据

```
# 首先执行SELECT语句从数据库中读取数据
cursor.execute("SELECT * FROM students")
```

- 展示数据
  - 读取之后，数据在哪呢？如何展示出来？
  - `cursor.fetchall()`

```
#打印数据的两种方式：
直接print：    print(cursor.fetchall())
逐个print：    for record in cursor.fetchall():
                   print(record)
```

问题：fetchall()中的顺序是怎样的？

- 关闭数据库
  - 访问完数据后，必须关闭数据库连接
  - `db.close()`

想一想，练一练

- 增加和更新students 表的数据
- 至少有3名同学（周边）

Students 学生信息表
(id, name, gender, age, major)

- 在university 数据库中建立class表
- 增加和更新表的数据

Class 课程表
(class_id, class_name, lecturer, credit)

- 至少插入三门以上课程
- （请大家投稿）

```python
# 插入数据:
genders = ['male', 'female']
majors = ['Math', 'CS', 'Finance', 'Economics']

for i in range(20):
    name = ''.join(random.sample(string.ascii_lowercase, 5))
    gender = genders[random.randint(0, 1)]
    age = random.randint(12, 24)
    major = majors[random.randint(0, 3)]
    cursor.execute('insert into students (id, name,gender,age,major) \
        values ({},"{}","{}",{},"{}")'.format(i+1, name, gender, age, major))
db.commit()
```

```python
# 执行一条SQL语句: 创建class表
cursor.execute('create table if not exists class( \
    class_id integer primary key, \
    class_name text, \
    lecture text, \
    credit integer )')

classes = ['Python', 'Java', 'C++', 'C', 'R', 'Go']
lectures = ['Adam', 'Bob', 'Cyrus', 'Dan', 'Eric', 'Frank']
Credits = [3, 2, 3, 2, 1, 1]
for i in range(6):
    cursor.execute('insert into class (class_id, class_name,lecture,credit) \
        values ({},"{}","{}",{})'.format(i+1, classes[i], lectures[i], Credits[i]
db.commit()
```

Example code

```
cursor.execute("SELECT * FROM students")
for x in cursor.fetchall():
  print(x)
```
Displays all the data from the `students` table and displays each record on a separate line

```
cursor.execute("SELECT * FROM students ORDER By name")
for x in cursor.fetchall():
  print(x)
```
Selects all the data from the `students` table, sorted by name and displays each record on a separate line.

```
cursor.execute("SELECT * FROM students WHERE grade>50")
```
Selects all the data from the `students` table where the grade is over 50.

```
cursor.execute("SELECT * FROM students WHERE class = 'Python'")
```
selects all the data from the `students` table where the class is "Python".

```
cursor.execute("""SELECT students.id, students.name,
students.lecturer
 FROM students, class WHERE students.class=class.class
 AND students.grade > 70""")
```
Selects the ID and name fields from the `students` table and the lecturer field from the `class` table if the grade is over 70.

```
cursor.execute("SELECT id, name, grade FROM students")
```
Selects the ID, name and grade from the `students` table.

Example code

```
whichClass = input(Enter a class: ”)
cursor.execute(“SELECT * FROM employees WHERE class=?”, [whichClass])
for x in cursor.fetchall():
   print (x)
```
allows the user to enter a class and displays the records of all the students in that class.

```
cursor.execute(“””SELECT students.id, students.name, class.lecturer
  FROM students,class WHERE students.class= class.class”””)
```
selects the ID and name fields from the students table and the lecturer filed from the class table, using the class filed to link the data. If you do not specify how the tables are linked, Python will assume every students takes every class and you will not get the results you are expecting.

```
cursor.execute(“UPDATE students SET name = ‘Richard’ WHERE id =1”) db.commit()
```
updates the data in the table(overwriting the original) to change the name to “Richard” for student ID 1

```
cursor.execute(“DELETE students WHERE id=1”)
```
deletes any data in the `students` table where the id is 1

想一想，练一练

- 在university 数据库中创建选课表enrolled表
- 增加和更新表的数据
- 至少包括：学生ID，选课的课号，考试分数。
- （请大家投稿）

Enrolled选课表
(student_id, class_id, score)

参考答案

```
# 执行一条SQL语句：创建enrolled表
cursor.execute('create table if not exists enrolled( \
    student_id integer,  \
    class_id integer, \
    credit integer, \
    score integer, \
    primary key(student_id,class_id) )')

for i in range(20):
    student_id = i+1
    for j in range(random.randint(1, 6)):
        class_id = j+1
        credit = Credits[j]
        score = random.randint(0, 100)
        cursor.execute('insert into enrolled (student_id,class_id, credit,score)
            values ({},{},{},{})'.format(student_id, class_id, credit, score))
db.commit()
```

Tables

- Students 信息表
  - (id, name,gender,age..)
- Class 课程表
  - (class_id,class_name, lecturer, credit)
- Enrolled选课表
  - (student_id,class_id,score)

想一想，练一练

- 使用 INNER JOIN 查询所有学生的个人信息和所选的每个课程的分数：
  - id, name, gender, class_name, credit, score

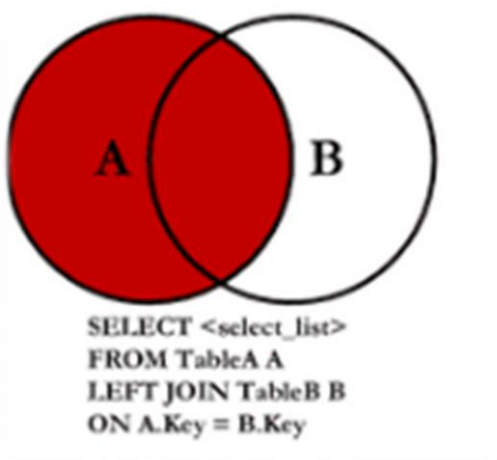
- (请大家投稿)

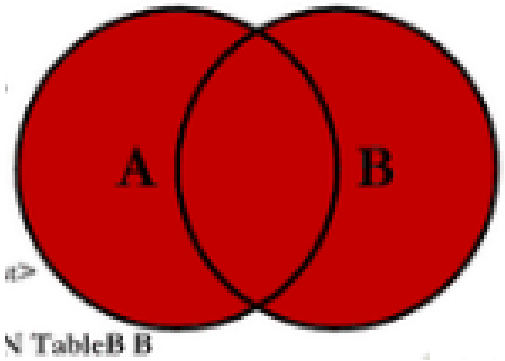
参考答案

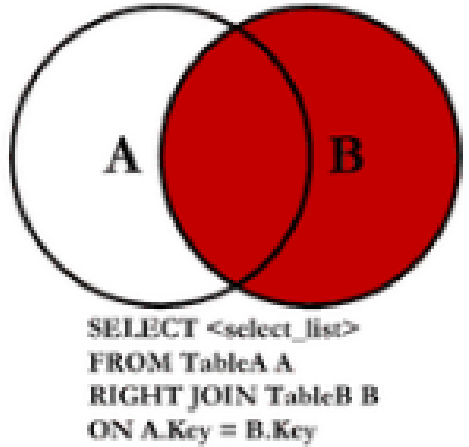
```
# 查询学生的个人信息及分数
cursor.execute('select id,name,gender,age,major,class_name,credit,score \
                from students s inner join\
                    (select student_id,class_name,c.credit,score from class c
                        inner join enrolled e on c.class_id = e.class_id)tc \
                on s.id =tc.student_id')
print('---------------inner join--------------------')
for x in cursor.fetchall():
    print(x)
```

想一想，练一练

- 尝试outer join操作， left join, right join 和full JOIN.
- 思考： sqlite 只有left join,如何实现 right join和full join







Python

PYTHON PANDAS SQLITE

Pandas-本地数据库SQLite访问

- Pandas读取SQLite的表

```
import pandas as pd
from sqlalchemy import create_engine
db = create_engine('sqlite:///:memory:')
with db.connect() as conn, conn.begin():
    data = pd.read_sql_table('data', conn)
data.to_sql('data', db)
```

- https://pandas.pydata.org/docs/user_guide/io.html#sql-queries

想一想，练一练

- Pandas直接读取在university 数据库中enrolled表
- （请大家投稿）

参考答案

Pandas直接读取在university 数据库中enrolled表

[14]:
```
import pandas as pd
from sqlalchemy import create_engine
db = create_engine('sqlite:///university.db')
with db.connect() as conn, conn.begin():
    data = pd.read_sql_table('enrolled', conn)
    print(data)
```

```
    student_id  class_id  credit  score
0            1         1       3     37
1            1         2       2      1
2            1         3       3     71
3            1         4       2     60
4            1         5       1     45
..         ...       ...     ...    ...
76          18         6       1     38
77          19         1       3     36
78          20         1       3     95
79          20         2       2     29
80          20         3       3     62

[81 rows x 4 columns]
```

谢谢指正！