# 第十四章 数据库模块操作

本节所讲内容：

14.1 pymysql

14.2 peewee数据库模块映射

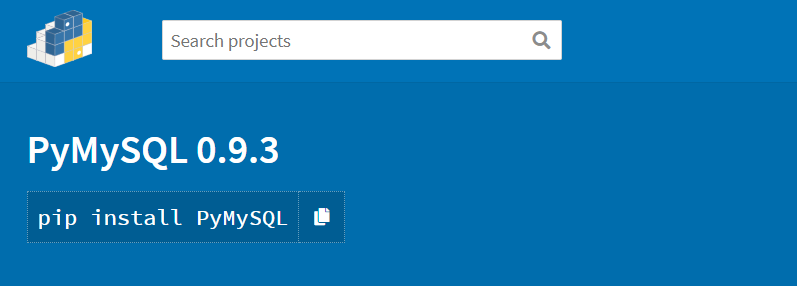
## 14.1 pymysql

### 14.1.1 介绍

PyMySQL 是在 Python3.x 版本中用于连接 MySQL 服务器的一个库，Python2中则使用mysqldb。

python2.x 版本当中运用 MySQLdb

python3.x 版本当中运用 pymysql



### 14.1.2 安装

1、普通安装

pip3 install pymysql

或者是

pip3 install mysql-python

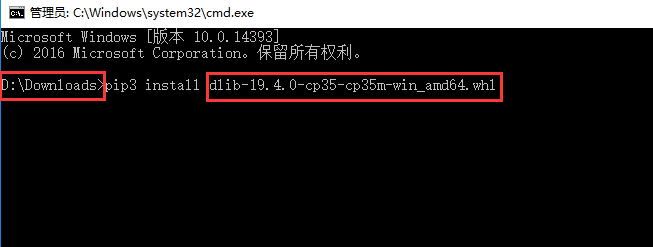
安装以后在cmd的python交互命令窗口下进行导入

>>>import pymysql

完成导入没有报错说明安装成功。

2、轮子安装

1. 轮子网站：<https://www.lfd.uci.edu/~gohlke/pythonlibs/#wordcloud>
2. 安装wheel 模块  
   pip3 install wheel（安装whl文件必备模块）
3. 启动cmd ，cd到whl文件的下载目录。  
   pip3 install whl文件名



3、换源安装

网络不好：换源一步解决

网络是不是完好

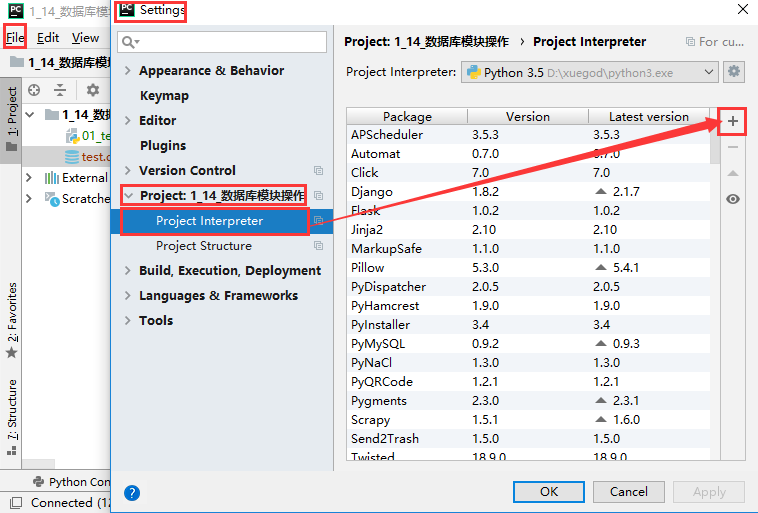
我们可以用pip install pymysql –i 国内源

阿里云 <http://mirrors.aliyun.com/pypi/simple/>  
中国科技大学 [https://pypi.mirrors.ustc.edu.cn/simple/](https://pypi.mirrors.ustc.edu.cn/simple/%20)  
豆瓣(douban) <http://pypi.douban.com/simple/>   
清华大学 <https://pypi.tuna.tsinghua.edu.cn/simple/>  
中国科学技术大学 <http://pypi.mirrors.ustc.edu.cn/simple/>

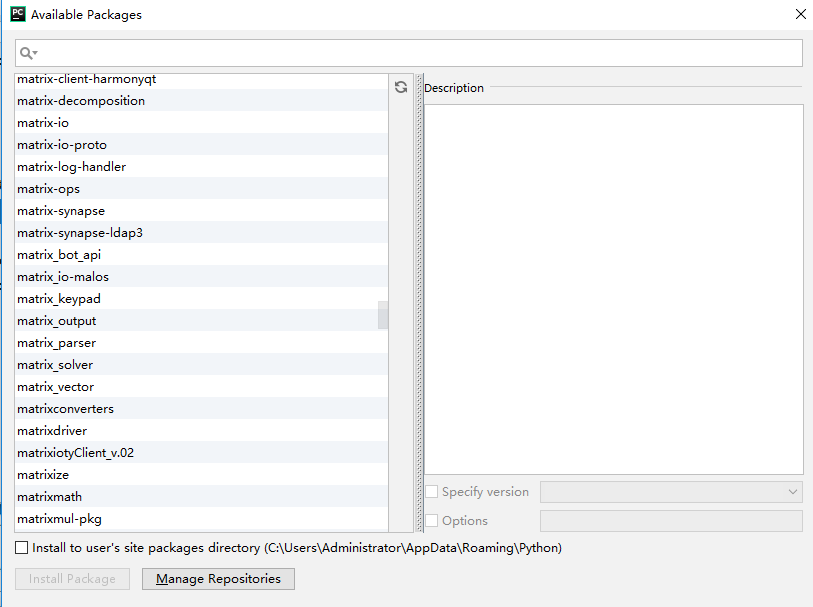
注意：轮子网站没有pymysql，只有mysql-python，脚本禁止起名为MySQLdb，否则模块属性错误，等第三模块重名的文件名。

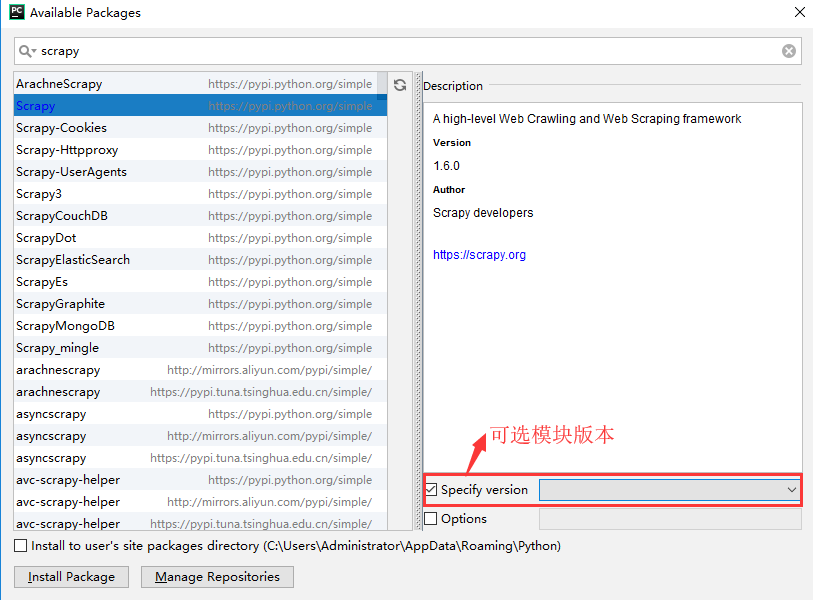
4、pycharm下安装

Pycharm 安装的方法比较方便，在编写程序的时候，可以让它智能安装，但是这个方法可能会出现一些的错误，大家看自己的需求来找到不同模块的比较合适的安装方式。



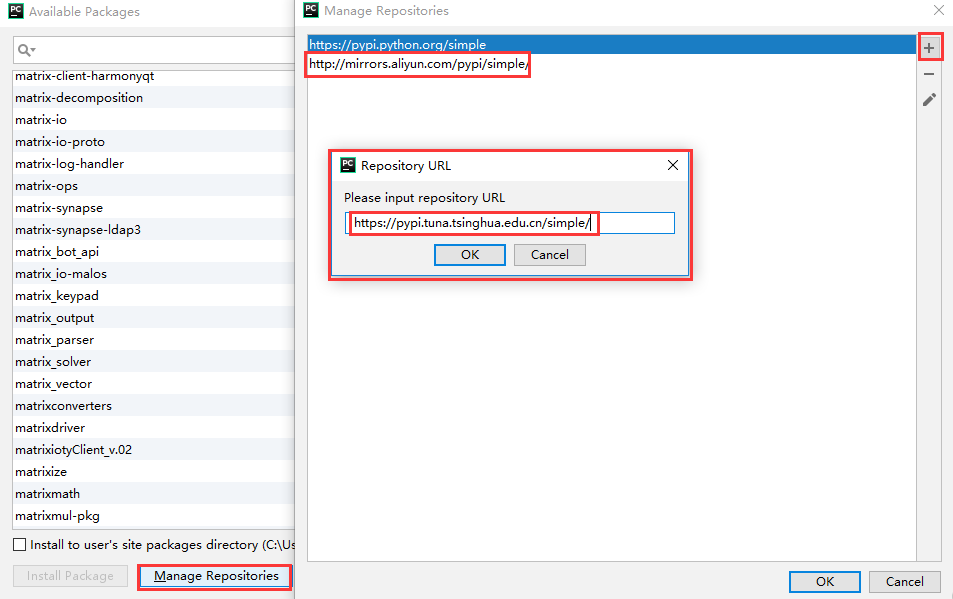
点击加号会出现这个模块安装界面，可以在索引地方搜索下自己想要的模块





为了更好更加快速的安装自己想要的模块，我们可以简单的调入一些国内源。

阿里云 <http://mirrors.aliyun.com/pypi/simple/>  
中国科技大学 [https://pypi.mirrors.ustc.edu.cn/simple/](https://pypi.mirrors.ustc.edu.cn/simple/%20)  
豆瓣(douban) <http://pypi.douban.com/simple/>   
清华大学 <https://pypi.tuna.tsinghua.edu.cn/simple/>  
中国科学技术大学 <http://pypi.mirrors.ustc.edu.cn/simple/>

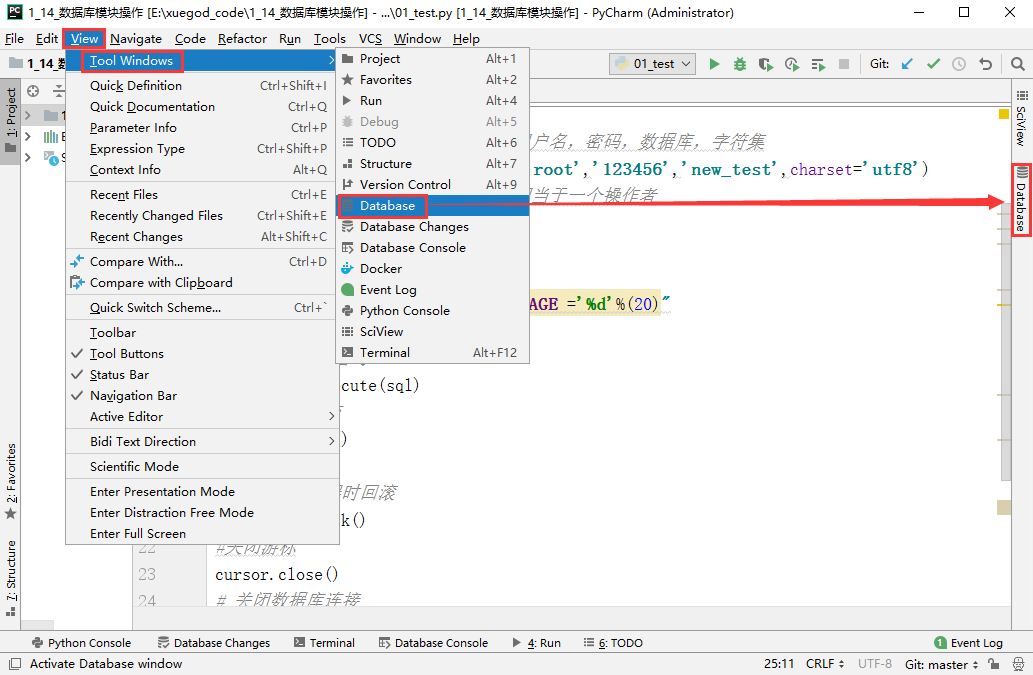


添加国内源让模块下载更加快速.

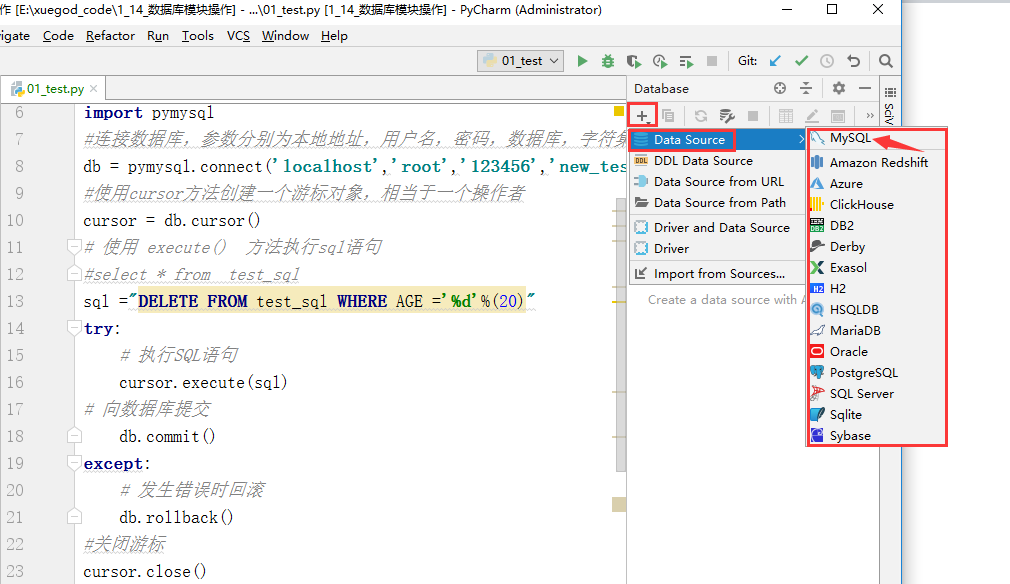
### 14.1.3 Pymysql 的操作步骤

1、pycharm展示数据库配置

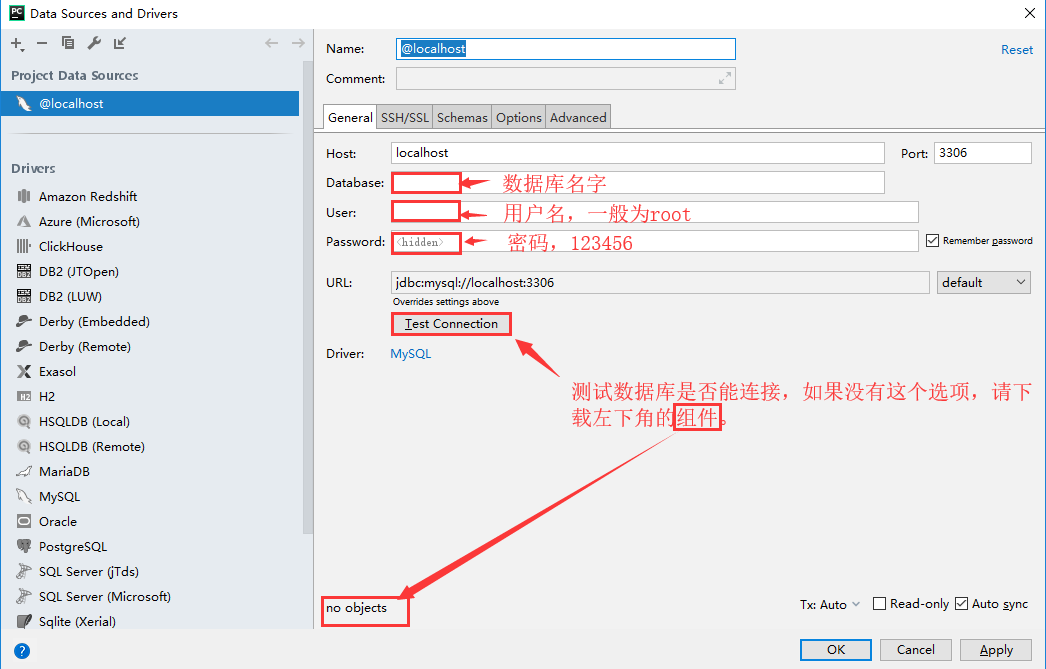
在我们学习这个模块之前，我们需要让自己的pycharm链接好数据库，方便我们进一步操作，让我们更加直观化的了解，pymysql模块操作后达到的效果。配置具体如下：



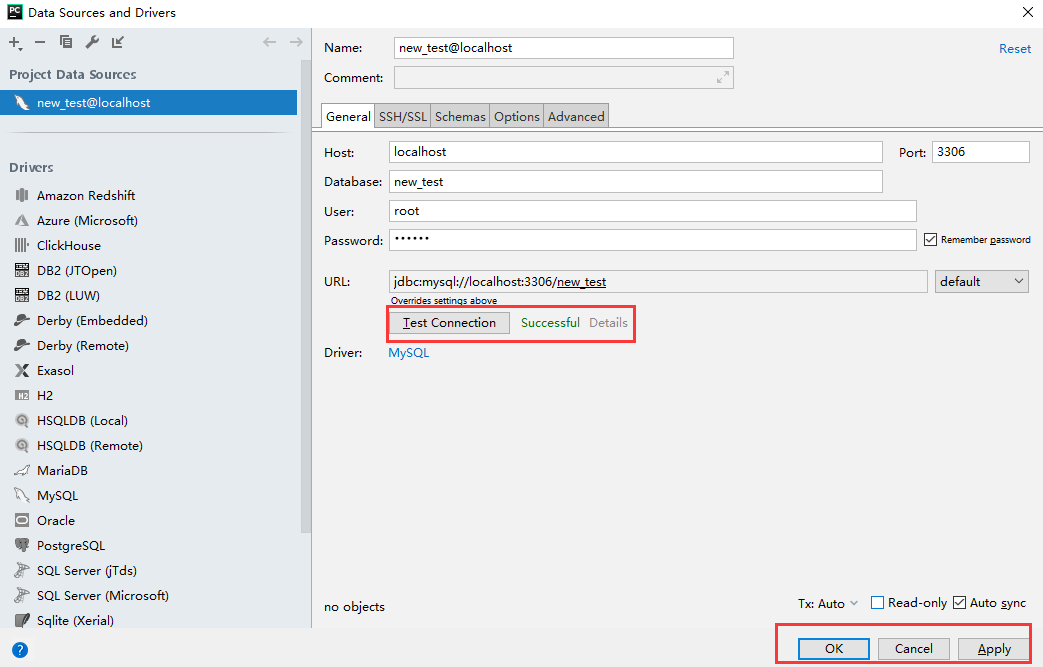
点击databases会弹出这个界面，我们点击+号，然后找到data source 点击后会出现各个数据库

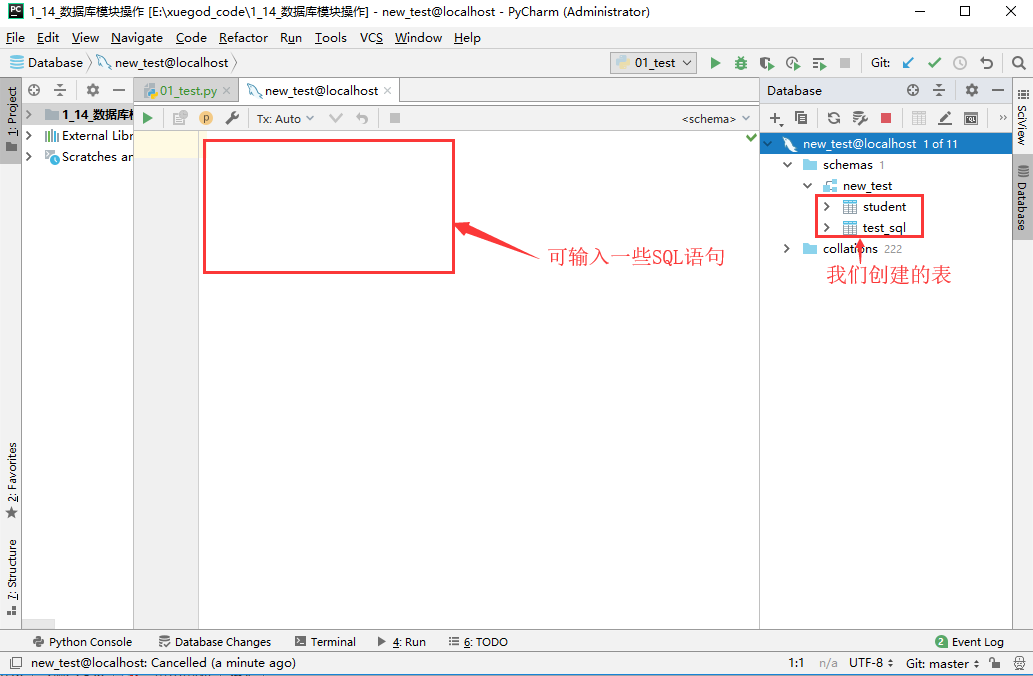


找到自己想要的数据库，比如说MySQL点击



然后测试成功点击apply之后ok就可以了！





注意这是数据库连接的状态，如果不想连接请点击  按钮。

2、操作关键字

创建连接 connect

实例化游标 cursor

执行sql语句 execute

提交修改 commit

事务回滚 rollback

关闭游标和链接 close

3、脚本连接运行

import pymysql  
*#连接数据库，参数分别为本地地址，用户名，密码，数据库，字符集*db = pymysql.connect('localhost','root','123456','new\_test',charset='utf8')  
*#使用cursor方法创建一个游标对象，相当于一个操作者*cursor = db.cursor()  
*#编写的sql语句*sql = '''create table test\_sql(  
id INT PRIMARY KEY auto\_increment,  
name VARCHAR (30),  
age INT   
) '''  
*#使用execute方法执行sql语句，相当于操作者在mysql命令中输入SQL语句并回车*cursor.execute(sql)  
*#关闭游标*cursor.close()  
*#关闭数据库连接*db.close()

4、connect 参数详解：

 host=None, # 要连接的主机地址

 user=None, # 用于登录的数据库用户

 password='', # 密码

 database=None, # 要连接的数据库

 port=0, # 端口，一般为 3306

 charset='', # 字符编码

 conv=None, # 转换字典

 use\_unicode=None, # 是否使用 unicode 编码

 init\_command=None, # 连接建立时运行的初始语句

 connect\_timeout=10, # 连接超时时间，(default: 10, min: 1, max: 31536000)

 autocommit=False, # 是否自动提交事务

 db=None, # 同 database，为了兼容 MySQLdb

 passwd=None, # 同 password，为了兼容 MySQLdb

 local\_infile=False, # 是否允许载入本地文件

 read\_timeout=None, # 读取超时时间

 write\_timeout=None, # 写入时间

5、增添数据

import pymysql  
*#连接数据库，参数分别为本地地址，用户名，密码，数据库，字符集*db = pymysql.connect('localhost','root','123456','new\_test',charset='utf8')  
*#使用cursor方法创建一个游标对象，相当于一个操作者*cursor = db.cursor()  
*#使用execute方法执行sql语句，相当于操作者在mysql命令中输入SQL语句并回车*cursor.execute("insert into test\_sql(name,age) value ('for',19)")  
*#提交给数据库，主要配合为增加删除修改这些操作*db.commit()  
*#关闭游标*cursor.close()  
*#关闭数据库连接*db.close()

6、删除

import pymysql  
*#连接数据库，参数分别为本地地址，用户名，密码，数据库，字符集*db = pymysql.connect('localhost','root','123456','new\_test',charset='utf8')  
*#使用cursor方法创建一个游标对象，相当于一个操作者*cursor = db.cursor()  
*# 使用 execute() 方法执行sql语句  
#drop的时候提不提交都行，delect 的时候要commit  
#delete from tab\_name where id =1*cursor.execute("drop table test\_sql")  
*#提交给数据库，主要配合为增加删除修改这些操作*db.commit()  
*#关闭游标*cursor.close()  
*#关闭数据库连接*db.close()

7、自主更改：

import pymysql  
*#打开数据库连接，charset ='utf-8',  
#本地地址，用户名，密码，数据库名*db = pymysql.connect("127.0.0.1","root","123456","new\_test")  
*# db = pymysql.connect  
#使用游标，创建一个游标对象*cursor = db.cursor()  
*#普通的sql语句  
# sql = "create table meinv\_test(id int PRIMARY KEY auto\_increment,name VARCHAR (30),age INT )"*sql = "insert into test\_sql(name,age) VALUE (%s,%d)"  
while True:  
 name = input('请输入姓名：')  
 age = int(input('输入年龄:'))  
 *#execute()这行SQL语句* cursor.execute(sql,[name,age])  
 *#提交* db.commit()  
*#关闭游标*cursor.close()  
*#关闭数据库*db.close()

8、更改

import pymysql  
*#连接数据库，参数分别为本地地址，用户名，密码，数据库，字符集*db = pymysql.connect('localhost','root','123456','new\_test',charset='utf8')  
*#使用cursor方法创建一个游标对象，相当于一个操作者*cursor = db.cursor()  
*#使用execute方法执行sql语句，相当于操作者在mysql命令中输入SQL语句并回车*cursor.execute("update test\_sql set name ='django'where id = 1")  
*#提交给数据库，主要配合为增加删除修改这些操作*db.commit()  
*#关闭游标*cursor.close()  
*#关闭数据库连接*db.close()

9、数据库查询操作

Python查询Mysql使用 fetchone() 方法获取单条数据, 使用fetchall() 方法获取多条数据。

fetchone(): 该方法获取下一个查询结果集。结果集是一个对象

fetchall(): 接收全部的返回结果行.

rowcount: 这是一个只读属性，并返回执行execute()方法后影响的行数。

import pymysql  
*#连接数据库，参数分别为本地地址，用户名，密码，数据库，字符集*db = pymysql.connect('localhost','root','123456','new\_test',charset='utf8')  
*#使用cursor方法创建一个游标对象，相当于一个操作者*cursor = db.cursor()  
*# 使用 execute() 方法执行sql语句  
#select \* from test\_sql*cursor.execute("select *\** from test\_sql ")  
*# 使用 fetchone() 方法获取单条数据.  
# fetchall() 查取所有，以元组的形式返回*data=cursor.fetchone()  
*#打印data*print(data)  
*#关闭游标*cursor.close()  
*# 关闭数据库连接*db.close()

fetchone()方法的运行结果如下:

(1, 'django', 19)

Fetchall()方法的运行结果如下:

((1, 'django', 19), (2, 'for', 19), (3, '小龙女', 14))

10、事务回滚rollback()

事务应该具有4个属性：原子性、一致性、隔离性、持久性。这四个属性通常称为ACID特性。

原子性（atomicity）。一个事务是一个不可分割的工作单位，事务中包括的诸操作要么都做，要么都不做。

一致性（consistency）。事务必须是使数据库从一个一致性状态变到另一个一致性状态。一致性与原子性是密切相关的。

隔离性（isolation）。一个事务的执行不能被其他事务干扰。即一个事务内部的操作及使用的数据对并发的其他事务是隔离的，并发执行的各个事务之间不能互相干扰。

持久性（durability）。持续性也称永久性（permanence），指一个事务一旦提交，它对数据库中数据的改变就应该是永久性的。接下来的其他操作或故障不应该对其有任何影响。

Python DB API 2.0 的事务提供了两个方法 commit 或 rollback。

案例：

sql ="DELETE FROM test\_sql WHERE AGE ='%d'%(20)"  
try:  
 *# 执行SQL语句* cursor.execute(sql)  
*# 向数据库提交* db.commit()  
except:  
 *# 发生错误时回滚* db.rollback()

mysql中比较常用搜索引擎？(面试)

在这里咱们主要了解一下，MySQL有两大搜索引擎，一个是InnoDB 、MyISAM他们的具体区别是：MyISAM类型不支持事务处理等高级处理，而InnoDB类型支持。MyISAM类型的表强调的是性能，其执行速度比InnoDB类型更快，但是不提供事务支持，而InnoDB提供事务支持以及外部键等高级数据库功能。

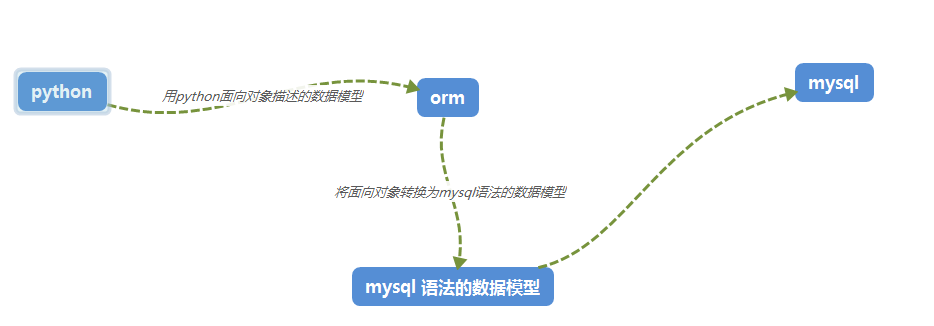
mysql>show engines； 查看数据库引擎

## 14.2 peewee数据库模块映射

### 14.2.1 peewee介绍

我们用上面的代码进行数据库操作是可行的，但是也存在很多的问题,假如开发人员不懂mysql,有时候我们随着业务需求的变换，而转换另一种数据库，进行操作,我们当前的pymysql对数据库数据的插入、删除和修改不方便。

在这种情况下，我们提出orm(Object Relational Mapping数据库映射)概念，是一种程序技术，用于实现面向对象编程语言里不同类型系统的数据之间的转换。从效果上来说，它其实是创建了一个可在编程语言里使用的“虚拟对象数据库”。



### 14.2.2 peewee安装

Peewee模块是第三方模块需要进行pip安装；

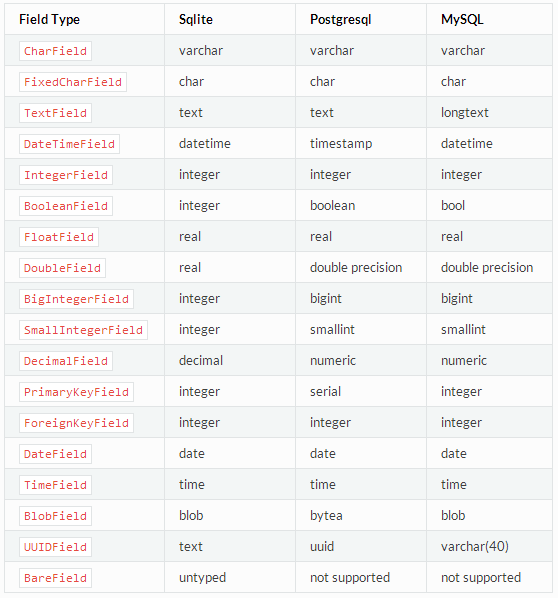
peewee 支持三种数据库

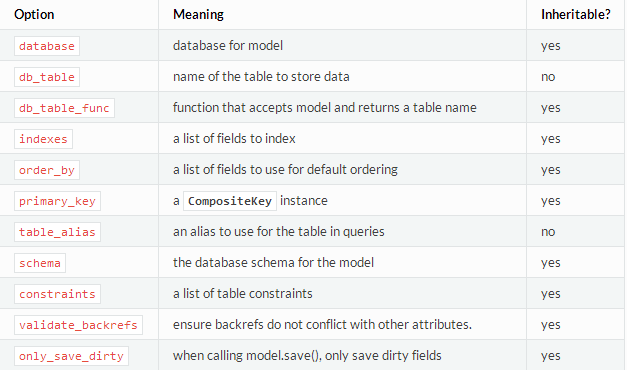
Mysql

Sqllite

Postgresql（了解一下）

### 14.2.3 类型转换



Class Meta  


### 14.2.4 peewee连接mysql

Peewee和mysql数据库进行连接，生成数据库

import peewee  
import datetime  
*#连接数据库*connect = peewee.MySQLDatabase(  
 database = 'new\_test',  
 host = '127.0.0.1',  
 user = 'root',  
 passwd = '123456'  
)  
*#继承peewee.Model，创建一个表。  
#peewee创建数据库的时候，默认会添加主键id  
#peewee创建数据库字段默认不可为空*class School(peewee.Model):  
 name = peewee.CharField(max\_length=20,default='for')  
 address = peewee.CharField(max\_length=30,default='xuegod')  
 age = peewee.IntegerField(default=18)  
 birthday = peewee.DateTimeField(default=datetime.datetime.now())  
 *#将表和数据库连接* class Meta:  
 database = connect  
if \_\_name\_\_ == '\_\_main\_\_':  
 *#创建表，创建多个表的时候用列表[]* School.create\_table()  
 *#插入数据* s = School.create(name='for',age=12,birthday = '2017-10-10')  
 s.save()  
 *# 第二种插入方法* School.insert(name = '小龙女',age = 18,birthday = '2018-6-12').execute()  
  
 *#更新数据  
 # School.update(name = '杨过',age = 10,birthday ='2018-5-10').where(School.id==1).execute()  
  
 #删除数据  
 # s = School.get(name = '杨过')  
 # s.delete\_instance()  
 #第二种删除数据  
 # School.delete\_by\_id(2)  
 # School.delete().where(School.id == 5).execute()  
  
 #查询语句  
 # s = School.select()  
 # for i in s:  
 # print(i.name,i.age)  
  
 # s = School.get(School.id == 3)  
 # print(s.name,s.age)  
  
 #有条件的查询  
 # s = School.select().where(School.id == 3)  
 # for i in s:  
 # print(i.name)  
  
 #正序查询，倒序查询* s = School.select().order\_by(School.id.asc())  
 s = School.select().order\_by(School.id.desc())  
 for i in s:  
 print(i.age)

### 14.2.5 peewee和 sqlite3 使用

Sqllite3 是一款nosql的轻量级数据库，python 2.7之后自动携带sqllite3，让我们直接上代码给大家展示。

import peewee  
*#直接运行会自动创建寻找*db = peewee.SqliteDatabase('sql.db')  
*#继承peewee.model创建表*class Teacher(peewee.Model):  
 *#创建具体的字段* name = peewee.CharField(max\_length=20,default='for')  
 age = peewee.IntegerField()  
 *#连接指向数据库db* class Meta:  
 database = db  
if \_\_name\_\_ == '\_\_main\_\_':  
 *#创建表* Teacher.create\_table()  
 *#增加* T = Teacher()  
 T.name = 'for'  
 T.age = 18  
 T.save()  
 *#增加* T = Teacher().insert(  
 name = '小龙女',  
 age = 18,  
 )  
 T.execute()  
 *#删除* T = Teacher.delete().where(Teacher.id == 1)  
 T.execute()  
 *#修改* T = Teacher.update(name = 'for').where(Teacher.id ==1)  
 T.execute()  
 T = Teacher().get(id = 2)  
 T = Teacher().get\_by\_id(2)  
 T.name = '杨过'  
 T.save()  
 *#查找* T\_list = Teacher.select()  
 for i in T\_list:  
 print(i.name,i.age)  
 T\_list = Teacher.select().order\_by(Teacher.age)  
 for i in T\_list:  
 print(i.name,i.age)  
 *#查一条* T\_list = Teacher.select().where(Teacher.age == 18)  
 for i in T\_list:  
 print(i.name,i.age)  
 T = Teacher.get(id = 2)  
 print(T.name,T.age)

总结：

14.1 pymysql

14.2 peewee数据库模块映射