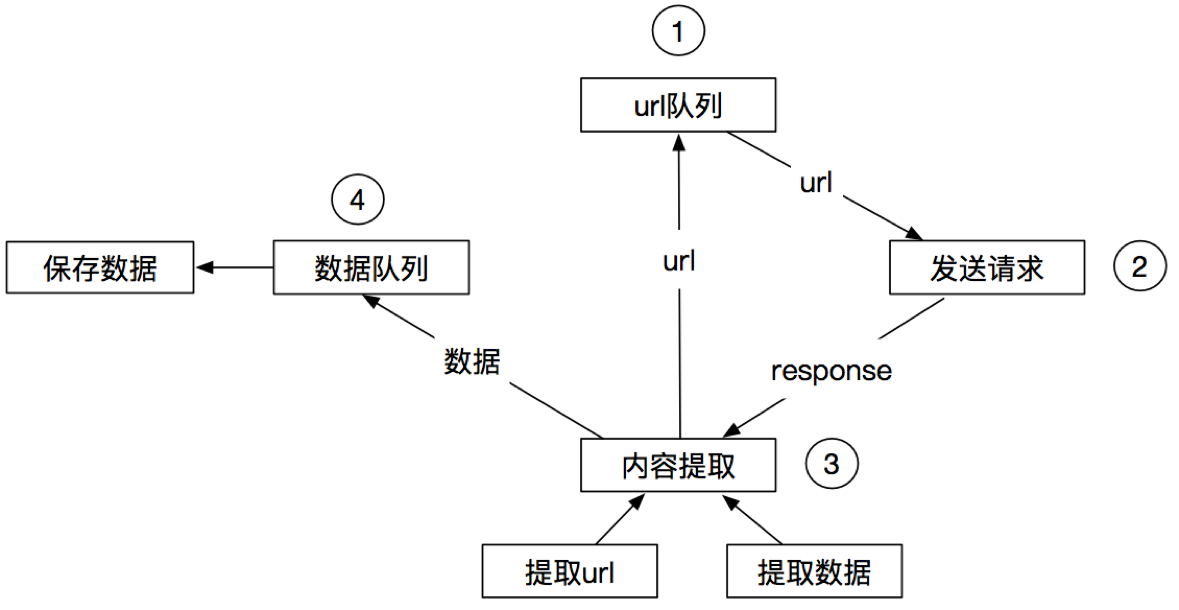
**实验 豆瓣动漫类电影评论爬取**

## 一、实验目的

1. 熟悉Python编程语言，学会安装Python开发环境，编写第一个Python程序，并学会使用IDE（比如PyCharm）进行Python开发；
2. 了解网络爬虫，掌握爬虫的原理；
3. 能够根据需求独立开发爬虫代码，掌握sql基础语法，将爬取的数据存储到数据库中；
4. 了解如何管理开发环境，使用pip等环境管理工具或者使用虚拟环境进行工程作业；
5. 了解关于爬虫的最新研究热点技术，比如反爬措施与防反爬措施等。

## 二、实验原理

**1. 实验流程图：** ：

**2. 爬虫原理：**

答：网络爬虫又名“网络蜘蛛”，是通过网页的链接地址来寻找网页，从网站某一个页面开始，读取网页的内容，找到在网页中的其它链接地址，然后通过这些链接地址寻找下一个网页，这样一直循环下去，直到按照某种策略把互联网上所有的网页都抓取完为止的技术。爬取的数据可以用来进行数据分析、机器学习数据训练等。

## 三、基础知识

1. **计算机网络（在浏览器中输入一个url后回车，后台会发生什么？）**

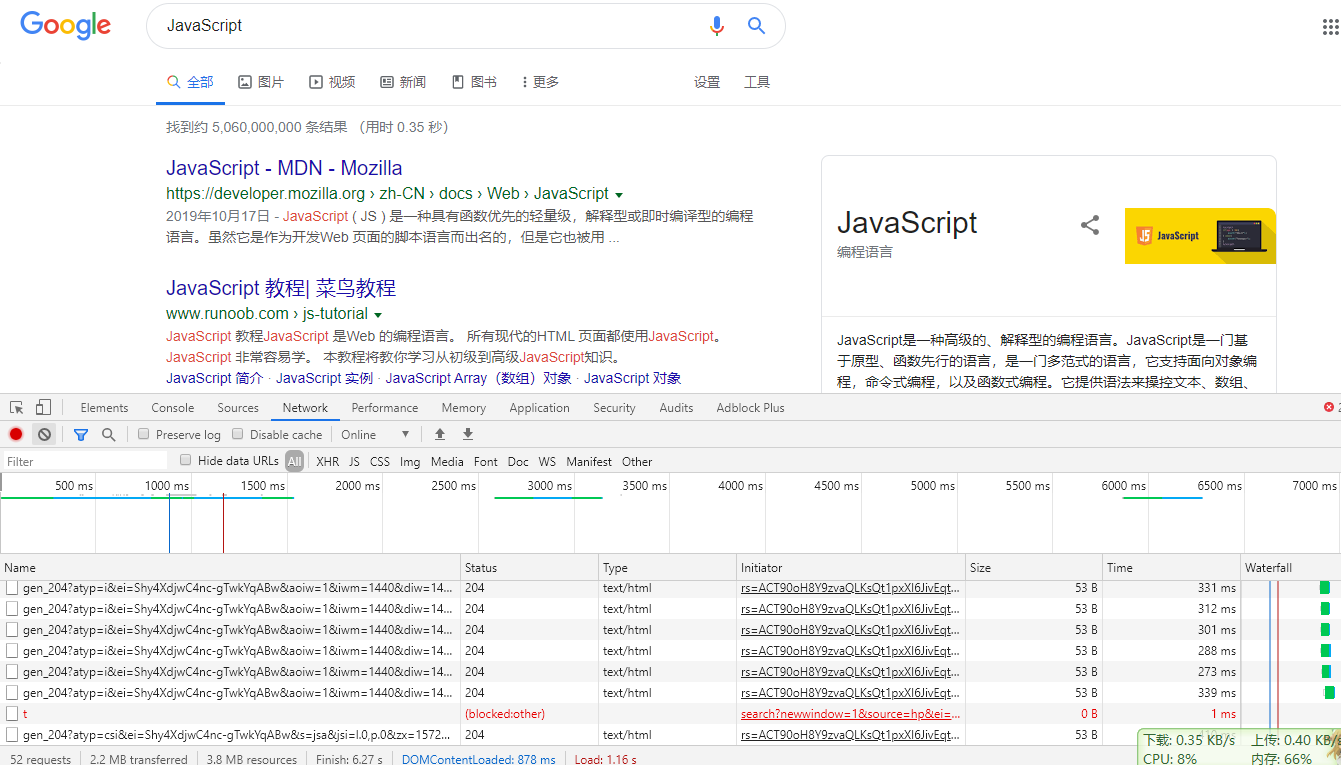
**答**：主要分为以下几个过程：

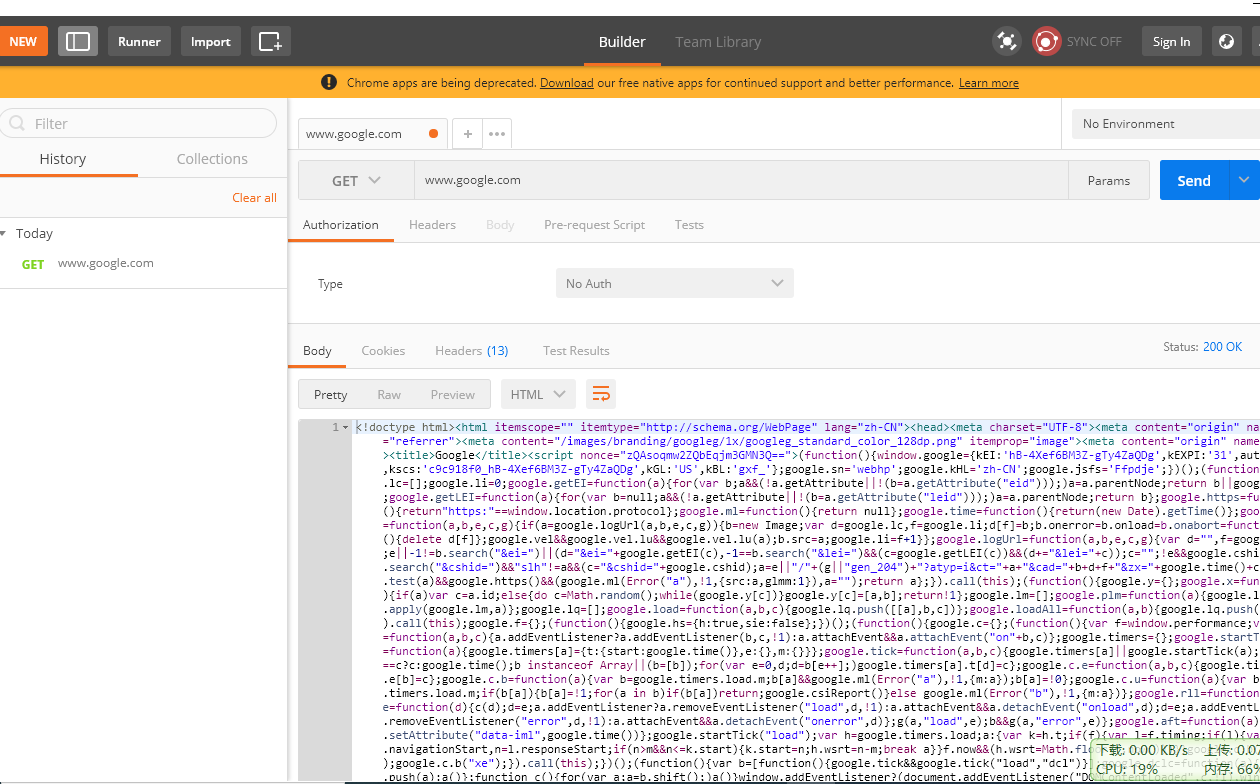
1. DNS 解析:将域名解析成 IP 地址；
2. TCP 连接：TCP 三次握手；
3. 发送 HTTP 请求；
4. 服务器处理请求并返回 HTTP 报文；
5. 浏览器解析渲染页面，到前端展示出来；
6. 断开连接：TCP 四次挥手；
7. **HTML/CSS/JS：**

**答：**这些主要是前端开发的核心基础知识，其中HTML是超文本标记语言，构成了网页的结构。CSS是层叠样式表，规定了网页的样式。而JavaScript 是一种脚本，一门编程语言，它可以在网页上实现复杂的功能，网页展现给你的不再是简单的静态信息，而是实时的内容更新，交互式的地图，2D/3D 动画，滚动播放的视频等等。

1. **Chrome浏览器开发者工具的使用：**

**答：**Chrome浏览器是谷歌公司开发的浏览器，也是现行前端开发的浏览器通用标准，我们可以通过F12来查看所有的网络请求情况，也可以编写我们自己的JavaScript脚本执行。同时可以安装Chrome插件来强化浏览器的功能，比如postman插件，可以模拟浏览器发送get、post等请求。





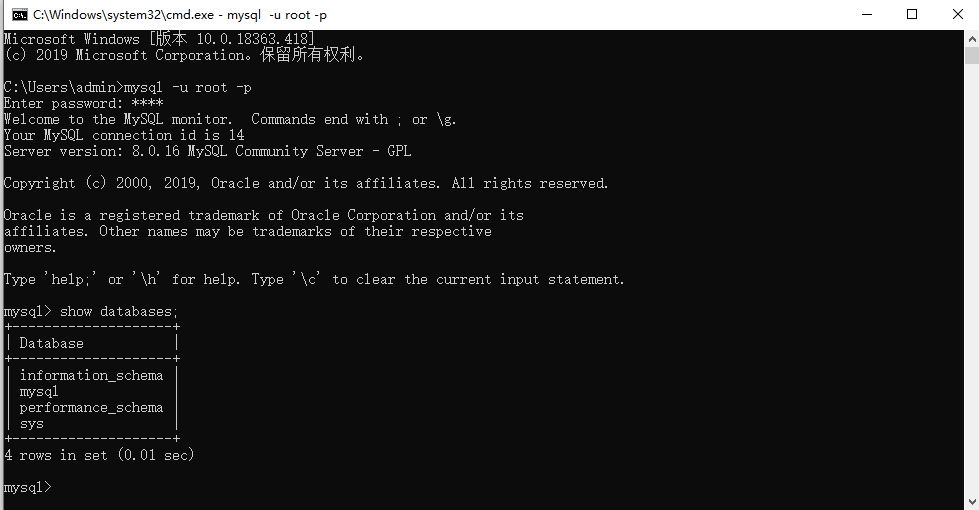
1. **Python基础语法及常用库：**

**答：**通过阅读Python官方文档，可以详细了解Python的语法细节情况。Python 是一种解释型、面向对象、动态数据类型的高级程序设计语言。Python中常见的数据结构可以统称为容器（container）。序列（如列表和元组）、映射（如字典）以及集合（set）是三类主要的容器。

与爬虫有关的库： Requests、 BeautifulSoup、 lxml、 numpy、 pandas等。

1. **MySQL数据库：**

**答：**MySQL 是最流行的关系型数据库管理系统，在 WEB 应用方面 MySQL 是最好的 RDBMS(Relational Database Management System：关系数据库管理系统)应用软件之一，也是目前使用最广泛的数据库。在非关系型数据库领域，常用的有MongoDB、Redis等。其中MongoDB是文档型数据库，扩展非常方便。Redis是内存型基于Key-Value的数据库，速度很快，常用来做缓存、消息队列等。自己经常用到的主要是以上三个。安装方法可以参考网上教程，安装成功后可以配置一下环境变量，MySQL端口号一般设置为3306。



1. **开发环境选择：**

**IDE：**Pychram 2019.02

**包管理方式:** Pip

**编辑器：**Python 3.7版本

**数据库：**MySQL 8.0.16版本

**数据库可视化工具：**Navicat 12版本

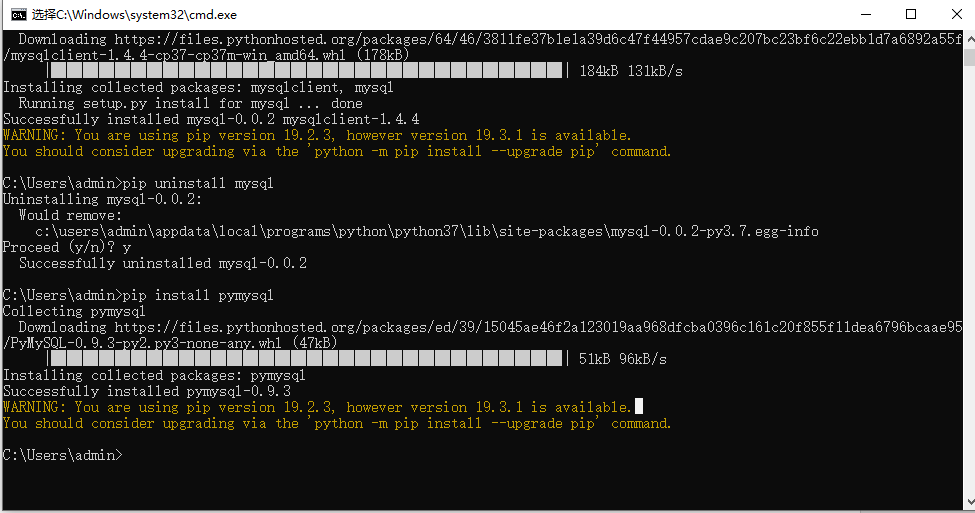
先通过Pip的方式安装常见的爬虫库：

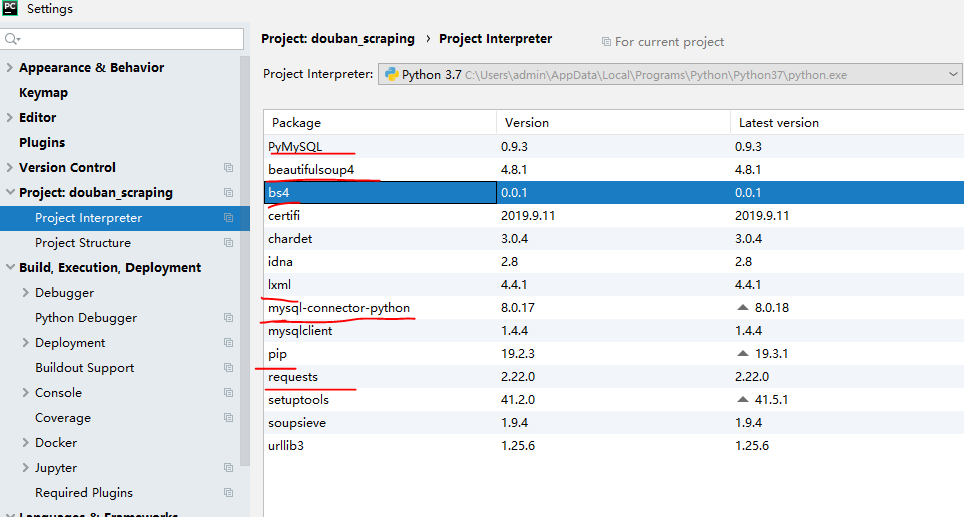
* pip install requests
* pip install bs4
* pip install lxml
* pip install re

由于Python 没有内置的 MySQL 支持工具。不过，有很多开源的库可以用来与 MySQL 做交互，Python 2.x 和 Python 3.x 版本都支持。最有名的一个库就是 PyMySQL，然后安装pymysql。

* pip install pymysql

然后测试本地数据库的连接，即可。



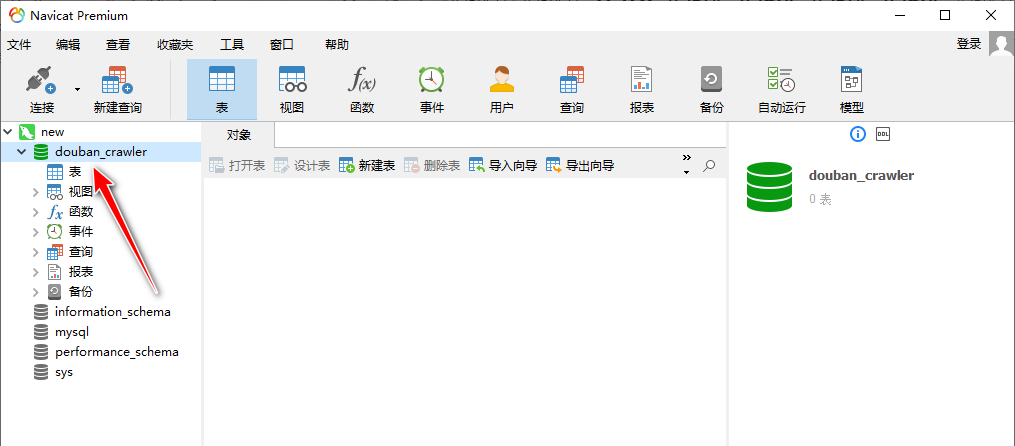




## 四、实验过程

先创建数据库表，然后再分析具体的爬取步骤：

CREATE DATABASE  `douban\_crawler` DEFAULT CHARACTER SET utf8 COLLATE utf8\_general\_ci;



再新建数据表“douban”：

    CREATE TABLE IF NOT EXISTS `douban` (

    `name` VARCHAR ( 100 ) NOT NULL,

    `comment\_content` VARCHAR ( 255 ) DEFAULT NULL,

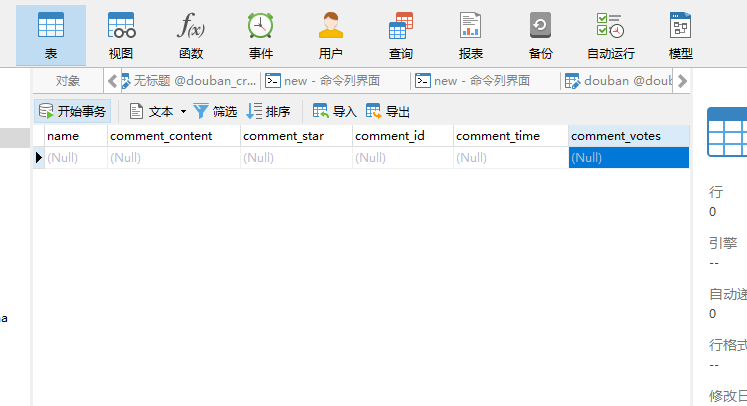
    `comment\_star` VARCHAR ( 50) DEFAULT NULL,

    `comment\_id` VARCHAR ( 50 ) DEFAULT NULL,

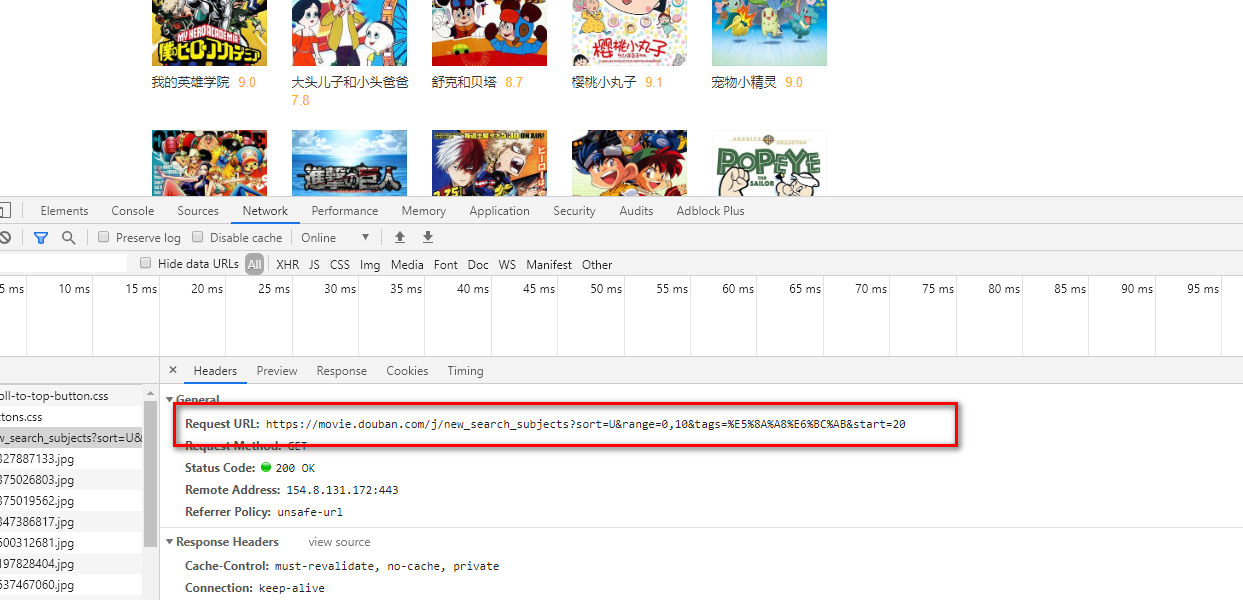
    `comment\_time` VARCHAR ( 50 ) DEFAULT NULL,

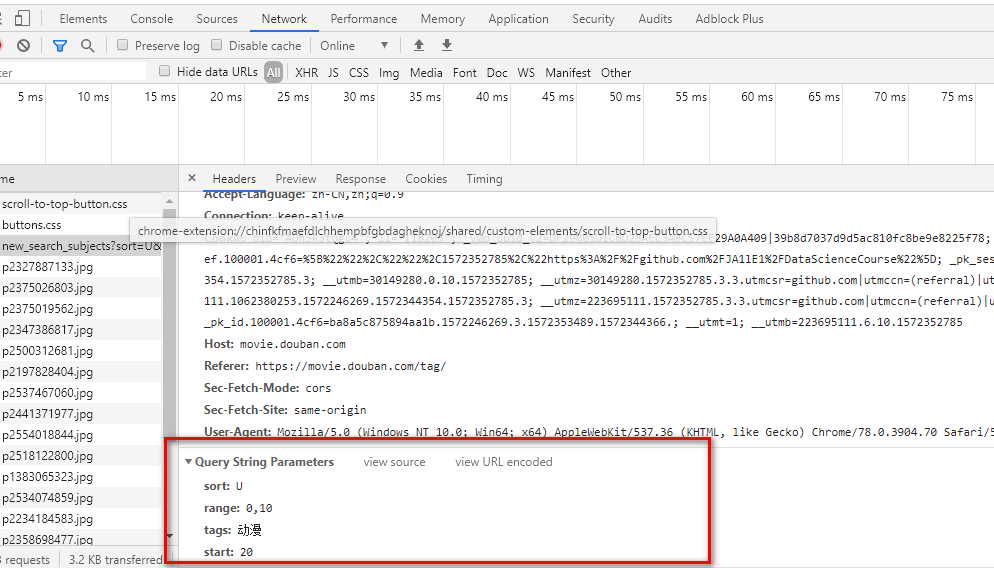
    `comment\_votes` VARCHAR ( 10 ) DEFAULT NULL

    ) ENGINE = INNODB DEFAULT CHARSET = utf8;



分析网页结构，通过打开F12，然后刷新网页，分析请求头部的数据，很容易找到网页异步加载的规律：





第一页内容为：<https://movie.douban.com/j/new_search_subjects?sort=U&range=0,10&tags=%E5%8A%A8%E6%BC%AB&start=0>

第二页内容为：

<https://movie.douban.com/j/new_search_subjects?sort=U&range=0,10&tags=%E5%8A%A8%E6%BC%AB&start=20>

通过分析多次异步加载的网页请求可以看出，

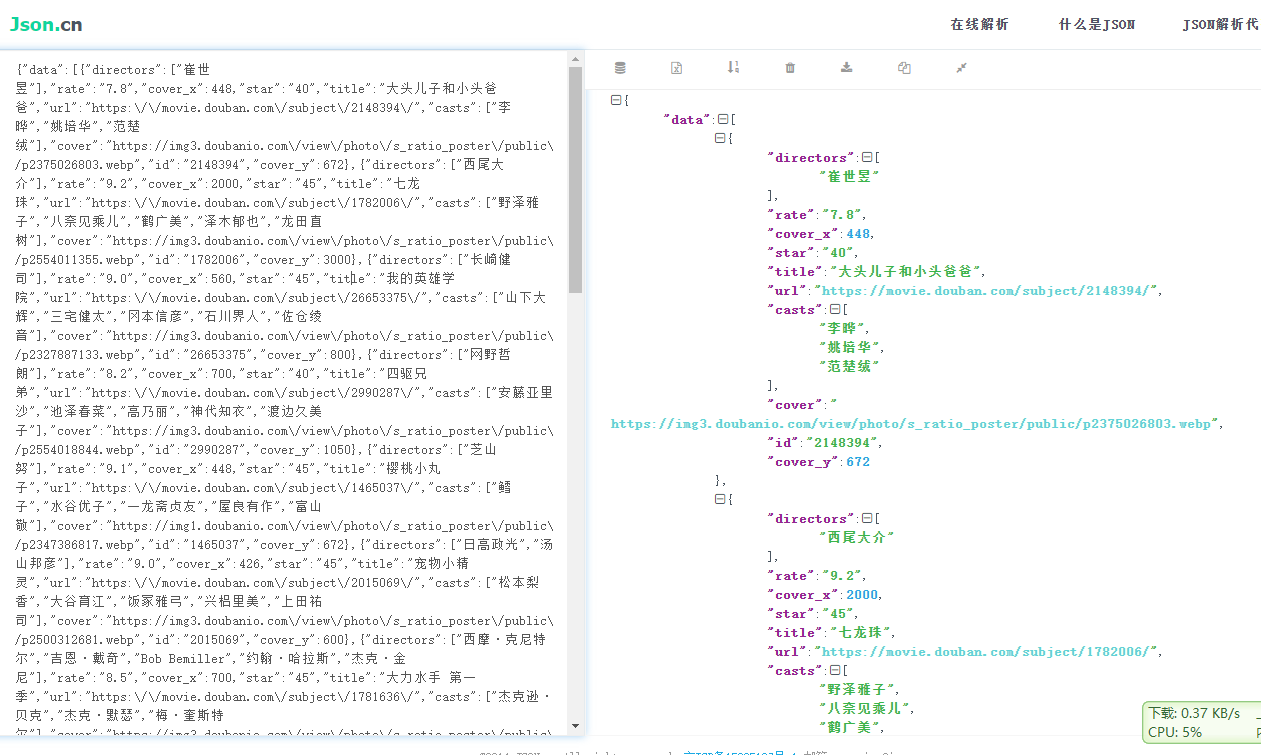
任意i页面的请求url是<https://movie.douban.com/j/new_search_subjects?sort=U&range=0,10&tags=%E5%8A%A8%E6%BC%AB&start>= （i-1）\*20，实际请求的页面url由前面的基网址加后面的字符串拼接而成。

实际电影评论的url地址：

"https://movie.douban.com/subject/"+ str(id)+**"**/comments?status=P"，其中id为电影的url对应的id；

要获取每个电影的ID，需要对其所在的列表页发出get请求，返回解析json字符串，来获取ID。





对于每个电影，爬取第一页的短评，进行写入数据库的操作。

 movieItem\_baseUrl = "https://movie.douban.com/subject/"

    base\_url = "https://movie.douban.com/j/new\_search\_subjects?sort=U&range=0,10&tags=%E5%8A%A8%E6%BC%AB&start="

    for i in **range**(0,1):

        req = base\_url + str(i\*20)

        movie\_ids = []

        response = requests.get(req, headers = headers)

        soup = BeautifulSoup(response.text,'lxml')

        singlePage\_jsonList = soup.find('p').text  *#返回的response为json字符串*

        json\_str = json.loads(singlePage\_jsonList,encoding='utf-8')['data']  *#将一个JSON编码的字符串转换回一个Python数据结构字典*

        json\_str\_list = list(json\_str)

        for item in json\_str\_list: *#获取每页20个电影的ID*

            movie\_ids.append(item['id'])

        for **id** in movie\_ids:

            movie\_url = movieItem\_baseUrl + str(**id**)+"/comments?status=P" *#实际的电影评论的url地址*

            movie\_response = requests.get(movie\_url, headers=headers)

            movie\_soup = BeautifulSoup(movie\_response.text, 'lxml')

            comment\_list = movie\_soup.find\_all('div',class\_='comment') *#获取单个电影的首页评论列表*

            lists = []  *# 存放每条评论的列表*

            for item in comment\_list:

                    movie = {}

                    movie['name'] = movie\_soup.find(id="content").h1.get\_text()[0:-2]  *# 获取电影标题*

                    try:

                        movie['comment\_content'] = item.find('span', class\_='short').get\_text() *# 获取评论实体*

                    except:

                        movie['comment\_content'] = ""

                    try:

                        movie['comment\_star'] = \

                            item.find('span', class\_='comment-info').find('span', {'class': re.compile("allstar")}).get(

                                'class')[0][-2:-1]  *# 获取评论星级*

                    except:

                        movie['comment\_star'] = ""

                    try:

                        movie['comment\_id'] = item.find('span', class\_='comment-info').find('a').get\_text()  *# 获取评论者ID*

                    except:

                        movie['comment\_id'] = ""

                    try:

                        movie['comment\_time'] = item.find('span', class\_='comment-info').find('span',

                                                                                              class\_="comment-time").get\_text()  *# 获取评论时间*

                    except:

                        movie['comment\_time'] = ""

                    try:

                        movie['comment\_votes'] = item.find('span', class\_='votes').get\_text()  *# 获取评论点赞数*

                    except:

                        movie['comment\_votes'] = ""

爬取出数据后，先写到字典中，然后写入到数据库中。这里通过PyMySQL的connect()方法声明一个MySQL连接对象db，此时需要传入MySQL运行的host（即IP）。由于MySQL在本地运行，所以传入的是localhost。如果MySQL在远程运行，则传入其公网IP地址。后续的参数user即用户名，password即密码，port即端口（默认为3306）。

连接成功后，需要再调用cursor()方法获得MySQL的操作游标，利用游标来执行SQL语句。这里我们执行了两句SQL，直接用execute()方法执行即可。

*#写入数据库部分*

                    db = pymysql.connect("localhost", "root", "\*\*\*\*\*\*", "douban\_crawler")

                    cursor = db.cursor()

                    table = 'douban'

                    keys = ', '.join(movie.keys())

                    values = ', '.join(['%s'] \* **len**(movie))

                    sql = 'INSERT INTO {table}({keys}) VALUES ({values})'.format(table=table, keys=keys, values=values)

                    try:

                        if cursor.execute(sql, tuple(movie.values())):

**print**('Successful')

                            db.commit()

                    except:

**print**('Failed')

                        db.rollback()

                    db.close()

这里我们传入的数据是字典，并将其定义为movie变量。表名也定义成变量table。接下来，就需要构造一个动态的SQL语句了。

首先，需要构造插入的字段name、comment\_content、comment\_star、comment\_id、comment\_time、comment\_votes。这里只需要将data的键名拿过来，然后用逗号分隔即可。所以', '.join(data.keys())的结果就是id, name, age，然后需要构造多个%s当作占位符，有几个字段构造几个即可。比如，这里有三个字段，就需要构造%s, %s, %s。这里首先定义了长度为1的数组['%s']，然后用乘法将其扩充为['%s', '%s', '%s']，再调用join()方法，最终变成%s, %s, %s。最后，我们再利用字符串的format()方法将表名、字段名和占位符构造出来。

由于豆瓣有反爬虫机制，我们需要设置请求头部和time.sleep进行延时爬取，防止被封IP

*#定义请求头*

    headers = {

        'User-Agent': 'Mozilla / 5.0(Windows NT 6.1;WOW64) AppleWebKit / 537.36(KHTML, likeGecko) Chrome / 45.0.2454.101Safari / 537.36'

    }

设置单个电影页面的爬取时间间隔为0.5s，避免IP被封。

  for **id** in movie\_ids:

            time.sleep(0.5)  *#设置单个电影页面的爬取时间间隔为0.5s，避免IP被封*

            movie\_url = movieItem\_baseUrl + str(**id**)+"/comments?status=P" *#实际的电影评论的url地址*

            movie\_response = requests.get(movie\_url, headers=headers)

            movie\_soup = BeautifulSoup(movie\_response.text, 'lxml')

            comment\_list = movie\_soup.find\_all('div',class\_='comment') *#获取单个电影的首页评论列表*

最后分析电影的列表页数量，一共大概450-500页，为了方便，我们爬取前450页的数据，一共9000条电影数据入库。

    for i in **range**(0,450):  *#爬取前450页列表页的所有数据*

        req = base\_url + str(i\*20)

        movie\_ids = []

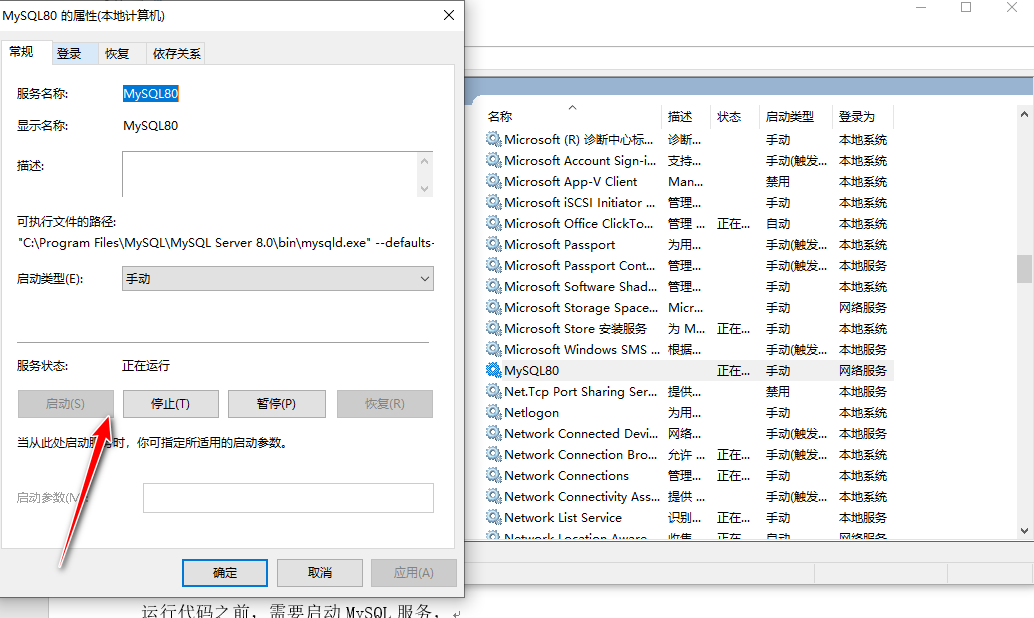
        response = requests.get(req, headers = headers)

        soup = BeautifulSoup(response.text,'lxml')

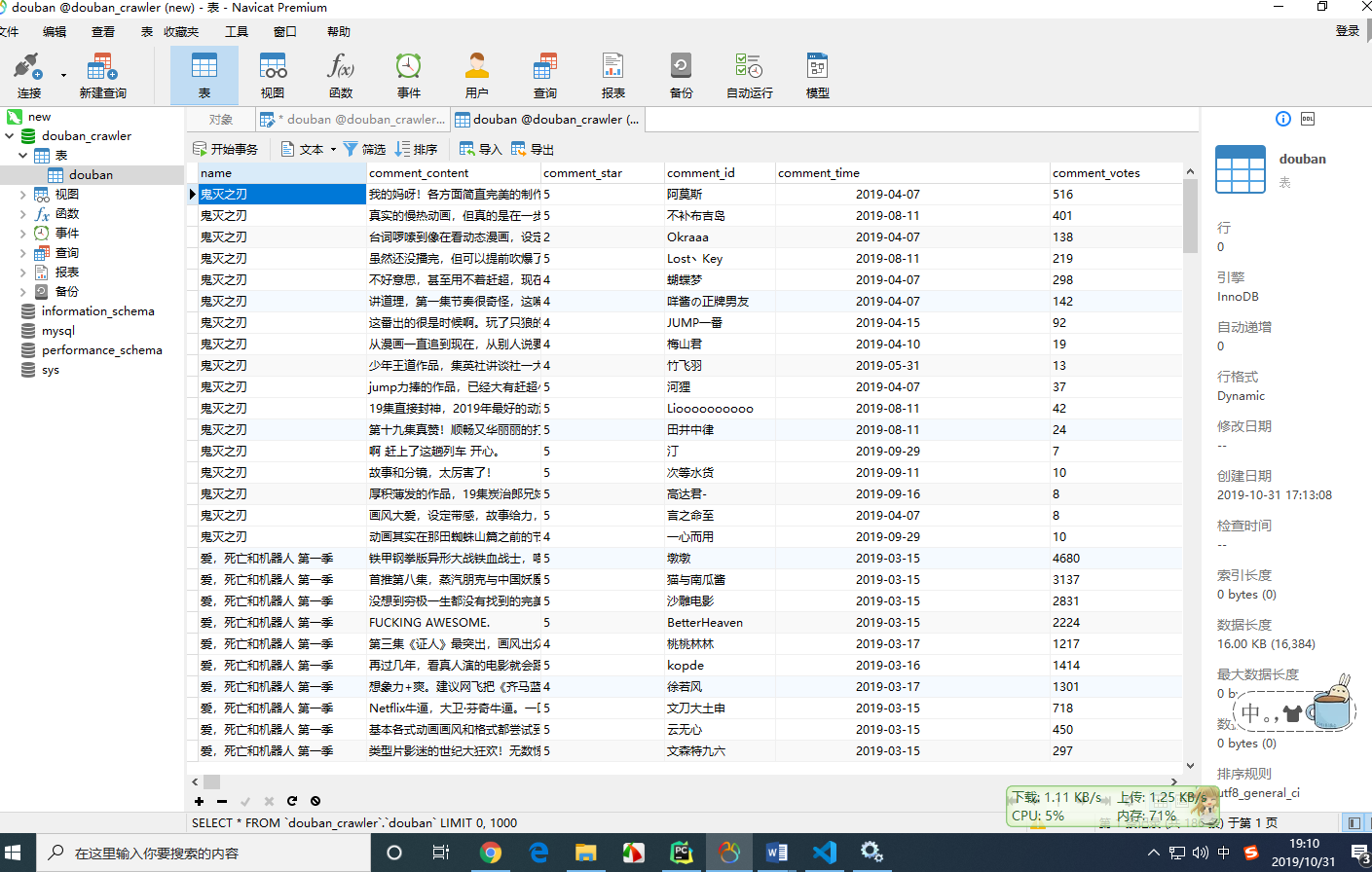
        singlePage\_jsonList = soup.find('p').text  *#返回的response为json字符串*

        json\_str = json.loads(singlePage\_jsonList,encoding='utf-8')['data']  *#将一个JSON编码的字符串转换回一个Python数据结构字典*

        json\_str\_list = list(json\_str)

运行代码之前，需要启动本地MySQL服务，然后运行代码。

打开navicat可视化数据界面，连接数据库，可以看到数据全部写入数据库中。



至此，数据的爬取和持久化工作已经全部完成。

项目源码已经上传至GitHub：<https://github.com/jhyscode/bigdata_douban_crawler>

## 五、小结讨论

爬虫是一门非常实用的技术，涉及到前端知识、后端知识、浏览器的调试等。对于爬取豆瓣电影类网站，反爬虫机制相对比较弱，我们只需要设置模拟浏览器头部和设置间隔时间爬取，基本上不会被封IP等，所以对于爬虫技术，我们的分析和经验也非常重要。此次爬虫任务我使用的是常用的BeautifulSoup、Requests爬虫库等，采用的是单线程爬取。在以前写过的爬虫中，我经常使用Scrapy框架来写一些爬虫，scrapy是多线程的，可以加快我们爬虫爬取的速度。

对于一些反爬虫比较厉害的网站来说，他们经常会设置登录反爬、验证码反爬、js加密等，以此来减轻爬虫对服务器的压力。对于我们技术人员来说，应该恪守职业道德，爬取数据的时候应该遵守robots协议，而不是一味的索求数据。