


## Neo4j 简单例子——豆瓣 Top250

一. 爬取目标网站，生成 json 格式的数据。

目标网站: <https://movie.douban.com/top250> 这个页面上的电影相关说明



肖申克的救赎 / The Shawshank Redemption / 月黑高飞(港) / 刺激1995(台) [可播放]

导演: 弗兰克·德拉邦特 Frank Darabont 主演: 蒂姆·罗宾斯 Tim Robbins /...

1994 / 美国 / 犯罪 剧情

★★★★★ 9.7 1655242人评价

“ 希望让人自由。 ”

代码在 spiders 文件的 spider.py。

环境: python3.6 所导入的包:

```
import csv
```

```
import json
```

```
import requests
```

```
from lxml import etree
```

```
import json
```

此代码跑完后在当前目录会生成 data.csv

rank	名字	评价人数	类别						
1	肖申克的救赎	1655242人	1994 / 美国 / 犯罪 剧情						
2	霸王别姬	1223183人	1993 / 中国大陆 中国香港 / 剧情 爱情 同性						
3	阿甘正传	1281877人	1994 / 美国 / 剧情 爱情						
4	这个杀手不太冷	1475545人	1994 / 法国 / 剧情 动作 犯罪						
5	美丽人生	749293人	1997 / 意大利 / 剧情 喜剧 爱情 战争						
6	泰坦尼克号	1220892人	1997 / 美国 / 剧情 爱情 灾难						
7	千与千寻	1310660人	2001 / 日本 / 剧情 动画 奇幻						
8	辛德勒的名单	661555人	1993 / 美国 / 剧情 历史 战争						
9	盗梦空间	1256627人	2010 / 美国 英国 / 剧情 科幻 悬疑 冒险						
10	忠犬八公	846370人	2009 / 美国 英国 / 剧情						
11	机器人总动员	832723人	2008 / 美国 / 爱情 科幻 动画 冒险						
12	三傻大闹宝莱坞	1139021人	2009 / 印度 / 剧情 喜剧 爱情 歌舞						
13	放牛班的春天	795280人	2004 / 法国 瑞士 德国 / 剧情 音乐						
14	楚门的世界	889959人	1998 / 美国 / 剧情 科幻						
15	海上钢琴师	917764人	1998 / 意大利 / 剧情 音乐						
16	星际穿越	909277人	2014 / 美国 英国 加拿大 冰岛 / 剧情 科幻 冒险						
17	大话西游	890159人	1995 / 中国香港 中国大陆 / 喜剧 爱情 奇幻 古装						
18	龙猫	777377人	1988 / 日本 / 动画 奇幻 冒险						
19	熔炉	531685人	2011 / 韩国 / 剧情						
20	无间道	730877人	2002 / 中国香港 / 剧情 犯罪 悬疑						
21	教父	566421人	1972 / 美国 / 剧情 犯罪						
22	疯狂动物城	1035356人	2016 / 美国 / 喜剧 动画 冒险						
23	当幸福来敲门	922545人	2006 / 美国 / 剧情 传记 家庭						
24	怦然心动	1037856人	2010 / 美国 / 剧情 喜剧 爱情						
25	触不可及	601517人	2011 / 法国 / 剧情 喜剧						
26	蝙蝠侠：黑暗骑士	803784人	2008 / 美国 英国 / 剧情 动作 科幻 犯罪 惊悚						

二.data.csv 导入 neo4j。

把 data.csv 放入 neo4j-community-3.3.1 (版本号可能不同) 文件夹下的 import 文件夹中。

neo4j-community-3.3.9\bin 下打开命令行

输入 neo4j.bat console

开启 neo4j, 开启浏览器, 地址看 cmd 的提示。

通过输入命令行进行数据导入

(如果您 neo4j 中已有别的数据, 可以先删除或改名 neo4j-community-3.3.9\data 下的 databases, 再启动 neo4j, 会自动生成一个新的库)

这里把电影名字单独作为一个类别, 把电影的三个属性放在了一起, 利用图

数据库的关系导入，让数据变得可视化起来。

批量加载，演示步骤

①

```
LOAD CSV WITH HEADERS FROM "file:///data.csv" AS line
CREATE(:电影名字 {name:line.名字,class:line.类别})
CREATE(:电影属性 {name:line.rank})
CREATE(:人数 {name:line.评价人数})
```

②

```
LOAD CSV WITH HEADERS FROM "file:///data.csv" AS line
MATCH (entity1:电影名字{name:line.名字}), (entity2:电影属性{name:line.rank})
CREATE(entity1)-[:rank{type:line.relation,name:'排名'}]->(entity2)
```

③

```
LOAD CSV WITH HEADERS FROM "file:///data.csv" AS line
MATCH (entity1:电影名字{name:line.名字}), (entity2:人数{name:line.评价人数})
CREATE(entity1)-[:评价人数{type:line.relation}]->(entity2)
```

neo4j 命令:

#这里因为 neo4j 版本低问题，不能在页面批量使用命令，需要逐条输入

一. 插入节点

插入一个 Person 类别的节点，且这个节点有几个属性值

```
CREATE (:Person {name : '邓超', birthdate:'1978 年',born_in:'江西省南昌'})
CREATE (:Person {name : '孙俪', birthdate:'1982 年',born_in:'上海市'})
CREATE (:Person {name : '鹿晗', birthdate:'1990 年',born_in:'北京市海淀区'})
```

2.插入边。插入一条 a 到 b 的有向边，且边的类别为 Follow r 后面是关系

```
MATCH (a:Person),(b:Person)
WHERE a.name = '邓超' AND b.name = '孙俪'
CREATE (a)-[:妻子]->(b);
```

```
MATCH (a:Person),(b:Person)
WHERE a.name = '邓超' AND b.name = '孙俪'
CREATE (b)-[:丈夫]->(a);
```

```
MATCH (a:Person),(b:Person)
WHERE a.name = '邓超' AND b.name = '鹿晗'
CREATE (a)-[:朋友]->(b);
```

注意:

CREATE (a)-[r:4]->(b);      r 后面不能直接是数字  
CREATE (a)-[r:"4"]->(b);      加双引号、单引号都不行  
CREATE (a)-[r:权重 4]->(b);      这样可以

3.更新节点。更新一个 Person 类别的节点，设置新的属性值。

```
MATCH (n:Person { name:'邓超' })  
SET n.birthdate = '1979 年';
```

4.删除节点。删除这个节点和这个节点有关的联系

```
MATCH (n:Person { name:'鹿晗' })  
DETACH DELETE n;
```

5.查询两个节点之间的关系。

```
MATCH (a:Person { name:'邓超' })-[r]->(b:Person { name:'孙俪' })  
RETURN type(r);
```

6.查询一个节点的其中一个关系的所有节点。

```
MATCH (:Person {name:'邓超'})-[r:妻子]->(Person)  
RETURN Person.name;
```

7.删除边

```
MATCH (a:Person)-[r:妻子]->(b:Person)  
WHERE a.name = '邓超' AND b.name = '孙俪'  
DELETE r;
```