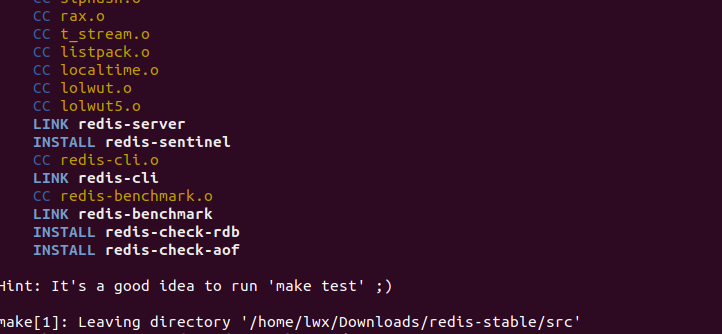
单人作业。根据 Redis、MongoDB、Neo4j 的指导文档完成三个数据库的安装与基本使用。之后利用给出的数据集student.csv，将数据分别存入三种数据库中，分析和比较三者在批量增、改、查方面上效率的优劣。  
    报告内容：请在报告中详细写明你的实验步骤、技术方法、实习体会等，必要时附上相应的代码段或截图。  
    截止时间：请同学们于北京时间2019年5月20日23:59前，在北大教学网课程页面，提交pdf形式的上机报告，文件名格式为："报告5-张三-1500012345.pdf"。  
    注意事项：请按时提交。每延迟一天当次作业成绩扣除20%，三天以上记为0

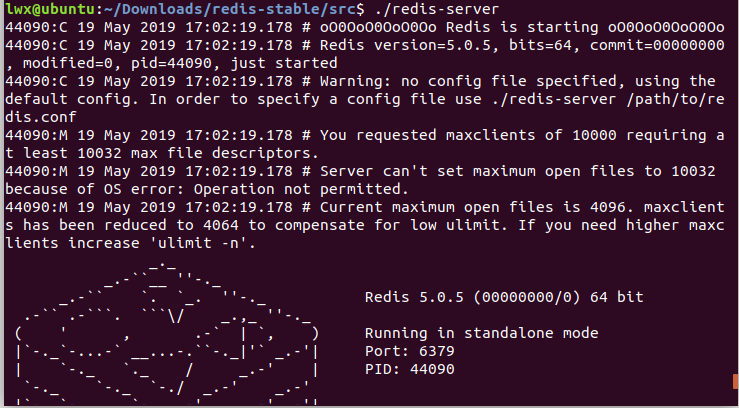
# Redis

## 安装

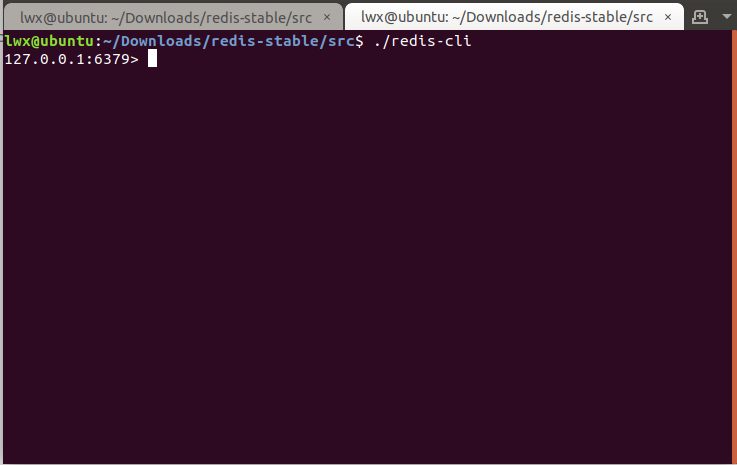
make



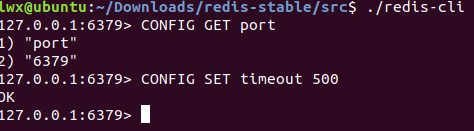
运行redis server



启动新终端



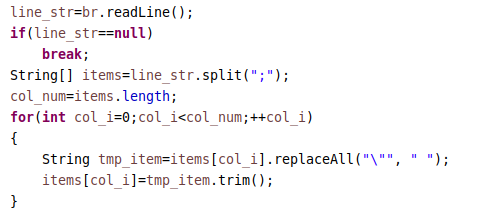
查看配置，更改配置，设置客户端闲置500s后关闭连接



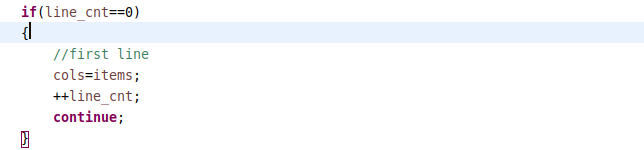
## 导入数据

使用java API 的Jedit，用哈希表存储

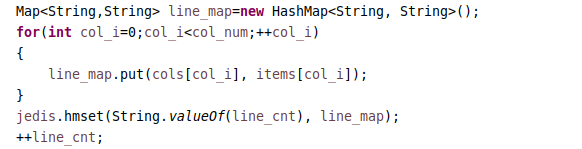
按行读取csv文件，以分号为分隔符分成string的数组，去掉string前后可能的空格和分号



如果是第一行，则得到属性名称数组



如果不是第一行，则以行号为key，每个属性名称为field,属性值为value写入redit。先创建field和value的键值对hashmap

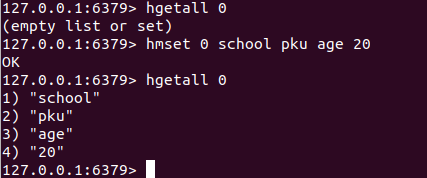


代码详见附录

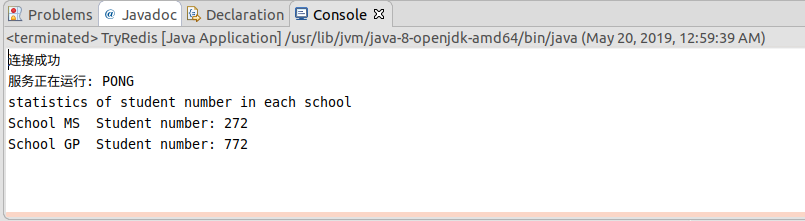
## 基本使用

### 增

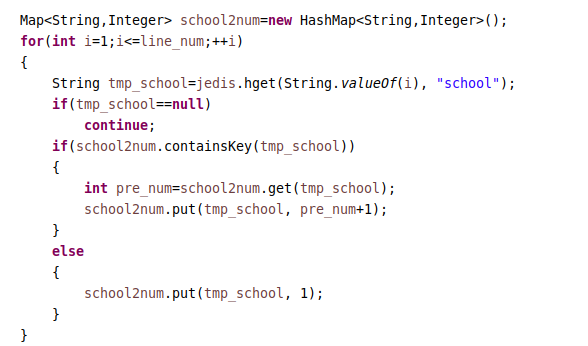
新增0号学生，定义其school和age属性



统计该表格中的学校和每个学校的学生数，代码详见附录

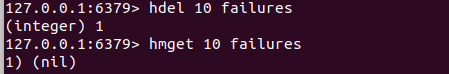


jedis.hget得到每个student对应的school，如果hashmap中存在该学校则数目加一，如果不存在则增加这项。



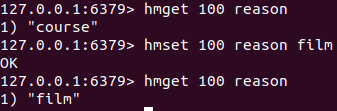
### 删

删除10号学生的failures域，可见该值变成空值



### 改

将100号学生的reason改成film



### 查

查找1号学生的信息，这里的哈希表的key为csv的行号，field为各个属性名称



查找100号学生的famsize



## 心得

redis安装时jemalloc报错，需要使用make MALLOC=libc命令。

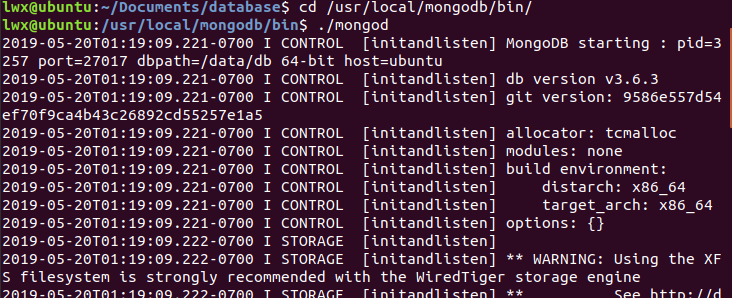
redis客户端默认0号数据库，本题中无需切换数据库

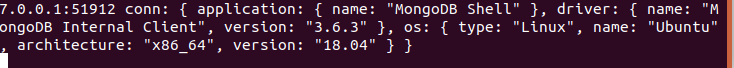
redis表的概念存在不同，这里的哈希是键值对映射表，一个key对应一组field和value对。

# MongoDB

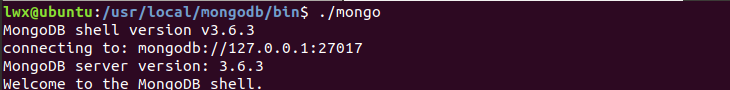
## 安装

启动mongodb服务



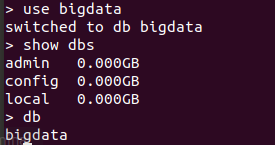


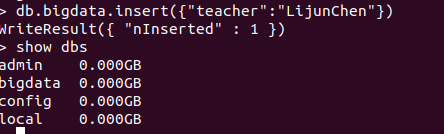
连接MongoDB，进入Shell模式





### 数据库操作

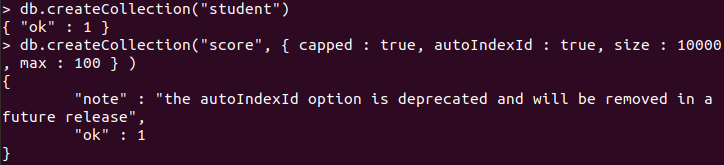






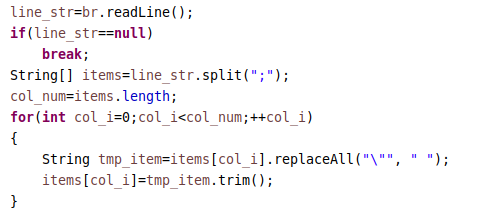
### 集合操作

创建固定集合

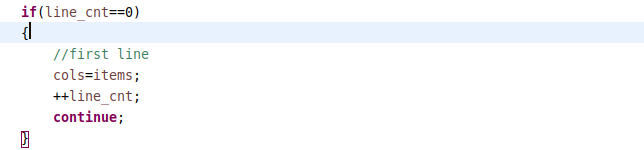


## 导入数据

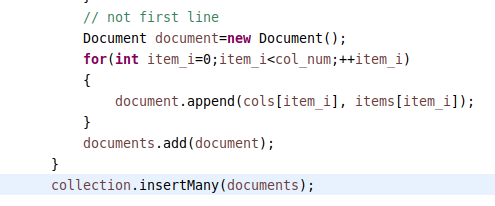
按行读取csv文件，以分号为分隔符分成string的数组，去掉string前后可能的空格和分号



如果是第一行，则得到属性名称数组



如果不是第一行，则将该行的属性值作为value，属性名称作为key，将这行的键值对作为一个document，所有行构成一个document的list，并加进数据库的一个集合中



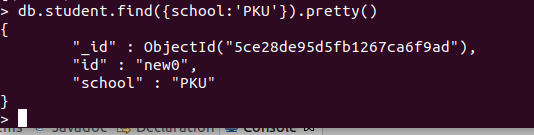
代码详见附录

## 基本使用

### 增

新建一个学校为PKU的学生

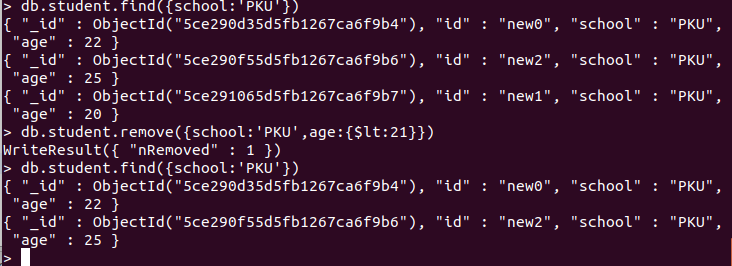




### 删

共三个pku的学生，年龄20，22，25，删去学校为PKU，年龄小于21的文档。

结果剩下两个文档。



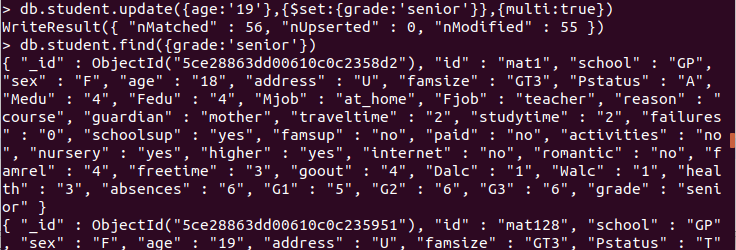
### 改

将19岁的学生的年级改成senior

如果不加multi:true，则只有最多改一个文档

加上multi:true后，mathch56个，改55个，因为有一个在前一步已改

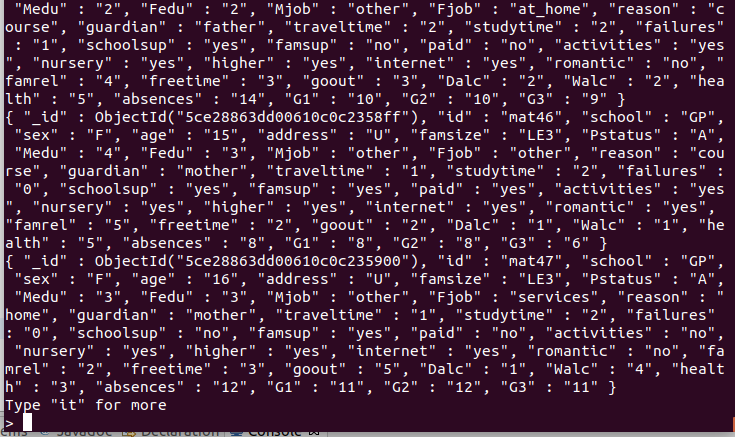




### 查

查找GP学校的所有女生





## 心得

update操作如果不指定{multi:true}则最多改一个

使用条件查询的比大小时，需要int类型的对象，数字字符串不参与匹配，创建时不带引号的数字是int，带引号则为string

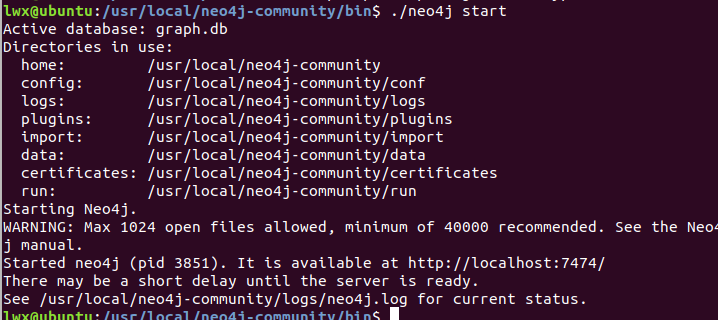
mongodb的每个“行”是一个文档，文档有键值对组成，一个集合中有多个文档，一个数据库中有多个集合

某个数据库中的集合都为空时，show dbs不显示该数据库

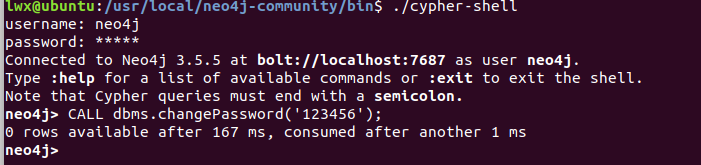
# Neo4j

## 安装

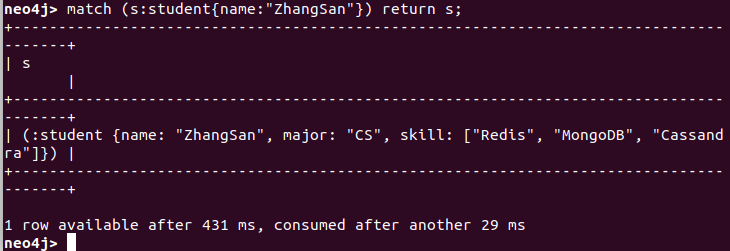
安装，启动服务

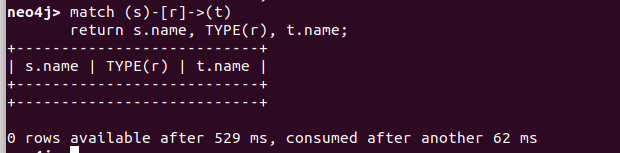


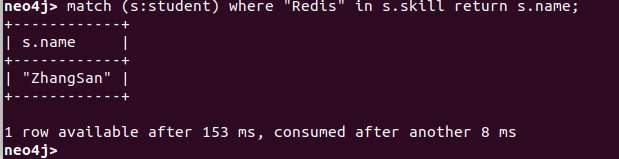
进入shell，修改密码



简单查询

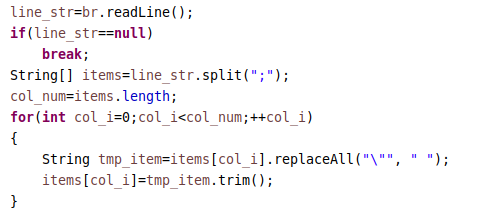




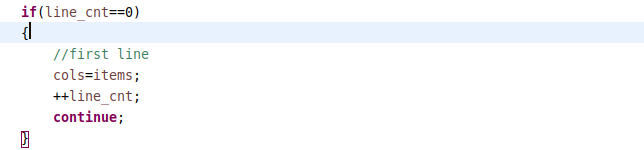


## 导入csv

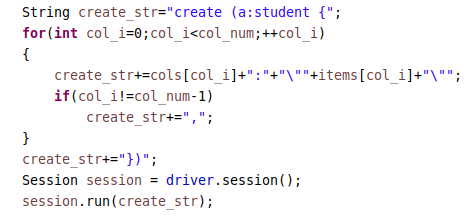
按行读取csv文件，以分号为分隔符分成string的数组，去掉string前后可能的空格和分号



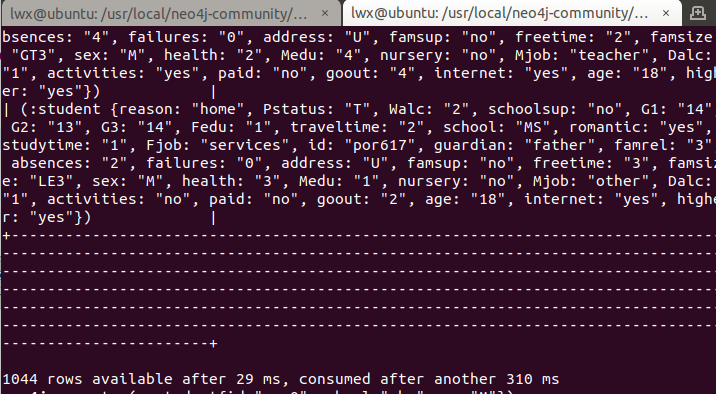
如果是第一行，则得到属性名称数组



如果不是第一行，则每行对应一个节点，构造create命令字符串，在session中运行命令。



代码详见附录



## 基本使用

### 增

新增一个student

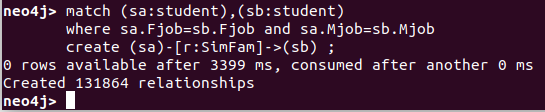


创建关系，父母职业均相同的学生之间创建关系：相似家庭，

match (sa:student),(sb:student)

where sa.Fjob=sb.Fjob and sa.Mjob=sb.Mjob

create (sa)-[r:SimFam]->(sb) ;



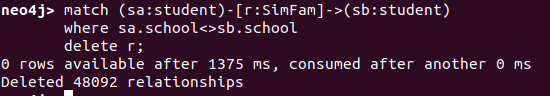
### 删

在相似家庭关系中，如果两人所在学校不同，不再认为是相似家庭

match (sa:student)-[r:SimFam]->(sb:student)

where sa.school<>sb.school

delete r;



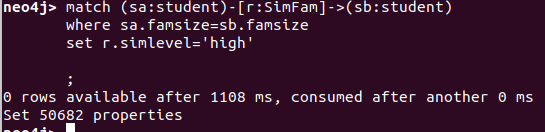
### 改

相似家庭中，如果家庭人数相同，则设置相似程度属性为high

match (sa:student)-[r:SimFam]->(sb:student)

where sa.famsize=sb.famsize

set r.simlevel='high'



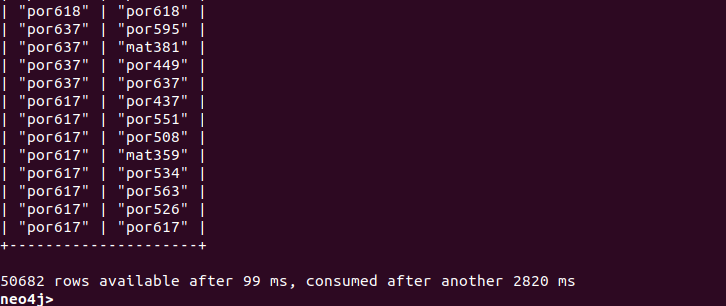
### 查

查看相似家庭关系的学生

match (sa:student)-[r:SimFam]->(sb:student)

where sa.famsize=sb.famsize

return sa.id,sb.id;



## 心得

neo4j的java api直接将命令字符串作为参数，命令返回值作为返回值

neo4j与之前两个不同，是图数据库，以点和关系为基本元素

where语句中<>为不等于，AND为且

创建关系必须为有向关系

# Appendix

## 导入csv到redit中的哈希表

package hw5;

import redis.clients.jedis.Jedis;

import java.io.BufferedReader;

import java.io.\*;

import java.util.ArrayList;

import java.util.List;

import java.util.\*;

public class TryRedis

{

public static void load\_csv(String csv\_path)

{

try

{

//连接本地Redis服务

Jedis jedis = new Jedis("localhost");

System.out.println("连接成功");

//查看服务运行状态

System.out.println("服务正在运行: "+jedis.ping());

System.out.println("read csv into redis");

BufferedReader br = new BufferedReader(new FileReader(csv\_path));

// ArrayList<String> cols=new ArrayList<String>();

String[] cols=null;

int col\_num=0;

int line\_cnt=0;

String line\_str=null;

while(true)

{

line\_str=br.readLine();

if(line\_str==null)

break;

String[] items=line\_str.split(";");

col\_num=items.length;

for(int col\_i=0;col\_i<col\_num;++col\_i)

{

String tmp\_item=items[col\_i].replaceAll("\"", " ");

items[col\_i]=tmp\_item.trim();

}

if(line\_cnt==0)

{

//first line

cols=items;

++line\_cnt;

continue;

}

// not first line

// write into hashtable

Map<String,String> line\_map=new HashMap<String, String>();

for(int col\_i=0;col\_i<col\_num;++col\_i)

{

line\_map.put(cols[col\_i], items[col\_i]);

}

jedis.hmset(String.valueOf(line\_cnt), line\_map);

++line\_cnt;

}

}

catch(Exception e)

{

e.printStackTrace();

}

}

public static void main(String[] args)

{

load\_csv("student.csv");

}

}

## redis统计学校和学生

public static void redis\_query\_school()

{

try

{

int line\_num=1044;

//连接本地Redis服务

Jedis jedis = new Jedis("localhost");

System.out.println("连接成功");

//查看服务运行状态

System.out.println("服务正在运行: "+jedis.ping());

System.out.println("statistics of student number in each school");

Map<String,Integer> school2num=new HashMap<String,Integer>();

for(int i=1;i<=line\_num;++i)

{

String tmp\_school=jedis.hget(String.valueOf(i), "school");

if(tmp\_school==null)

continue;

if(school2num.containsKey(tmp\_school))

{

int pre\_num=school2num.get(tmp\_school);

school2num.put(tmp\_school, pre\_num+1);

}

else

{

school2num.put(tmp\_school, 1);

}

}

Set set=school2num.entrySet();

Iterator iterator = set.iterator();

while(iterator.hasNext())

{

Map.Entry tmp\_entry = (Map.Entry)iterator.next();

System.out.print("School "+ tmp\_entry.getKey() + " ");

System.out.print("Student number: "+tmp\_entry.getValue());

System.out.println();

}

}

catch(Exception e)

{

e.printStackTrace();

}

}

## mongodb导入csv

public static void load\_csv(String \_csv\_path)

{

try

{

// 连接到 mongodb 服务

MongoClient mongoClient = new MongoClient( "localhost" , 27017 );

// 连接到数据库

MongoDatabase mongoDatabase = mongoClient.getDatabase("test");

//创建集合

mongoDatabase.createCollection("student");

//选择集合

MongoCollection<Document> collection = mongoDatabase.getCollection("student");

List<Document> documents = new ArrayList<Document>();

BufferedReader br = new BufferedReader(new FileReader(\_csv\_path));

String[] cols=null;

int col\_num=0;

int line\_cnt=0;

String line\_str=null;

while(true)

{

line\_str=br.readLine();

if(line\_str==null)

break;

String[] items=line\_str.split(";");

col\_num=items.length;

for(int col\_i=0;col\_i<col\_num;++col\_i)

{

String tmp\_item=items[col\_i].replaceAll("\"", " ");

items[col\_i]=tmp\_item.trim();

}

if(line\_cnt==0)

{

//first line

cols=items;

++line\_cnt;

continue;

}

// not first line

Document document=new Document();

for(int item\_i=0;item\_i<col\_num;++item\_i)

{

document.append(cols[item\_i], items[item\_i]);

}

documents.add(document);

}

collection.insertMany(documents);

}

catch(Exception e)

{

e.printStackTrace();

}

}

## neo4j导入csv

public void load\_csv(String \_csv\_path)

{

try

{

// neo\_obj.remove\_all();

BufferedReader br = new BufferedReader(new FileReader(\_csv\_path));

String[] cols=null;

int col\_num=0;

int line\_cnt=0;

String line\_str=null;

while(true)

{

line\_str=br.readLine();

if(line\_str==null)

break;

String[] items=line\_str.split(";");

col\_num=items.length;

for(int col\_i=0;col\_i<col\_num;++col\_i)

{

String tmp\_item=items[col\_i].replaceAll("\"", " ");

items[col\_i]=tmp\_item.trim();

}

if(line\_cnt==0)

{

//first line

cols=items;

++line\_cnt;

continue;

}

// not first line

String create\_str="create (a:student {";

for(int col\_i=0;col\_i<col\_num;++col\_i)

{

create\_str+=cols[col\_i]+":"+"\""+items[col\_i]+"\"";

if(col\_i!=col\_num-1)

create\_str+=",";

}

create\_str+="})";

Session session = driver.session();

session.run(create\_str);

}

}

catch(Exception e)

{

}

}