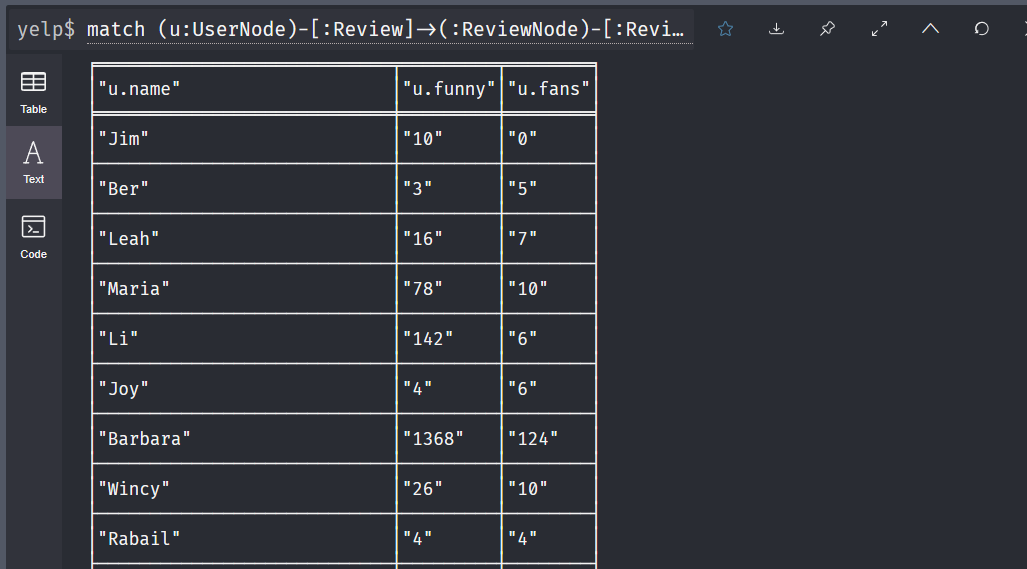
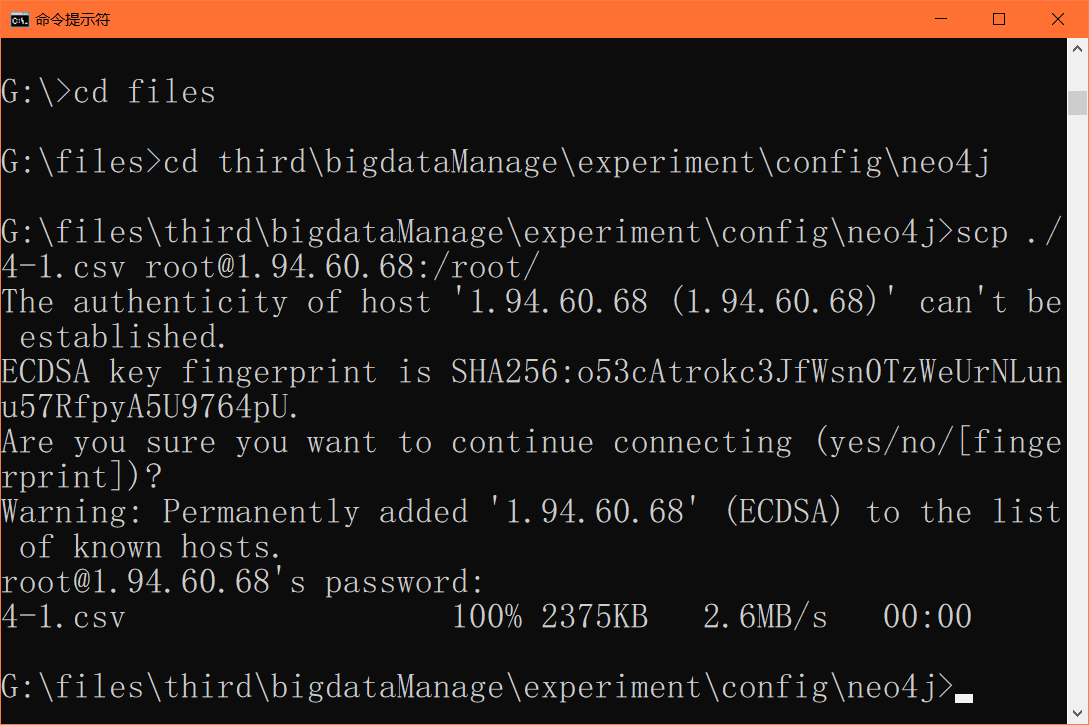
**1. 使用 Neo4j 查找：找出评论过超过 5 家不同商户的用户，并在Neo4j以表格形式输出满足以上条件的每个用户的信息：name, funny, fans。**

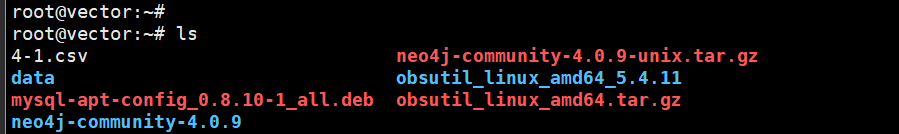
match (u:UserNode)-[:Review]->(:ReviewNode)-[:Reviewed]->(b:BusinessNode) with u,count(distinct b) as cnt where cnt > 5 return u.name,u.funny,u.fans



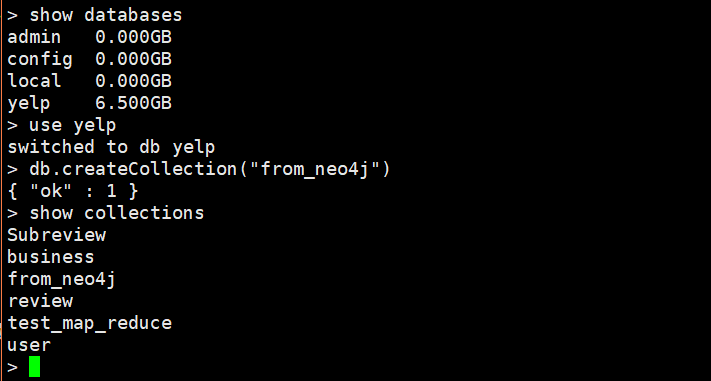
**2. 将 1 得到的结果导入 MongoDB，并使用该表格数据，**



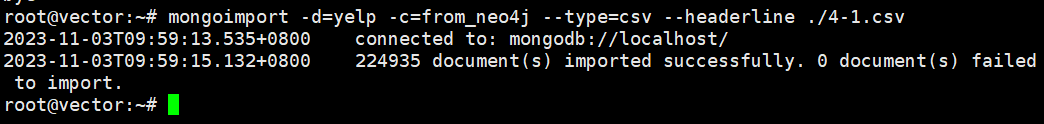
4-1.csv文件上传



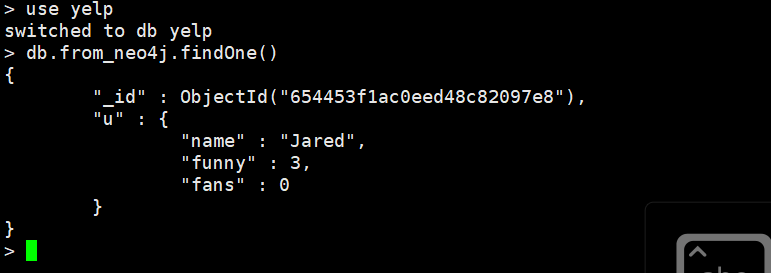
创建from\_neo4j集合：



将/root/里4-1.csv文件上传到mongodb

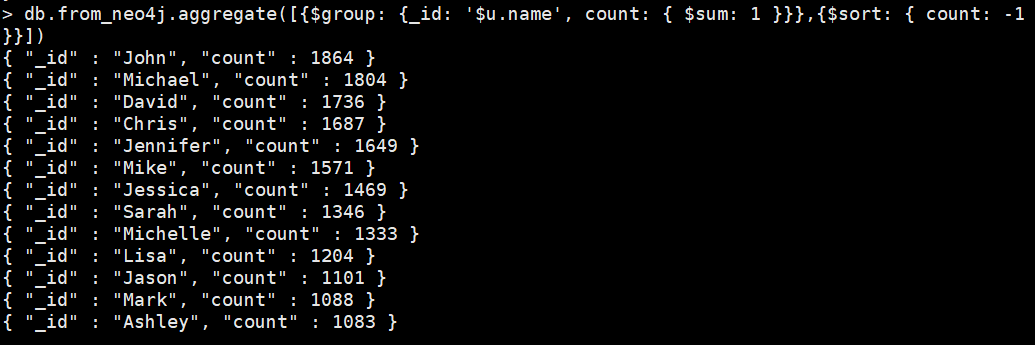


检查是否正常导入

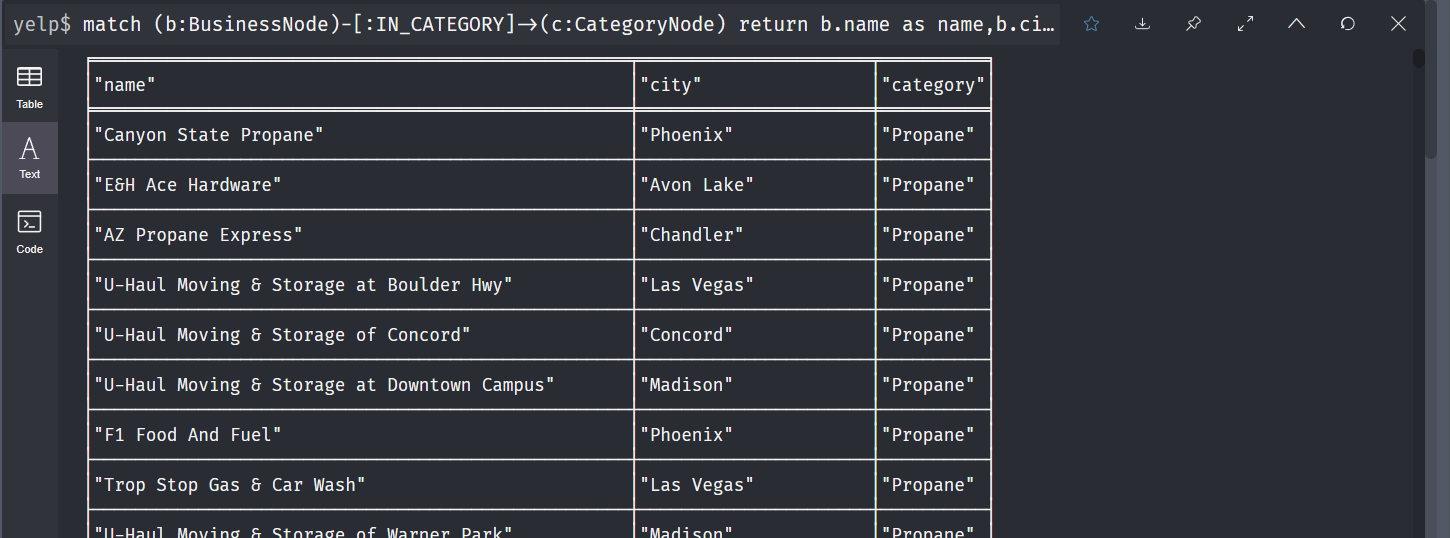


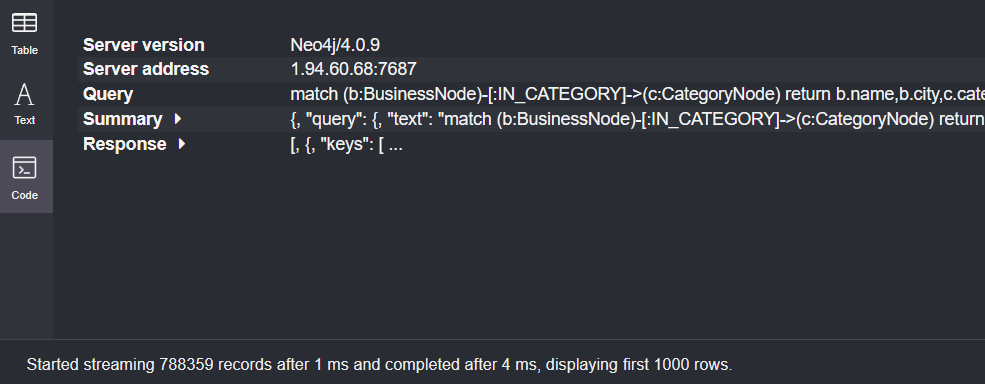
**3. 统计其中所有出现的 用户名及该用户名对应的出现次数，并按照出现次数降序排序,使用 aggregate 实现**

db.from\_neo4j.aggregate([{$group: {\_id: '$u.name', count: { $sum: 1 }}},{$sort: { count: -1 }}])

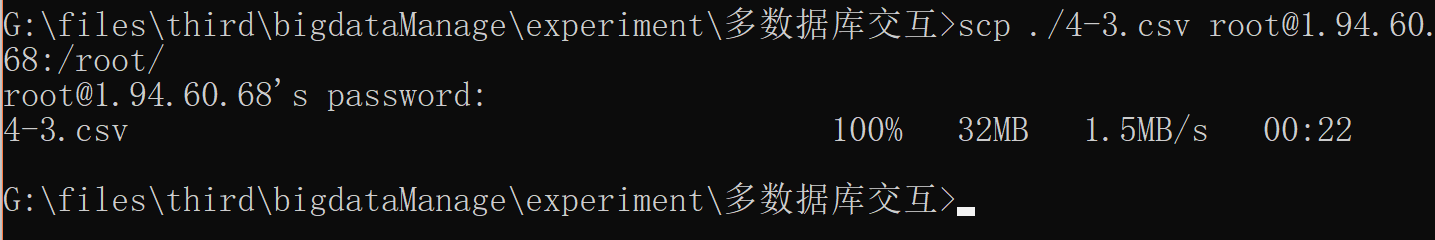


**4. 在 Neo4j 中查找所有商家，要求返回商家的名字，所在城市、商铺类**

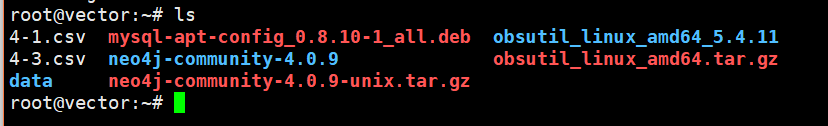




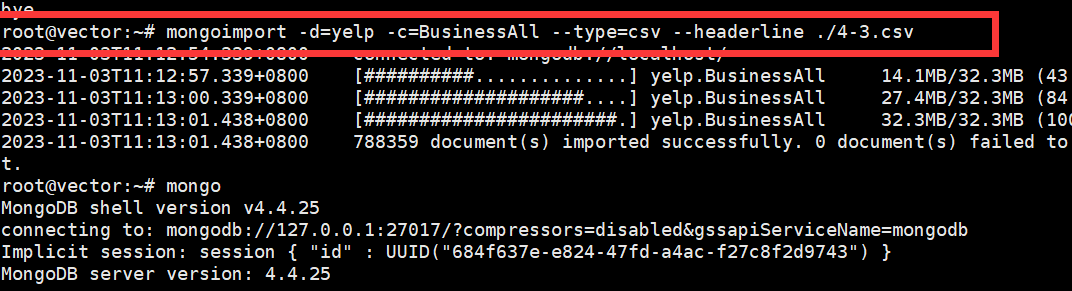
将文件下载为4-3.csv，从本地上传到/root/：

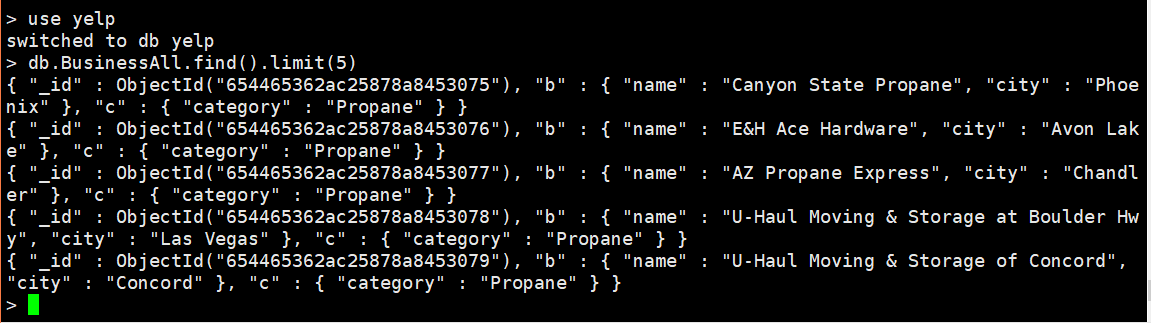


在/root/使用ls可以查看：



将4-3.csv上传到mongodb的新集合BusinessAll



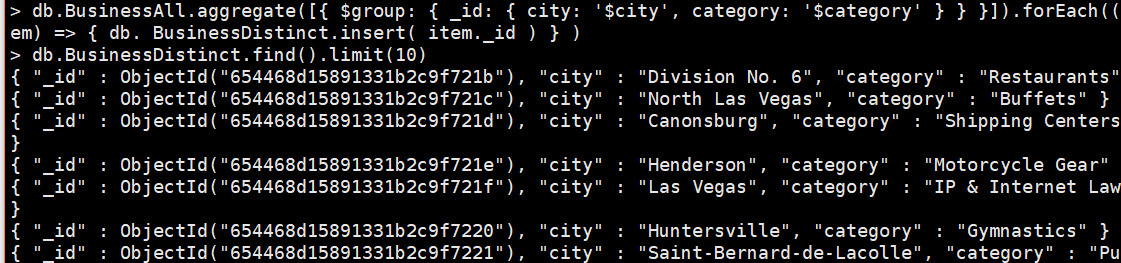


**将查找结果导入MongoDB中实现对数据的去重（提示：使用aggregate，仅保留城市、商铺类型即可）**

创建BusinessDistinct集合用于保存结果：

db.BusinessAll.aggregate([{ $group: { \_id: { city: '$city', category: '$category' } } }]).forEach((item) => { db. BusinessDistinct.insert( item.\_id ) } )

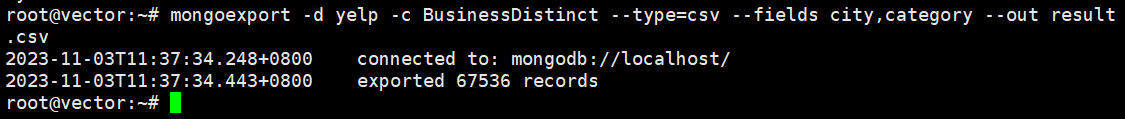
查看结果：



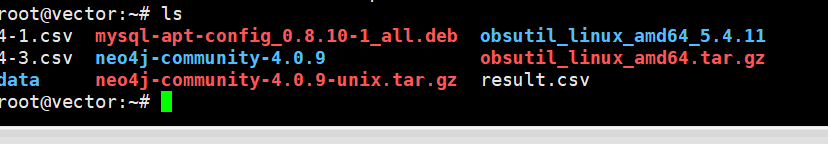
**5.将去重后的结果导入Neo4j中的新库result中，完成（City-[Has]->Category）图谱的构建。**

导出mongodb中的数据：

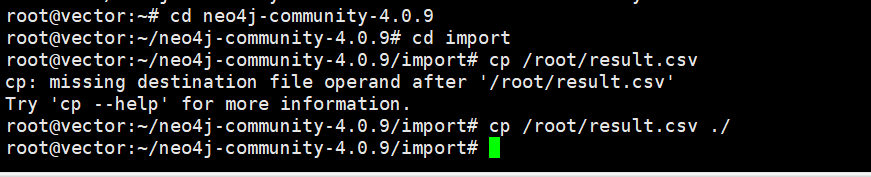
mongoexport -d yelp -c BusinessDistinct --type=csv --fields city,category --out result.csv



ls查看result.csv



将result.csv复制到import文件夹中



在neo4j中，输入：

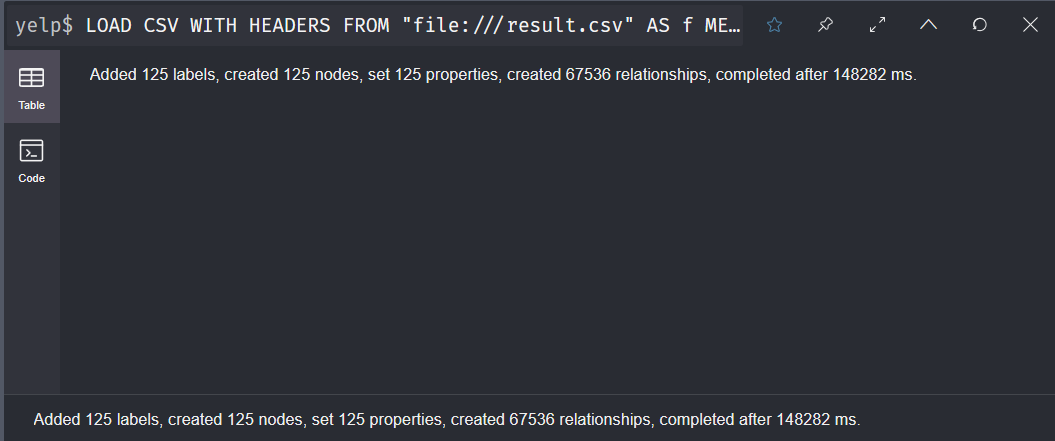
load csv with headers from "file:///result.csv" AS file

merge (c:CityNode {city: coalesce(file.city, "")})

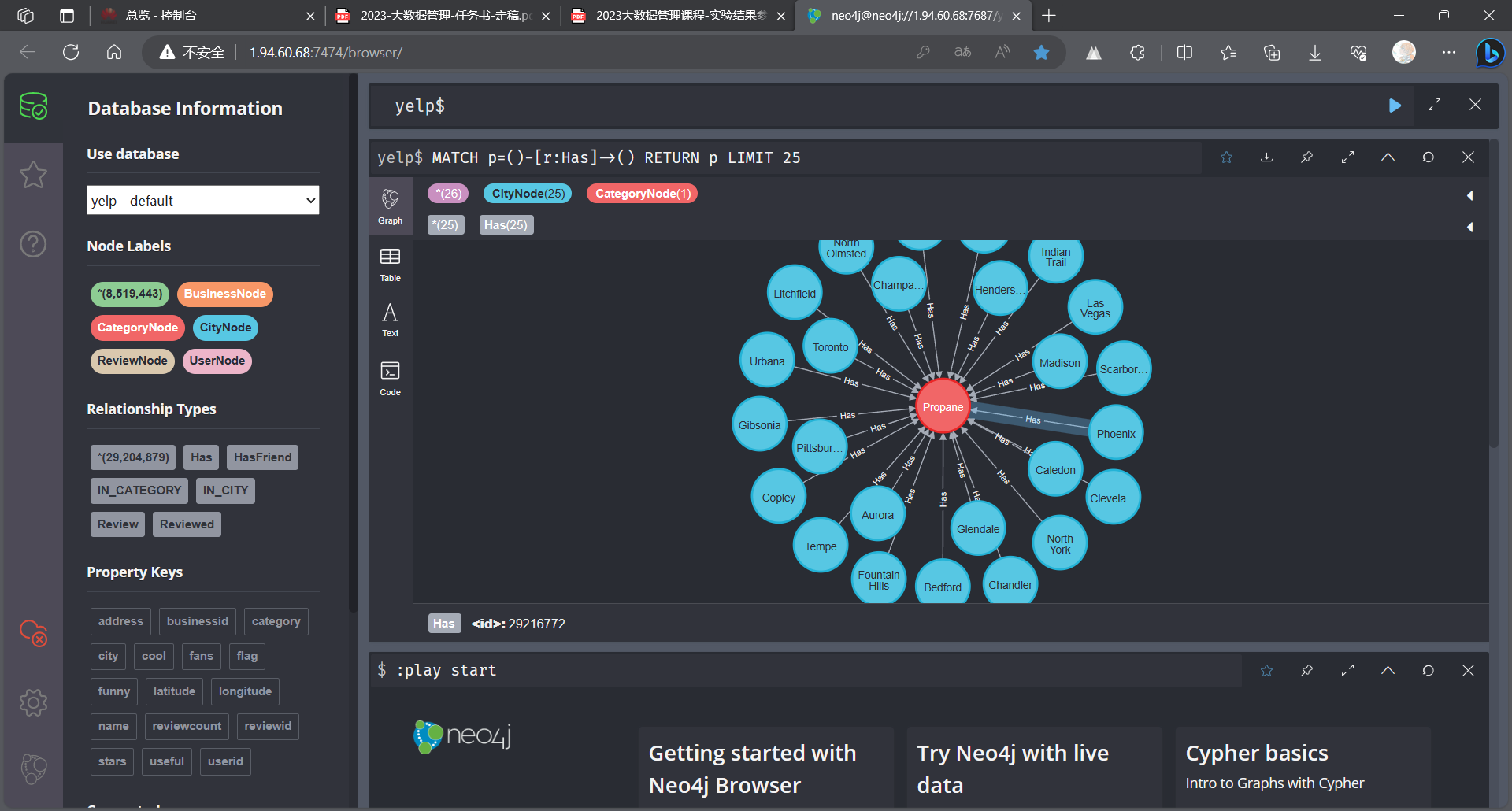
merge (a:CategoryNode {category: coalesce (file.category, "")})

create (c) -[:Has]-> (a)

merge创建结点或者关系



刷新neo4j的页面可以查看Has关系的图数据库：



任务5：不同类型数据库 MVCC 多版本并发控制对比实验