《软件实践》课程实验报告

暑期学校实验项目：高考志愿填报助手

小组名称	知识图谱构建 A 组						
姓　　名	罗琦晴	专业	人工智能	班级	091181	学号	09118133
实验时间	2020.8.31-2020.9.23		指导教师	孔祥龙		成绩	

一、 实验背景和目的

高考是人生的转折点，合理并正确地高考志愿填报是高考完后重要的一个环节，但历年来，高考志愿填报仍是大多数考生和家长的一个难题，面对不同的学校和不同的专业，以及海量的录取分数数据，需要付出大量时间和精力去对比分析，任务量繁琐，且无法确保结果的有效性。

因而，我们设计并实现了基于 web 的高考志愿填报系统，为考生在进行志愿填报时避免大量的信息查找、筛选和对比等工作，并依据考生的成绩和需求，提供合理的报考志愿推荐，具有一定的社会意义。

二、小组任务和个人任务

小组任务：利用已有的数据，通过 Neo4j 构建一个小型的报考知识图谱，展现不同高校的基本信息和其开设专业的近 3 年来的录取分数。

任务 1：数据源

本项目需要用到的数据源：是第一组清洗的包含学校，专业，省份，分数，年份的 csv 文件。

任务：从之前的小组获取报考省份及相应分数线的讯息

任务 2：知识图谱设计与优化

利用已有的数据构建一个小型的报考知识图谱(知识库)，通过调用该图谱可以实现如下功能：

1.已知自己某分数能上什么学校

2.某个特定的专业哪个学校分数最高

3.已知自己的分数判断自己能学什么样的专业

4.查询某学校的特定专业

5.我只想学 XX 专业，能去什么学校？

任务 3：知识图谱数据准备

1.对专业名称进行消歧处理

2.为有需要的实体生成标识符

任务 4：创建可以导入Neo4j 的 csv 文件

在第一个任务里，我们已经分获取了了包含所有信息的 csv，但这些文件不能直接导入到 Neo4j 数据库。所以需要做一些处理，并生成能够直接导入 Neo4j 的 csv 格式。 我们需要生成这几个文件：暨设计环节决定的实体及这些实体之间相互关系对应的 csv 文件。

任务 5：利用上面的 csv 文件生成数据库

使用 neo4j 命令把所有的数据导入到 Neo4j 中，数据默认存在 graph.db 文件夹中。
重启 Neo4j 服务，通过 localhost:7474 观察知识图谱。使用自带的命令进行简单查询的测试。

任务 6：基于构建好的知识图谱，构建显示网页

个人任务:

1. 分析知识图谱和已小组成员清洗的数据
2. 创建可以导入Neo4j 的 csv 文件。

三、个人任务需求分析

个人任务:

1、 分析已有的数据和设计好的知识图谱框架，检查知识图谱设计的可实施性。
2、 检查数据正确性和完整性。
3、 创建可以导入Neo4j 的 csv 文件——分别处理实体文件：province、subject、year 和关系文件：located_in、need_score。

需求分析:

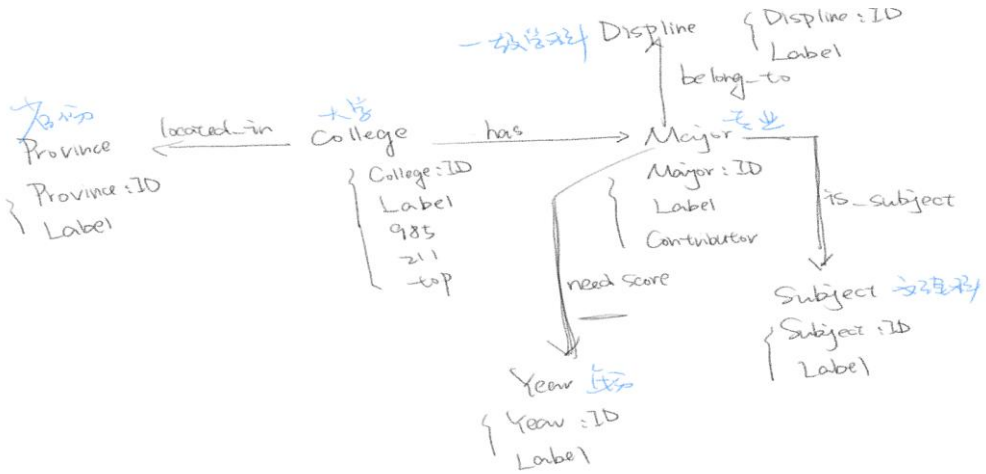
1、 设计合理的知识图谱框架。
2、 对专业名称进行消除歧义处理的数据。
3、 实体数据有唯一标识符，实体与实体间关系明确。

四、实验过程（需附上关键代码及相关说明）

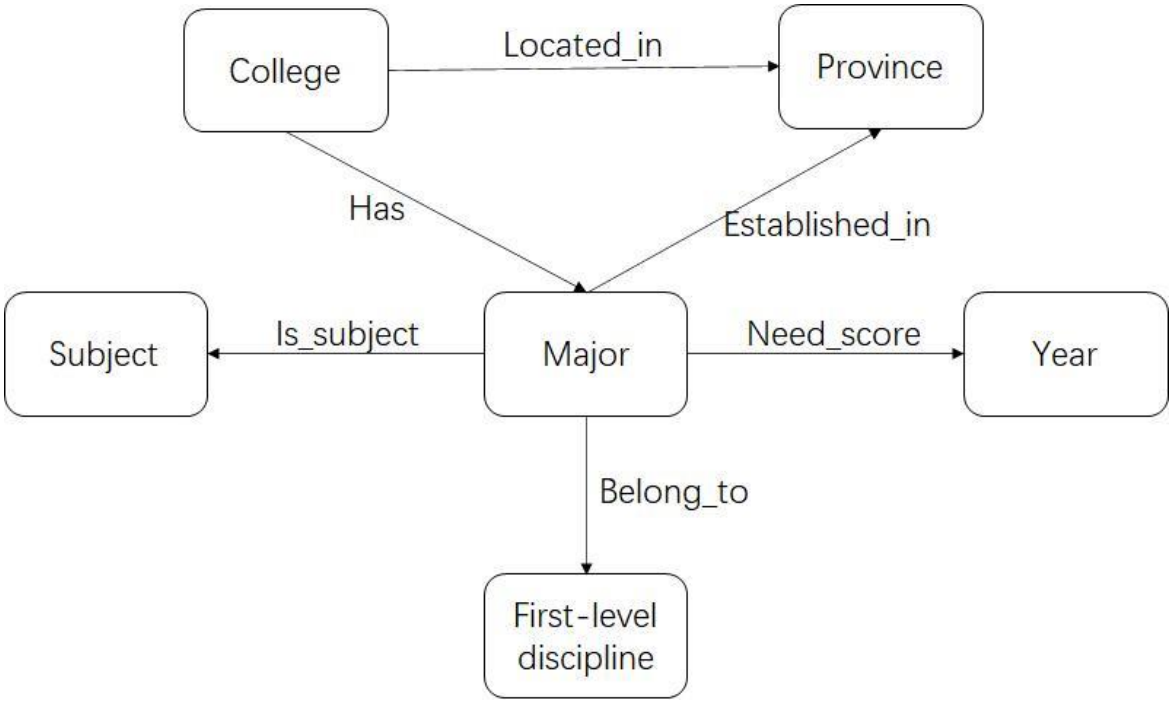
本次实验，我和王明颢、王靖婷在组内再组成一个小组，主要负责创建可以到导入 Neo4j 的文件，并生成知识图谱。具体实验过程如下：

1、 分析数据和知识图谱设计

获得了已经数据清洗完了的 csv 文件，其包括实体文件夹 entity 和关系文件夹 relation：其中，entity 文件夹包含 college、displine、major、province、subject 和 year 等实体表；relation 文件夹包含 belong_to、has、is_subject、located_in 和 need_score 等关系表，和由朱佳涛、白劭宸设计最初的知识图谱：



经过分析和与王靖婷、王明颢的讨论，我们一致认为省份不应该作为专业的属性，我们在此基础上增加一个 major 和 province 的关系 established_in，我们修改后的设计图如下：



2、清洗脏数据

在处理数据时，我们发现 has.csv 关系表中的一部分脏数据，如'c 北京理科大学'、'c?北京航空航天大学'、'c 华东理科大学'，因为其名称与原本名称有出入，导致标识符不能与其对应，生成关系时也未匹配到对应标识符。我编写代码 Process_dirtydata.py 来处理，代码如下（关键代码已标红）：

```
import csv
# 处理关系表 has 里的脏数据
input = open(r'has.csv/', 'r', encoding='utf-8')
output = open(r'../has.csv/', 'w', newline='', encoding='utf-8')
data = csv.reader(input)
writer = csv.writer(output)

flag = 0
for j in data:
    arr = j[:]
    if flag == 0:
        writer.writerow(arr)
        flag += 1
        continue

    if arr[0]=='c 北京理科大学':
        arr[0]=='c10007'
    if arr[0]=='c?北京航空航天大学':
        arr[0]=='c10006'
    if arr[0]=='c 华东理科大学':
        arr[0]=='c10561'

    writer.writerow(arr)

input.close()
output.close()
```

3、创建可以导入Neo4j 的 csv 文件

因为 Neo4j 需要固定的数据格式才能使用，我们将我们清理后的数据转换为可导入 Neo4j 的文件，即：修改实体表和关系表的的 title，并增加:LABEL 属性，其编码格式统一为 utf-8。我处理了 entity 文件夹的 province、subject 和 year 的实体表和 relation 文件夹 located_in 和 need_score 的关系表，代码如下（关键代码已标红）：

1） Process_entity_罗琦晴.py

```
import csv


# 处理实体的 title 并增加:LABEL 属性
csv_files = [r"province.csv", r"subject.csv", r"year.csv"]
labels = ["province", "subject", "year"]
title = [['Province:ID','Name',':LABEL'],
         ['Subject:ID','Name',':LABEL'],
         ['Year:ID','Name',':LABEL']]
i = 0
for file in csv_files:
    input = open(file, 'r')
    output = open('../import./'+file, 'w', newline='', encoding='utf-8')

    data = csv.reader(input)
    writer = csv.writer(output)

    flag = 0
    for j in data:

        arr = j[:]

        if flag == 0:
            writer.writerow(title[i])
            flag += 1
```

```
                continue


                arr.append(labels[i])#增加 label
                writer.writerow(arr)


            i += 1
            input.close()
        output.close()
```

2） Process_relation_罗琦晴.py

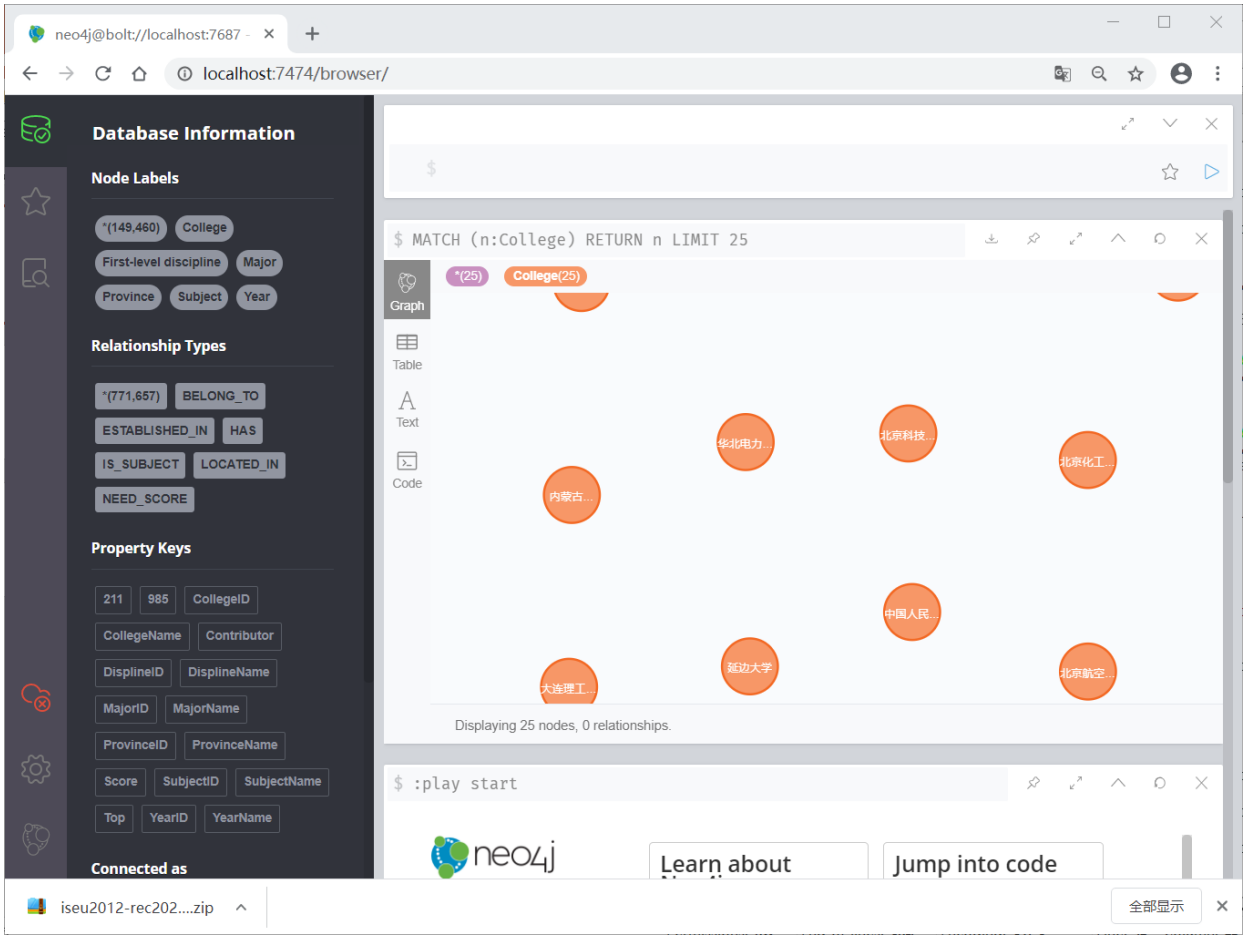
```
        import csv

        # 处理 need_score 表（title)
        input = open(r'need_score.csv', 'r', encoding='utf-8')
        output = open(r'../import./need_score.csv', 'w', newline='', encoding='utf-8')

        data = csv.reader(input)
        writer = csv.writer(output)

        flag = 0
        for j in data:

            arr = j[:]

            if flag == 0:
                writer.writerow([':START_ID','need_score',':END_ID',':TYPE'])# 重新处理 title
                flag += 1
                continue

            writer.writerow(arr)
```

```
    input.close()
    output.close()



    # 处理 located_in 表(1、title 2、删除不需要的列)
    input = open(r'located_in.csv', 'r', encoding='utf-8')
    output = open(r'../import./located_in.csv', 'w', newline='', encoding='utf-8')

    data = csv.reader(input)
    writer = csv.writer(output)

    flag = 0
    for j in data:

        arr = j[:]

        if flag == 0:
            writer.writerow([':START_ID',':END_ID',':TYPE']) # 修改 title
            flag += 1
            continue

        arr = [arr[0],arr[2],arr[3]] # 删除不需要信息
        writer.writerow(arr)

    input.close()
output.close()
```

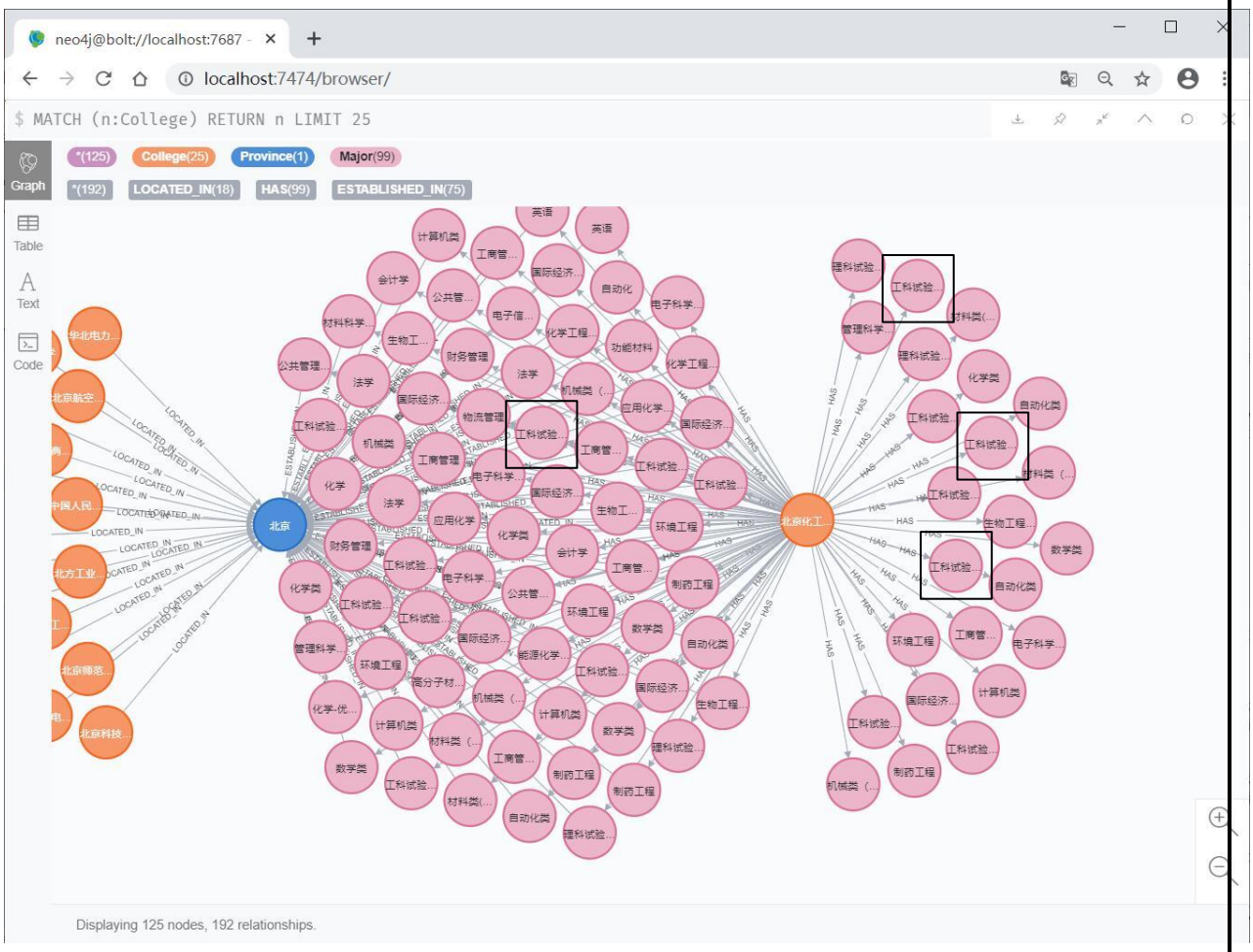
五、实验结果与分析

经过对数据和知识图谱设计分析、脏数据的清洗和可以导入Neo4j 的 csv 文件的创建等工作后，我们最终生成了知识图谱的第一个版本：

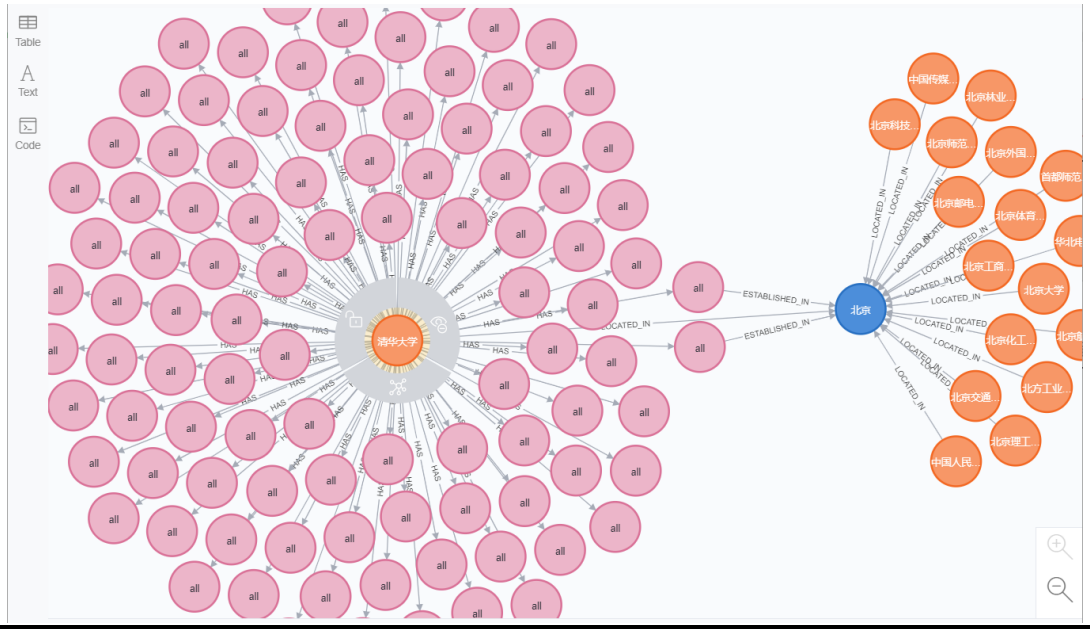
首先，进入首页时，我们可以看到诸多学校



选择其中一个学校，我们可以看到该校拥有的所有专业和该校所属省份，以及该省份可报考的所有高校信息。但由于知识图谱设计的问题——将不同年份、不同省份的同一专业作为不同的实体，所以会出现该学校有多个相同名称的专业，如下图所示的北京化工大学有多个“工科试验”的专业，这使用户体验感受较差，难以快速找到所需的信息。



更为不方便的地方是有的学校对于不同年份和省份的招生专业为所有专业的最低分数线，使所有专业的名称 name 属性全为“all”，以至于难以进行信息查找。



我们小组基本解决了实体标识符还有实体间关系等数据问题，但我们发现的诸多不合理的地方总结如下：

1、 数据问题

1） 有的实体因为数据清洗疏忽的原因，名称与原本名称有出入，导致生成关系时未匹配到对应标识符。

2） 处理一对多的关系时，将一个实体分配了多个不同的标识符，导致标识符不唯一。

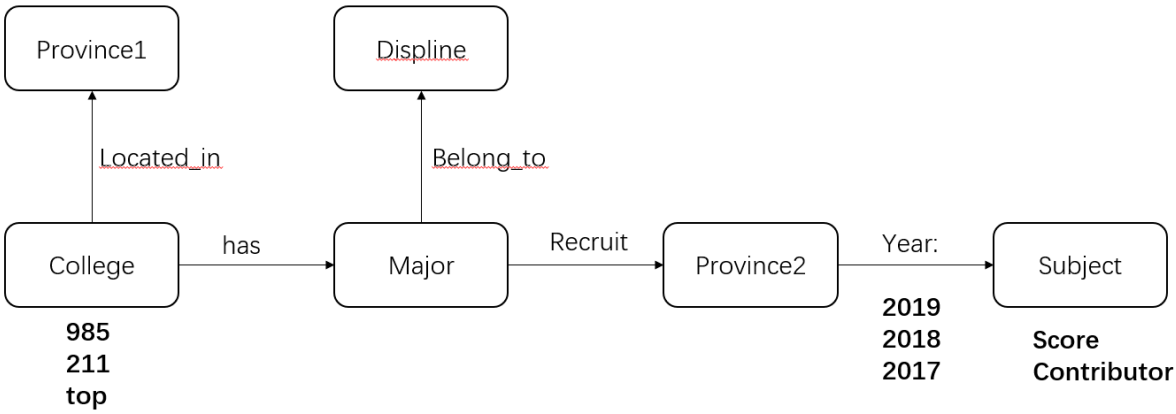
2、 知识图谱设计问题

1） 将一条记录作为一个 major 实体，直接与学校连接关系，导致对于某一学校的不同省份不同年份的同一专业名多次出现，视觉感受上十分混乱。

2） 建议在学校和专业间增加省份的实体和年份的关系，使查询时更加直观形象。

因最终生成知识图谱后，知识图谱设计问题明显，我们在小组内提出了我们发现的问题，积极参与讨论，并且给出知识图谱设计意见，最后返还给设计人员重新处理。

我们提出的改进知识图谱如下所示：



我们提出的知识图谱，将查找过程设定为对于某一大学，首先显示有哪些专业，这些专业分别在哪些省份招生，之后可以对于同一省份查看不同年份的理科和文科招生分数。我们重新的设计图谱优点如下：

1） 增加一个省份 Province2 的实体，使专业实体 major 的名称唯一，避免名称混乱的情况出现。

2） 将年份 Year 作为身份 Province2 和 Subject 的关系，关系名称分别为 2019、2018、2017，解决了分数是由年份和科目共同决定的问题。

3） 查询方式与用户思维方式相符合，用户体验感增强。

六、实验总结与心得体会

1、 第一次接触有关知识图谱的设计、生成和使用的实验，我自主学习了知识图谱的相关知识，研读 Neo4j 的使用说明文档和实例，并加以运用。活学活用，收获满满。

2、 第一次参与大项目的实践，体会到了项目集体配合的过程，小组成员工作的并行和串行关系，互相牵连，互相影响，在任务和工作交接能力方面，有所提升。切实感受到了我们的工作从最开始的一盘散沙、摸不着头脑，变得有条有序、密切合作，对于一个集体合作新项目，团队精神是十分重要的，只有我们不断配合、调整，最终才能完成满意的结果。

3、 对于本次自己的任务，感觉主要起了中间过程中的纠错的作用。我和王靖婷、王明颢在组内在成立一个小组，主要负责创建可以导入 Neo4j 的文件并生成图谱，在饰演的过程中，遇到了很多错误点，磕磕碰碰，我们一起发现并指出了诸多数据清洗和知识图谱设计的问题，给设计组提供了有价值的修正意见，给之后的实践组提供了宝贵的方向和新的图谱设计。

2020 年 9 月制