****

**《知识工程项目1：COVID-19知识图谱系统开发》**

**项目报告**



**题目：COVID-19知识图谱系统开发**

**学 院 环境科学与工程学院**

**专 业 环境科学**

**年 级 2022**

**姓 名**

**学 号 2022214034/2022214038**

目录

[第一章 项目背景 1](#_Toc3172)

[1.1 知识图谱背景 1](#_Toc26377)

[1.1.1 概念 1](#_Toc16092)

[1.1.2 发展历史 1](#_Toc17424)

[1.1.3 知识图谱应用的优点 2](#_Toc663)

[1.2 新冠疫情的背景 2](#_Toc16395)

[1.3 病毒与蛋白质研究的内在联系 2](#_Toc12868)

[第二章 需求分析 3](#_Toc23247)

[2.1 问题分析 3](#_Toc9913)

[2.1.1问题分析的对象 3](#_Toc26825)

[2.1.2具体对象分析 3](#_Toc3547)

[2.1.3用例分析 3](#_Toc1745)

[第三章 项目设计 4](#_Toc9630)

[3.1 架构设计 4](#_Toc9623)

[3.1.1 目的 4](#_Toc25657)

[3.1.2 读者对象 4](#_Toc12702)

[3.1.3 约束设计 4](#_Toc7872)

[3.1.4 系统总体架构 4](#_Toc17370)

[3.2 概要设计 5](#_Toc2493)

[3.3详细设计 5](#_Toc26731)

[3.3.1目的 5](#_Toc2958)

[3.3.2流程图 5](#_Toc13328)

[第四章 系统实现 8](#_Toc22143)

[4.1 数据库实现 8](#_Toc26175)

[4.1.1 Neo4j简介 8](#_Toc12550)

[4.1.2 Neo4j环境搭建 8](#_Toc1296)

[4.1.3 将数据集导入Neo4j数据库 9](#_Toc17542)

[4.1.4 知识图谱查询 11](#_Toc2388)

[4.1.5 知识图谱数据挖掘 13](#_Toc20357)

[4.2 前端实现 16](#_Toc9362)

[4.3 后端实现 19](#_Toc15932)

[4.3.1 axios 19](#_Toc14272)

[4.3.2 页面初始化 19](#_Toc1246)

[第五章 系统测试 28](#_Toc2289)

[5.1 实体 28](#_Toc12574)

[5.2 实体的关系 28](#_Toc30548)

[5.3 关系 28](#_Toc5290)

[5.4 查询 29](#_Toc10288)

[5.5 数据挖掘 30](#_Toc14520)

[5.6 属性 32](#_Toc15557)

[第六章 项目总结 33](#_Toc2784)

# 第一章 项目背景

## 知识图谱背景

### 概念

知识图谱（Knowledge Graph）是人工智能的重要分支技术，它在2012年由谷歌提出，是结构化的语义知识库，用于以符号形式描述物理世界中的概念及其相互关系，其基本组成单位是“实体—关系—实体”三元组，以及实体及其相关属性—值对，实体间通过关系相互联结，构成网状的知识结构。

### 发展历史

知识图谱的起源可以追溯至1960年，在人工智能的早起发展中，有两个主要的分支，也就是两派系，一个是符号派，注重模拟人的心智，研究如何用计算机符号表示人脑中的知识，以此模拟人的思考、推理过程；一个则是连接派，注重模拟人脑的生理结构，由此发展了人工神经网络。这个时候提出了Semantic Networks，也就是语义网络，作为一种知识表示的方法，主要用于自然语言理解领域。

1970年，随着专家系统的提出和商业化发展，知识库(Knowledge Base)构建和知识表示得到重视。专家系统的主要思想认为专家是基于脑中的知识来进行决策的，所以为了实现人工智能应该用计算机符号来表示这些知识，通过推理机来模仿人脑对知识进行处理。早期的专家系统常用的知识表示方法有基于框架的语言(Frame-based Languages)和产生式规则(Production Rules)。框架语言用来描述客观世界的类别、个体、属性等，多用于辅助自然语言理解；产生式规则主要用于描述逻辑结构，用于刻画过程性知识。

1980年，哲学概念“本体”(Ontology)被引入人工智能领域来刻画知识，本体可以说是知识的本体，一条知识的主体可以是人，可以是物，可以是抽象的概念，本体就是这些知识的本体的统称。1989年，Tim Berners-Lee在欧洲高能物理研究中心发明了万维网，人们可以通过链接把自己的文档链入其中，在万维网概念的基础上，1998年又提出了语义网(Semantic Web)的概念，与万维网不同的是，链入网络的不止是网页，还包括客观实际的实体（如人、机构、地点等）。2012年谷歌发布了基于知识图谱的搜索引擎。

### 知识图谱应用的优点

知识图谱可以应用于搜索和问答，也可以用于辅助进行数据分析与决策。基于知识图谱的搜索更加便捷与准确，不同来源的知识通过知识融合进行集成，通过知识图谱和语义技术增强数据之间的关联，用户可以更直观地对数据进行分析。此外知识图谱也被广泛用于作为先验知识从文本中抽取实体和关系，也被用来辅助实现文本中的实体消歧，指代消解等。

## 新冠疫情的背景

由严重急性呼吸综合征冠状病毒-2（SARS-CoV-2，简称新冠病毒）引起的2019冠状病毒病（COVID-19）大流行已进入第3年，在这2年多时间内新冠病毒持续变异，出现多个适应性强的变异株，造成多波全球性疫情高峰。根据世界卫生组织2022年9月发布的数据，全球超过6.1亿人感染，超过650万人死亡。尽管科学界进行了深入研究，但关于其分子机制的许多重要问题仍未得到解答。

## 病毒与蛋白质研究的内在联系

病毒是细胞内的病原体，依靠与宿主蛋白的相互作用来确保存活和繁殖。蛋白质相互作用组研究不仅为“小科学” (small science) 研究提供了重要的功能机制“线索”, 而且是系统生物学研究的重要“基石”。 内源的蛋白质异常 (包括表达量过高或不足、突变和异常定位等) 导致蛋白质相互作用网络的改变, 是机体发生疾病的内在决定因素;外源物质 (致癌物、过敏原和病原体毒力分子等) 的扰动引起蛋白质相互作用网络的变化, 诱发炎症、肿瘤等各种表型异常。全面地获得细胞、组织和器官等蛋白质相互作用组的“组成规律”, 系统地理解其在生长、发育等过程中的动态“调控规律”, 深入地剖析其在异常状态下的“变化规律”, 将有助于系统揭示生物学表型的基本分子 (群) 机制、有利于疾病的防治和人类的健康生活。

因此，研究SARS-CoV-2与人类蛋白质之间的蛋白质-蛋白质相互作用（PPI）对于了解SARS-CoV-2病毒感染的机制和开发治疗方法非常重要。本文提供了一种基于新冠肺炎病毒有关蛋白质间相互作用的知识图谱的可视化分析，研究者可输入蛋白质名称或序列进行查询该蛋白质的相互作用网络，结果以可点击的互动网络图进行展示，并且可根据需要选取特定来源的数据或扩展的网络进行重新绘图。

# 第二章 需求分析

## 2.1 问题分析

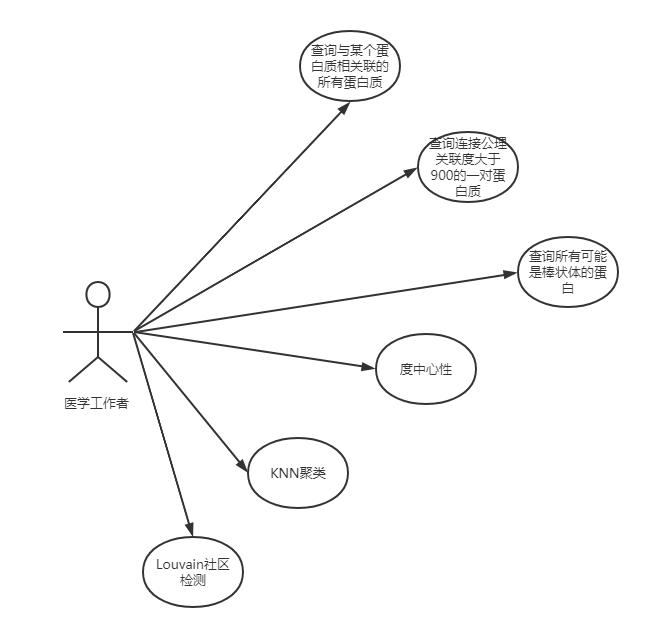
### 2.1.1问题分析的对象

研究COVID-19的医学工作者

### 2.1.2具体对象分析

为了对抗COVID-19，临床医生和科学家需要阅读科学文献中大量医学相关典籍，以了解病毒特征，我们构建COVID-19知识图谱来提取病毒相关的知识元素，方便医学工作者便捷快速地搜索与病毒相关的知识。

### 2.1.3用例分析



在本系统中，具备查询与某个蛋白质相关联度前10的蛋白质，查询连接公理关联度大于X的一对蛋白质，查询所有可能是棒状体的蛋白质三个查询功能，还具备度中心性，KNN聚类，Louvain社区检测三个数据挖掘功能。

# 第三章 项目设计

## 3.1 架构设计

### 3.1.1 目的

COVID-19知识图谱系统体系架构设计报告主要描述系统的整体技术架构，帮助模块设计人员、开发人员对系统有个整体认知。它是系统分析人员将需求转换成为开发人员所比较容易理解的结构描述；并且，高屋建瓴的指导开发人员的开发方向。

### 3.1.2 读者对象

COVID-19知识图谱系统体系架构设计报告的读者主要分为三部分人：用户、开发人员和测试人员。其中，主要读者是开发人员和测试人员。开发人员主要是对技术体系架构有整体认识，确保其在进行模块开发之时不至于偏离；测试人员主要对设计约束进行了解，以构建在测试的时候，对系统整理架构的测试基准。

### 3.1.3 约束设计

设计约束是系统在架构设计的时候，应该遵循的规范准则。其详细如下：

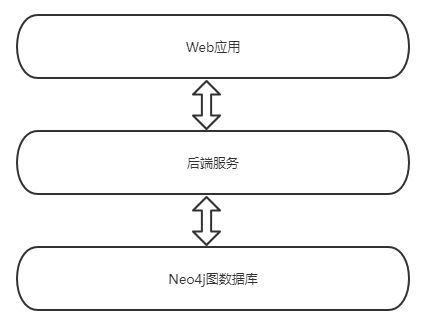
1. 需求约束：系统在设计之时，严格遵循需求分析中所约定的需求范围。

2. UI 设计约束：在进行 UI 设计时，将充分考虑使用者的计算机应用水平，尽可能的整体形成统一的操作规范风格。

本章详细说明体系结构设计人员根据产品的需求与发展战略，确定的设计策略。在本系统设计时，主要涉及两类策略：扩展策略和复用策略。其中扩展策略主要偏重于业务上的延伸，而复用策略来自于底层技术实现的接口复用。

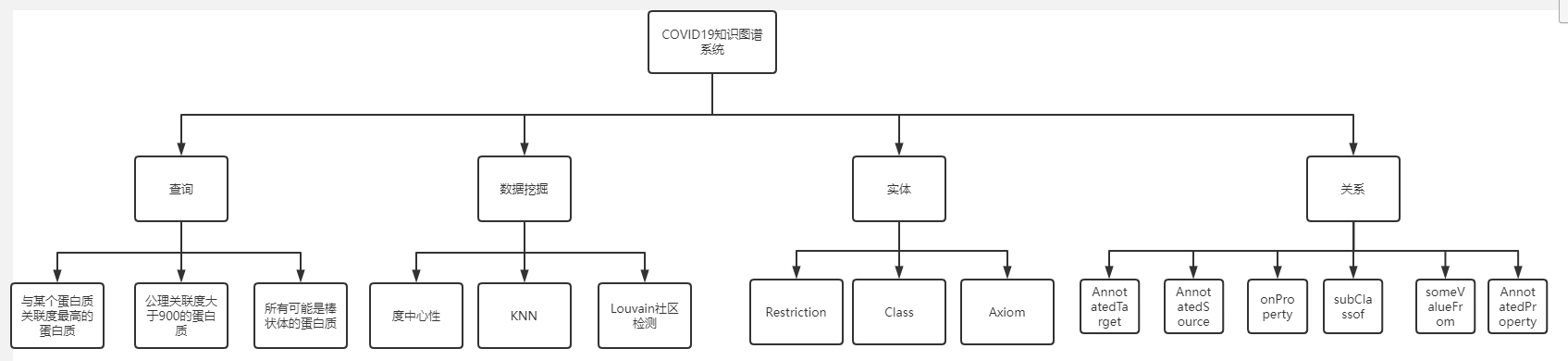
### 3.1.4 系统总体架构

本系统在整体架构上，分为三个层次：数据库层，后端服务层，web应用层，其具体架构图如下：



## 3.2 概要设计

概要设计更多从宏观的角度确定软件系统的总体布局，如软件的功能模块如何划分，各模块之间的关系是什么样子的，软件的逻辑结构是什么，本系统概要设计图如下：



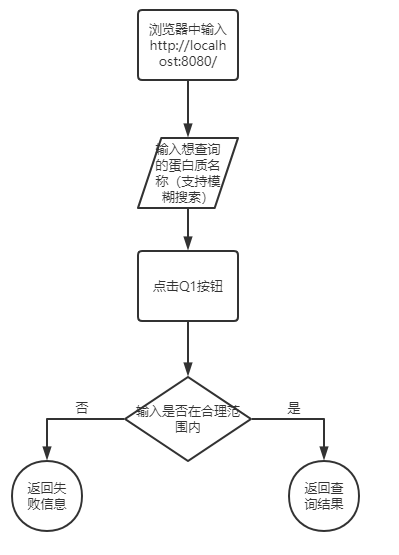
## 3.3详细设计

### 3.3.1目的

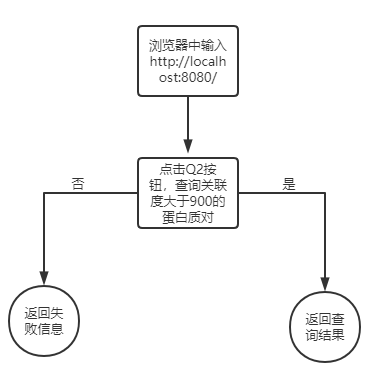
详细设计阶段的任务就是把解法具体化，解决具体应怎样实现这个系统。也称为模块设计，详细地设计每个模块，确定实现模块所需的功能需要的算法和数据结构。

### 3.3.2流程图

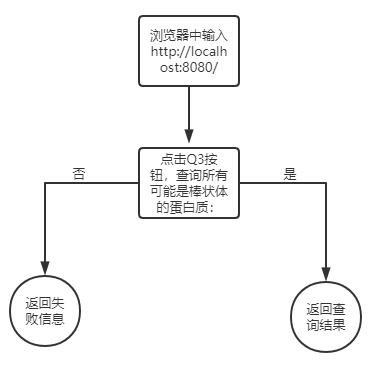
（1）查询与某个蛋白质相关联度前10的蛋白质



（2）查询连接公理关联度大于900的一对蛋白质（限制100条）



（3）查询所有可能是棒状体的蛋白质



度中心性，KNN聚类，Louvain社区检测等数据挖掘实例的操作流程与上述流程一致，在web界面上点击相应的按钮，触发底层数据挖掘算法，完成相应查找。

# 第四章 系统实现

## 4.1 数据库实现

### 4.1.1 Neo4j简介

随着社交、电商、金融、零售、物联网等行业的快速发展，现实社会织起了一张庞大而复杂的关系网，传统数据库很难处理关系运算。大数据行业需要处理的数据之间的关系随数据量呈几何级数增长，急需一种支持海量复杂数据关系运算的数据库，图数据库应运而生。Facebook, Twitter，Linkedin使用图数据库来管理社交关系，实现好友推荐；eBay，沃尔玛使用它实现商品实时推荐，给买家更好的购物体验；图数据库是基于图论实现的一种NoSQL数据库，其数据存储结构和数据查询方式都是以图论为基础的，图数据库主要用于存储更多的连接数据。Neo4j是世界上最先进的开源NoSQL图形数据库之一，提供原生的图数据存储，检索和处理，采用属性图模型，极大地完善和丰富图数据模型。为其专门的设计的Cypher查询语句直观易懂且非常高效，故本项目采用Neo4j作为图数据库。

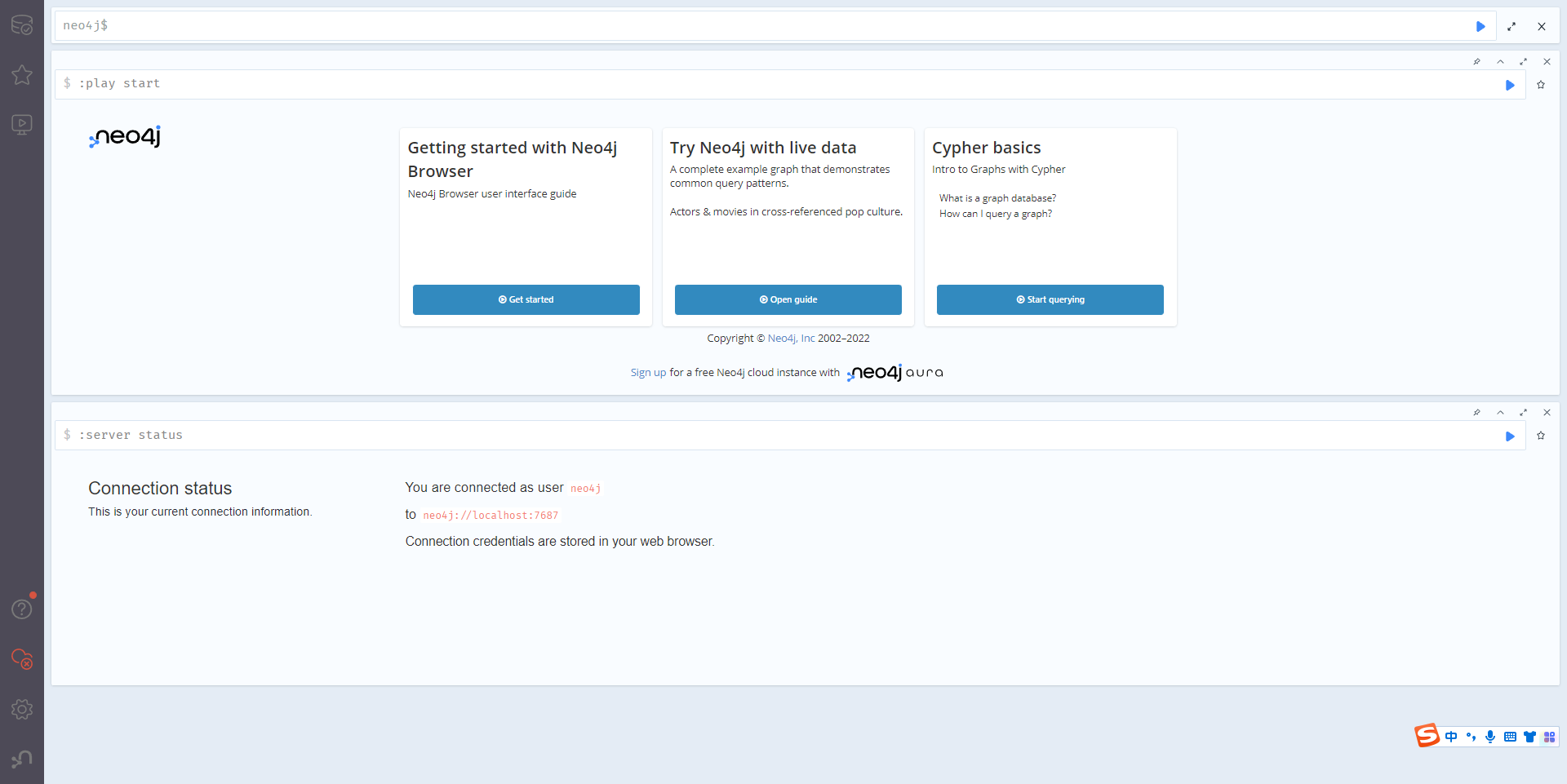
Neo4j图数据库主要有节点，属性，关系，标签，数据浏览器五种构建元素。节点是图数据库的基本元素，用来表示一个实体记录，类似于关系型数据库中的一条记录，一个节点可以包含多个属性和多个标签。节点之间可以通过关系相互连接。属性是用于描述图节点和关系的键值对，其中key是一个字符串，值可以是任何一种Neo4j支持的数据类型，属性可以被索引和约束，并可以从多个属性创建复合索引。关系也是图数据库的基本元素。关系用来连接两个节点，关系也称为图论的边，关系和节点一样可以包含多个属性，但只能有一个类型。标签将一个公共名称与一组节点或关系相关联，每一个节点或关系可以包含一个或多个标签。可以通过对标签进行索引以加速在图中查找节点。

### 4.1.2 Neo4j环境搭建

（1）下载Neo4j安装包

（2）在bin目录下执行./neo4j.bat console，即可启动Neo4j服务

（3）输入http://localhost:7474/，访问Neo4j数据浏览器，界面如下：



### 4.1.3 将数据集导入Neo4j数据库

本项目采用String human PPI数据集，该数据库存储蛋白质之间的相互作用网络。下载的PPI数据集是rdf格式，但Neo4j本身不支持导入RDF文件，因此需要借助neosemantics来导入。

（1）下载neosemantics-4.1.0.1.jar文件

（2）修改neo4j/neo4j.conf文件，添加以下内容：

dbms.unmanaged\_extension\_classes=n10s.endpoint=/rdf

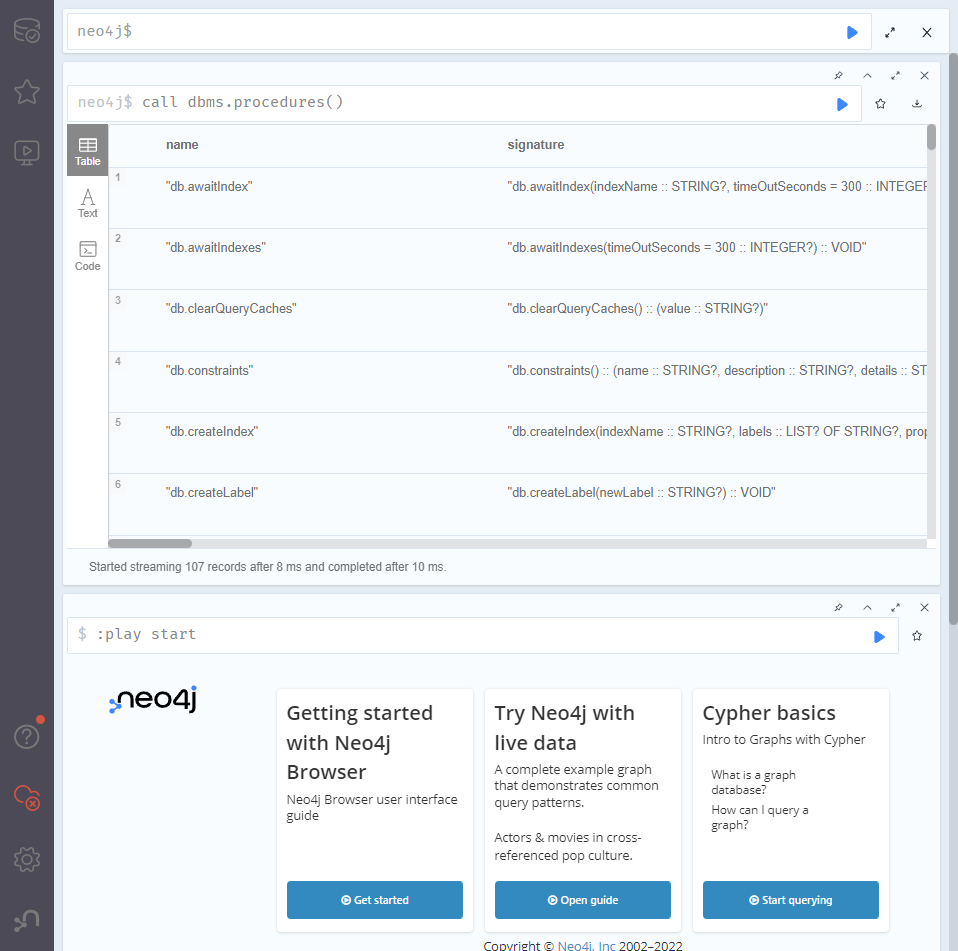
（3）输入如下命令，重新启动Neo4j：

./bin/neo4j restart

（4）浏览器访问Neo4j，输入如下命令，查看是否支持导入RDF

call dbms.procedures()

返回截图如下，即代表已支持导入RDF文件



（5）初始化，设置图形的配置

CALL n10s.graphconfig.init();

返回截图如下：



（6）创建唯一性约束

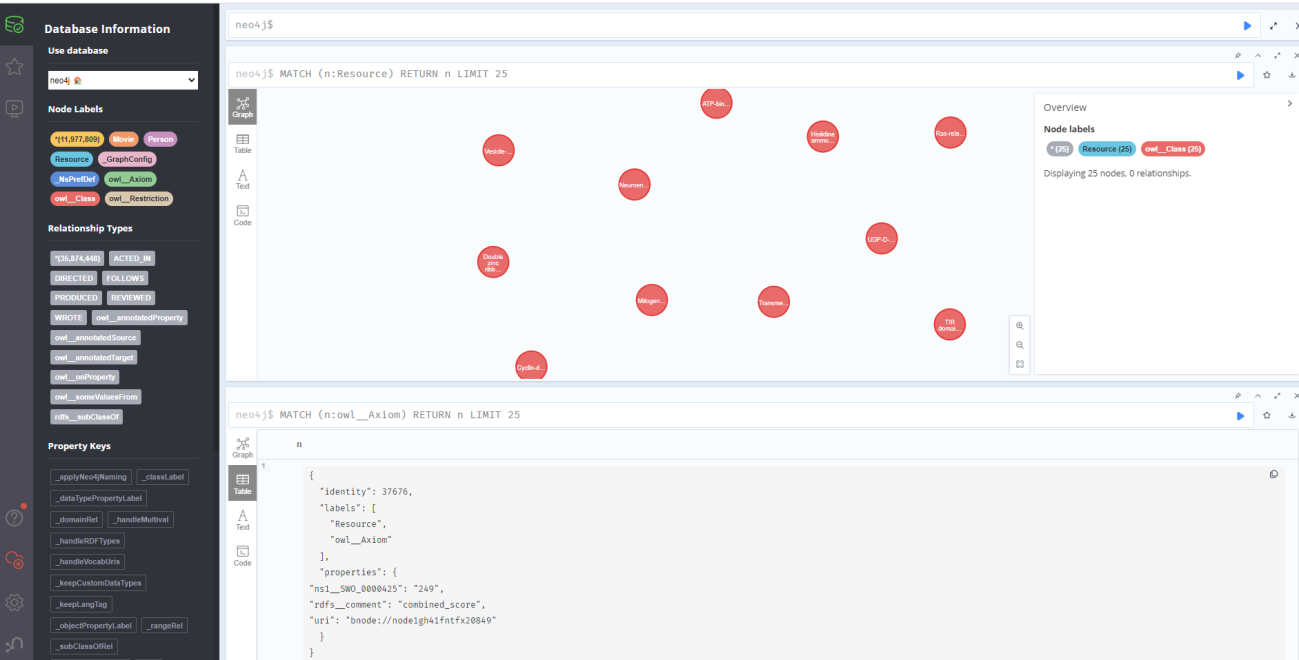
将数据持久化到Neo4j中的所有方法都有一个模式级的先决条件：这是在带有标签资源的节点的属性URI上存在唯一性约束，这个约束的目的是通过URI保证资源的唯一性，并通过将资源添加到索引来加速获取过程

CREATE CONSTRAINT n10s\_unique\_uri ON (r:Resource) ASSERT r.uri IS UNIQUE;

（7）导入数据，命令如下：

CALL n10s.rdf.import.fetch('file:///D:/Documents/stringhuman.rdf', 'Turtle')

导入成功后，Neo4j的浏览器界面展示如下：

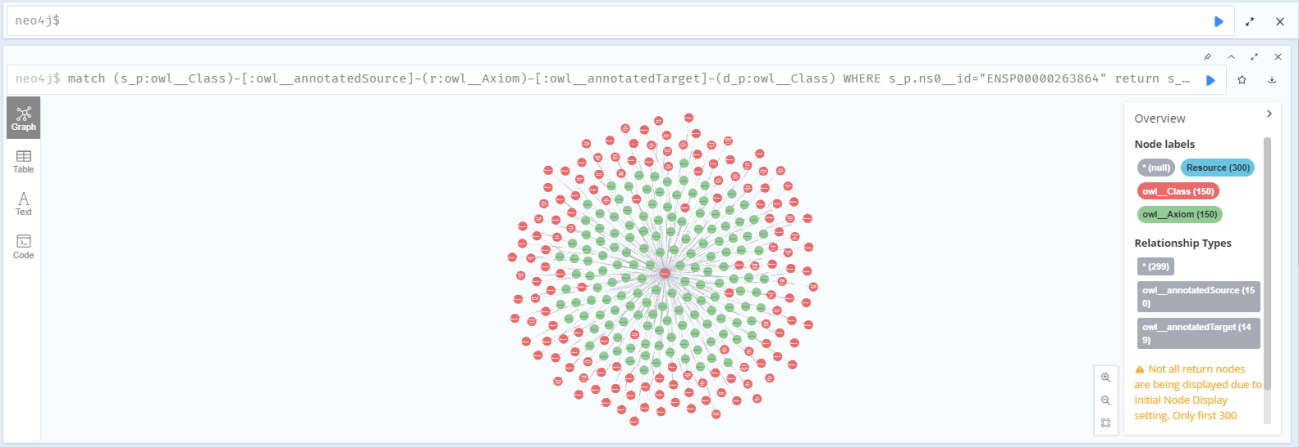


### 4.1.4 知识图谱查询

（1）查询与某个蛋白质相关联的所有蛋白质，即某个蛋白质的相互作用网络，并根据combine\_score降序排列,输出排名前1000的结果

|  |
| --- |
| SQL match (s\_p:owl\_\_Class)-[:owl\_\_annotatedSource]-(r:owl\_\_Axiom)-[:owl\_\_annotatedTarget]-(d\_p:owl\_\_Class) WHERE s\_p.ns0\_\_id="ENSP00000263864" return s\_p,r,d\_p Order by r.ns1\_\_SWO\_0000425 DESC LIMIT 1000 |

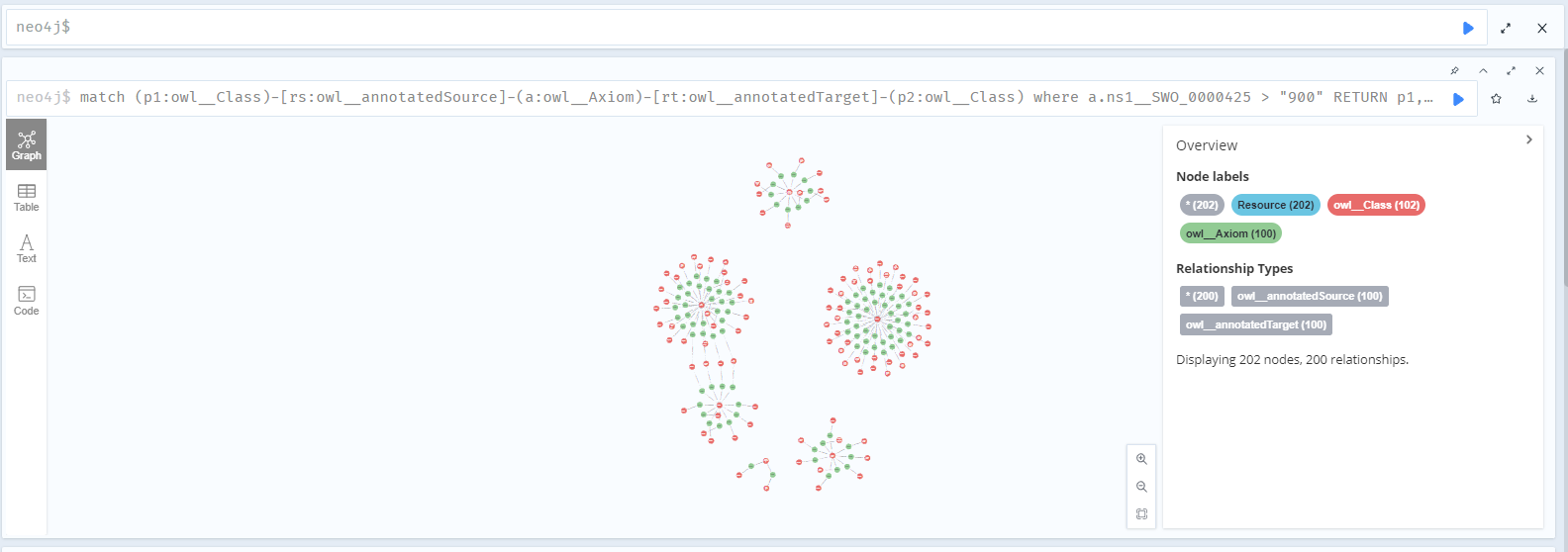
结果如下：



（2）查询连接公理关联度大于900的一对蛋白质（限制100条）

|  |
| --- |
| SQL match (p1:owl\_\_Class)-[rs:owl\_\_annotatedSource]-(a:owl\_\_Axiom)-[rt:owl\_\_annotatedTarget]-(p2:owl\_\_Class) where a.ns1\_\_SWO\_0000425 > "900" RETURN p1, labels(p1), rs, type(rs), a, labels(a), rt, type(rt), p2, labels(p2) LIMIT 100 |

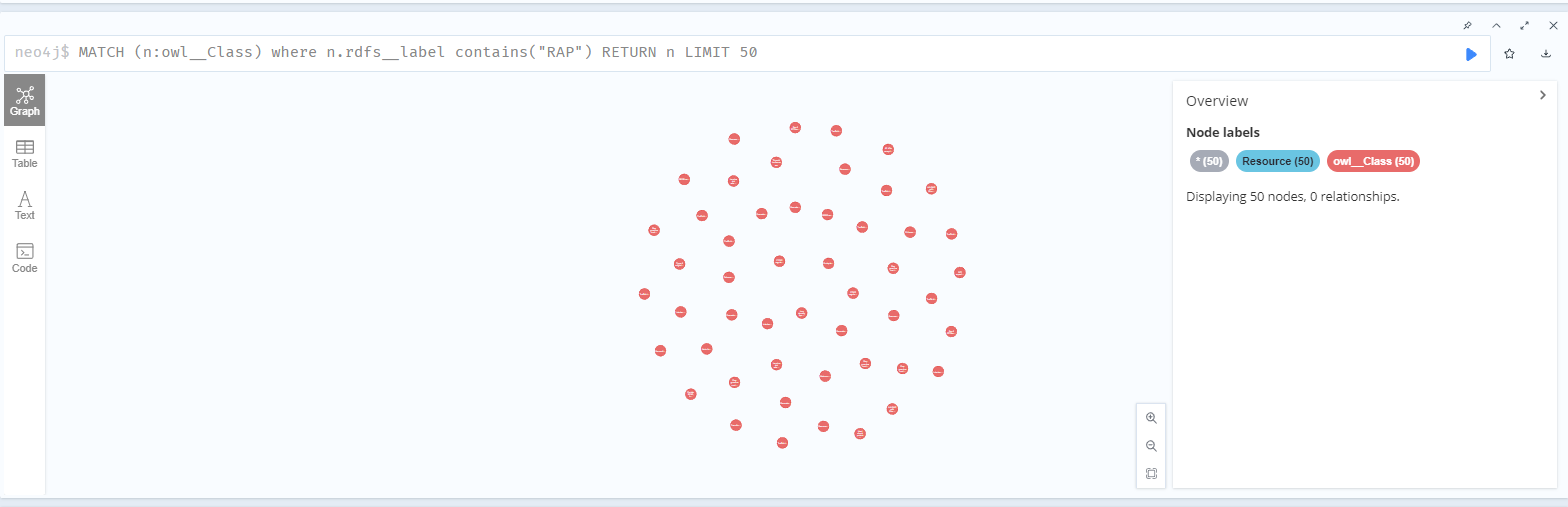
结果如下：



（3）查询所有可能是棒状体的蛋白，RAP蛋白是棒状体相关，因此包含RAP字段的蛋白很可能是棒状体相关蛋白。

|  |
| --- |
| SQL MATCH (n:owl\_\_Class)  where n.rdfs\_\_label contains("RAP")  RETURN n LIMIT 50 |

结果如下：



### 4.1.5 知识图谱数据挖掘

本项目知识图谱的数据挖掘部分的实现使用了Neo4j的图数据科学库（Graph Data Science，GDS）

（1）创建图映射

创建名称为“myGraph”的图映射，包含的节点类型包括“owl\_\_Class”类和“owl\_\_Axiom”类，关系类型包括“owl\_\_annotatedSource”和“owl\_\_annotatedTarget”，前者的方向按原始关系方向反向，后者方向和原始关系方向一致

|  |
| --- |
| SQL CALL gds.graph.project(  "myGraph",  ["owl\_\_Class", "owl\_\_Axiom"],  {owl\_\_annotatedSource: {orientation: "REVERSE"},  owl\_\_annotatedTarget: {orientation: "NATURAL"}  } ) |

结果如下：

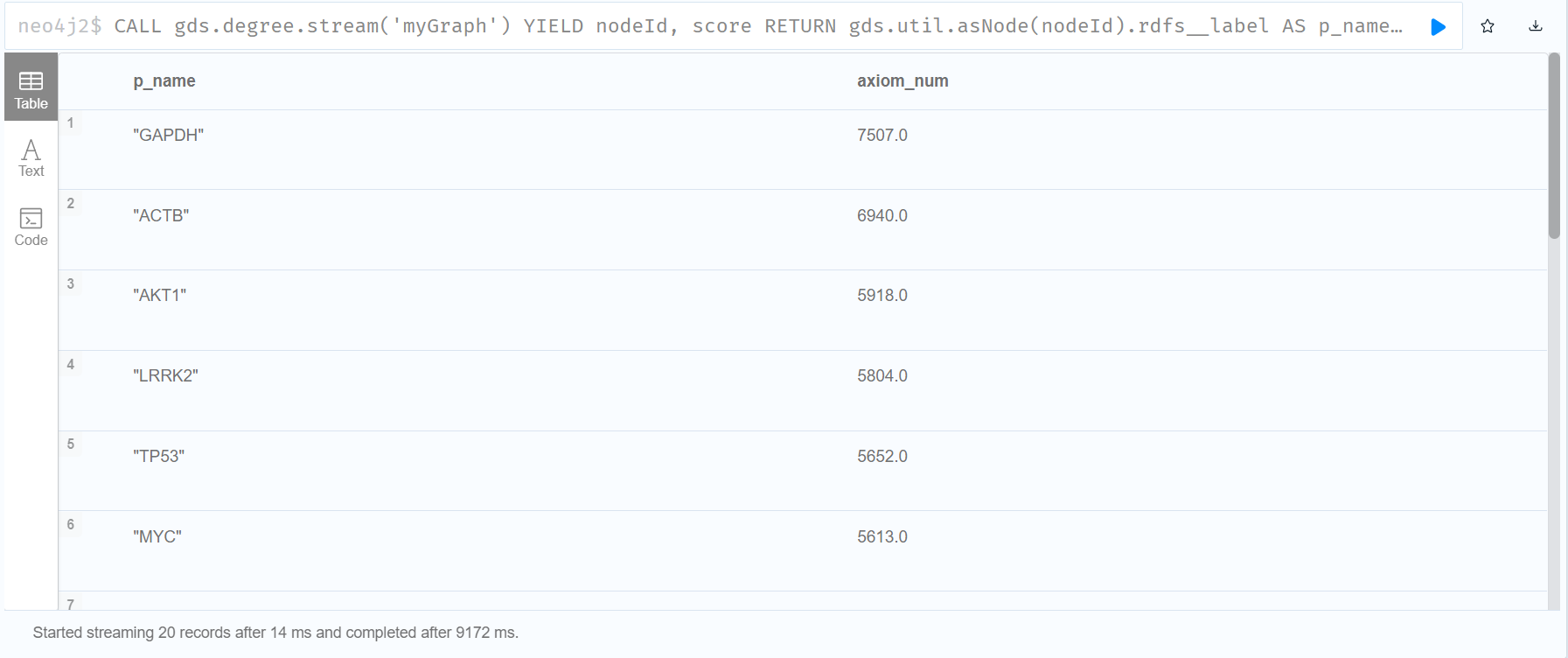


（2）数据挖掘算法一：度中心性

度中心性是最普遍和最简单的中心性算法之一，它计算一个节点拥有的关系数量，我们按照与每个Class节点相连接的Axiom数量进行排序，展示出Axiom数最多的前20个节点

|  |
| --- |
| SQL CALL gds.degree.stream('myGraph') YIELD nodeId, score RETURN gds.util.asNode(nodeId).rdfs\_\_label AS p\_name, score AS axiom\_num ORDER BY axiom\_num DESCENDING LIMIT 20 |

结果如下：



（3）数据挖掘算法二：KNN聚类

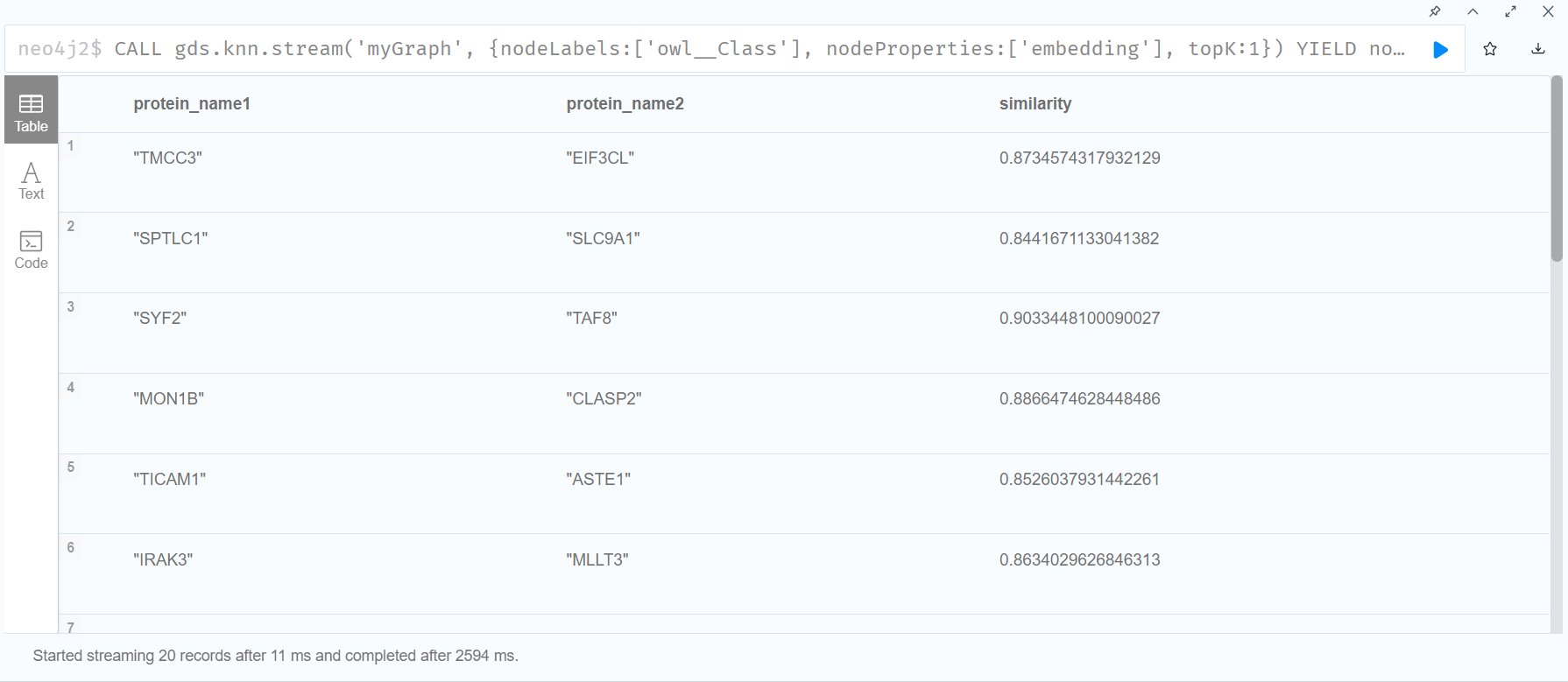
首先，需要在将图数据输入到机器学习算法之前，进行图嵌入。图嵌入是一种将图数据（通常为高维稠密的矩阵）映射为低微稠密向量的过程，能够很好地解决图数据难以高效输入机器学习算法的问题。GDS提供了一种叫做快速随机投影的节点嵌入技术的个性化实现，简称 FastRP。FastRP 利用概率抽样技术生成图的稀疏表示，允许极快地计算嵌入向量，其质量与传统的随机游走和神经网络技术 产生的向量相当。刚开始在 GDS 中探索图的嵌入时，FastRP 将是很好的选择。代码如下：

|  |
| --- |
| SQL CALL gds.fastRP.mutate('myGraph', {  embeddingDimension:16,  randomSeed:7474,  mutateProperty:'embedding' }) |

然后再运行KNN聚类算法，并使用默认的余弦度量，并将topK设置为1，从而查找每个节点的最相似的配对

|  |
| --- |
| SQL CALL gds.knn.stream('myGraph', {nodeLabels:['owl\_\_Class'], nodeProperties:['embedding'], topK:1}) YIELD node1, node2, similarity RETURN gds.util.asNode(node1).rdfs\_\_label AS protein\_name1,  gds.util.asNode(node2).rdfs\_\_label AS protein\_name2,  similarity LIMIT 20 |

运行结果如下：



（4）数据挖掘算法三：Louvain社区检测

社区检测算法被用来评估节点组在图中的聚类或分区情况。GDS 的大部分社区检测功能集中在区分和分配这些节点组的 ID，以便进行下游分析，可视化或其他处理。一个常见的社区检测算法是 Louvain 算法.。Louvain 算法对每个社区的模块化程度进行了最大化。模块化度量了将节点分配给社区的质量。就是度量一个节点在社区中联系的紧密程度比在随机网络中的联系紧密多少。

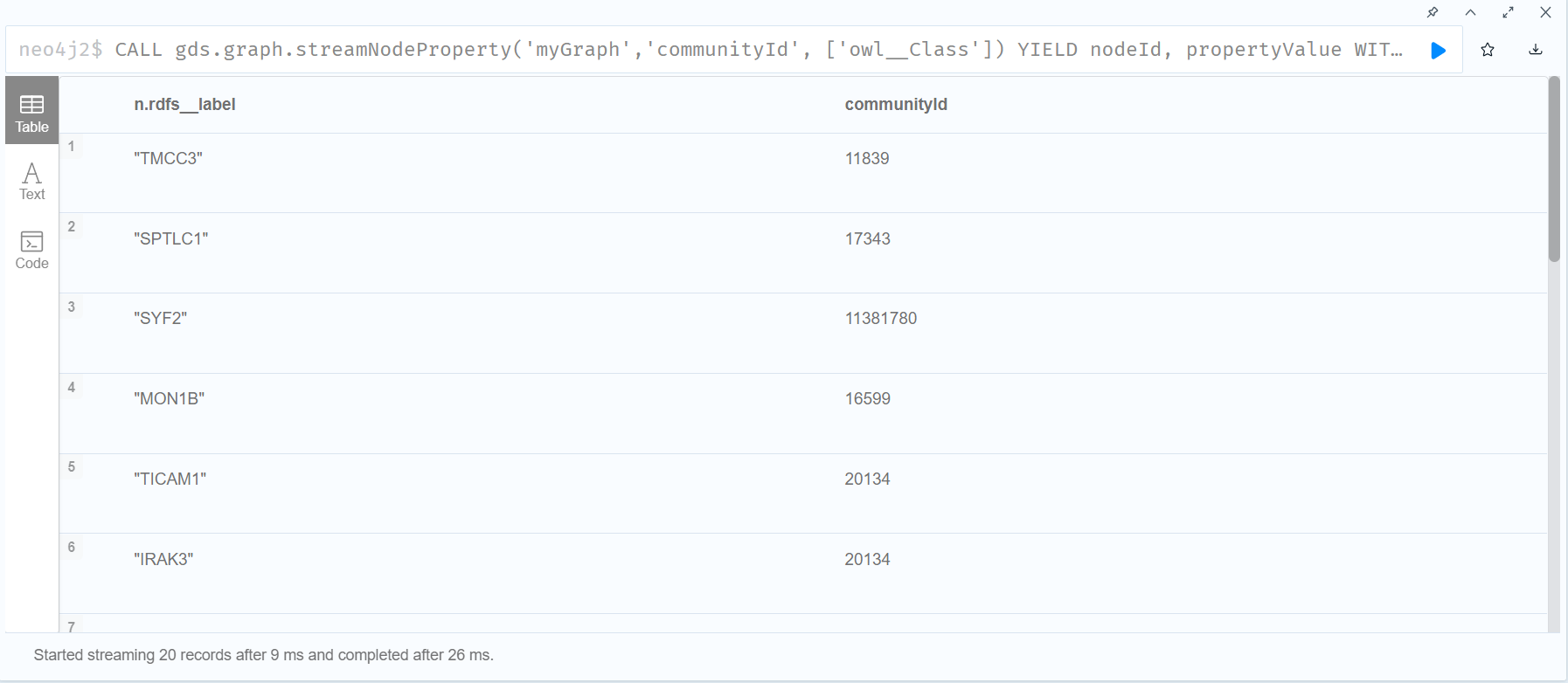
然后运行Louvain算法。这里，我们将在mutate模式下运行 Louvain 算法，以保存社区 ID, 并返回社区数量，分布，模块化程度的统计数据，以及 Louvain 算法如何处理图形的信息。

|  |
| --- |
| SQL CALL gds.louvain.mutate('myGraph', {mutateProperty:'communityId'}) |

然后再通过stream操作验证投影中的communityId节点属性。

|  |
| --- |
| SQL CALL gds.graph.streamNodeProperty('myGraph','communityId', ['owl\_\_Class']) YIELD nodeId, propertyValue WITH gds.util.asNode(nodeId) AS n, propertyValue AS communityId WHERE n:owl\_\_Class RETURN n.rdfs\_\_label, communityId LIMIT 20 |

运行结果如下：



可见其为每一个节点都分配了communityID

## 4.2 前端实现

前端页面的实现通过html语言和css语言实现，部分代码如下：

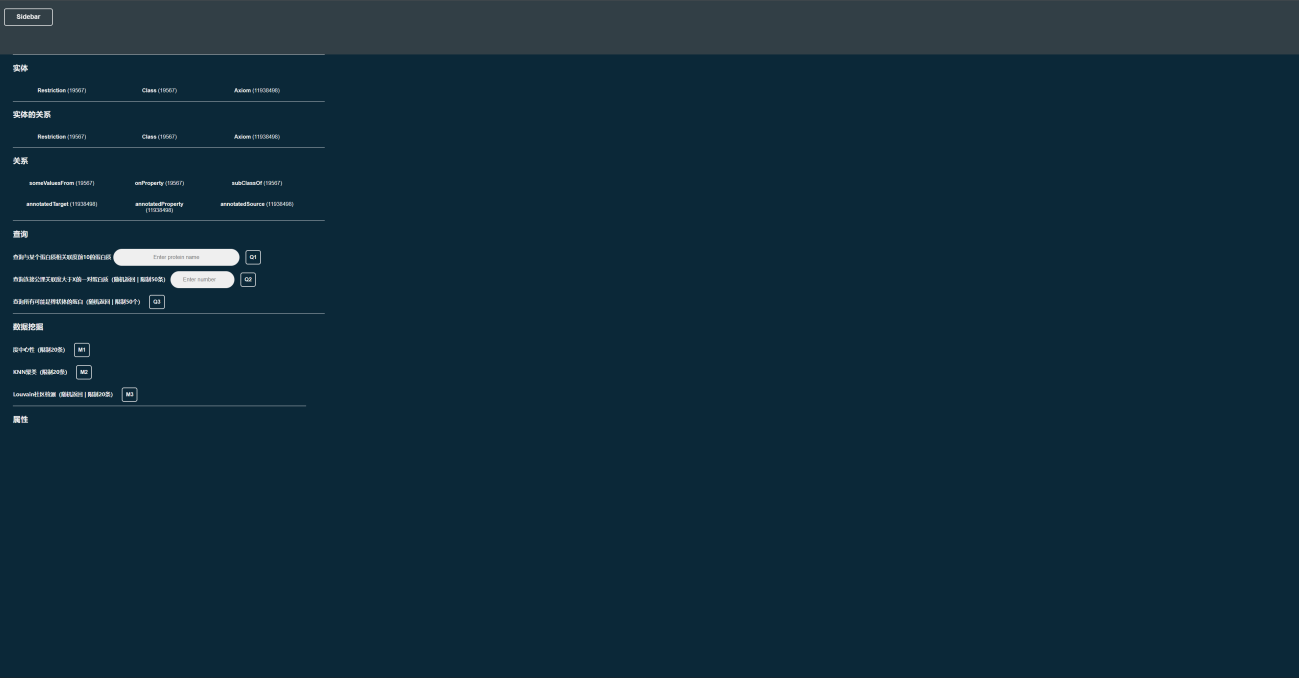
通过html的页面布局：



通过css设置页面样式：



页面展示如下：



左边为查询区域，右边为展示区域。

左边从上到下依次为实体集合、每个实体所拥有的的关系、关系集合、查询方法、数据挖掘方法以及属性。

## 4.3 后端实现

本系统后端基于javascript语言实现，使用axios网络请求库实现前后端通信。

### 4.3.1 axios

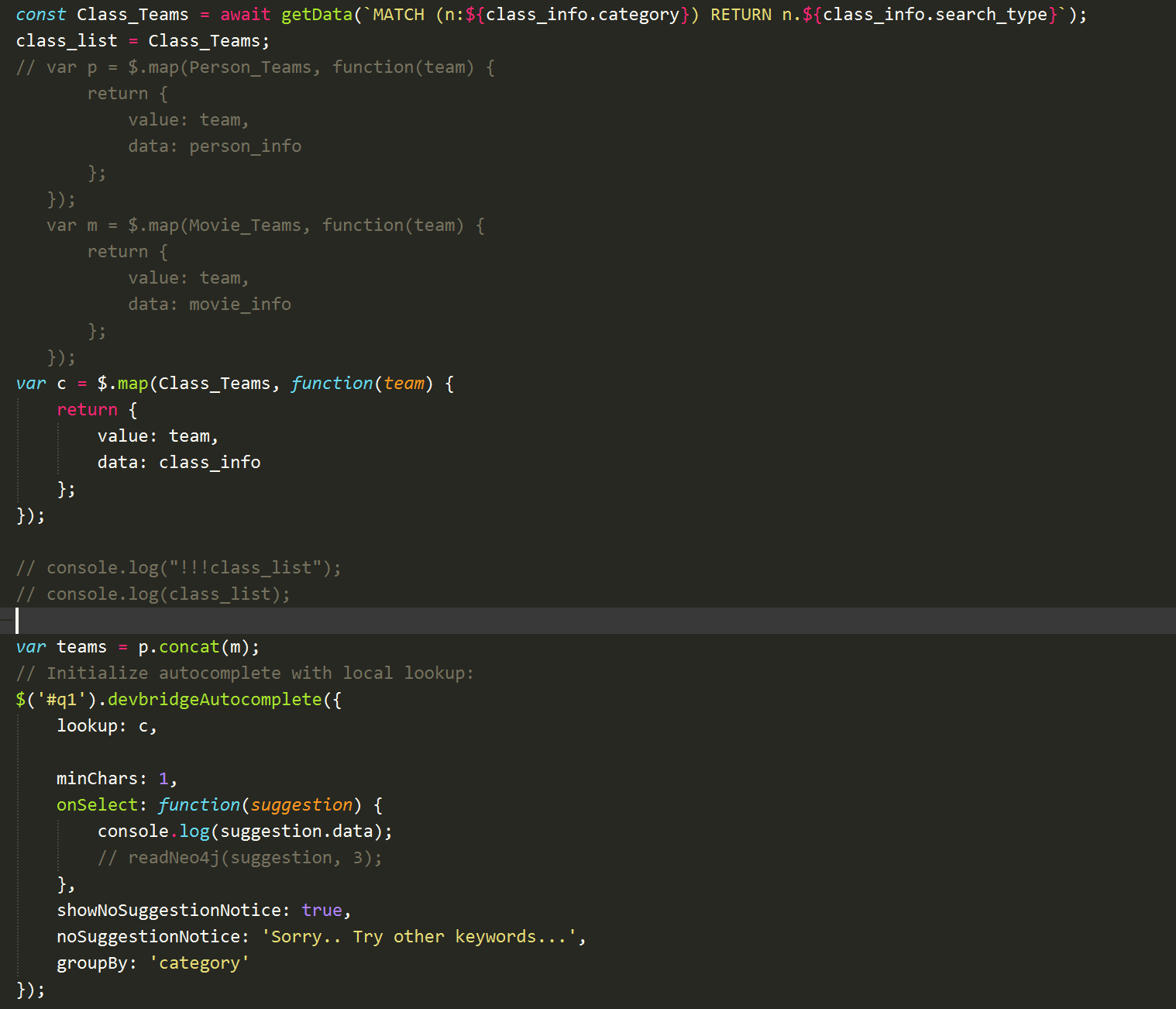
axios是一个基于promise的异步ajax请求库。前端发送get，post，put，delete等请求，与后端交互。axios在web端和node端均可使用，在浏览器端使用XMLHttpRequests对象发送ajax请求，并接收返回的数据。在本系统中，将cypher查询语言作为axios请求的参数，发送给后端，后端执行相应的cypher语句，并将结果返回前端。

### 4.3.2 页面初始化

（1）向Neo4j发出创建图映射的请求，并创建图嵌入fastRP用于KNN聚类，并请求Neo4j运行Louvain算法



（2）提前获取所有Class的实体集合，用于查询一的模糊搜索和补全

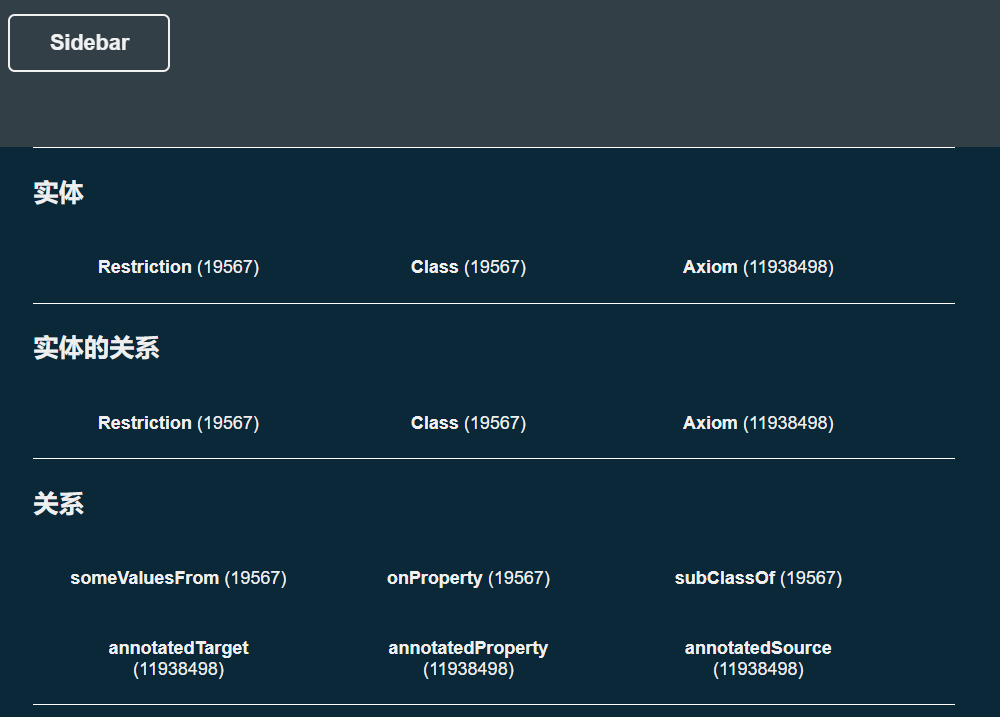


（3）提前获取所有的实体和关系类型，用于前端页面的展示

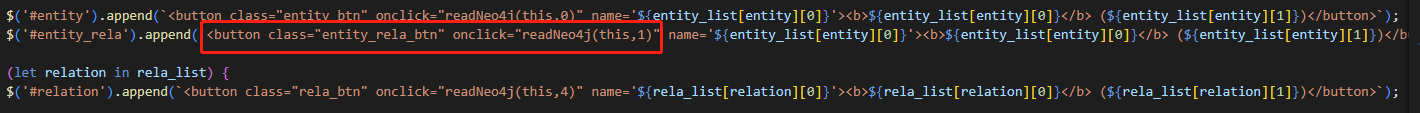


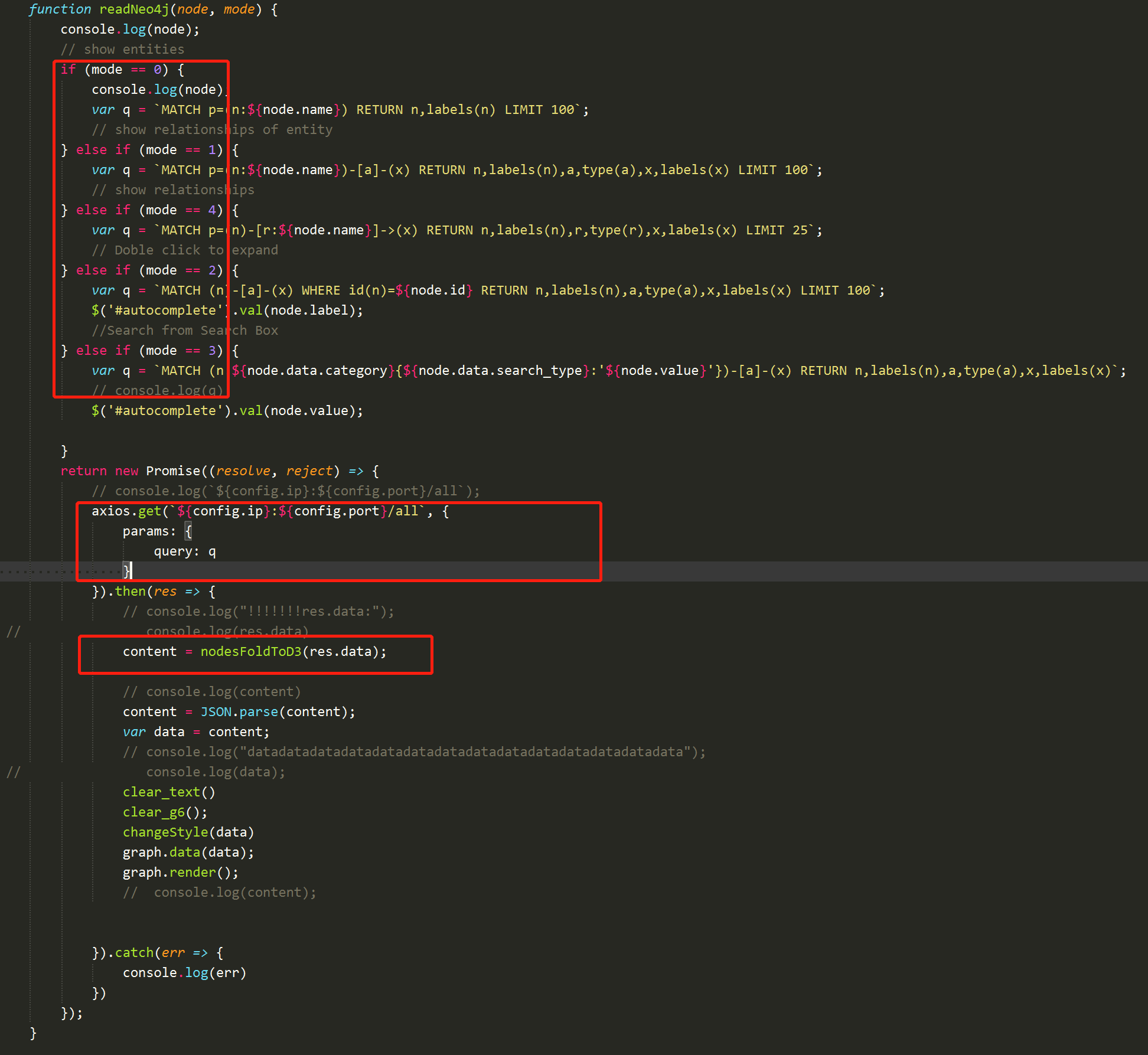
4.3.3 实体与关系的展示和实现

界面如下：



通过点击各个实体和关系的类型，在后端调用readNeo4j()。readNeo4j()根据点击的不同类型来生成不同的cypher查询语句，并通过nodesFoldToD3()将neo4j传回的数据转换成G6可视化库可以使用的格式。







Neo4j返回结果的格式如下：

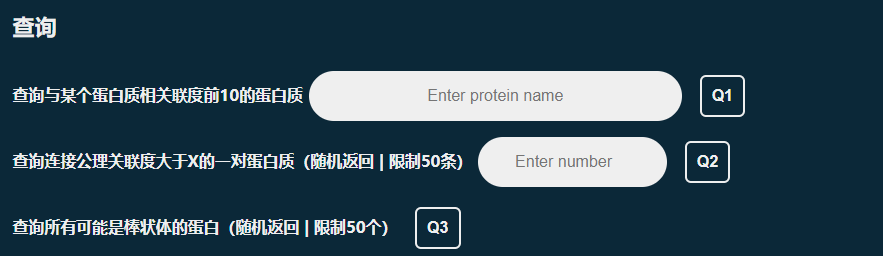


后端解析、处理格式，以result = {'columns': columns, 'data': data}的格式使用数据

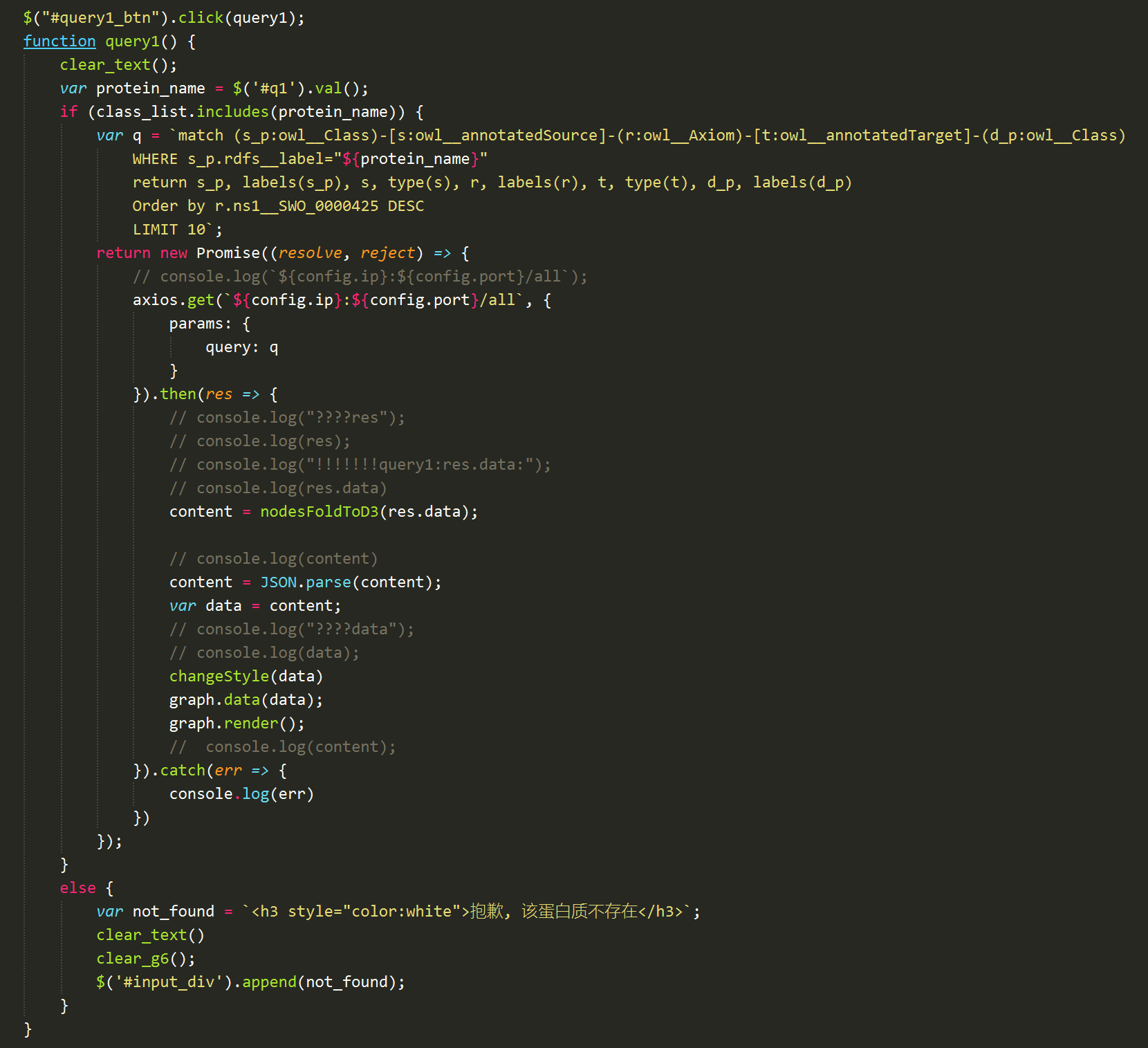


4.3.4 查询实现

用户通过点击相应的查询类型按钮

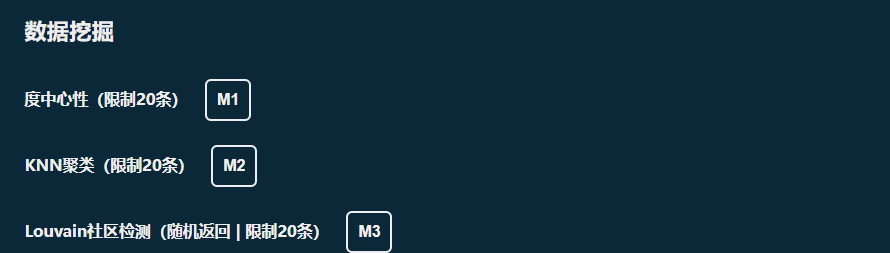


触发后端的查询函数，向Neo4j发出cypher查询语句，并根据4.3.3的过程拿到数据、转化数据、展示数据



4.3.5 数据挖掘实现

同理，用户通过点击相应的数据挖掘类型按钮



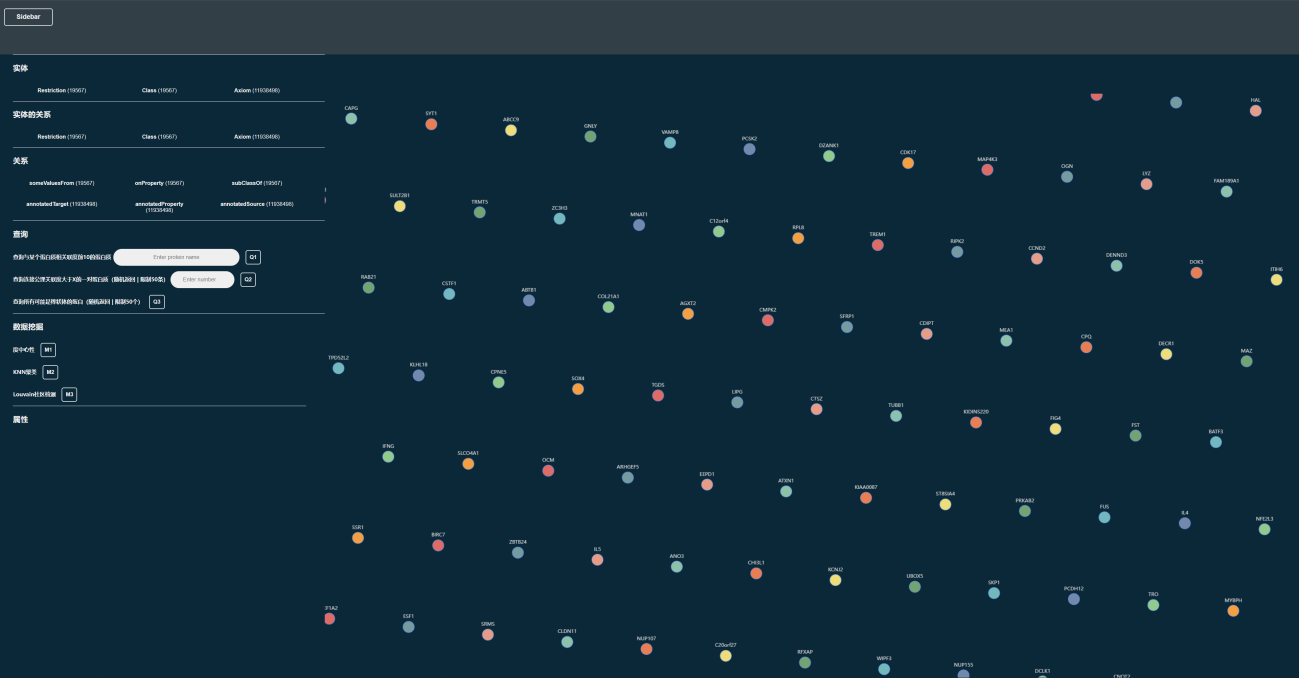
触发相应的数据挖掘函数，从而向neo4j发出请求，拿到数据。此处以表格的形式在前端展现。



# 系统测试

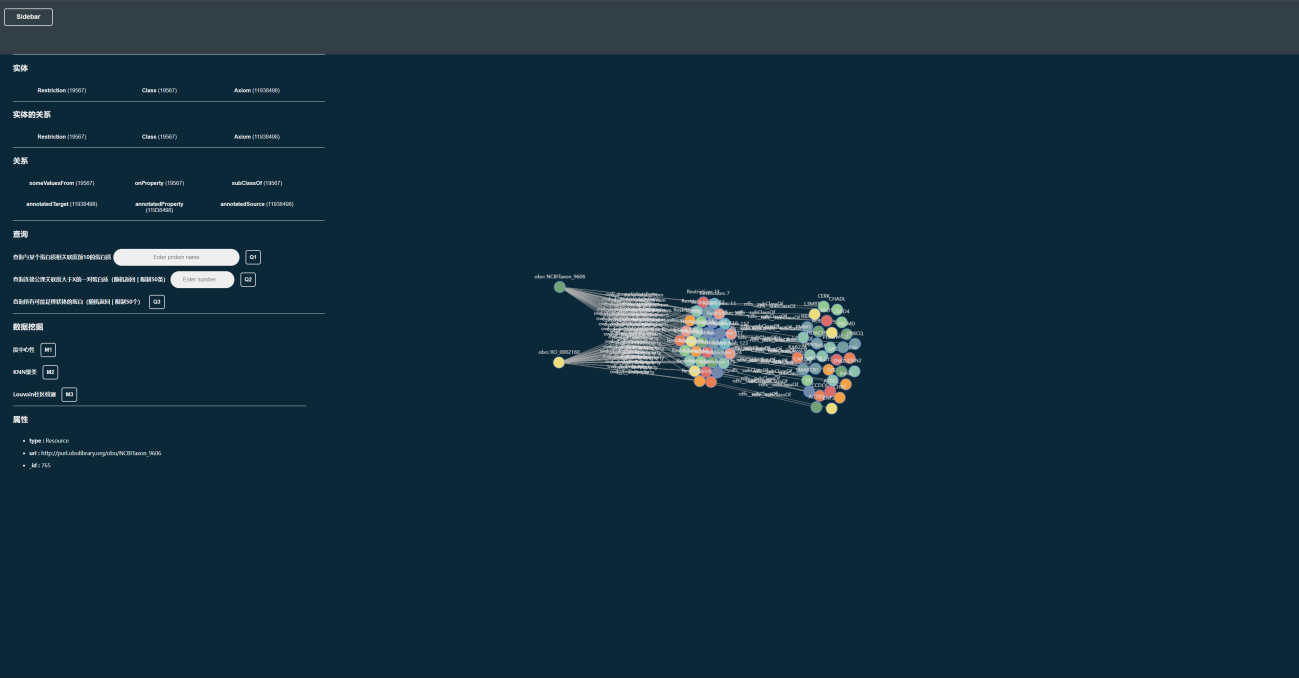
## 5.1 实体

点击每个实体集合，右边会展示一定数目的相应的实体。比如点击Class实体，右边会展示所有的蛋白质，如下图所示：



## 5.2 实体的关系

点击每个实体关系，右边会展示一定数目的与该实体相关的属性。比如点击Restriction，展示情况如下：



## 5.3 关系

点击每个关系集合，右边会展示一定数目的与该关系有关的两个实体。比如点击annotatedTarget，展示情况如下：

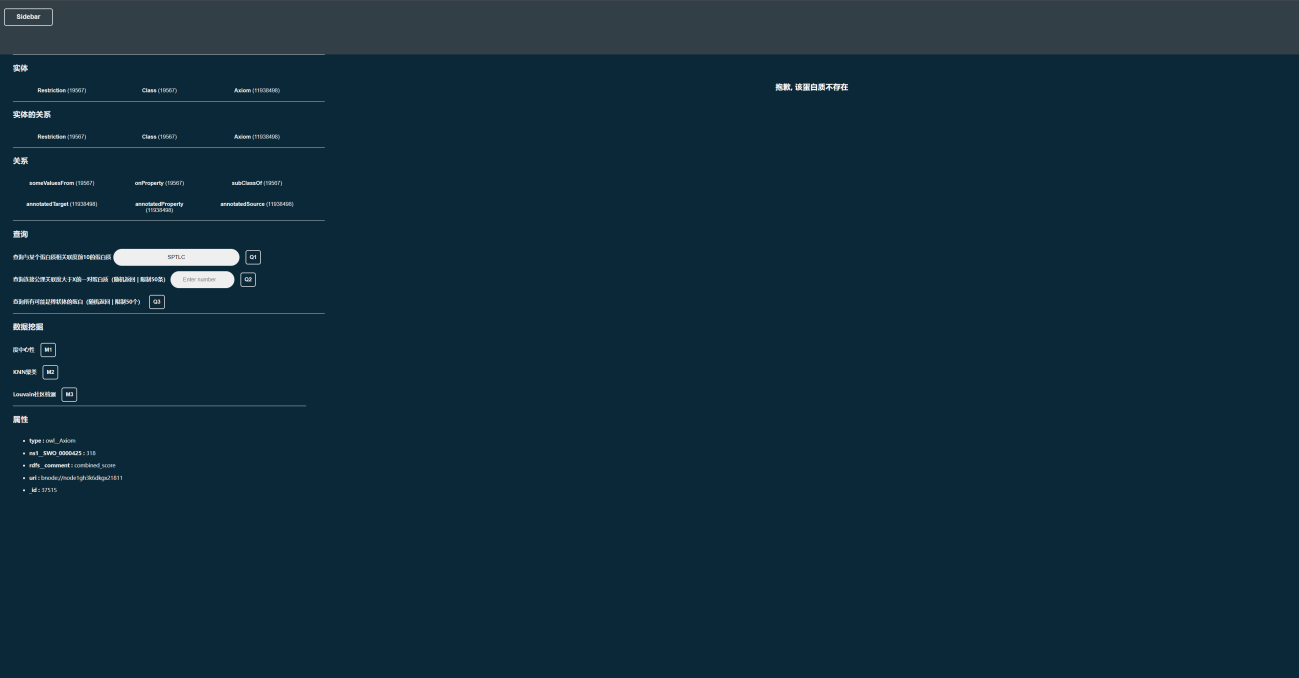


## 5.4 查询

查询部分一共分成三个查询

（1）查询一：查询与某个蛋白质相关联度前10的蛋白质

该查询允许用户在查询框内输入蛋白质名称，并支持模糊搜索，如果查询失败会在右侧显示相应失败信息：

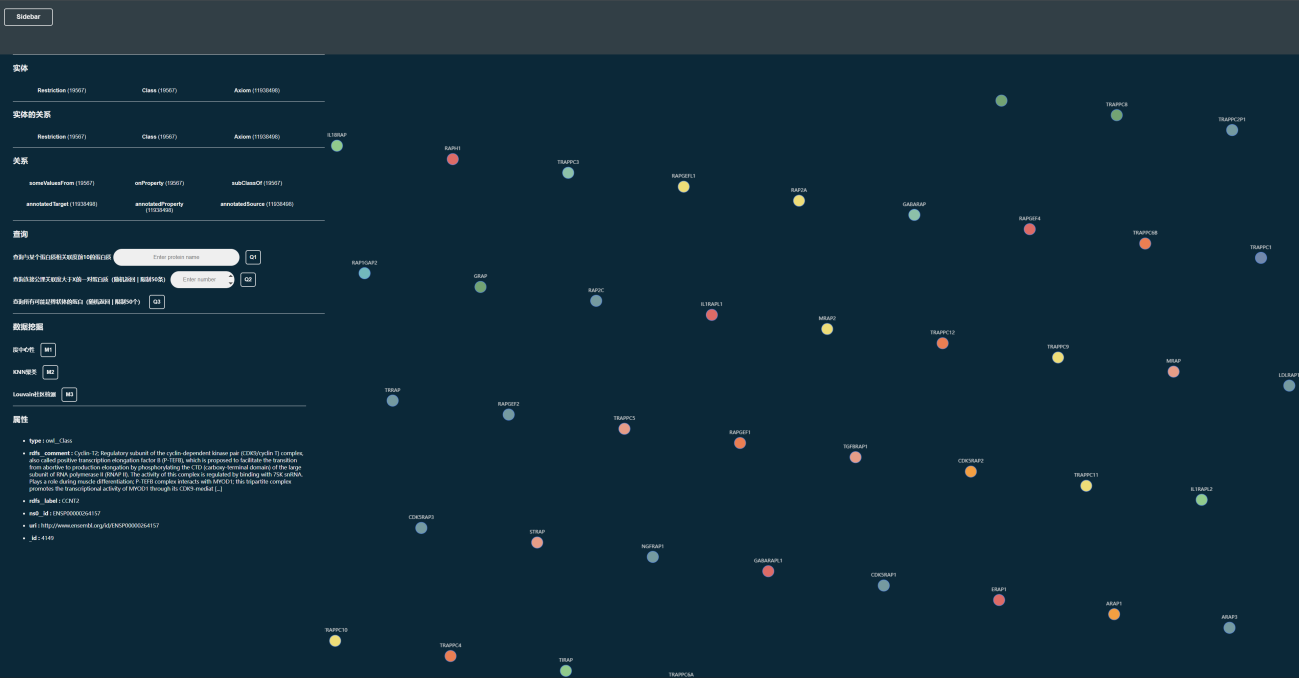


（2）查询二：查询连接公理关联度大于X的一对蛋白质（限制并随机返回50条数据）



并且为输入的数字做了限制，不允许输入的数小于0或者大于最大值

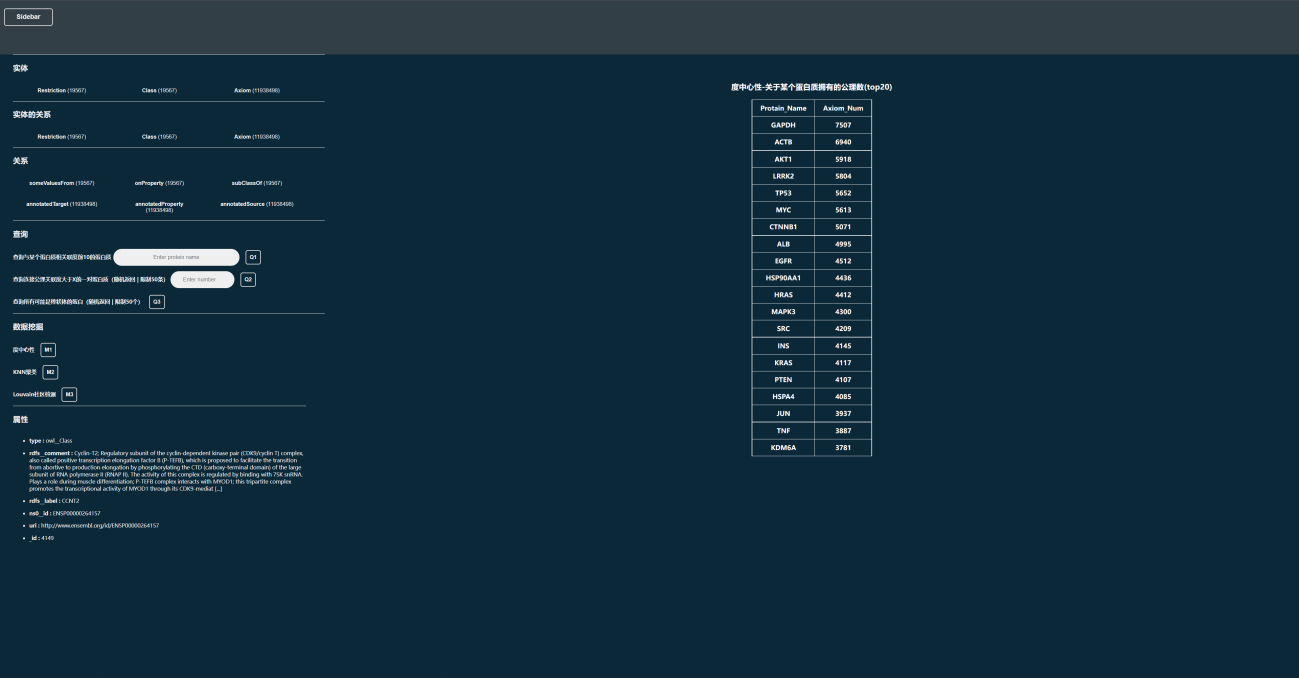
（3）查询三：查询所有可能是棒状体的蛋白质：



## 5.5 数据挖掘

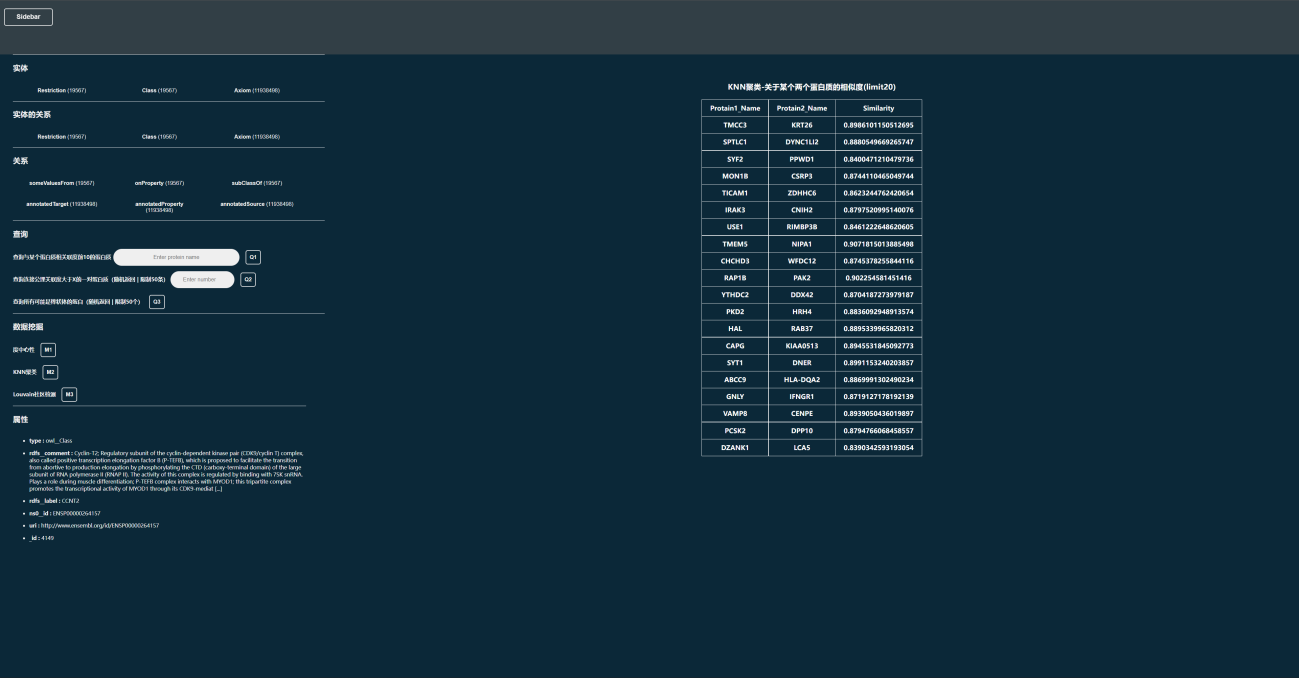
（1）数据挖掘算法一：度中心性

以表格的形式展示了度中心性最高的前20种蛋白质以及其所连接的公理数



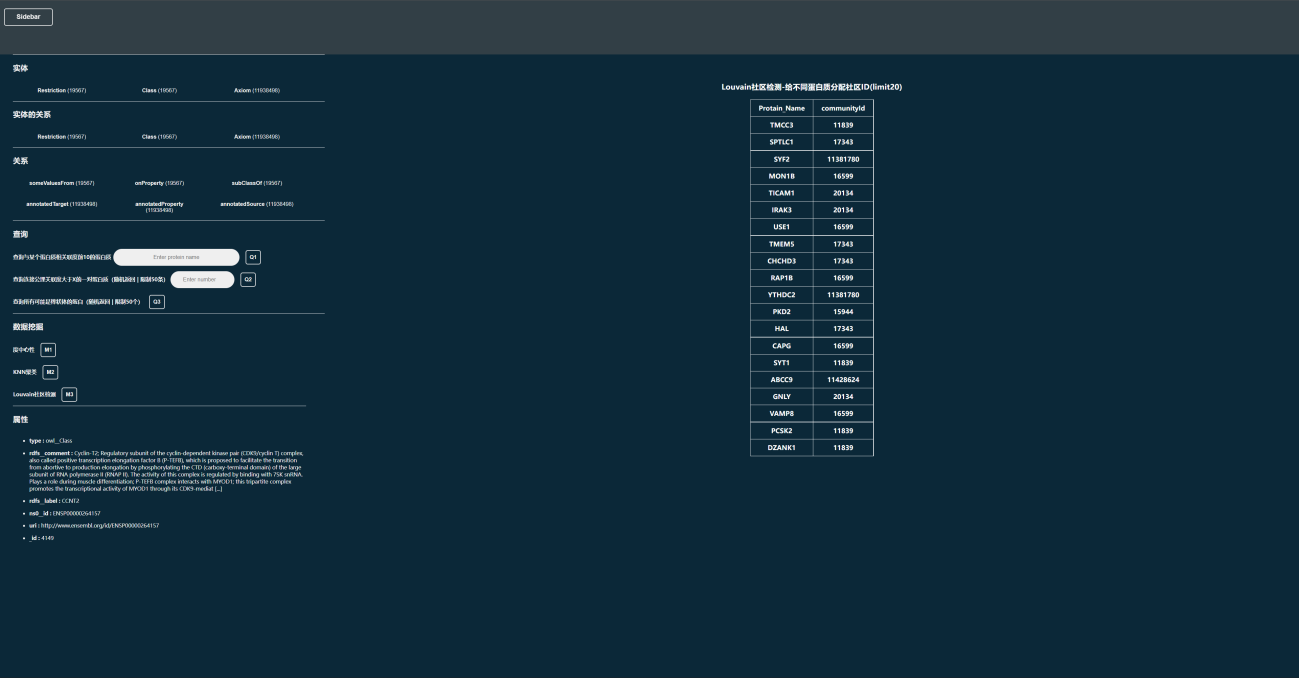
（2）数据挖掘算法二：KNN聚类

以表格的形式展示了20对蛋白质的相似度



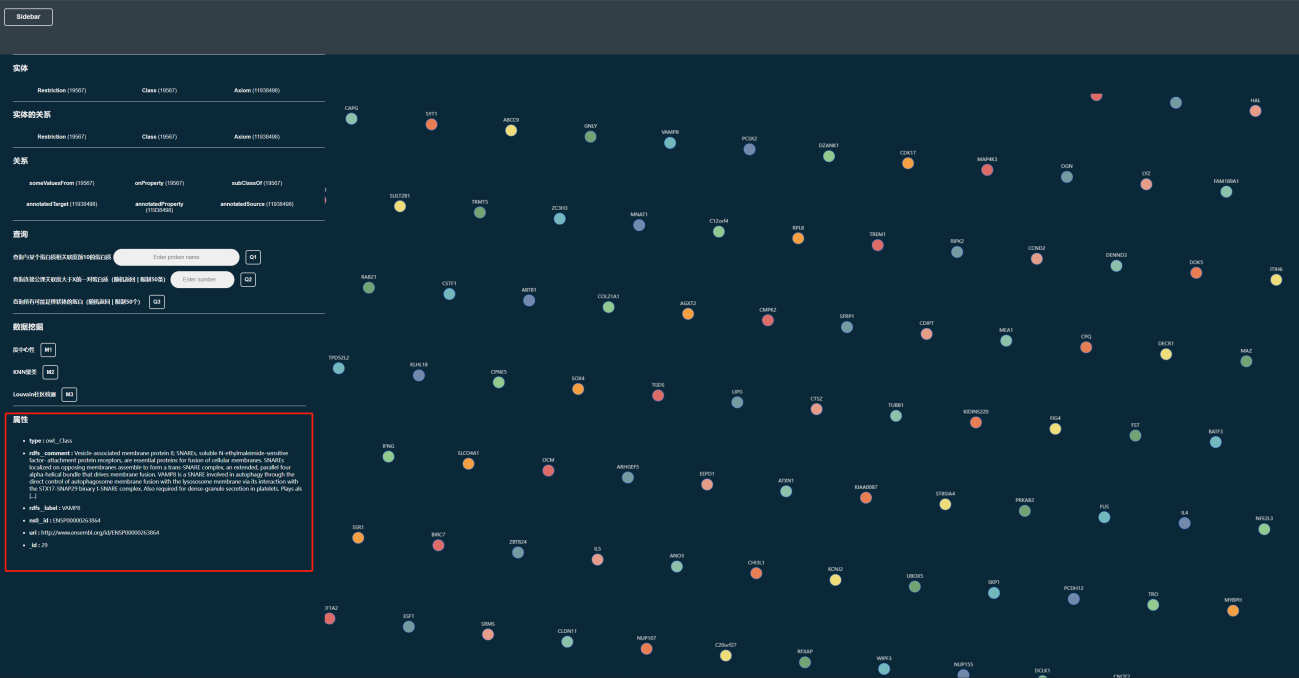
（3）数据挖掘算法三：Louvain社区检测

以表格的形式随机展示了20种蛋白质所属的社区ID



## 5.6 属性

当鼠标移动到任何一个实体上时，属性栏会显示所有与该实体相关的信息



# 项目总结

本项目基于Neo4j数据库，使用了基于JavaScript的Node.js的前台用户界面技术，使用了基于G6的可视化组件工具，并基于Neo4j Graph Data Science组件，实现了一个具有后台知识图谱数据库和前台用户界面的COVID-19中人类蛋白质相互影响关系的知识图谱管理信息系统，并支持有实际意义的、包括“查询用户指定蛋白质所有相关联的其他蛋白质”的知识图谱查询，支持知识图谱的可视化，支持类如度中心性、KNN聚类算法等知识图谱数据挖掘算法，并且具有一定的鲁棒性、用户友好性。

通过本系统所提供的功能，能够快速获取人类蛋白质之间的相互作用以及作用的程度，在某些场景下可以为生物医药人员提供一定的指导作用。

本系统也具有一定的拓展性，其内核所支持的功能远不止前端所展现出来的。所以如果有需求，仍然可以通过在后端进行简单的修改，去实现自己想要的功能。