

**2021年****同济大学软件学院**

**AI与商务智能课程项目报告**

基于路透社公开金融数据集的数据研究项目

**小组成员**

1854116朱明志

1851008刘文朔

1853434谭梓煊

撰写日期：2021年6月20日

目录

[基于路透社公开金融数据集的数据研究项目 1](#_Toc75120133)

[1 项目架构分析 3](#_Toc75120134)

[2 数据引入 3](#_Toc75120135)

[2.1 源数据集简介 3](#_Toc75120136)

[2.2 本项目使用数据 3](#_Toc75120137)

[2.3 数据处理及导入 3](#_Toc75120138)

[3 数据应用 4](#_Toc75120139)

[3.1 数据库介绍 4](#_Toc75120140)

[3.1.1 Neo4j 4](#_Toc75120141)

[3.1.2 Redis 4](#_Toc75120142)

[3.2 功能实现 4](#_Toc75120143)

[3.2.1 单节点及关系查询 4](#_Toc75120144)

[3.2.2 按特定属性模糊查询 5](#_Toc75120145)

[3.2.3 自定义沿图扩展查询​ 5](#_Toc75120146)

[3.2.4 两节点最短路径查询 5](#_Toc75120147)

[4 自定义评分系统 5](#_Toc75120148)

[4.1 企业评分体系 5](#_Toc75120149)

[4.1.1 企业结点数据重述 5](#_Toc75120150)

[4.1.2 雅虎财经数据引入 6](#_Toc75120151)

[4.1.3 企业评分规则建立 7](#_Toc75120152)

[4.1.4 企业评分规则嵌入 8](#_Toc75120153)

[4.2 个人评分体系 8](#_Toc75120154)

[4.2.1 个人结点数据重述 8](#_Toc75120155)

[4.2.2 企业评分数据引入 9](#_Toc75120156)

[4.2.3 个人评分规则建立 9](#_Toc75120157)

[4.2.4 个人评分规则嵌入 10](#_Toc75120158)

[5 项目思考 10](#_Toc75120159)

[5.1 实时变化 10](#_Toc75120160)

[5.2 性能提升 10](#_Toc75120161)

[6 团队成员 10](#_Toc75120162)

[7 参考文献 11](#_Toc75120163)

# 项目架构分析

本项目是一个知识图谱展示网站，查询结果以图的方式向用户展示。

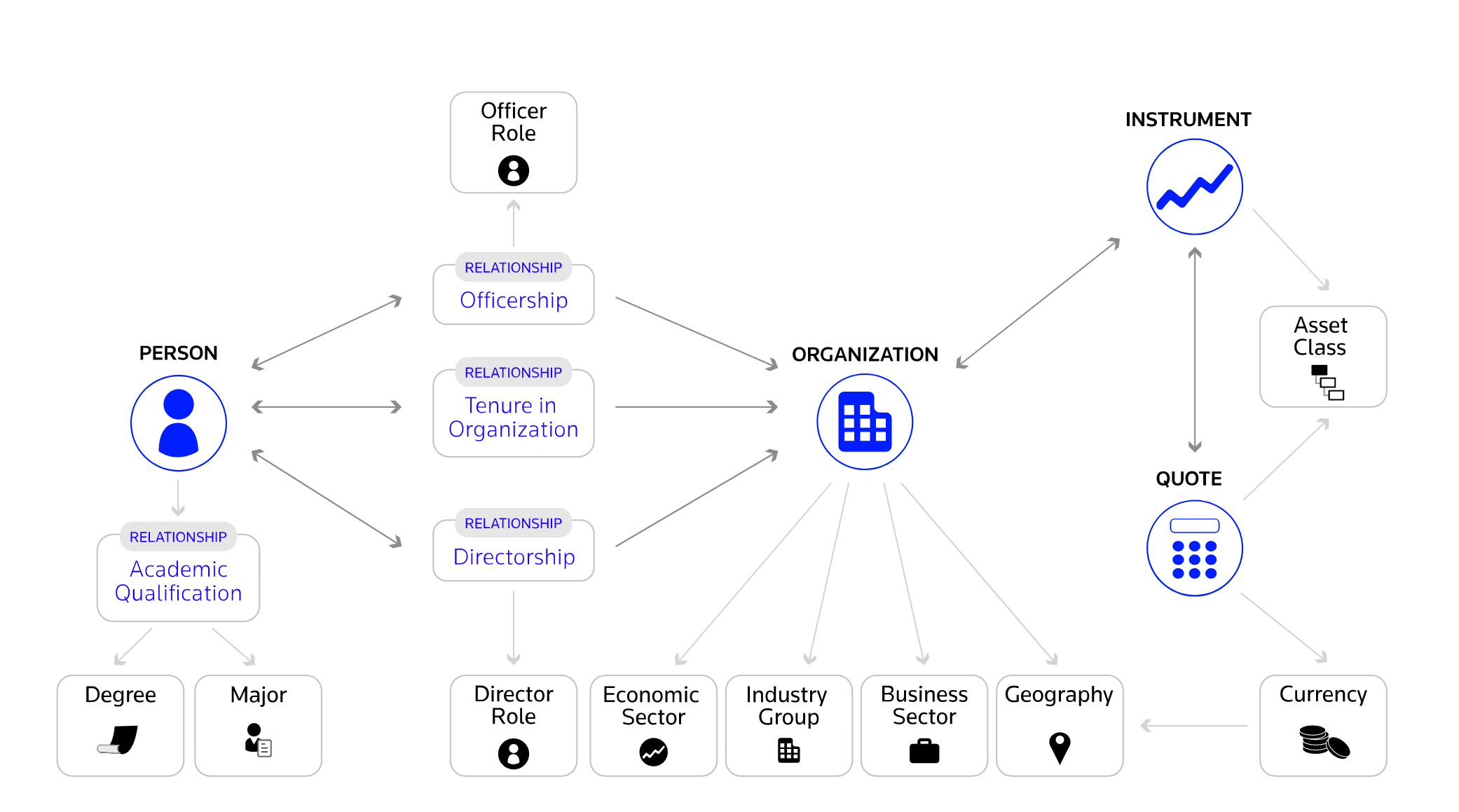
前端使用vue+D3框架实现前端基本逻辑、样式设定以及数据可视化。

后端采用Python的flask框架，实现分发逻辑、图算法等功能。

数据库方面使用Neo4j存储节点及关系数据；使用Redis作为检索结果的缓存。

# 数据引入

## 源数据集简介



数据在可以共享或打开时具有更大的潜在值。它可以被企业内部或外部的许多不同的利益相关者和合作伙伴用于各种应用程序，以获得新的分析洞察力，并构建新的产品和服务。为了有效地使用数据，了解数据如何与现实世界连接是很重要的。这就是为什么Refinitiv将其永久标识符或权限以及相关实体主控和元数据提供给市场的原因。PermID是开放的、永久的和通用的标识符，其中底层属性捕获它们各自表示的标识的上下文。本项目的所有数据均来自路透社的金融数据集，他们的PermID的唯一的。

## 本项目使用数据

## 数据处理及导入

在做数据清理的时候，我们有两种方案：

1.原来使用数据清理导出成csv

优点：清理完数据比较小，可以通过neo4jimport直接导入

缺点：在清洗数据的过程中，会有一定的数据缺失

2.使用neosemantics直接导入

优点：数据比较完整

缺点：数据较大，比较难导入

我们对于源数据进行了一定的分析，发现源数据采用rdf数据框架，其中URI作为语义知识图谱的唯一标示，不仅可以存储数据，还可以存储该数据含义的明确描述，例如：

http://permid.org/ontology/common/hasPermId节点代表了规则

http://www.w3.org/1999/02/22-rdf-syntax-ns#type存储了相关时间信息

原始数据本就为知识图谱类型，因此可以直接导入而无需额外的处理。只是person数据中有一个属性表示该结点是否已经被废弃，查询时若有则我们将该结点直接过滤去掉。

因此我们决定使用第二种方法进行数据导入，保证数据信息最大程度的完整性，为了加快读写速度，增加增删改查的速率，我们选择使用固态硬盘进行数据存储。

# 数据应用

## 数据库介绍

### Neo4j

Neo4j是一种基于图的数据库，以图（节点+关系）的结构存储数据。这种存储数据的方式使得它在处理存在复杂关系的数据时，能够清晰高效地理顺它们之间的关系。

本项目使用Neo4j存储路透社提供的公开金融数据集，其中包含许多金融实体（如阿里巴巴公司）及实体之间的关系。我们使用Neo4j为本项目构建知识图谱模型并存储，且在此基础上提供一些检索服务。

#### 运行配置

运行Neo4j的物理机器内存为16G，Neo4j会根据当前运行状态动态占用所需要的内存。

#### 数据统计

图数据库中共存有29.50GiB图数据，其中：

节点数：24,787,186

关系数：69,212,963

属性数：73,442,896

### Redis

Redis是一个开源的，存储于内存中的键-值对数据结构存储系统，它可以用作数据库、缓存和消息中间件，它支持存储多种类型的数据结构。Redis含有事务（transactions）和不同级别的磁盘持久化（persistence）功能，并通过Redis哨兵（Sentinel）和自动分区（Cluster）提供高可用性（highavailability）。

在本项目中，我们使用Redis存储用户先前查询过的结果，期限为10分钟。这样可以节约高频内容检索花费时间与短时间内重复查询时的检索时间。

## 功能实现

### 单节点及关系查询

单节点查询是Neo4j中最基本的查询，用于获取一个实体及相关实体的属性，语句如下：

MATCHp=(n:${label})-[]-()wheren.hasPermId=${PermId}returnplimit${limit}

对于节点a，给定四个参数a的PermId、a的类型label、返回的路径数量限制，直接返回其周围的关系和节点组成的结果集。

但是在数据量如此大的库里面，定位到某节点并进行扩展也会变得很慢。于是我们做了以下几种提升性能的方法：

首先，考虑到Neo4j在节点定位上很吃力，为了避免遍历时全文搜索，我们在Person节点和Organization节点的hasPermId属性上创建了索引，大大增快了节点定位的速度；

其次，我们使用Redis对查询记录进行缓存，当短期内执行同样的查询时，会从Redis中摘取记录并返回，查询的速度基本是在ms级别，同时我们设定了记录失效时间10分钟，防止结果过于陈旧。

### 按特定属性模糊查询

在实际应用中，我们经常需要查询具有特定属性的节点。

例如查询名称中包含name的企业：

MATCH(n:Organization)WHEREANY(nameINn.\`organization-name\`WHEREnameCONTAINS"${name}")RETURNnLIMIT250

查询名称中包含"Zhu","mingzhi"的人：

MATCH(n:Person)WHEREn.\`family-name\`="Zhu"ANDn.\`given-name\`CONTAINS"mingzhi"RETURNnLIMIT250

### 自定义沿图扩展查询​

在实际应用中，我们经常需要沿图中的节点查询其邻居节点。

首先进行任意一个查询，查询结果将会被图形化展示。双击图中任意一个节点，该节点的邻居节点将会被添加至图中。

### 两节点最短路径查询

最短路径我们采用了Neo4j内置的shortestPath算法进行查找，其使用快速双向广度优先搜索算法，为避免出现暴力穷举的情况，我们在配置中设置了cypher.forbid\_exhaustive\_shortestpath为true。由于我们的关系上不存在权重这一说，因此我们的最短路径就是指最短跳数。

查询语句如下：

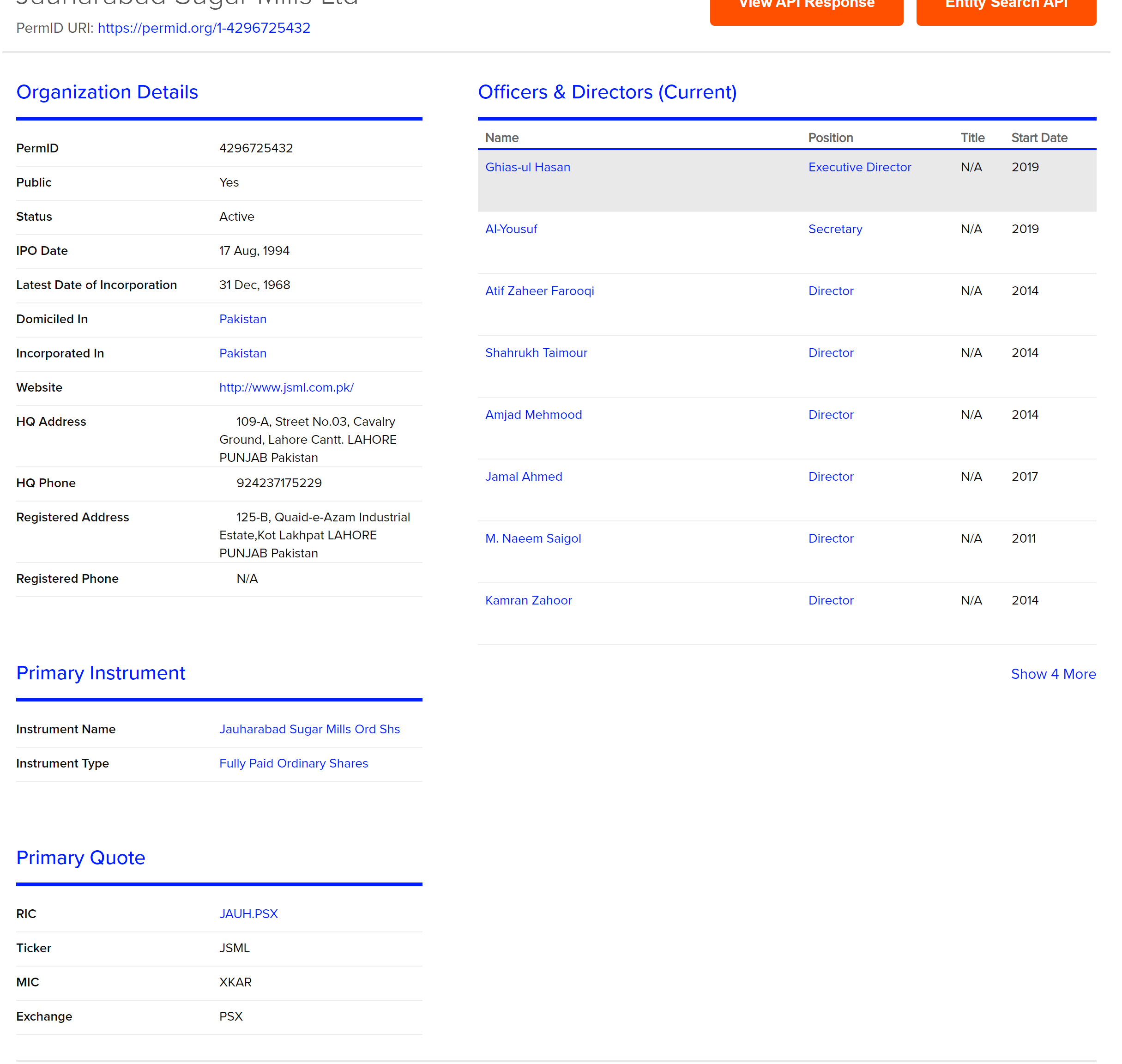
MATCH(source:${sourceType}),(target:${targetType})WHEREhasPermId(source)=${sourceId}ANDhasPermId(target)=${targetId}MATCHp=shortestPath((source)-[\*]-(target))returnp

# 自定义评分系统

## 企业评分体系

### 企业结点数据重述

一个典型的与一个企业结点有关联的数据如下，除了企业节点自身的一些属性信息以外，还有企业结点连接的成员、器械和报价结点信息。项目对一个企业的打分应基于这些数据产生。特别要说明的是，成员结点的评分结果可以依赖于企业结点的评分结果，企业评分的结果也可以依赖于成员结点的评分结果。可见，这两个评分结果必然是互相依赖的。考虑企业数量较少，企业的属性较多，以及企业的额外数据更容易获得。最后选择企业评分作为基线，并以此确定成员评分。



### 雅虎财经数据引入

依据小节的数据综述，项目组发现这些数据集中的原始数据并不能很好的对一个企业评分。如果依据一个企业的位置、企业电话、或者企业ID这些信息给一个企业打分这未免过于强行，而企业的设备是租赁还是全资持有这些指标似乎也不能反反映企业当下的运营状况。对于企业成员这一个参数，上文提及本评分系统是以企业的得分决定个人得分的，因此也不能作为评分依据。

可见这些数据是有限的，不能完全满足评分的需求。但是在primaryquote结点中有一个属性，是这个公司的股票代码，即Ticker字段。这个股票代码是全球唯一的，可以利用这个字段去查询到更多关于这个公司在当下的经营状况，这些是能代表该企业当下的发展水平。

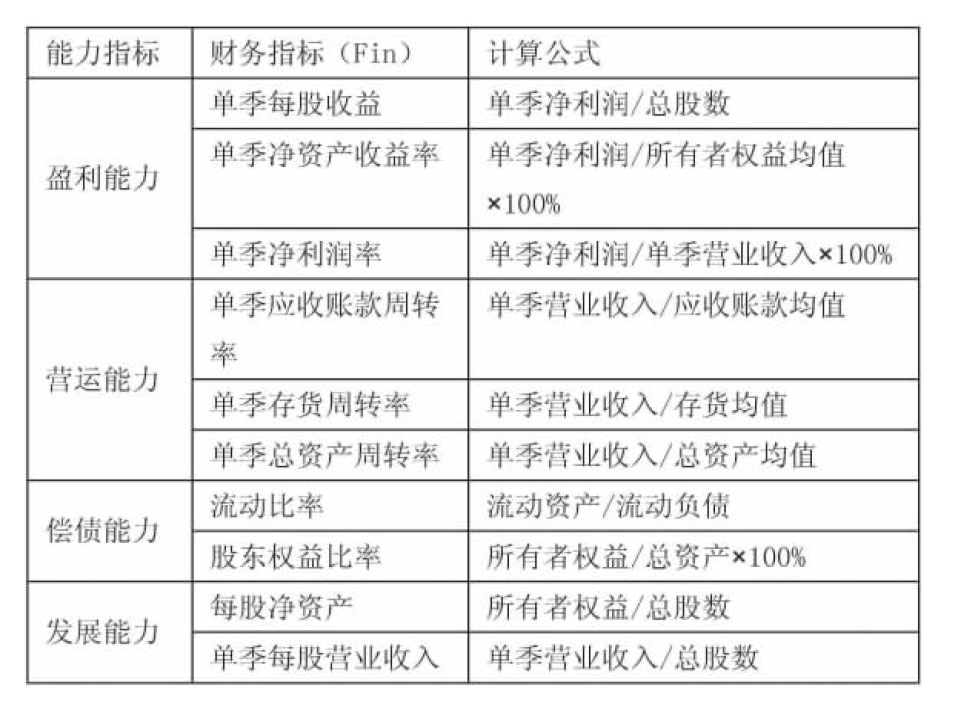
幸运的是雅虎财经提供了这样一个接口，可以依据企业的股票代码查询企业的几乎所有财报数据。[1]引入这些企业最新的财报数据后，企业经营水平变得可以被量化了。雅虎财经提供的数据包括但不限于：企业的股票历史信息、企业的控股结构、企业的股权人、企业分红信息、企业股权划分、企业完整财报、企业现金流等。

### 企业评分规则建立

杨洋，张大方曾提出一种基于上市企业财报的企业评分方法,可用上市企业的一些财报信息对企业进行综合评分[2]。本项目的评分规则便参考了他们研究的成果。首先是热数据的利用，基于论文的描述，与当下间隔较远的企业财经数据对研究企业的发展水平意义不大，故本项目组仅利用雅虎财经API获取了最近1年待评分企业的金融数据，数据为季度数据。首先计算出上市公司近一年所有财务指标；其次将计算出来的各财务指标进行排序，得到最大值和最小值，按照如下公式进行线性插值，可以得到所有上市公司的财务指标赋值。

公式（1）中Xj表示财务指标，计算结果表示该财务指标插值得分。通过线性插值之后，财务指标的取值区间为0到100，为了保证线性度，本文在取最大值和最小值时，剔除了个别极大值和极小值。

本评分标准用到的各项财务指标见下：



根据论文中的研究结果结合本数据集的实际情况，评分规则对企业各项能力指标分配权重如下表所示：

|  |  |
| --- | --- |
| 能力指标 | 权重系数 |
| 盈利能力 | 0.55 |
| 运营能力 | 0.15 |
| 偿债能力 | 0.10 |
| 发展能力 | 0.20 |
| 合计 | 1 |

基于上述分析和沃尔评分法的原理，本项目构建的综合评分模型见公式如下：

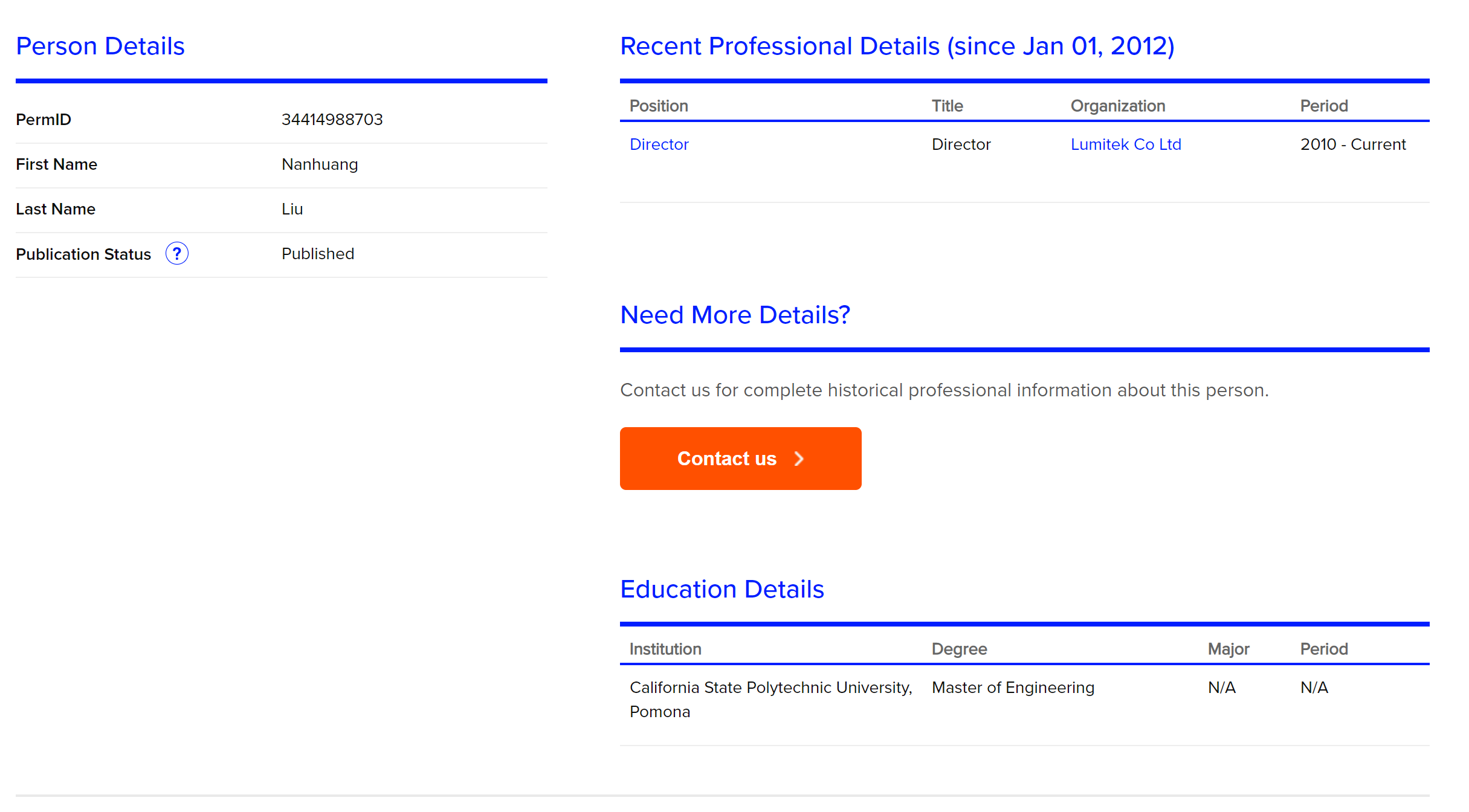
### 企业评分规则嵌入

对于企业评分的结果，项目选择在查询时候的动态生成评分结果。本项目将评分算法封装为一个函数，在查询指令发出后，调用该函数计算该企业的评分，并将这个评分作为企业结点的一个属性保存入数据库，这样保证每次查询时调用的企业财经数据都是最新的，但也牺牲了一定的系统响应速度。

## 个人评分体系

### 个人结点数据重述

一个典型的与一个个人结点有关联的数据如下，除了个人节点自身的一些属性信息以外，还有个人结点连接的工作公司和教育经历结点信息。上文提到本评分系统以企业结点的得分为基线，这一个思想也被运用到了个人评分的构建中。•观察大量的个人结点数据后，我们观察到个人结点基本上可以被分为四类：第一类人只有个人基本信息列；第二类人有第一类人的个人基础信息，也有个人在企业的任职经历；第三类人有第二类人的所有特点，还额外附上了个人的教育经历；第四类人有第三类人除企业任职经历以外的所有属性项。



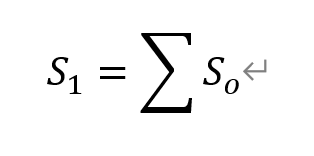
### 企业评分数据引入

由上文的叙述可知，本系统的评分评分基线是企业结点的评分。因此对于个人评分，系统引入了上步生成的企业评分数据。如果这个企业结点还没有评分数据但这个企业在待评分个人的就职经历中，就调用企业评分函数，先生成企业的评分数据。

### 个人评分规则建立

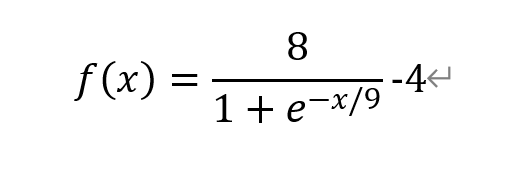
第一类个人结点的基本信息基本只有名字、性别和信息公开状态，用这些信息对个人评分几乎没有贡献。同企业的原因，不能用一个人的名字、性别来衡量他们水平，因此对这些人的评分只能放弃评分，设置为缺失值。对于第二类人，可以用他们的工作经历作为评分的一部分，对于第三类人，可以用他们的教育经历给评分加分。

一个简单的想法是，就职过的公司越厉害，这个人的水平相对就越高。因此可以用一个sum函数对一个就职过的企业得分进行累和，这样得到一个初始分S1，公式见下。



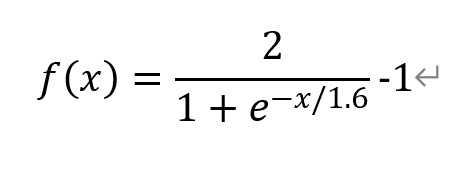
当然，不是就职的企业越多就能得到无限多的分，而且累积得分应具有边界效应，即基础得分S1转换得到最终得分S2的转换函数F(x)的导数，应随x的增大而不断减小，直至为0，当S1无限大的时候S2应收敛于一个定值。F‘（x)在x较小的时候取值应该较大，这也符合一两家大企业的就职经历，或者很多家不错的企业就职经历，对个人成长能力的成长有明显增益。而太多的工作经历对个人的成长反而影响有限，即就职经历这个指标是一个边际效用显著的评判依据。

本评分规则采用Sigmod函数作为前后得分的转换函数，并通过设置平移等比例放大等参数使转换函数收敛在4。Sigmod函数多被用于神经网络的激活函数，因为它当x较小时斜率较大，x取无穷时函数收敛于1，完美符合本评分标准转换函数的要求，本文使用的Sigmod函数式如下。



最后，对于有教育经历的第三类人，教育经历虽然不是对一个人工作能力或者工作成就的完全决定性因素，但是起到一定的影响是必然的。因此教育经历这项数据起到加分项的作用，有就加分，最多加1分。因此，个人评分上线是5分，取值范围是[0,5)。

对于教育经历，每有一条经历，如果是博士学位则+3，如果是硕士学位+2，其他+1，计算一个累积得分S3，同理用一个sigmod函数进行转换，转换得到最后的得分S4，这个转换函数的公式如下：



### 个人评分规则嵌入

同理企业评分嵌入，每次查询到一个人时，会利用实时企业评分计算他的个人得分，并返回给前端和存入数据库。

# 项目思考

## 实时变化

Q：在真实世界中，企业/个人的信息会频繁的发生变化，在你的解决方案中如何应对这种实时的变化？如何支持实时数据更新、查询结果的更新和评分系统的更新？

A：可以在查询时采用读写分离的存储架构。采用一个写库与多个读库。在数据更新时先写入写库，不影响读库的查询，再待读库闲时再同步数据，就可以实现实时数据与查询结果的更新。考虑到企业数据的庞大，略微的同步延迟是可以接受的。同时我们也可以采用将数据存储于关系型数据库（如MySQL）中，通过外键链接到neo4j中的混合存储方式，以提升数据更新的效率。

关于评分系统的更新，我们的评分系统使用Python脚本写成，每次查询分数时都会调用一次雅虎财经的API去查询一下当下的数据保证评分系统的更新。但是这样操作的缺点是牺牲的查询速度，虽然每次计算的评分都是最新的数据web数据生成的，但是web请求的速度是得到查询结果的决速步。考虑到系统使用的是季度数据，后期非原型系统可以考虑每三个月维护一次数据库，在维护数据库的时候统一生成评分，每次查询的时候不用再实时计算用户得分，提高了查询效率。但是这样也存在问题，数据库的规模实在太大，如果对每个企业结点都更新一次评分，这个数据库维护的时间可能很长，用户可能很长时间都不能使用这个系统。所以可以选择用原型系统先上线试运行一段时间，记录哪些结点是热结点，经常被用户查询得分。之后维护数据库的时候只维护这些热结点，冷结点被查询时，就现场计算这些结点的评分。

## 性能提升

Q：当考虑所有数据的时候，如何提高查询和评分计算的性能？

A：采用固态硬盘与高频率大容量内存存储数据，以提升数据读取速度。同时可以在数据库上建立索引，并使用查询约束规则以优化性能，此外可以使用缓存、预热、算法优化、后期处理等方式进行性能优化。

关于图数据的查询，在目前的算法中，我们使用双向快速BFS进行两节点关系查询。但实际工作中，若某结点的相邻接点过多，则cache、内存的存储压力会非常大，因此会出现查询效率受限于内存性能且速度较慢的结果，若内存量不够的话甚至会溢出乃至直接宕机。因此可以使用networkx复杂网络建模工具进行改进。搜索子图时将两个节点的子图保存，用networkx进行分析后返回结果，以优化两节点多跳关系查询的效率。

关于评分系统的查询，将计算完成的评分存储于缓存（如Redis）中以提升查询时的查询速度。在更新评分系统后，即立刻开始更新缓存中的内容以保证同步性。

# 团队成员

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名 | 学号 | 成绩比例 | 项目角色 |
| 朱明志 | 1854116 | 0.33 | 组长 |
| 刘文朔 | 1851008 | 0.33 | 组员 |
| 谭梓煊 | 1853434 | 0.33 | 组员 |

# 参考文献

[1]Yahoo!Financemarketdatadownloader,[https://github.com/ranaroussi/yfinance](https://github.com/ranaroussi/yfinance.)

[2]杨洋,张大方.基于改进的沃尔评分法综合评分与股价的相关性分析[J].中国乡镇企业会计,2021(02):45-48.