# 1. 知识图谱增删改查功能的实现

**1.1 知识图谱增删改查功能概述**

在实现了运用excel表格对于企业公司各类别数据进行导入后，如果我们需要对于知识图谱的每个节点进行一些主观的细小的调整，就需要引入增删改查功能对neo4j数据库中的内容进行即时修改。这里我们使用Python脚本+flask框架控制数据库进行对应的修改。这里我们选择调用py2neo的Python库联动neo4j进行间接控制，而没有选择使用Python脚本书写cypher语言对库进行直接操作。这样我们整个代码会有更好的封装性以及交互性。

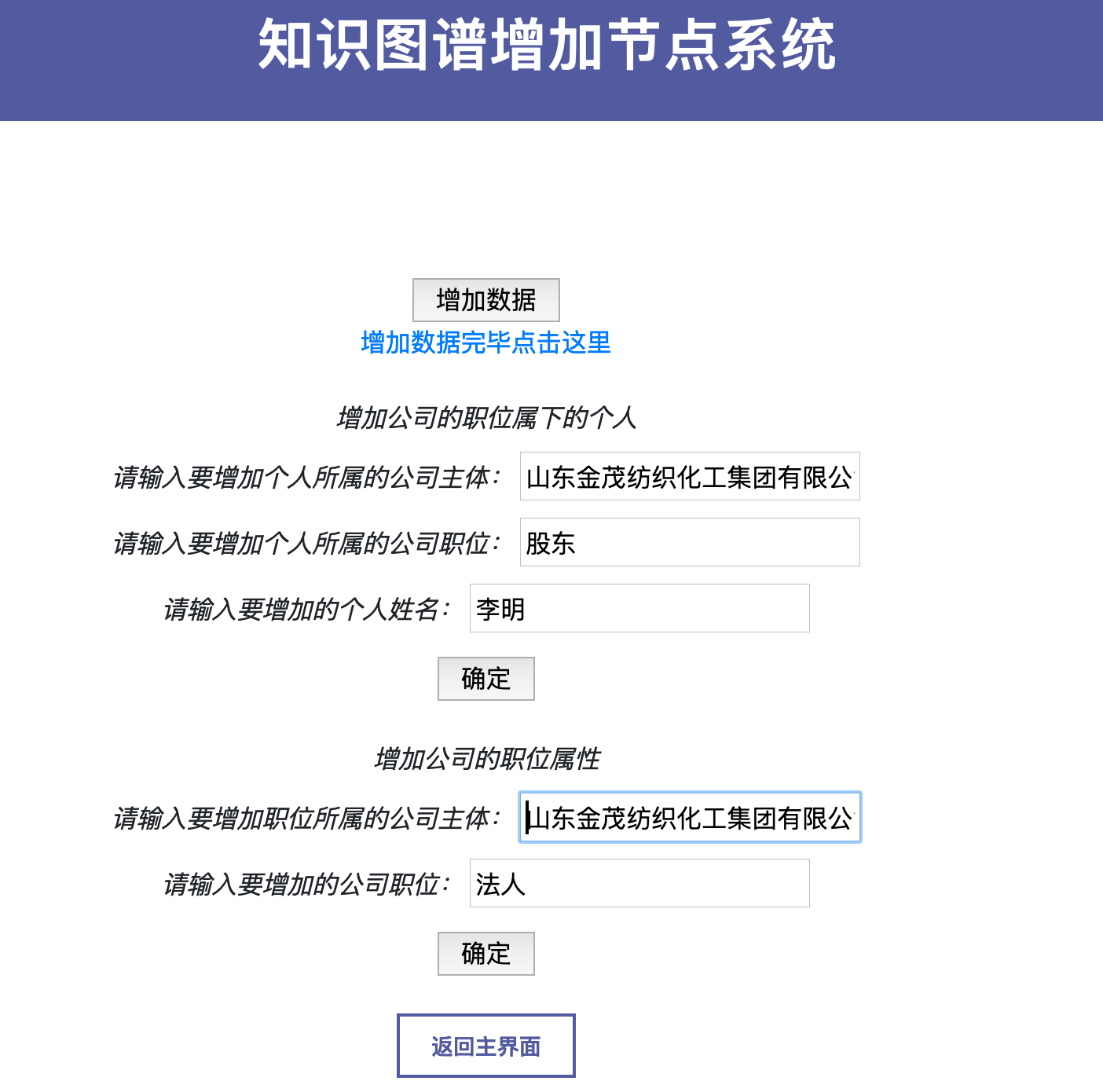
首先Py2neo是一个客户端库和工具包，可从Python应用程序内部和命令行使用Neo4j。 该库支持Bolt和HTTP，并提供高级API，OGM，管理工具，交互式控制台，用于Pygments的Cypher词法分析器以及许多其他功能。 与以前的版本不同，Py2neo v5不再需要启用HTTP的服务器，并且可以完全通过Bolt运行。与neo4j官方给出的Python驱动程序相比较，Py2neo提供了更高级别的API和OGM，但是官方驱动程序提供了与集群一起使用的机制，例如自动重试。我们在这次实现增删改查的功能中主要运用了data, database, matching, ogm组件。

在对于具体的功能实现介绍之前，我们先简单阐述一下使用neo4j数据库对于知识图谱进行构建的框架，框架下除了几个根据行业划分的大型主体节点之外，还有公司主体节点，以及对应公司下属二级职位节点（董监高），三级节点则用于存储对应的人物姓名，每个相关所属的节点之间都有关系相连接，以保证在具体使用Py2neo对于库进行操作的过程中可以成功地检索，在增删改查的功能的实现过程中主要是对于我们所谓的二级职位节点和三级姓名节点进行增删改查操作。

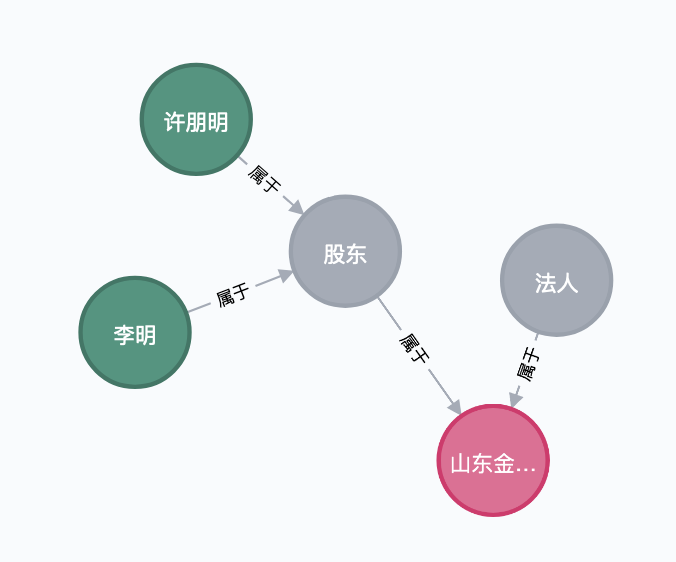
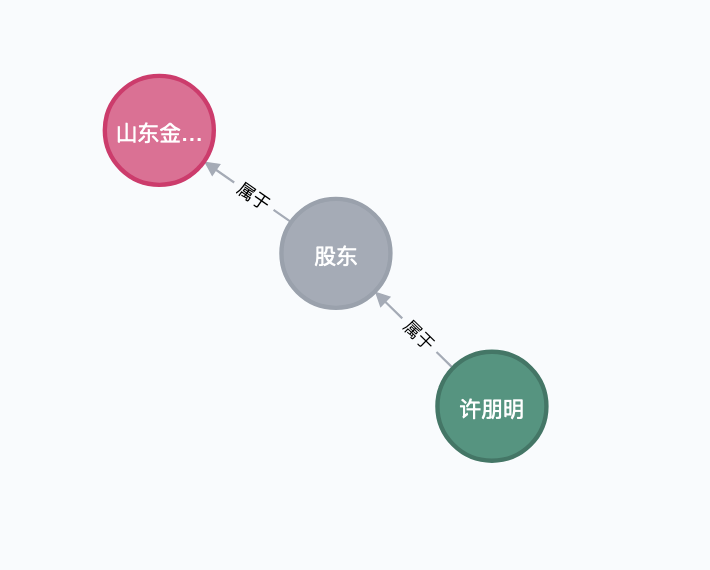
**1.2 知识图谱增删改查功能具体实现介绍**

**1.2.1 增加节点系统框架和实现**

对于增加节点这个功能我们根据实际的要求划分出了两个不同的板块，分别是增加二级职位节点以及增加三级姓名节点。前者是根据用户的需求对于企业下属的职位部分进行补充，在实现的过程中先matching用户在前端交互得到公司节点，py2neo.matching模块提供了根据某些条件匹配节点和关系的功能。对于每种实体类型，提供了Matcher类和Match类。Matcher可用于执行基本选择，返回可以评估或进一步完善的Match。NodeMatcher可用于查找满足一组特定条件的节点。通常，可以通过特定标签和属性键值对来标识单个节点。与此同时，允许任何数量的标签和Cypher WHERE子句支持的任何条件。运用graph.create()的方式创建新二级节点并建立关系。其中这里使用到的graph是Py2neo的一个类操作，py2neo.database软件包包含与Neo4j服务器交互所需的类和函数。 其中最重要的是Graph类，它代表Neo4j图形数据库实例，并提供对大部分最常用py2neo API的访问。通过Bolt或HTTP进行整个Neo4j图形数据库安装的访问器。 在py2neo对象层次结构中，数据库包含一个其中发生了大多数活动的图。 目前，Neo4j每个数据库仅支持一个图形。后者是对于企业某职位下的三级姓名节点进行补充。在实现的过程中先matching用户在前端交互得到公司节点以及下属的职位二级节点，运用graph.create()的方式创建新三级节点并建立关系。下图是演示的界面与效果。



图一：增加节点系统页面展示



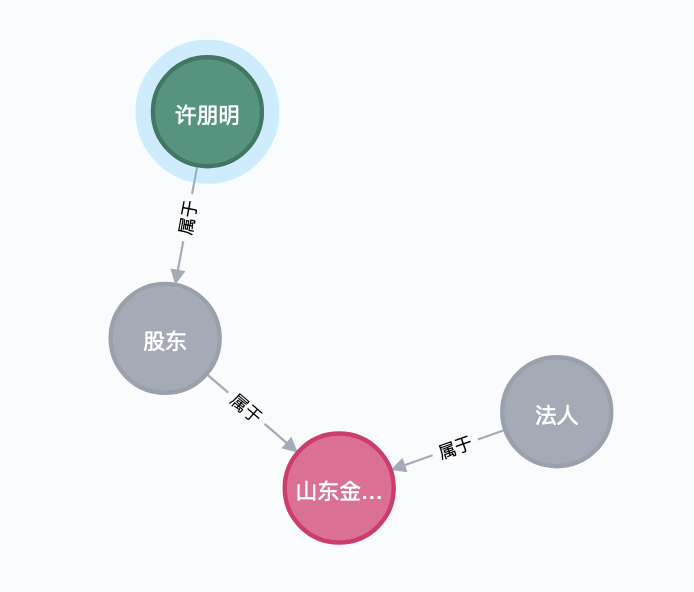
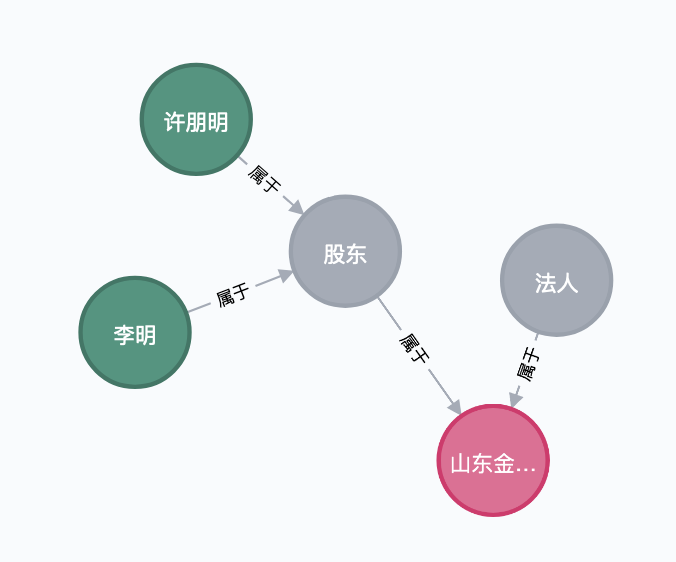
图二：修改前图谱显示及修改后图谱显示

**1.2.2 删除节点系统框架和实现**

删除节点功能主要是对于三级姓名节点的删除，因为实际情况下回存在企业管理人员离职或者转岗的情况，所以我们需要删除对应企业下属职位的对应人物并删除节点关联。这里具体实现方法是调用Py2neo库先matching用户需求删除的节点，再调用graph.delete()进行删除。由于调用py2neo的graph类进行删除操作的时候，节点删除会一并删除所有该节点建立的关系，因此我们在目前的知识图谱的背景应用下，不需要对于关系进行独立的删除操作。当然日后用户有对于知识图谱中关系部分新的应用也很方便解决。下图是演示的界面与效果。



图三：删除节点系统页面展示



图四：修改前图谱显示及修改后图谱显示

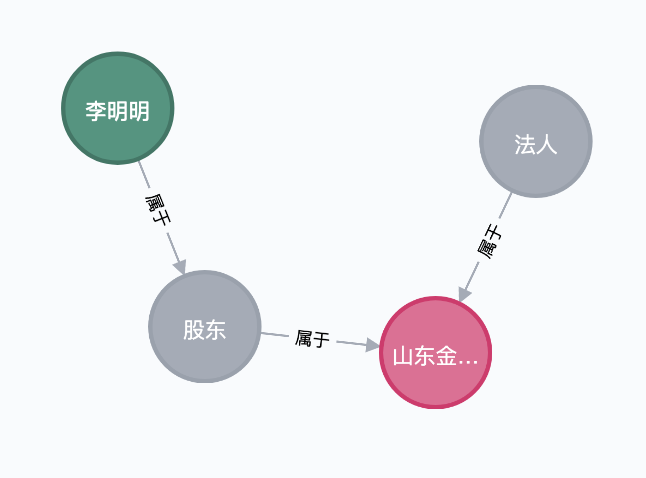
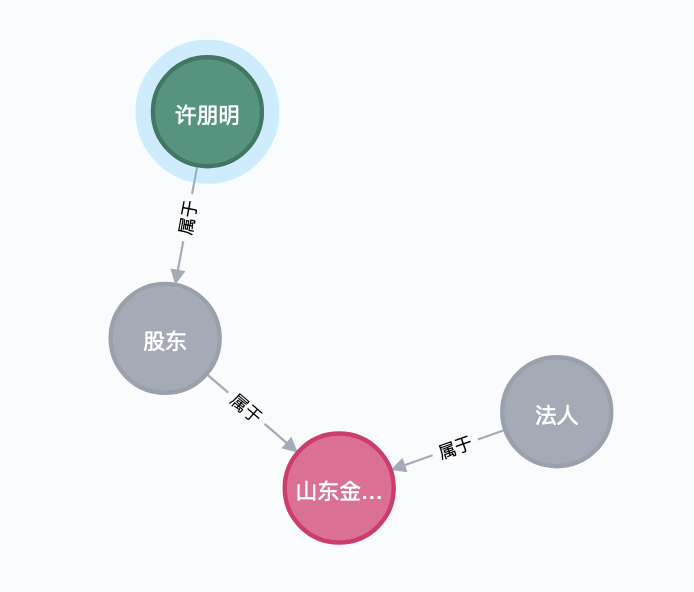
**1.2.3 修改节点系统框架和实现**

修改节点系统功能主要也是对于三级姓名节点进行信息修改，在实际情况下企业的管理人员可能经常会有变动，因此我们需要对于变动的人员进行姓名节点的调整。这里和删除节点系统功能的实现中，我们没有使用用户操作和实现较为简单的单节点匹配，因为在实际的应用中不能排除企业人员重名的情况，所以我们在使用姓名作为检索的这种方式必须要确保精准定位，因此也在前端要求用户必须输入需要修改/删除的三级姓名节点的上属企业节点和职位节点才能精确的匹配到我们需要操作的对应节点。其实这里提到的修改节点操作时前面所谓的删除操作和增加操作的叠加，只不过是我们使用了py2neo的ogm组件使得neo4j数据库与python端的新建类相联系，这样后续操作的可修改性就更强，py2neo中的ogm组件相当于从数据库到python类的一个映射，把neo4j中的对应标签信息作为Python中类的名字，把其下属的一些对应节点，当作类的成员，当然主体成员也是会按照cypher中的对应要素分为Node和Relationship两大类，但是在我们实际的应用汇总会淡化他们之间的区分，而会使用更加贴近用户使用的方法给用户一个更加贴近实际的描述。

py2neo.ogm软件包包含一组用于将Python对象绑定到底层图形数据的工具。类定义扩展了GraphObject，并包括属性和标签定义以及Related对象的详细信息。py2neo OGM框架的核心是GraphObject。 这是所有要映射到图形数据库的类的基类。 每个GraphObject都包装一个节点和一组指向RelatedObjects的指针以及连接它们的关系详细信息。可以像构造任何其他Python对象一样构造GraphObject实例，但也可以从数据库中对其进行匹配。 每个实例可能包含代表标签，节点或相关对象的属性。在GraphObject上定义的标签为基础中央节点的标签提供访问器。它显示为布尔值，该值的设置允许打开或关闭标签。标签在API中的显示方式与布尔属性相同。两者之间的区别在于值如何转换为图形数据库。 如果定期对该值进行匹配，则通常应使用标签。次要信息或支持信息可以存储在布尔属性中。下图是演示的界面与效果。



图五：删除节点系统页面展示



图六：修改前图谱显示及修改后图谱显示

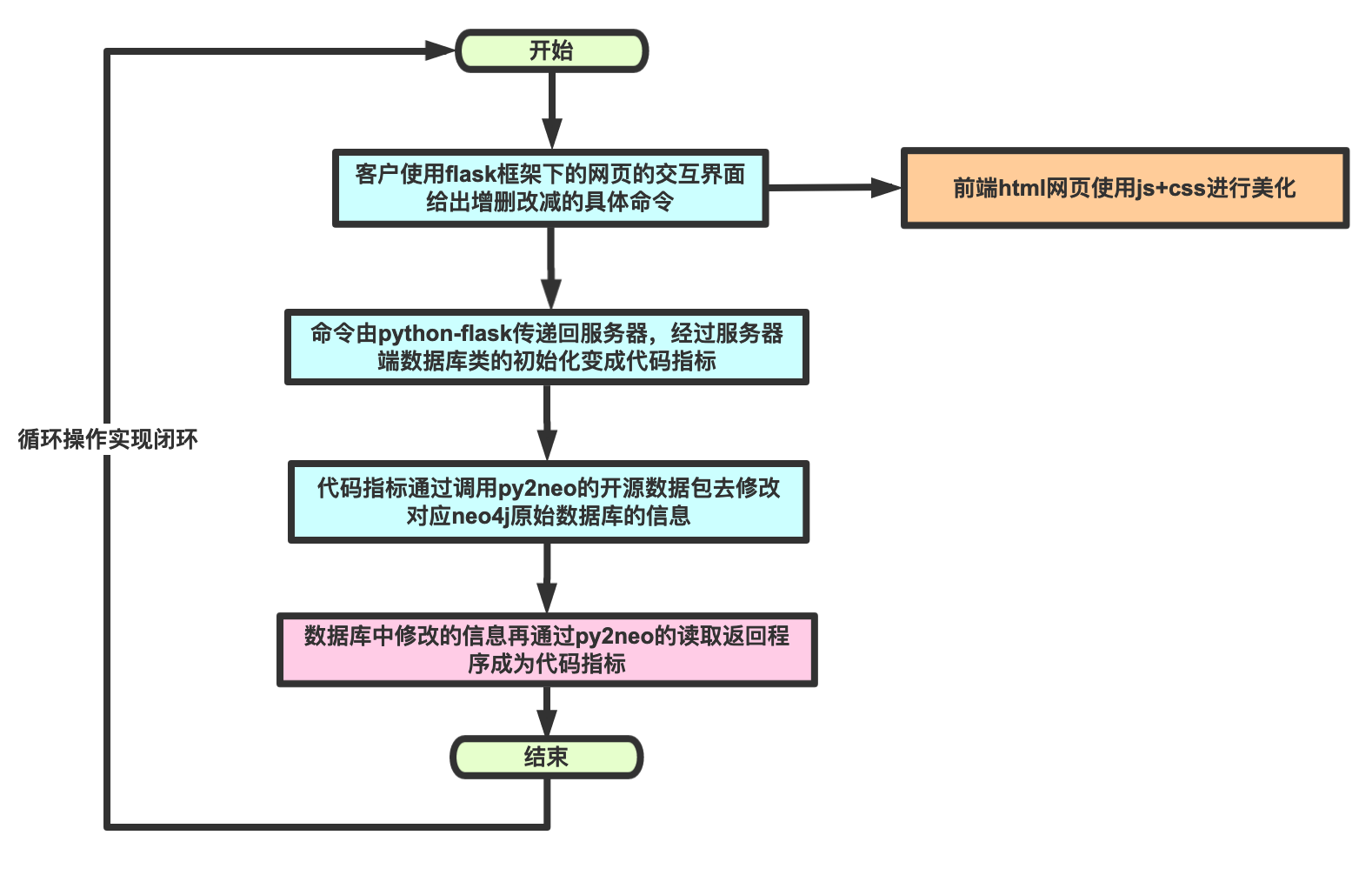
**1.3 知识图谱增删改查功能的flask串联实现**

最后关于知识图谱的查询系统我们结合了知识图谱的可视化操作进行一个具体的检索显示，在本报告的其他章节中有所涉及，这里就不详细叙述。

除此之外，我们给知识图谱的增删改查系统制作了一个用账号密码登陆的主界面，使得有权限的用户才可以使用这样一个平台进行操作。然后在主界面部分用户可以选择进入增、删、改、查的任何一个功能的界面，选择并登陆成功之后会跳转到相应的功能界面进行下一个具体的操作。完成每一个操作之后也会有提示你回到主界面。整体的主界面展示如下：



图七：增删改查主界面的显示



图八：增删改查整体框架的实现

上述提到的所用实现的功能和系统我们都是使用了flask框架作为交互通道，Web Application Framework（Web应用程序框架）或简单的Web Framework（Web框架）表示一个库和模块的集合，使Web应用程序开发人员能够编写应用程序，而不必担心协议，线程管理等低级细节。Flask自由、灵活，可扩展性强，第三方库的选择面广，开发时可以结合自己最喜欢用的轮子，也能结合最流行最强大的Python库。我们使用flask框架把以上提到的功能在服务器上进行部署，方便所有公网上的用户访问服务器对应端口对于网站进行访问从而完成对应的操作。这样以来，用户通过访问服务器对应的端口，通过flask框架与服务器进行交互，flask引出的html网页在post的条件下，request.form.get得到对应的信息反馈给服务器，服务器运行python脚本对neo4j数据库中的数据库的数据进行增删改查，然后用可视化的方法再回馈给用户，整体上实现了一个闭环操作，我们运用了neo4j这样一个强大的数据库平台构造企业知识图谱储存信息作为后台，使得交互又方便，可靠性又强。

当然，后续我们也有很多优化的方向，诸如在查找的部分增加关键字的检索，不需要用户对于公司企业的名称知道的很确凿，仍然可以精准的定位到想要修改的部分。除此之外，前端的CSS和JS组件需要再做一些优化，使得整个界面能够更加的美观漂亮，在功能实现的过程中还需要根据实际的用户需求不断调整输入的交互设计，避免使用过多专业化术语，能够让用户更容易上手。最后需要对于检测匹配节点的一些异常情况进行分析，优化从html端得到的输入转化为对应节点的代码，使得节点匹配环节又快又准。