## 简介

Neo4j 的数据由下面几部分构成：

* 节点：为一个字符串
* 边：为基本类型或者是基本类型数组
* 属性：为一个类似HashMap的数据
* 总结：所有的边和节点可以有任意多的属性

## 特点

* 它拥有简单的查询语言 Neo4j CQL
* 它遵循属性图数据模型
* 它通过使用 Apache Lucence 支持索引
* 它支持 UNIQUE 约束
* 它包含一个用于执行 CQL 命令的 UI：Neo4j 数据浏览器
* 它支持完整的 ACID（原子性，一致性，隔离性和持久性）规则
* 它采用原生图形库与本地 GPE（图形处理引擎）
* 它支持查询的数据导出到 Json 和 XLS 格式
* 它提供了 REST API，可以被任何编程语言（如 Java，Spring，Scala 等）访问
* 它提供了可以通过任何 UI MVC 框架（如 Node JS ）访问的 Java 脚本
* 它支持两种 Java API：Cypher API 和 Native Java API 来开发 Java 应用程序

### Py2Neo的基本操作（对接 Neo4j的Python库）

1. 导入和连接

from py2neo import Graph, Node, Relationship， NodeMatcher

graph = Graph(host = IP，http\_port = 7474，user = 账号，password = 密码)

1. 增
   1. 增加节点：

a = Node('Person', name='Alice')

graph.create(a)

* 1. 增加关系：

a = Node('Person', name='Alice')

b = Node('Person', name=Bob)

r = Relationship(a,” KNOWS”,b)

graph.create(r)

# 创建子图

a = Node('Person', name='A')

b = Node('Person', name=’B’)

c = Node('Person', name='c')

d = Node('Person', name=’d’)

r1 = Relationship(a,” KNOWS”,b)

r2 = Relationship(c,” KNOWS”,d)

r3 = Relationship(a,” KNOWS”,c)

s = a|b|c|r1|r2|r3

graph.create(s)

1. 删:
   1. 删除节点关系（必须要先删除关系，后删除节点）。

graph.delete(relationship)

graph.delete(node)

* 1. 删除整个图

graph.delete\_all()

1. 改
   1. 修改属性：

node[key] = value  
node[key] += value

graph.push(node)

1. 查
   1. 使用Cypher查询：

graph.run(Cypher语句).data()

* 1. 使用API查询：

节点查询：NodeMatcher(graph).match(”Person”, name=”A”)

节点正则查询：NodeMatcher(graph).match(”Person”).Where(“\_.name=~ ‘A.\*?’”)

节点排序查询：NodeMatcher(graph).match(”Person”).order\_by('\_.age')

单节点查询：NodeMatcher(graph).match(”Person”).first()

关系查询：graph.match\_one(r\_type=关系类型例例如：'KNOWS')

其他

* 1. 设置默认属性

node.setdefault('location', '北京')

* 1. 获取子图交集

s1 & s2（s1和s2是两个subgraph）

* 1. 插入节点技巧（该方法可以避免重复节点出现）：

gb=graph.begin()

r1 = Relationship(Node,关系类型, Node)

gb.merge(r1, primary\_label=实体类型, primary\_key=实体属性的key)  
gb.commit()