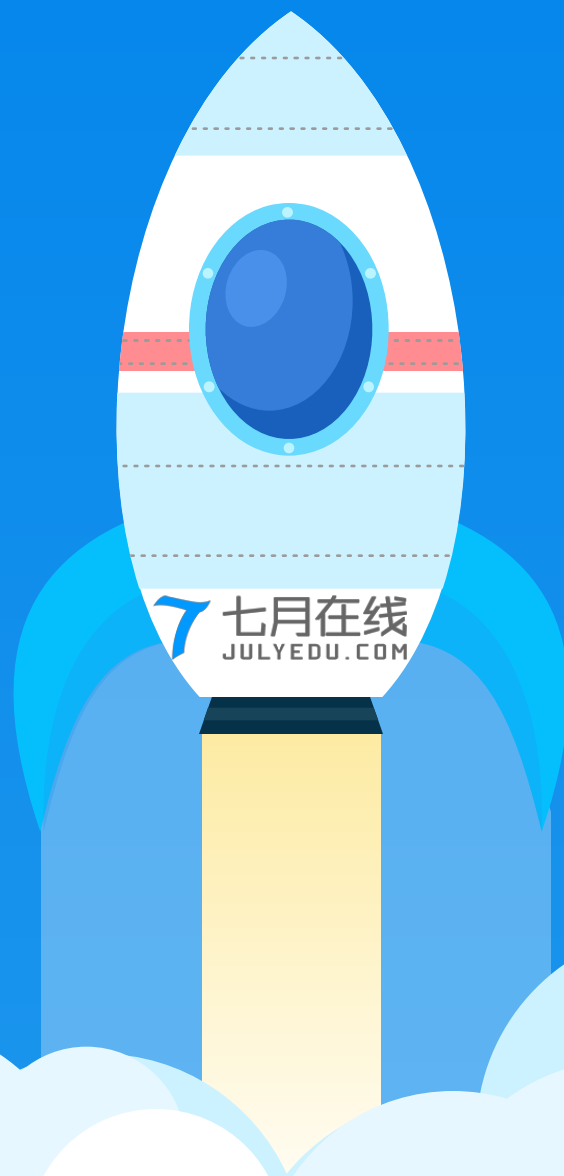


知识的存储与检索

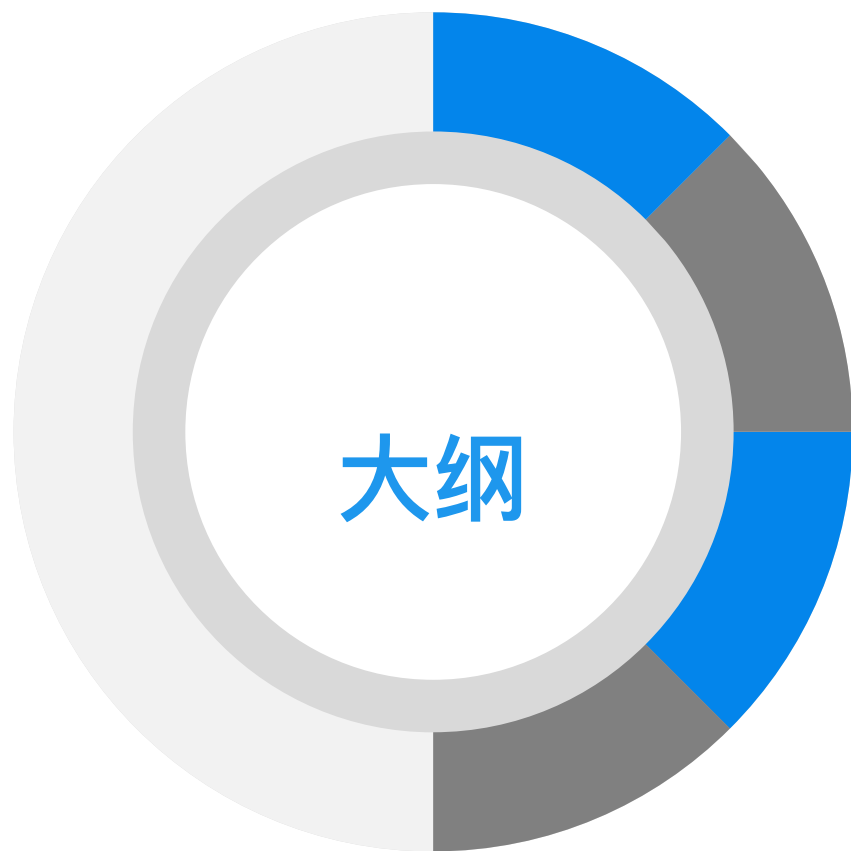
<https://www.julyedu.com/>



课程介绍

知识图谱实战课程表			
日期	时间	课程	备注
第一阶段 知识图谱的基础概念			
1月24日	晚20: 00~22: 00	第1课 课程介绍及知识图谱基础	在线直播
1月31日	晚20: 00~22: 00	第2课 知识的存储和检索	在线直播
第二阶段 非结构化数据的知识图谱构建			
2月6日	晚20: 00~22: 00	第3课 词汇挖掘与实体识别	在线直播
2月21日	晚20: 00~22: 00	第4课 关系抽取	在线直播

第三阶段 图表示与图算法			
2月28日	晚20: 00~22: 00	第5课 常用的图算法与图聚类	在线直播
3月7日	晚20: 00~22: 00	第6课 知识表示方法	在线直播
3月14日	晚20: 00~22: 00	第7课 图神经网络初步	在线直播
3月21日	晚20: 00~22: 00	第8课 图神经网络进阶	在线直播
第四阶段 图应用			
3月27日	晚20: 00~22: 00	第9课 基于知识图谱的问答系统	在线直播
3月28日	晚20: 00~22: 00	第10课 节点分类与关系推理相关项目	在线直播

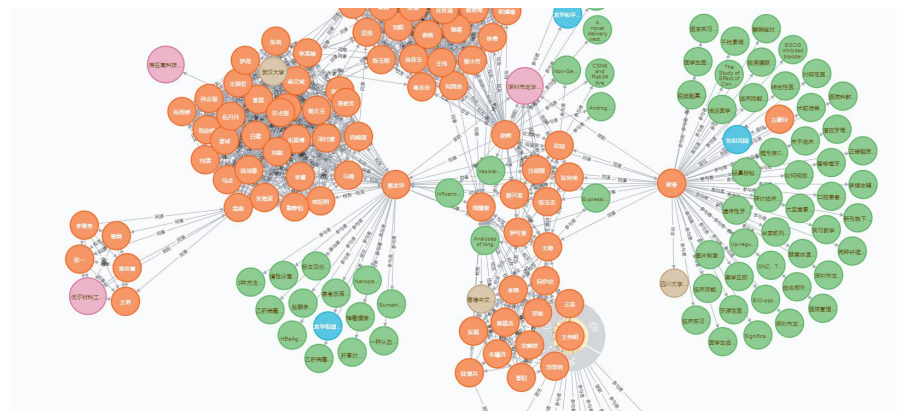


- ✓ neo4j简介及安装
- ✓ neo4j的基础语法
- ✓ neo4j案例

neo4j简介及安装

图形数据库（Graph Database）是NoSQL数据库家族中特殊的存在，用于存储丰富的关系数据，Neo4j 是目前最流行的图形数据库，支持完整的事务，在属性图中，图是由顶点（Vertex），边（Edge）和属性（Property）组成的，顶点和边都可以设置属性，顶点也称作节点，边也称作关系，每个节点和关系都可以由一个或多个属性。Neo4j创建的图是用顶点和边构建一个有向图，其查询语言cypher已经成为事实上的标准。

关系型数据库只对单个Join操作进行优化查询，而多重Join操作查询的性能显著下降。图形数据库适合查询关系数据，由于图形遍历的局部性，不管图形中由多少节点和关系，根据遍历规则，Neo4j只访问与遍历相关的节点，不受到总数据集大小的影响，从而保持期待的性能；相应地，遍历的节点越多，遍历速度越慢，但是变慢是线性的，这使得图形数据库不适合做海量数据统计分析。对与存在大量丰富关系的数据，遍历的性能不受图形数据量大小的影响，这使得Neo4j成为解决图形问题的理想数据库。



第一步：安装jdk

参考博文：<https://www.cnblogs.com/huzixia/p/10402200.html>

neo4j简介及安装

第二步：下载neo4j的安装包

地址：<https://neo4j.com/>

Download Neo4j

Experience Neo4j on Your
Desktop

Free. Get Started Today.

DOWNLOAD

Includes Neo4j Desktop and
Neo4j Enterprise Edition for Developers
[Learn more](#)

Get Started Now

Please fill out this form to begin your download

信息可以随便填，但是
要记住所填写的信息！！

之后会用到！！

*

*

*

*

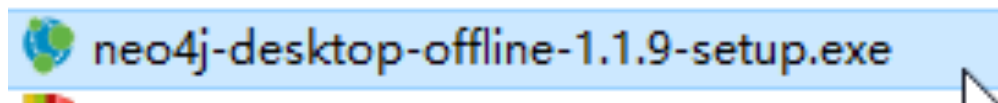
*

By downloading you agree to the [Neo4j License Agreement for Neo4j Desktop Software.](#)

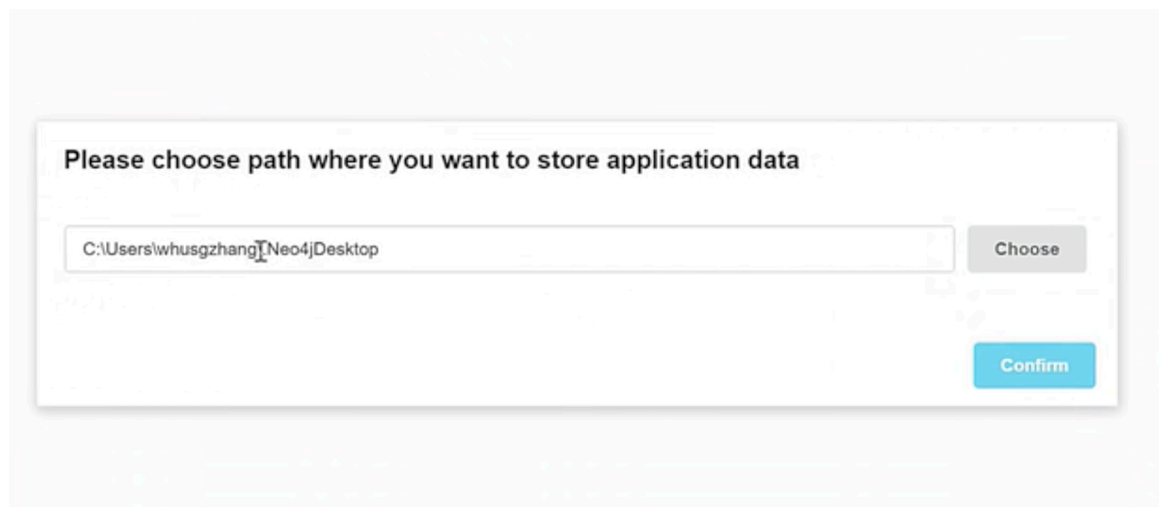
Download Desktop

The information you provide will be
used in accordance with the terms of

第二步：双击安装包

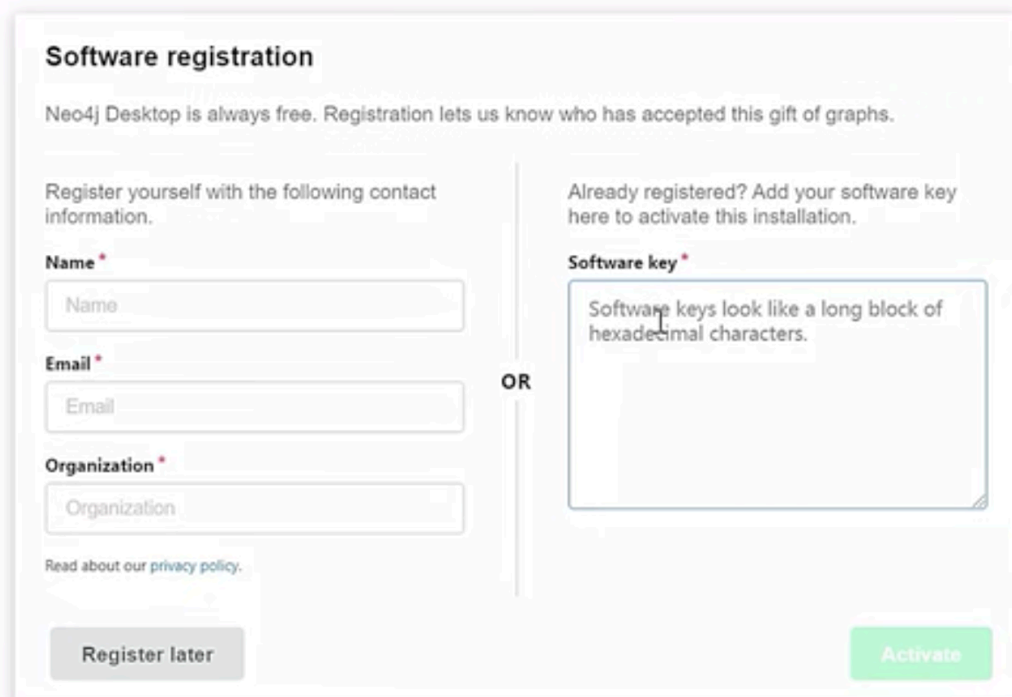


一步步地点击即可
可以更改安装目录及数据存储目录



第二步：填写信息

这里会用到刚刚所填写的信息！！



Software registration

Neo4j Desktop is always free. Registration lets us know who has accepted this gift of graphs.

Register yourself with the following contact information.

Name *

Email *

Organization *

[Read about our privacy policy.](#)

[Register later](#)

OR

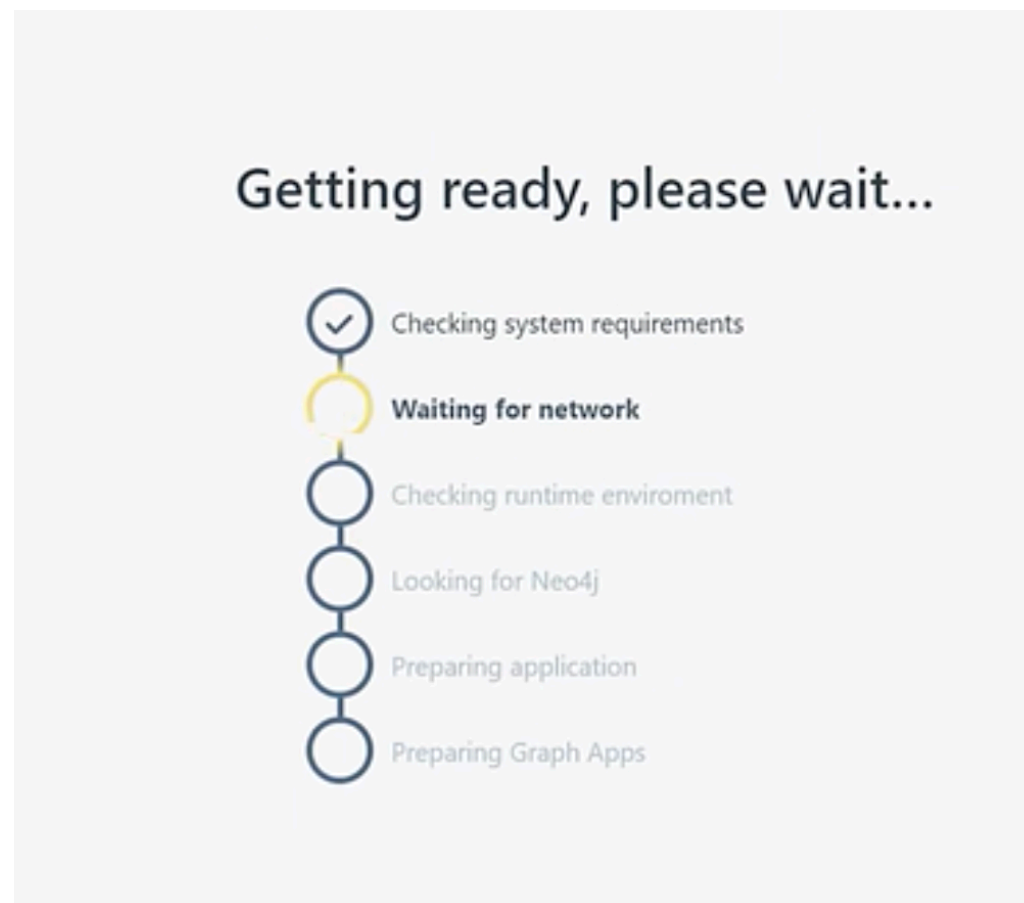
Already registered? Add your software key here to activate this installation.

Software key *

Software keys look like a long block of hexadecimal characters.

[Activate](#)

最后，出现如下进程，安装完成



小试牛刀

```
create(n:人工智能教育平台{name:'七月在线'})
```

```
create(n:小课{name:'知识图谱'})
```

```
MATCH(P1:`人工智能教育平台`)(P2:`小课`) create(P1)-[r:'开设']->(P2)
```

neo4j的基础语法

手工创建一个简单小型的电影知识图谱，包括：

- 创建电影节点
- 创建导演节点
- 创建演员节点
- 创建电影和导演的关系
- 创建电影和演员的关系

电影名称	类型	豆瓣评分
肖申克的救赎	犯罪	9.7
霸王别姬	剧情	9.6
辛德勒的名单	历史	9.5
盗梦空间	剧情	9.3
星际穿越	剧情	9.3
荆轲刺秦王	历史	8.2
绿里奇迹	剧情	8.9

创建一个节点的命令：

```
create (variable:label {key1:value1,key2:value2})
```

创建一个电影的命令：

```
create (n:film {name:'肖申克的救赎',type:'犯罪',score:'9.7'})
```

同时创建多部电影的命令：

```
create (:film {name:'肖申克的救赎',type:'犯罪',score:'9.7'}),  
(:film {name:'霸王别姬',type:'剧情',score:'9.6'}),  
(:film {name:'辛德勒的名单',type:'历史',score:'9.5'}),  
(:film {name:'盗梦空间',type:'剧情',score:'9.3'}),  
(:film {name:'星际穿越',type:'剧情',score:'9.3'}),  
(:film {name:'荆轲刺秦王',type:'历史',score:'8.2'}),  
(:film {name:'绿里奇迹',type:'剧情',score:'8.9'})
```

同时创建多个导演的命令：

```
create (:director {name:'弗兰克·德拉邦特'}),  
(:director {name:'陈凯歌'}),  
(:director {name:'史蒂文·斯皮尔伯格'}),  
(:director {name:'克里斯托弗·诺兰'})
```


同时创建多个演员的命令：

```
create (:actor {name:'蒂姆·罗宾斯'}),  
(:actor {name:'张国荣'}),  
(:actor {name:'连姆·尼森'}),  
(:actor {name:'马修·麦康纳'}),  
(:actor {name:'张丰毅'})
```

创建电影“肖申克的救赎”和导演“弗兰克·德拉邦特”关系的命令：

```
MATCH(a:director),(b:film)
```

```
WHERE a.name='弗兰克·德拉邦特' AND b.name='肖申克的救赎'
```

```
CREATE(a)-[r:direct]->(b)
```

创建电影和演员关系的命令：

```
MATCH(a:actor),(b:film)
```

```
WHERE a.name='张国荣' AND b.name='霸王别姬'
```

```
CREATE(a)-[r:play]->(b)
```

创建电影和演员关系的命令：

```
MATCH(a:actor),(b:film)
```

```
WHERE a.name='蒂姆·罗宾斯' AND b.name='肖申克的救赎'
```

```
CREATE(a)-[r:play]->(b)
```

创建电影和演员关系的命令：

```
MATCH(a:actor),(b:film)
```

```
WHERE a.name='张丰毅' AND (b.name='霸王别姬' or b.name='荆轲刺秦王')
```

```
CREATE(a)-[r:play]->(b)
```

查询节点与关系：

- 查询某个电影
- 查询某个标签下的所有节点
- 查询两个节点间的关系
- 通过函数type获取关系的类型

➤ 查询某个电影

```
MATCH(a:film)
WHERE a.name='霸王别姬'
RETURN a
```

neo4j的基础语法

➤ 查询某个标签下的所有节点

```
MATCH(a:director)
```

```
RETURN a
```


➤ 查询两个节点间的关系

```
MATCH (:director {name:'弗兰克·德拉邦特'})-[r]->(:film{name:'肖申克的救赎'})  
RETURN r
```

➤ 通过函数type获取关系的类型

```
MATCH ()-[r]->(film{name:'肖申克的救赎'})  
RETURN type(r)
```

删除节点和关系

- 删除某个节点
- 删除某个标签的节点
- 删除所有节点
- 删除两个节点的关系
- 删除某节点的关系
- 删除某个标签全部的关系

neo4j的基础语法

➤ 删除某个节点

```
MATCH(a:actor)
```

```
WHERE a.name='马修·麦康纳'
```

```
DELETE a
```

neo4j的基础语法

➤ 删除某个标签的所有节点

```
MATCH(a:actor)
```

```
DELETE a
```

neo4j的基础语法

➤ 删除所有节点

MATCH(n)

DELETE n

➤ 删除两个节点的关系

```
MATCH (:actor {name:'张国荣'})-[r]->(:film {name:'霸王别姬'})  
DELETE r
```

neo4j的基础语法

➤ 删除某节点的关系

```
MATCH (:actor {name:'张国荣'})-[r]->()  
DELETE r
```

neo4j的基础语法

➤ 删除某个标签全部的关系

```
MATCH ()-[r]->(film)
```

```
DELETE r
```

属性的相关操作：

➤ 增加节点属性

```
MATCH(n:film) WHERE n.name='盗梦空间' SET n.language='english'
```

➤ 删除节点属性

```
MATCH(n:film) WHERE n.name='盗梦空间' REMOVE n.language
```

neo4j案例

朋友圈

```
match n=(:Person{name:'小北'})-[:认识]->>() return n
```

```
match n=(:Person{name:'小北'})-[*..2]->>() return n
```

```
match n=shortestPath((:Person{name:'小Y'})-[*..6]-  
>(:Person{name:'小明'})) return n
```

欺诈环检测

//创建三个人的账户

```
create (accountHolder1:AccountHolder{
  FirstName:"John",
  LastName:"Doe",
  UniqueId:"JohnDoe"
})
```

```
create (accountHolder2:AccountHolder{
  FirstName:"Jane",
  LastName:"Appleseed",
  UniqueId:"JaneAppleseed"
})
```

```
create (accountHolder3:AccountHolder{
  FirstName:"Matt",
  LastName:"Smith",
  UniqueId:"MattSmith"
})
```

欺诈环检测

//创建地址

```
create (address1:Address{
  Street:"123 NW 1st Street",
  City:"San Francisco",
  State:"California",
  ZipCOde:"94101"
})
```

//把账户人1、账户人2、账户人3关联到地址上

```
create (accountHolder1)-[:HAS_ADDRESS]->(address1),
      (accountHolder2)-[:HAS_ADDRESS]->(address1),
      (accountHolder3)-[:HAS_ADDRESS]->(address1)
```

欺诈环检测

//创建电话号码

```
create (phoneNumber1:PhoneNumber{
  PhoneNumber:"555-555-555"
})
```

//把账户1、账户2关联到电话号码1上

```
create (accountHolder1)-[:HAS_PHONENUMBER]->(phoneNumber1),
      (accountHolder2)-[:HAS_PHONENUMBER]->(phoneNumber1)
```

欺诈环检测

//创建社会安码 SSN1

```
create (ssn1:SSN{  
  SSN:"241-23-1234"  
})
```

//把账户人2、账户人3关联到SSN1

```
create (accountHolder2)-[:HAS_SSN]->(ssn1),  
       (accountHolder3)-[:HAS_SSN]->(ssn1)
```

//创建社会安全码SSN2 关联到账户人1

```
create (ssn2:SSN{SSN:"241-23-4567"})<-[:HAS_SSN]-(accountHolder1)
```


欺诈环检测

//创建信用卡1 关联到账户1

```
create (creditCard1:CreditCard{
  AccountNumber:"1234567890123456",
  Limit:5000,
  Balance:1442.23,
  ExpirationData:'02-20',
  SecurityCode:'456'
})<-[:HAS_CREDITCARD]-(accountHolder1)
```

//创建信用卡2 关联到账户2

```
create (creditCard2:CreditCard{
  AccountNumber:"2345678901234567",
  Limit:4000,
  Balance:2345.56,
  ExpirationData:'02-20',
  SecurityCode:'456'
})<-[:HAS_CREDITCARD]-(accountHolder2)
```

欺诈环检测

```
///创建银行账户1 关联到账户1  
create (bankAccount1:BankAccount{  
  AccountNumber:"2345678901234567",  
  Balance:7054.43  
})<-[:HAS_BANKACCOUNT]-(accountHolder1)
```

```
//创建银行账户2 关联到账户2  
create (bankAccount2:BankAccount{  
  AccountNumber:"3456789012345678",  
  Balance:4231.12  
})<-[:HAS_BANKACCOUNT]-(accountHolder2)
```

```
//创建银行账户3 关联到账户3  
create (bankAccount3:BankAccount{  
  AccountNumber:"4567890123456789",  
  Balance:12345.45  
})<-[:HAS_BANKACCOUNT]-(accountHolder3)
```

欺诈环检测

```
//创建无抵押贷款2并关联到账户人2
create (unsecuredLoan2:UnsecureLoan{
  AccountNumber:'4567890123456789-0',
  Balance:90453,
  APR:0.0541,
  LoanAmount:12000
})<-[:HAS_UNSECUREDLOAN]-(accountHolder2)
```

```
//创建无抵押贷款3并关联到账户人3
create (unsecuredLoan3:UnsecureLoan{
  AccountNumber:'5678901234567890-0',
  Balance:16341.95,
  APR:0.0341,
  LoanAmount:22000
})<-[:HAS_UNSECUREDLOAN]-(accountHolder3)
```

```
//创建电话号码3 并关联到账户3
create (phoneNumber2:PhoneNumber
{PhoneNumber:'555-555-1234'
})<-[:HAS_PHONENUMBER]-(accountHolder3)
```

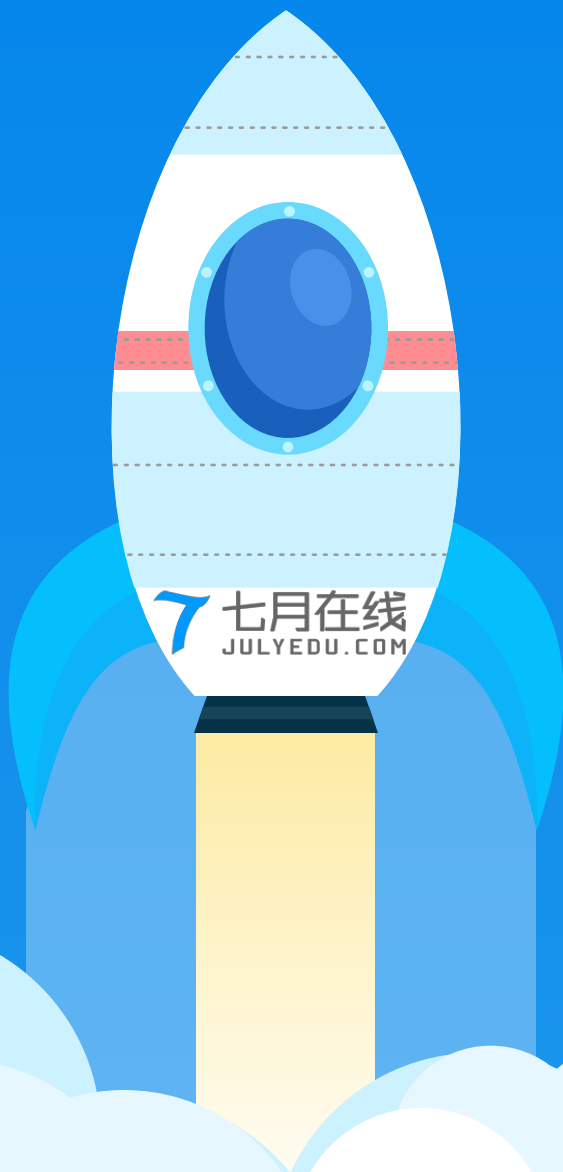
欺诈环检测

```
match (accountHolder:AccountHolder)-[]->(contractInformation)
with contractInformation,count(accountHolder) as RingSize,collect(accountHolder.UniqueId) as
AccountHolders
where RingSize>1
return AccountHolders as FraudRing,
           labels(contractInformation) as ContractType,
           RingSize
order by RingSize desc
```

✓ <https://blog.csdn.net/RHJlife/article/details/108586578>



微信扫一扫关注我们



THANKS

Speaker name and title

<https://www.julyedu.com>