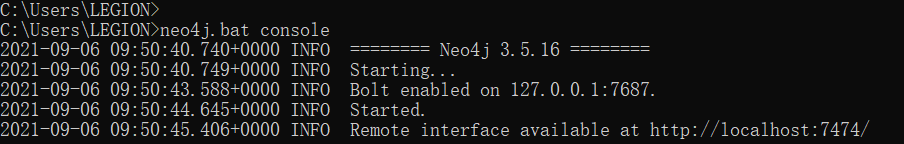
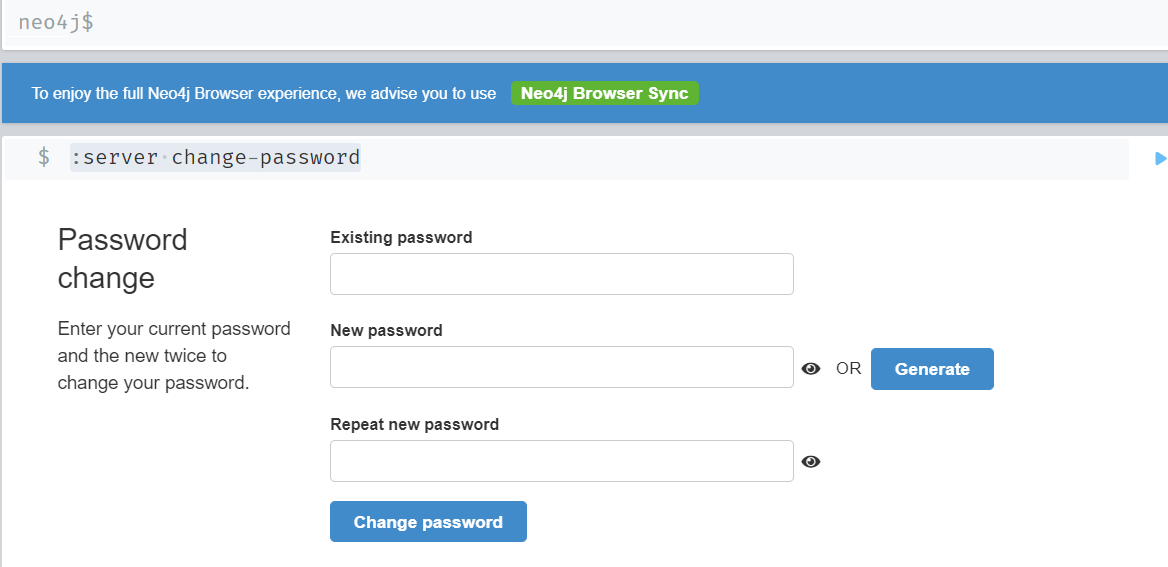
<https://www.cnblogs.com/future-dream/p/10908738.html>

<http://neo4j.com.cn/public/cypher/neo4j_cql_remove.html>

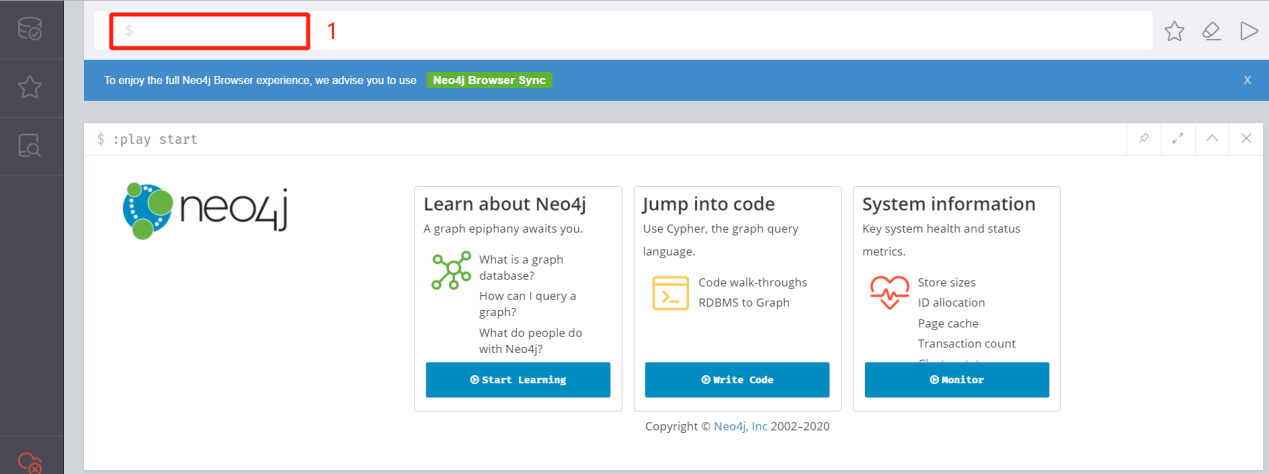


<http://localhost:7474/>



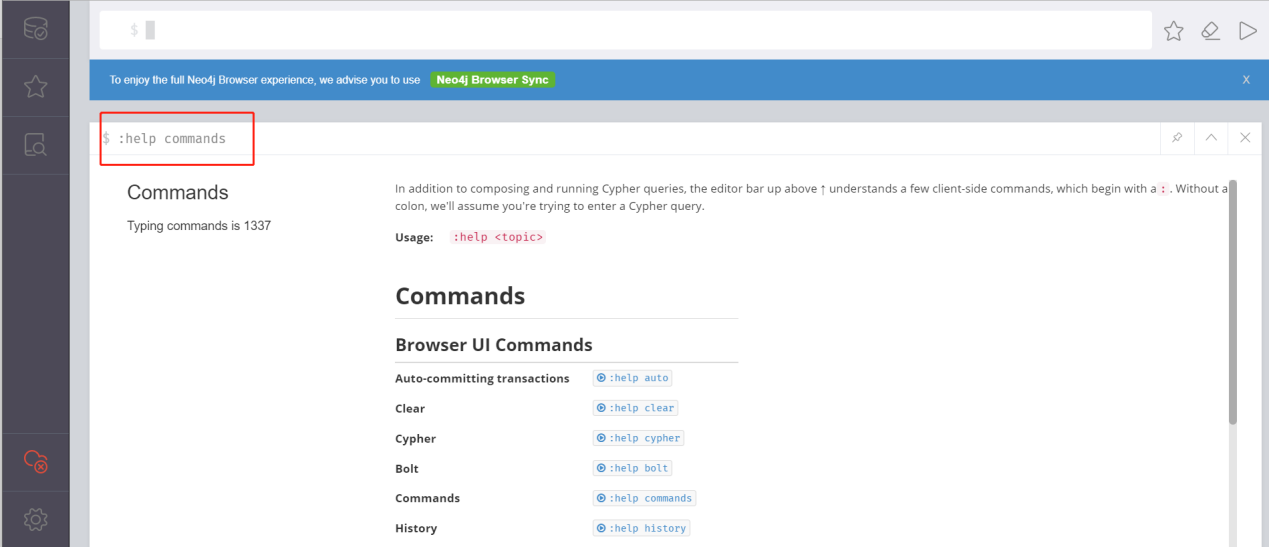
默认的用户名和密码均为：neo4j

首次登录的时候会有修改密码的需求



启动服务，在红框$处输入 :play start（包含标点）

查看帮助命令 :help commands



刚开始时，数据库中都是空的，是没有数据的，我尝试在数据库中加入一个节点，

输入命令

1. **创建节点（两个）**

create (n:西游记{name:"唐僧", skill:"南无阿弥陀佛", student:"孙悟空"})

create (n:西游记{name:"孙悟空", skill:"72变化", master:"唐僧"})

n为一个对象，可以任意的命名，key最后会作为property（属性）出现在property keys的内容中。

CREATE (

<node-name>:<label-name>

{

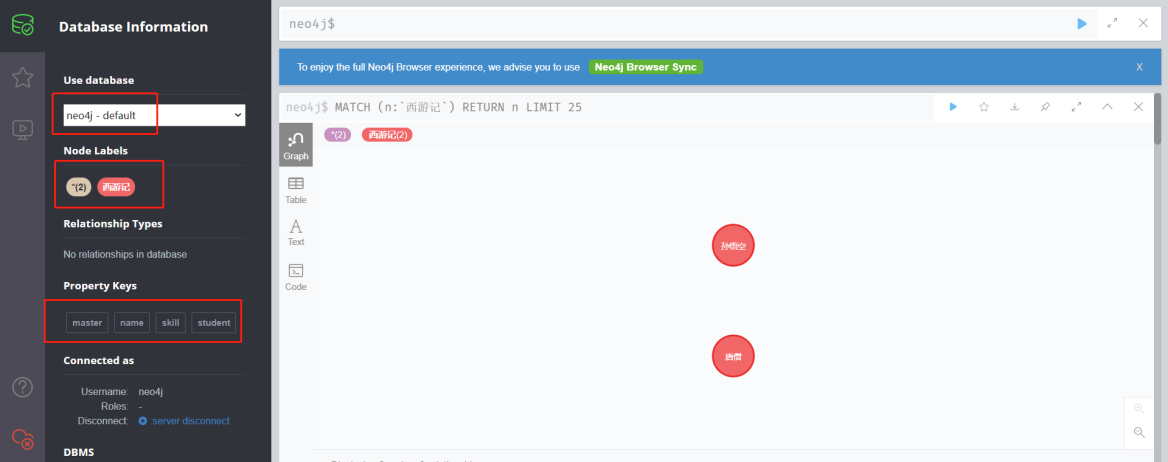
<Property1-name>:<Property1-Value>

........

<Propertyn-name>:<Propertyn-Value>

}

)

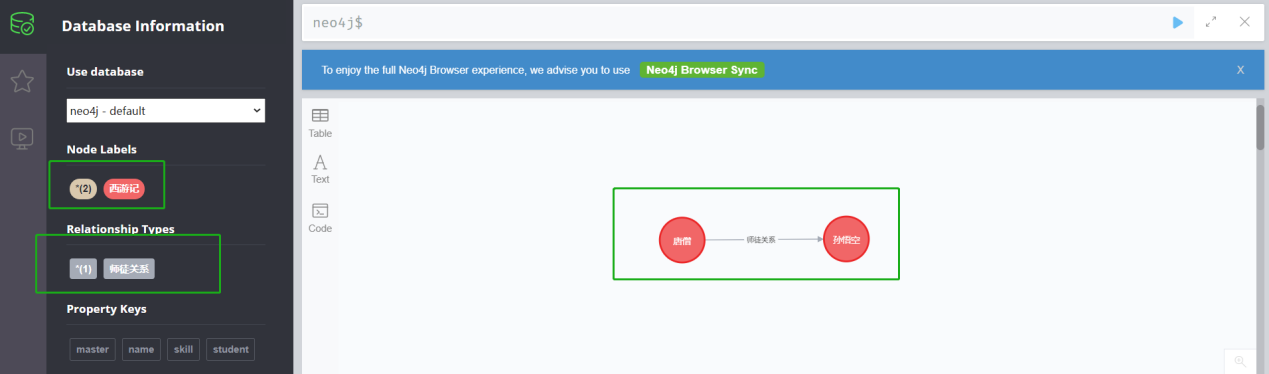


1. **添加节点之间的关系**

首先是获取两个关系对象，先建立两个对象，找出这两个点。在通过对象指明的标签，根据具体的属性值来查找出两个节点信息，最后用create (a) - [r: relationshipname] ->(b);来创建关系

（注:关系有具有方向性质的）创建的语句如下：

match (a:西游记), (b:西游记) where a.name = "唐僧" and b.name = "孙悟空" create (a)-[r:师徒关系]->(b);



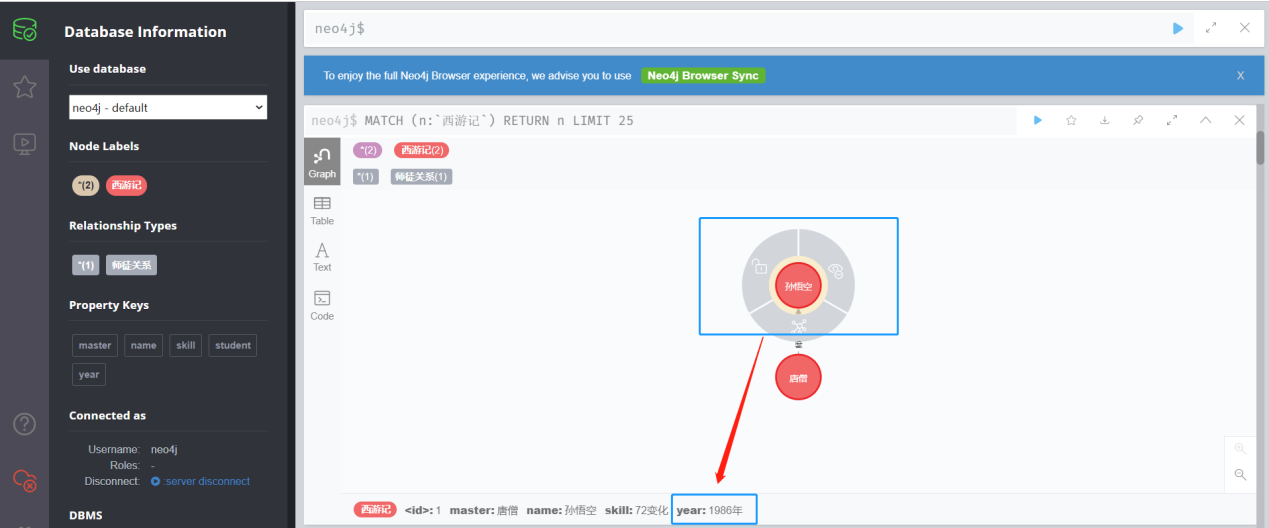
运行后进行查询可以看出两者之间存在了“师徒关系”的关系。关系从唐僧指向孙悟空。

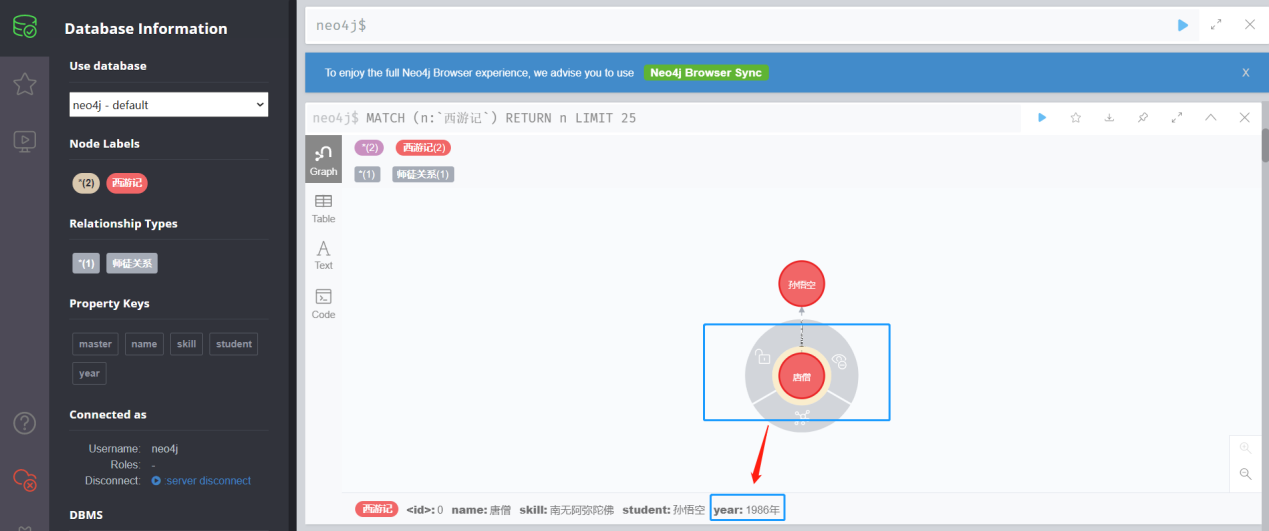
1. **新增属性**

确定节点的位置，然后使用set函数来增加和修改属性，如果属性存在就修改，如果属性不存在则添加

match (e:西游记) set e.year= "1986年";

这个方法会将属于“西游记”这个Node Label 下的节点都设置一个year属性。





再运行一次即可修改属性：

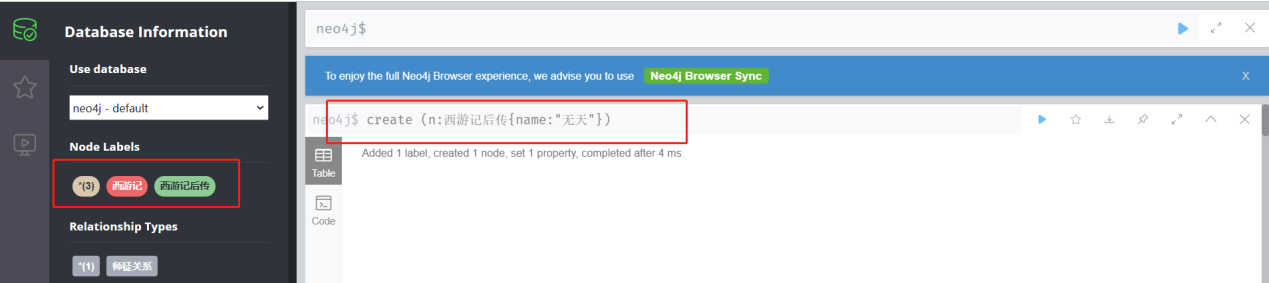
match (e:西游记) set e.year= "1986年的版本";

1. **跨标签建立关系**

建立两个对象，然后建立的同时限定对象，最后用 **create (object) - [r:relationship] -> (object)**，完成跨标签关系的建立

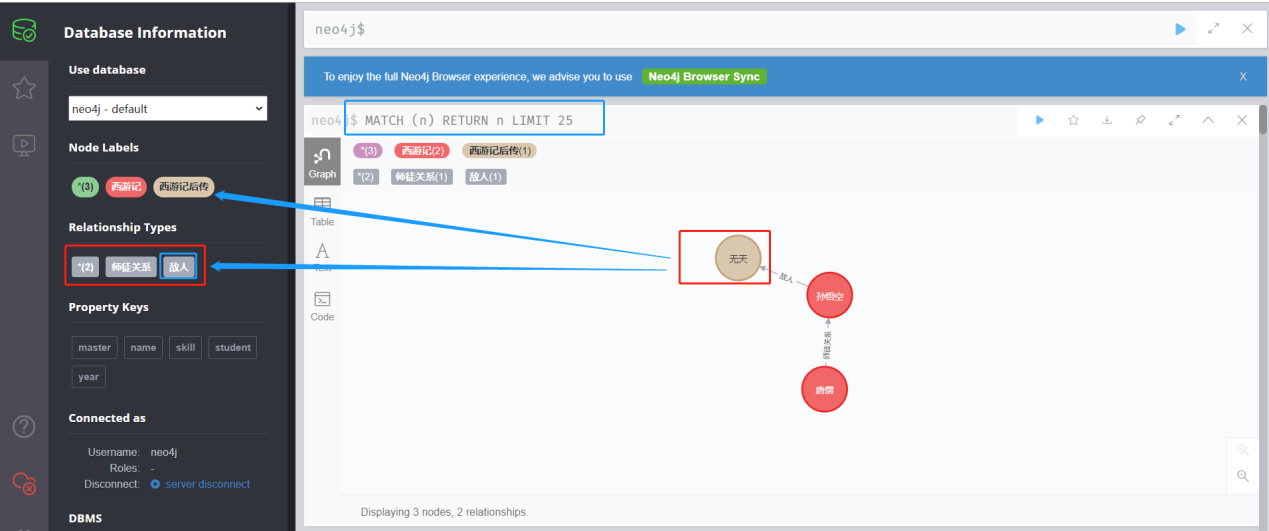
我们先创建一个新的对象节点：

create (n:西游记后传{name:"无天"})



再创建其他的关系：

match (n:西游记{name:"孙悟空"}), (m:西游记后传{name:"无天"}) create (n) - [r:敌人] -> (m);

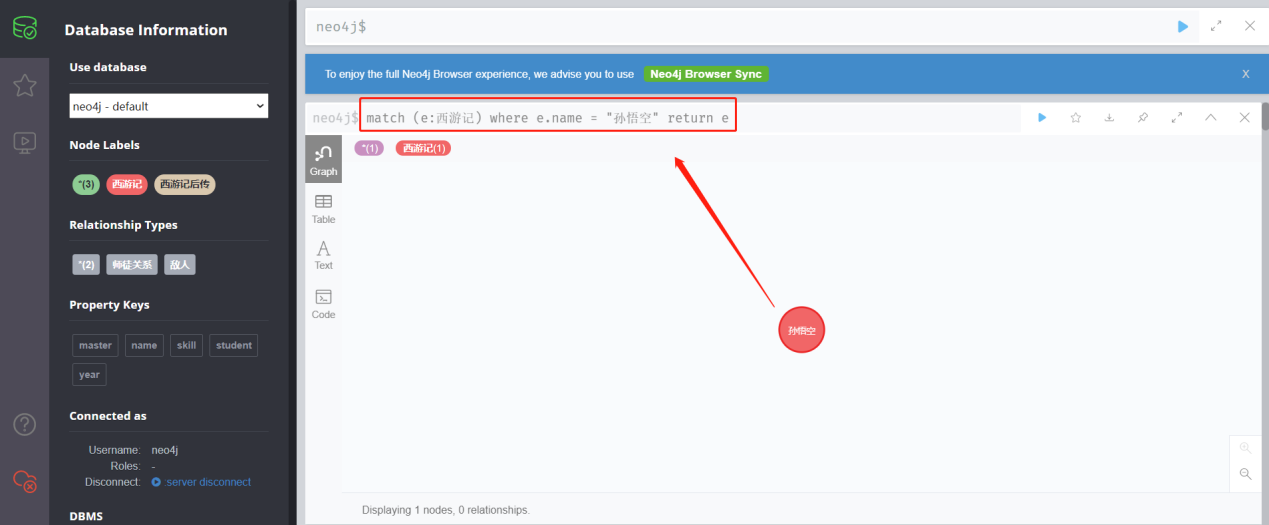


1. **单个节点的查询**首先建立查找对象然后通过标识找出对应的节点。

match (e:西游记) where e.name = "孙悟空" return e

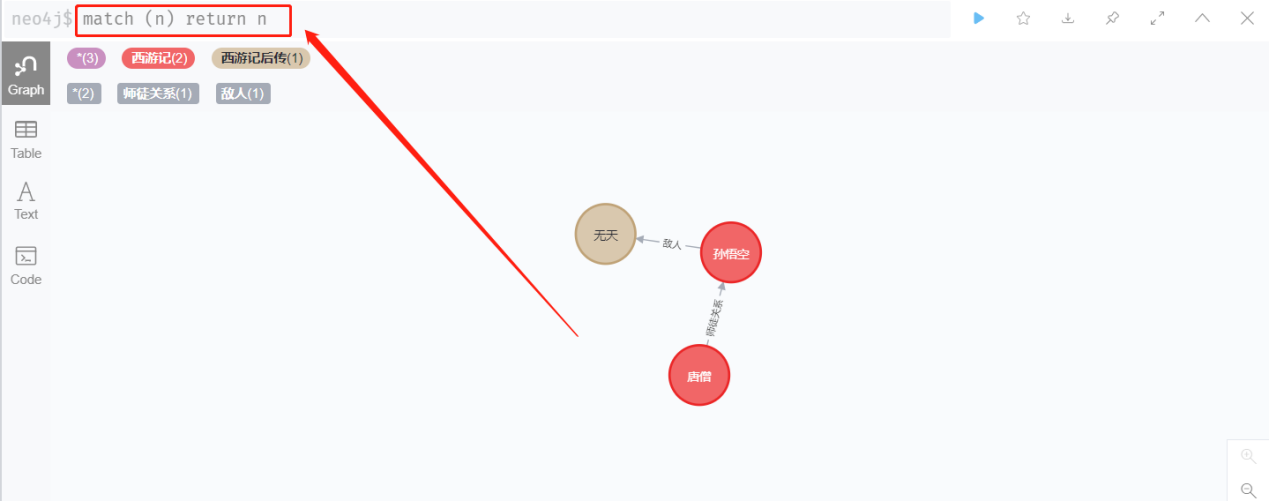
等价于：

match (n:西游记) where n.name = "孙悟空" return n



1. **整个label查询**  
   与前面的操作一样，首先要建立一个对象(n)后面接标签确定要查找的节点信息，最后返回信息。

match (n) return n



match (n:`西游记`) return  n

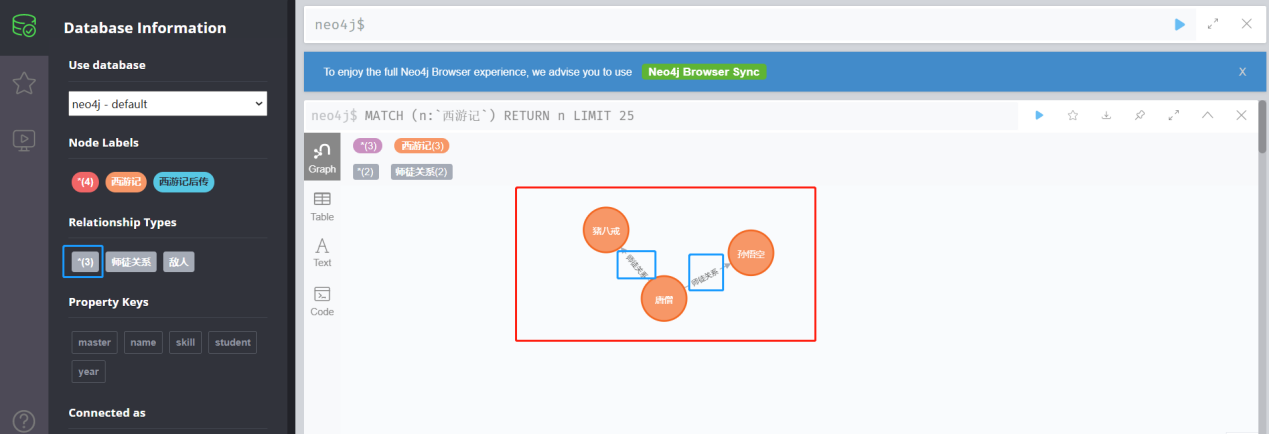


1. **整体关系查询**

我们先新建一个节点猪八戒并关系到唐僧上,执行下述两条语句：

CREATE (n:西游记{name:"猪八戒",skill:"eating"}) return n;

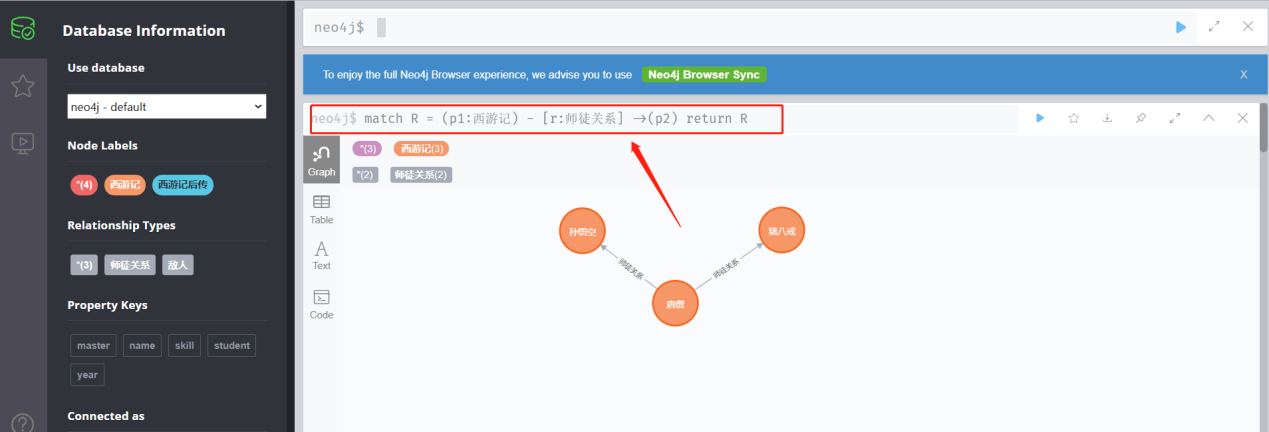
match (a:西游记), (b:西游记) where a.name = "唐僧" and b.name = "猪八戒" create (a)-[r:师徒关系]->(b) return r;



接下来我们基于“西游记这个”对象进行整体关系的查询：

match R = (p1:西游记) - [r:师徒关系] ->(p2) return R

即查找某一个Node label下的节点之间的关系，并返回结果。p1，p2为对象，同时用一个关系R来接收查询的结果。

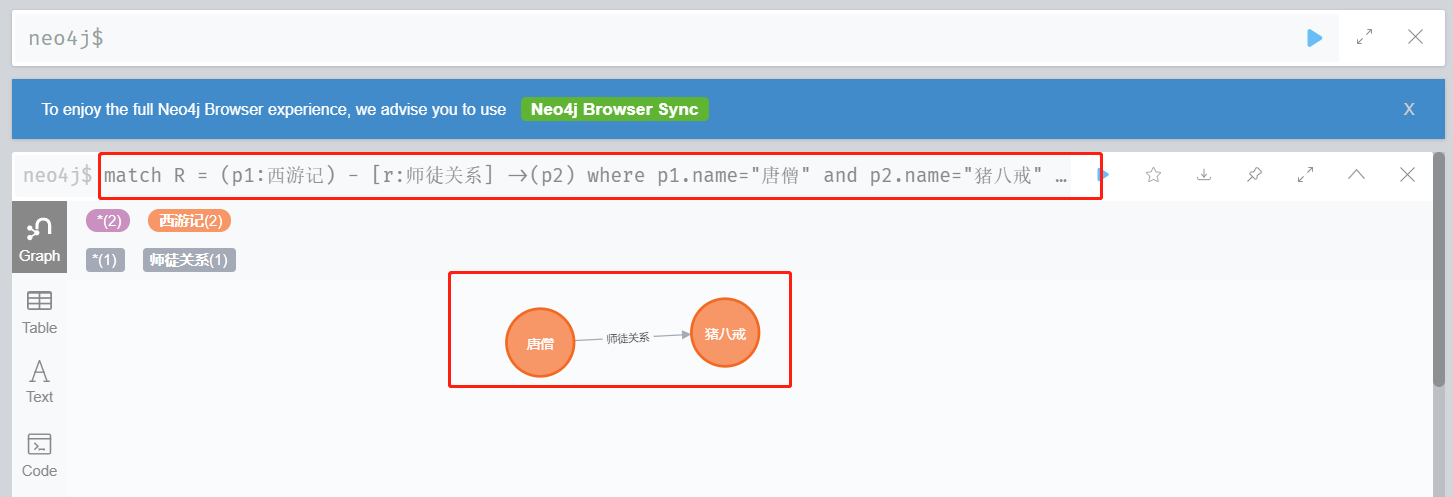


1. **单个关系的查询**

使用限定条件就可以完成单个数据查询(注:由于关系是有方向的因此不可以交换否则数据查找会出现none)

查询语句可以这样写：

match R = (p1:西游记) - [r:师徒关系] ->(p2) where p1.name="孙悟空" and p2.name="猪八戒" return R;



1. **删除节点关系，根据关系id值**

match (n) -[r]-() where id(r) =2 delete r

上述代码可以删除一个关系id为2的两个节点之间的关系，那么如何删除节点呢？

1. **删除节点**

删除节点首先得找到该节点，之后得先删除节点之间的关系，才能删除节点。

先查看某一个Node Label下的关系，比如“西游记”，可以看出：

**Started streaming 5 records after 1 ms and completed after 2 ms.**



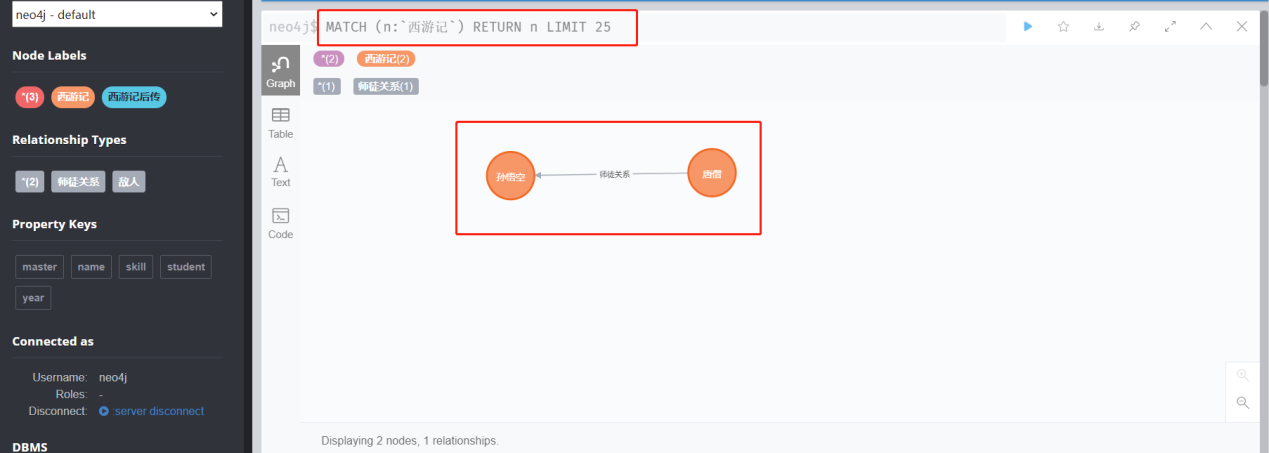
我们先删除“唐僧”与“猪八戒”之间的师徒关系，两者之间的id=3.

MATCH (n)-[r:师徒关系]-() where id(n)= 3 delete r

删除节点之间的关系以后，我们来删除节点：

MATCH (n:西游记{name:'猪八戒'}) delete n

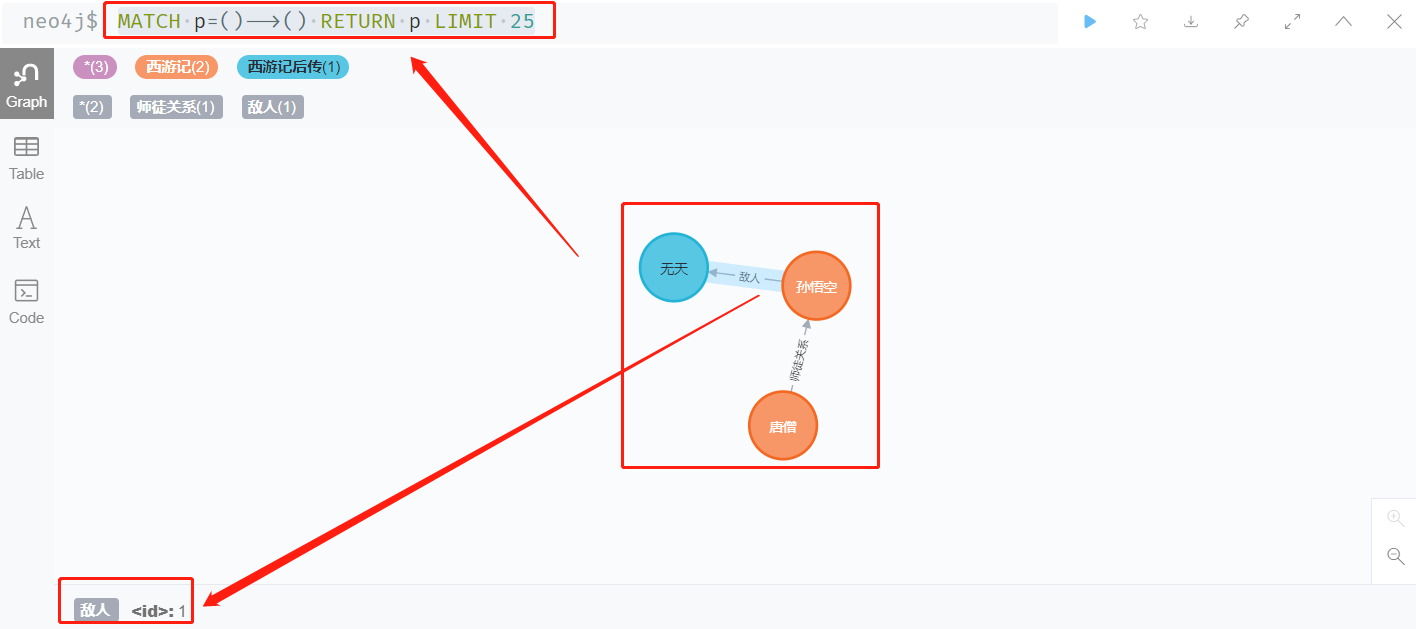
查询后只剩下一个关系了：



1. **删除与节点相连的节点和节点所有关系：**

这种删除不仅可以删除节点，还可以删除节点之间的关系，我们先查看一下里面的所有关系：

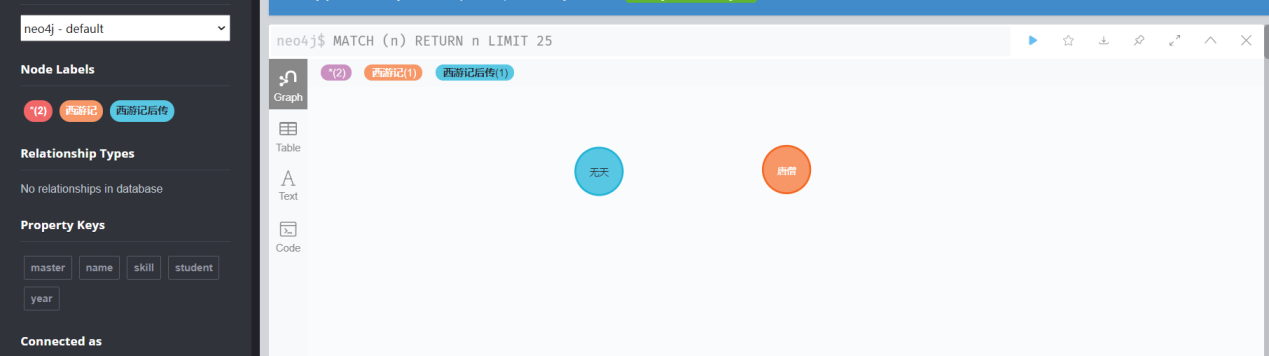
MATCH p=()-->() RETURN p LIMIT 25



然后我们删除关系id = 1的节点：

MATCH (n) -[r]-() where id(n) =1 DELETE n,r

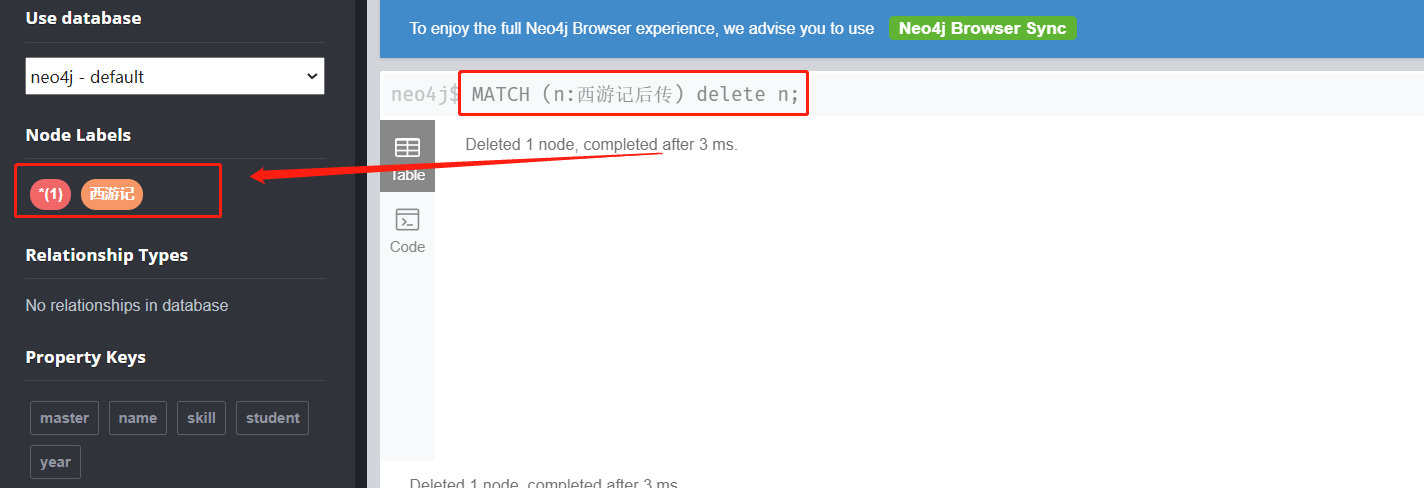
运行之后，只剩下独立的两个节点了：



1. **删除一个Node label：**

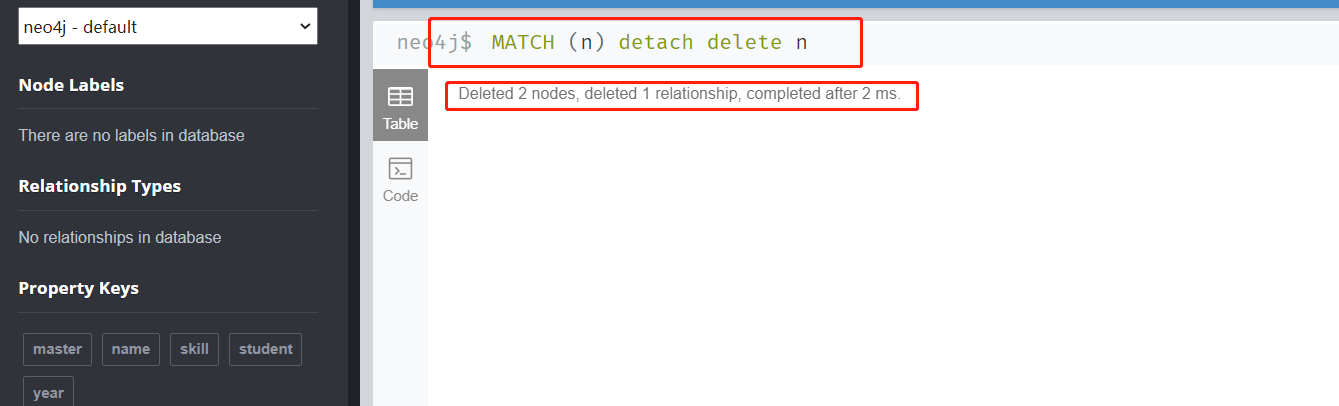
比如说删除西游记后传

MATCH (n:西游记后传) delete n;



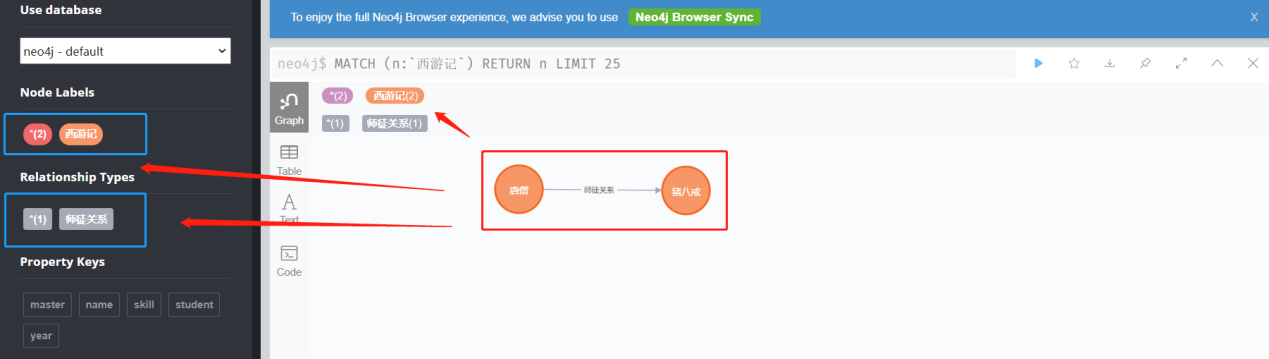
MATCH (n) detach delete n

会删除所有的数据



1. **查看存在哪些关系**

为了演示这个功能，我们在“西游记”中建立一猪八戒并关联“唐僧”为“师徒关系”，代码请参考第7小节。创建好之后如下：



查看所有关系的语句如下：

MATCH (n:西游记)-[r]-() RETURN r,type(r)

其中，type(r)返回的是关系的类型。

1. **属性标签的删除**

比如说我们想删除“唐僧”的一个“year”属性：

语句可以这样：

MATCH (n:西游记{name:"唐僧"}) remove n.year;



当然了修改属性可以这样：

MATCH (e:西游记{name:"唐僧"}) set e.skill = "嘛哩嘛哩哄"



1. **修改节点属性**

UNWIND [ {age: 3 0}, { addr :”s z ”} ] AS prop

MERGE (e:西游记{name:"唐僧"})

SET n += prop

1. **删除节点和属性**

一般来说，删除节点的时候如果使用下述方法：

MATCH (n) DELETE n

可能会报错，因为节点之间还存在着关系

可以这样将节点和关系同时删除：

MATCH () - [r:LOVES]->(m) DELETE r,m

使用下述语句来移除节点的时候, 并且不管节点与其他节点是否存在关系。

MATCH (n) REMOVE n:西游记

标签为“西游记”下的节点将被移除不管存在关系与否。

删除某一些节点的age属性

MATCH (n{name :”b ”}) REMOVE n.age

1. **For 循环创建节点**

WITH [”a ”,”b ”,”c ”] AS coll

FOREACH (value IN coll | CREATE (:Person{name:value}))

1. **索引**

下列查询语句将标签为 Person 的节点使用关键字的DEX ON 为节点的属性 name

建一个普通索引。

CREATE INDEX ON :Person(name)

查询的时候索引将被自动调用：

MATCH (n:Person) WHERE n.name IN [”a”,”b”] RETURN n as Person

通过关键字using 显示指定使用的索引：

MATCH (n:Person)  
USING INDEX n:Person(name)  
WHERE n.name =’a’  
RETURN n as Person

**删除索引：**

DROP INDEX ON : Person (name)