Neo4j多级数据导入查询

# 删除历史的所有数据

MATCH(n) DETACH DELETE n

**案例一：**

演示流程：

1. **先添加节点**

CREATE (PRODUCT:TABLES {title:'PRODUCT', tagline:'once upon a time'})  
CREATE (PROFIT\_RATE\_89:TABLE1 {name:'PRODUCT', id\_:89,column\_name:'PROFIT\_RATE'})  
CREATE (PROFIT\_RATE\_75:TABLE2 {name:'TMP\_TABLE', id\_:75,column\_name:'PROFIT\_RATE'})  
CREATE (COST\_50:TABLE3 {name:'RESULT\_OF\_FHJ', id\_:50,column\_name:'COST'})  
CREATE (PROFIT\_49:TABLE3 {name:'RESULT\_OF\_FHJ', id\_:49,column\_name:'PROFIT'})  
CREATE (TOTAL\_COST\_41:TABLE4 {name:'FACT\_BILL', id\_:41,column\_name:'TOTAL\_COST'})  
CREATE (TOTAL\_COST\_38:TABLE4 {name:'FACT\_BILL', id\_:38,column\_name:'TOTAL\_COST'})  
CREATE (TURNOVER\_39:TABLE4 {name:'FACT\_BILL', id\_:39,column\_name:'TURNOVER'})

1. **添加关系**

CREATE  
(TOTAL\_COST\_38)-[r:FLOW]->(PROFIT\_49),  
(TURNOVER\_39)-[r:FLOW]->(PROFIT\_49),  
(PROFIT\_49)-[r:FLOW]->(PROFIT\_RATE\_75),  
(PROFIT\_RATE\_75)-[r:FLOW]->(PROFIT\_RATE\_89),  
(TOTAL\_COST\_41)-[r:FLOW]->(COST\_50),  
(COST\_50)-[r:FLOW]->(PROFIT\_RATE\_75),  
(PROFIT\_RATE\_89)-[r:FLOW]->(PRODUCT)

**4.x版本的数据**

Match (a:TABLE4) ,(b:TABLE3) WHERE a.id\_=38 and b.id\_=49 create (a)-[r: FLOW]->(b)

Match (a:TABLE4) ,(b:TABLE3) WHERE a.id\_=39 and b.id\_=49 create (a)-[r: FLOW]->(b)

Match (a:TABLE3) ,(b:TABLE2) WHERE a.id\_=49 and b.id\_=75 create (a)-[r: FLOW]->(b)

Match (a:TABLE2) ,(b:TABLE1) WHERE a.id\_=75 and b.id\_=89 create (a)-[r: FLOW]->(b)

Match (a:TABLE4) ,(b:TABLE3) WHERE a.id\_=41 and b.id\_=50 create (a)-[r: FLOW]->(b)

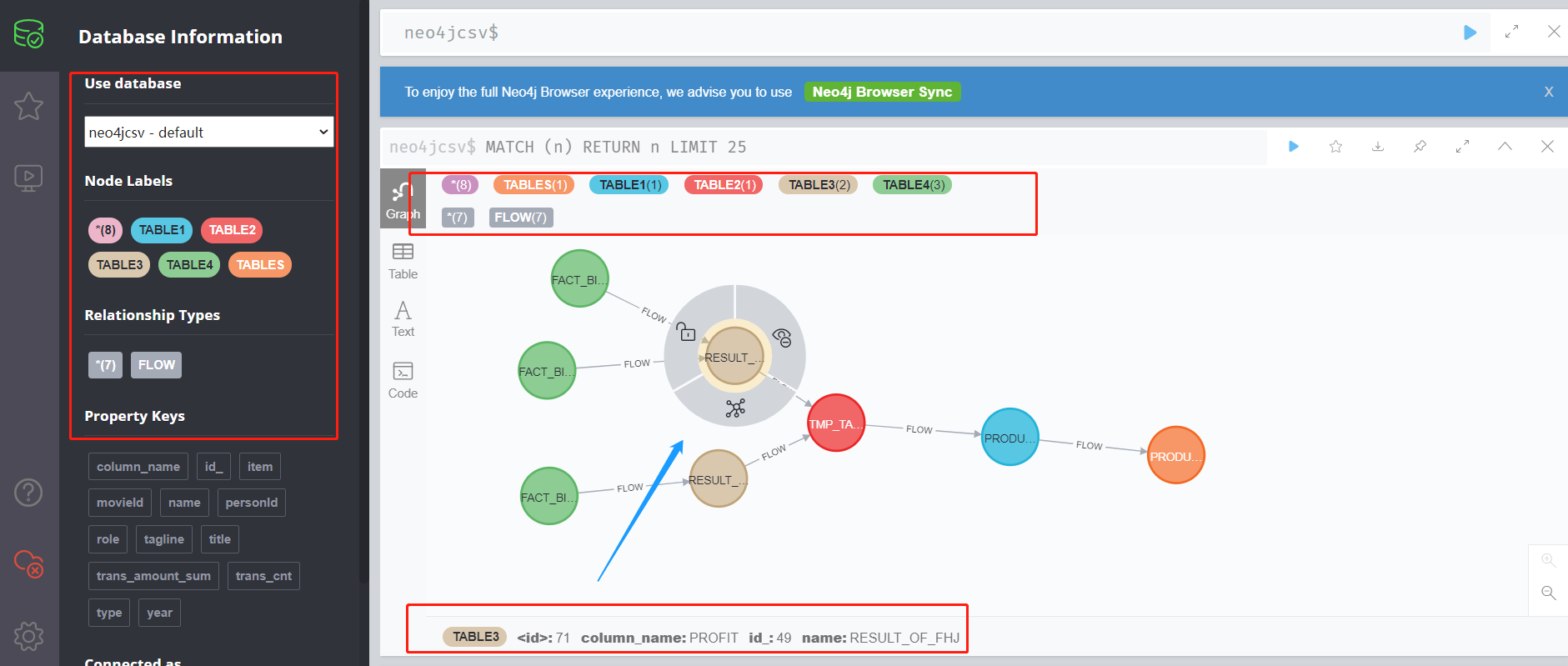
Match (a:TABLE3) ,(b:TABLE2) WHERE a.id\_=50 and b.id\_=75 create (a)-[r: FLOW]->(b)

Match (a:TABLE1) ,(b:TABLES) WHERE a.id\_=89 and b.title='PRODUCT' create (a)-[r: FLOW]->(b)

**3、建立好关系之后就进行相应的查询:**

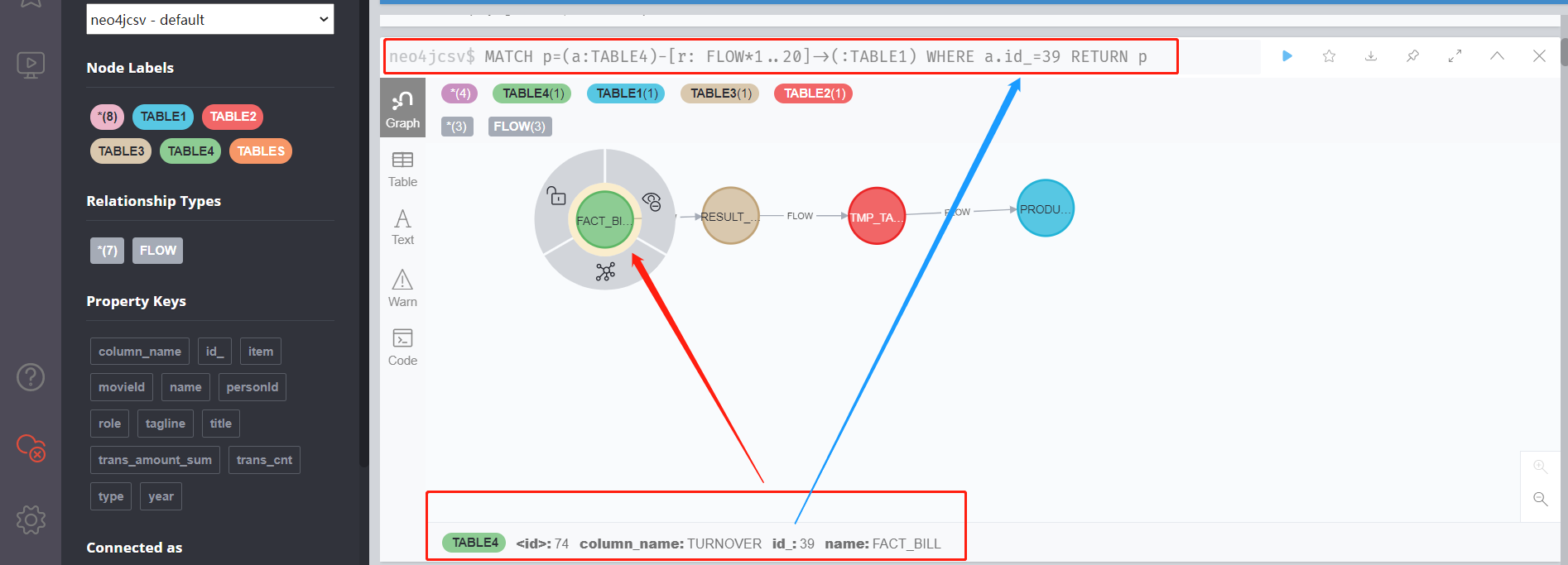
（1）整个关系的查询如下：

MATCH (a)-[r:FLOW]->(m) RETURN a,m



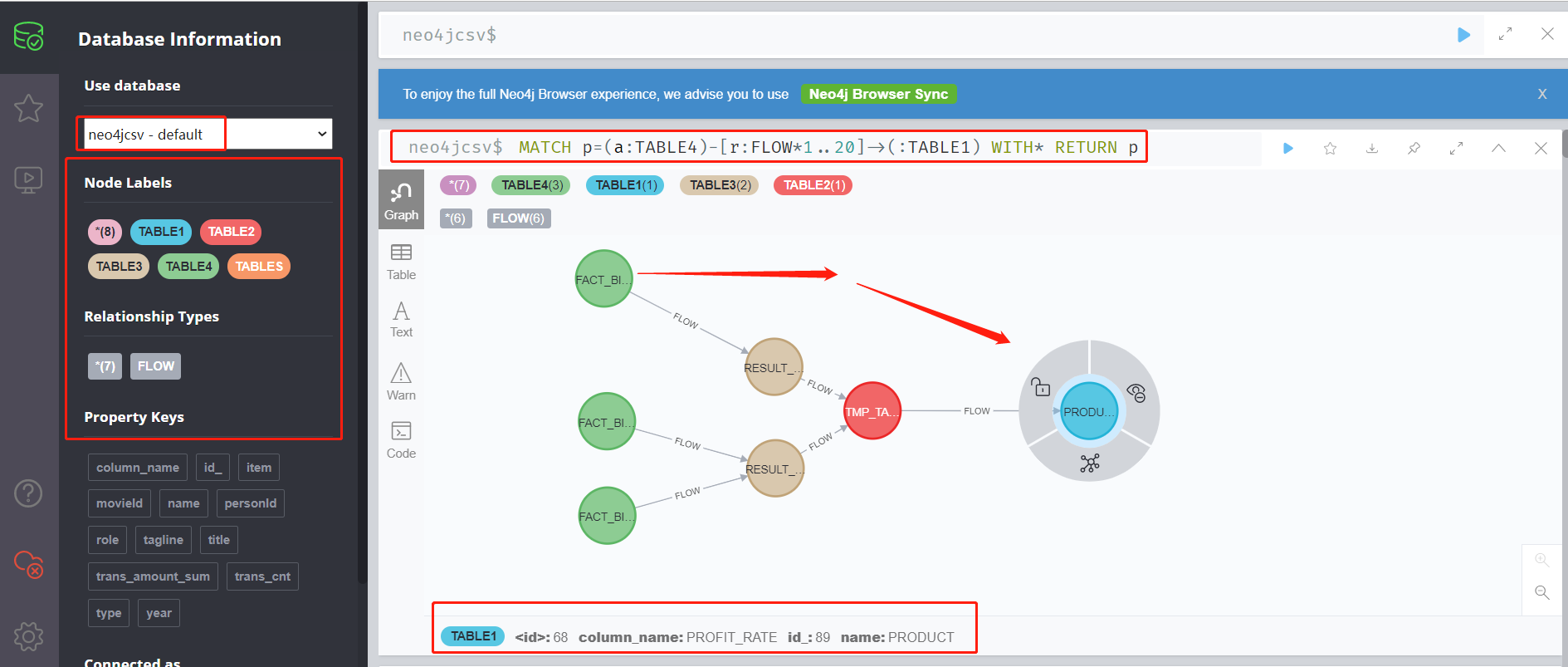
（2）查找两个节点之间的关系：查出来的是直连的关系，即本条路线上的关系

MATCH p=(a:TABLE4)-[r: FLOW\*1..20]->(:TABLE1) WHERE a.id\_=39 RETURN p



（3）两个node对象之间的关系（最多29层？）

MATCH p=(a:TABLE4)-[r:FLOW\*1..20]->(:TABLE1) WITH\* RETURN p



**案例二**

MATCH(n) DETACH DELETE n

1. 返回所有节点以及关系

MATCH p=()-[r:FLOW]->() RETURN p

1. 返回某一张表的依赖关系关系

MATCH p=(n:table)-[r: FLOW]->() where n.name = ‘name’ RETURN p

1. 返回所有跟某一张表有3层深度关系的关系

MATCH p=(n:table)-[r: FLOW\*..3]->(m:table) n.name = ‘name’ RETURN p

create (n:TABLE{name:"Math",SimchinaName:"数学表",id:1,Belongdb:"Train",type:"TABLE"});

create (n:TABLE{name:"English",SimchinaName:"英语表",id:2,Belongdb:"Train",type:"TABLE"});

create (n:TABLE{name:"Probability",SimchinaName:"概率论表",id:3,Belongdb:"Train",type:"TABLE"});

create (n:TABLE{name:"Computer",SimchinaName:"计算机科学表",id:4,Belongdb:"Test",type:"TABLE"});

create (n:TABLE{name:"AI",SimchinaName:"人工智能表",id:5,Belongdb:"Test",type:"TABLE"});

#一个简单的查询

#MATCH (n: table)

#RETURN n. name,n. SimchinaName

运行后进行查询如下：

建立好节点和关系之后，我们就可以进行查询的操作了：

1. **我们先查询一下表与表之间的血缘关系：**

MATCH (a:TABLE)-[r:TABLEFLOW]-(b:TABLE)

WHERE a.name="AI"

RETURN a,b

这与下述查询是等效的

MATCH (a:TABLE)-[r:TABLEFLOW]-(b:TABLE)

WHERE a.id = 5

RETURN a,b

这个查询我们是借属性来进行的。

1. **我们再次查询一下字段与字段之间的血缘关系：**

https://blog.csdn.net/zengbowengood/article/details/114904242?utm\_medium=distribute.pc\_relevant.none-task-blog-2~default~baidujs\_title~default-0.no\_search\_link&spm=1001.2101.3001.4242

https://guohongyuan.blog.csdn.net/article/details/115954121?utm\_medium=distribute.pc\_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7Edefault-2.no\_search\_link&depth\_1-utm\_source=distribute.pc\_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7Edefault-2.no\_search\_link

https://blog.csdn.net/ai\_1046067944/article/details/85342567?utm\_medium=distribute.pc\_relevant.none-task-blog-2~default~baidujs\_baidulandingword~default-4.no\_search\_link&spm=1001.2101.3001.4242