**实现mysql 和neo4j 数据库数据同步**

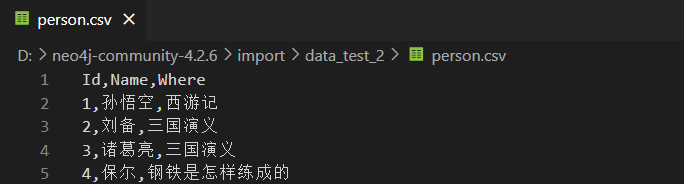
CSV方式

1、数据表简单：从数据库表MySQL中抽出节点表和关系表CSV导入

2、数据表多而杂 抽取MySQL节点关系数据，使用UNWIND字句配合导入过程进行同步 需要更新数据用：apoc.cypher.doIt，没有更新：apoc.merge.node/relationships

**情景一、仅仅导入节点数据**

我们导入person.scv中的数据，数据大概如下：

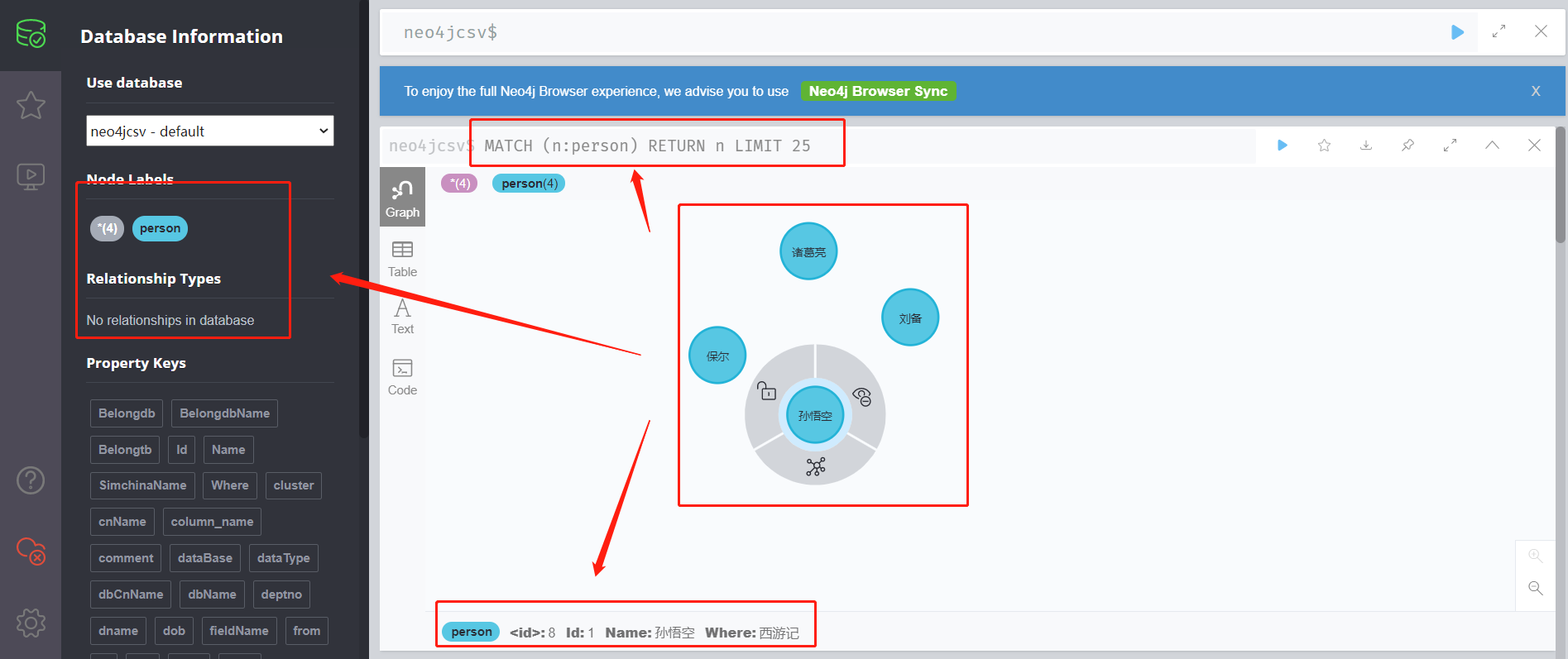


接着我们使用Cypher来导入数据，导入数据的时候我们设置了一些标签：

:auto USING PERIODIC COMMIT 500 LOAD CSV WITH HEADERS FROM "file:///data\_test\_2/person.csv" AS line

MERGE (p:person{Id:line.Id,Name:line.Name,Where:line.Where})

查看数据：

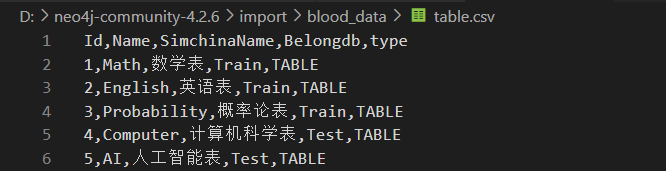


使用MERG较CREATE创建可以忽略重复的数据（再一次运行上述语句将不在有变化）

**情景二、以CSV方式导入节点和关系数据**

**2.1 table.csv(表)文件导入：**

假设我们的table.csv文件的内容如下：



我们将上述文件table.csv导入neo4j的cypher语句为：

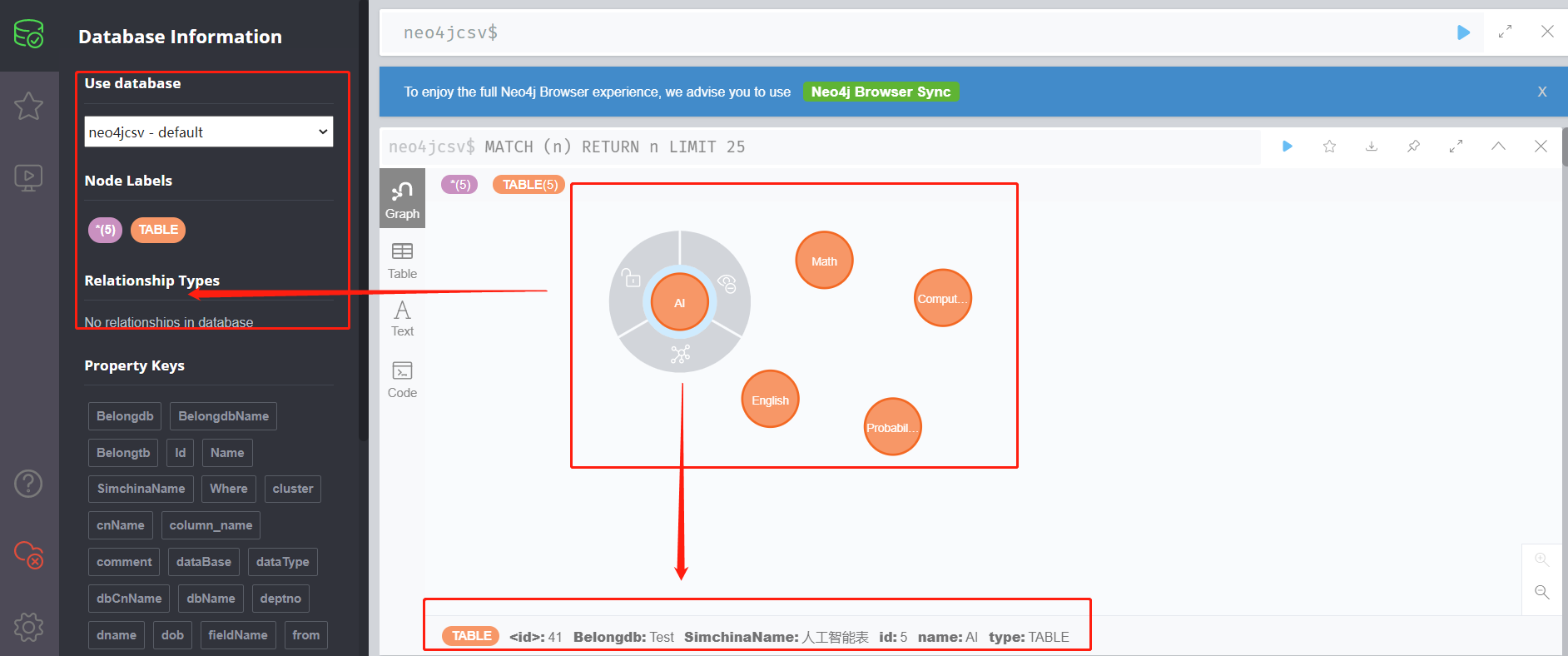
:auto USING PERIODIC COMMIT 500 LOAD CSV WITH HEADERS

FROM "file:///blood\_data/table.csv" AS line

MERGE(t:TABLE{id:line.Id,name:line.Name,SimchinaName:line.SimchinaName,

Belongdb:line.Belongdb,type:line.type})

导入后，我们查询如下：



**2.2 colunm.csv(字段)文件导入：**

**假设column.csv文件的内容大致如下：**



我们将上述文件(字段表)column.csv导入neo4j的cypher语句为：

:auto USING PERIODIC COMMIT 500 LOAD CSV WITH HEADERS

FROM "file:///blood\_data/column.csv" AS line

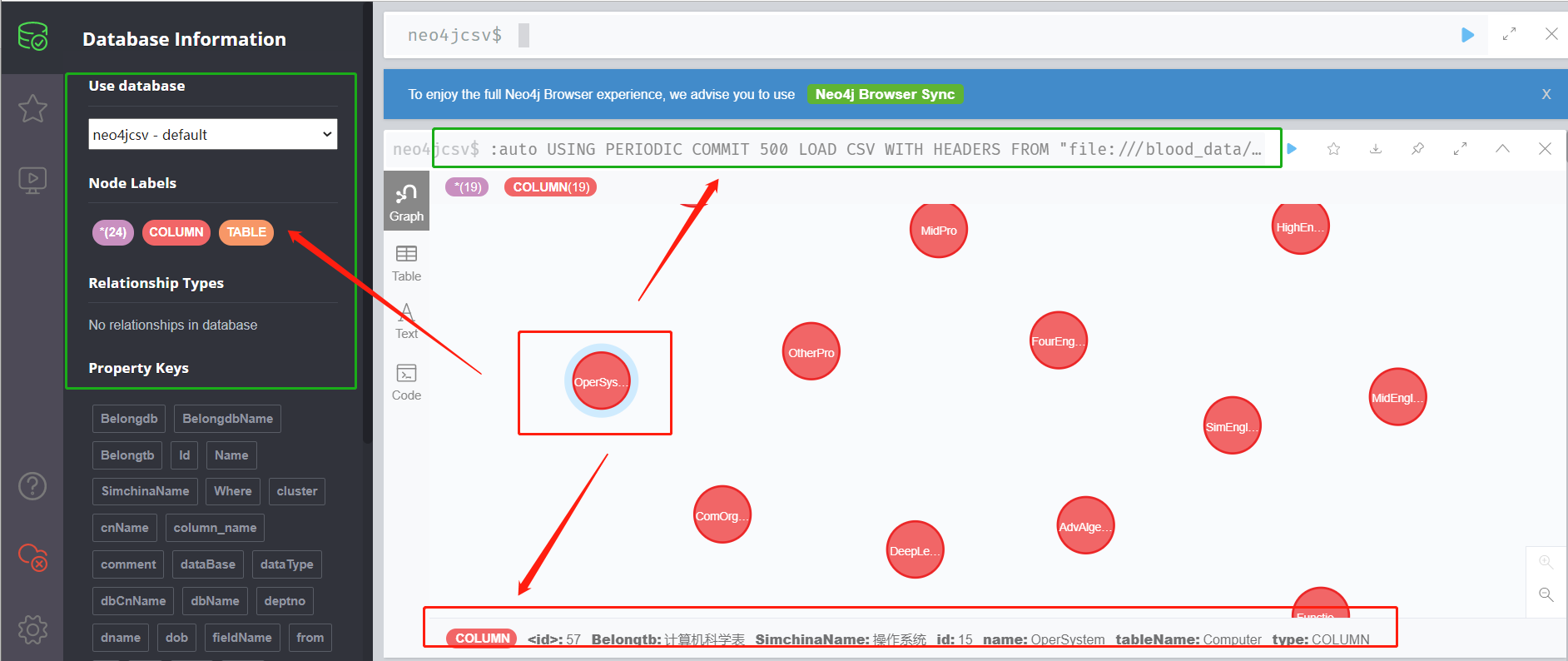
MERGE(c:COLUMN{id:line.Id,name:line.Name,SimchinaName:line.SimchinaName,

tableName:line.tableName,

Belongtb:line.Belongtb,type:line.type}) return c

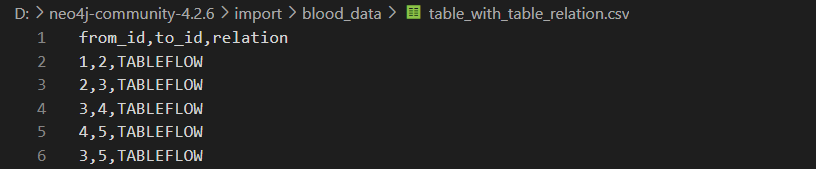
我们使用return c语句返回结果，这样就不需要进行查询了：

MATCH (n:COLUMN) RETURN n LIMIT 25



**2.3 表与表关系文件(csv)导入**

**假设table\_with\_table\_relation.csv文件的内容大致如下：**



我们将上述文件(表与表的关系)table\_with\_table\_relation.csv导入neo4j的cypher语句为：

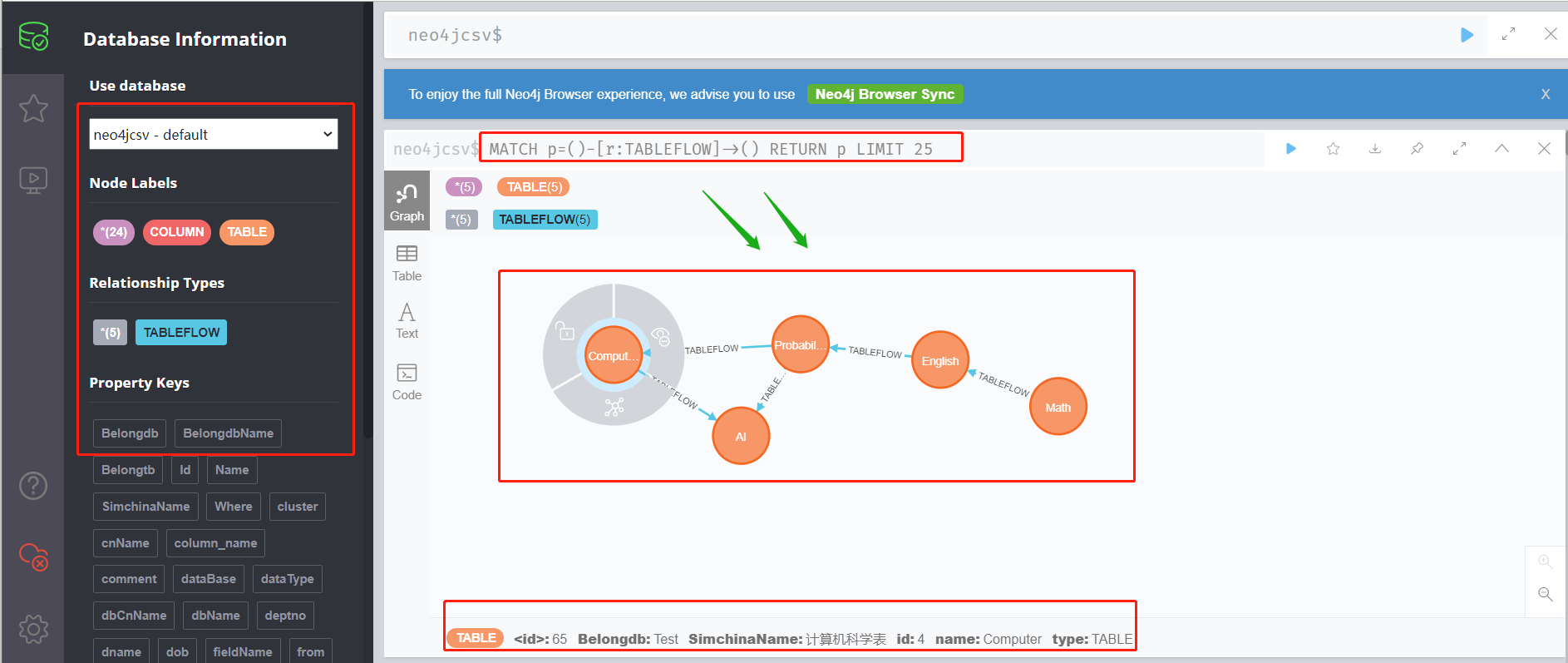
:auto USING PERIODIC COMMIT 500 LOAD CSV WITH HEADERS

FROM "file:///blood\_data/table\_with\_table\_relation.csv" AS line

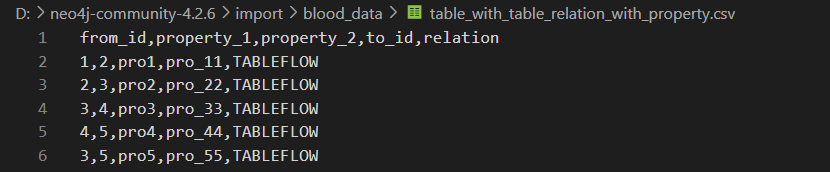
match (from:TABLE{id:line.from\_id}),(to:TABLE{id:line.to\_id})

merge (from)-[r:TABLEFLOW]->(to)

我们查询一下结果：



当我们表与表的关系存在属性的时候，即内容为下图所示的时候，我们的导入语句相应的调整即可：



:auto USING PERIODIC COMMIT 500 LOAD CSV WITH HEADERS

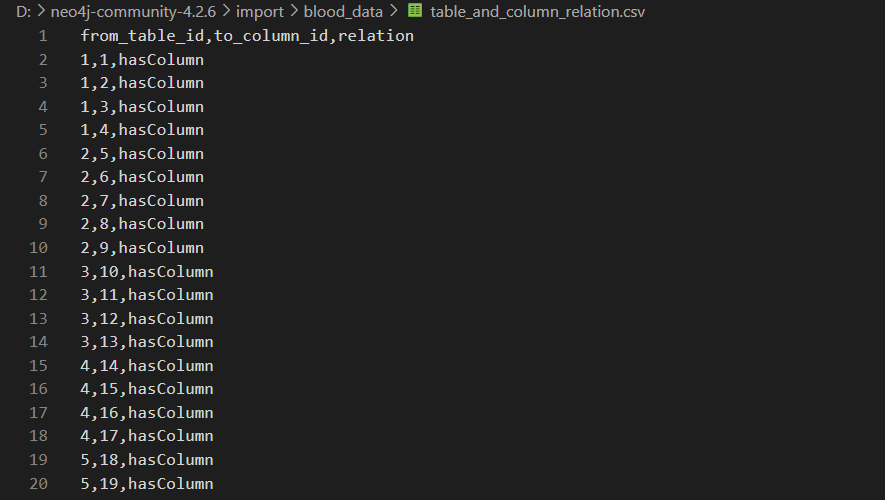
FROM "file:///blood\_data/table\_with\_table\_relation.csv" AS line

match (from:TABLE{id:line.from\_id}),(to:TABLE{id:line.to\_id})

merge (from)-[r:TABLEFLOW{property1:line.property\_1,property2:line.property\_2}]->(to)

**2.4 表与字段的关系文件(csv)导入**

**假设table\_and\_column\_relation.csv文件的内容大致如下：**



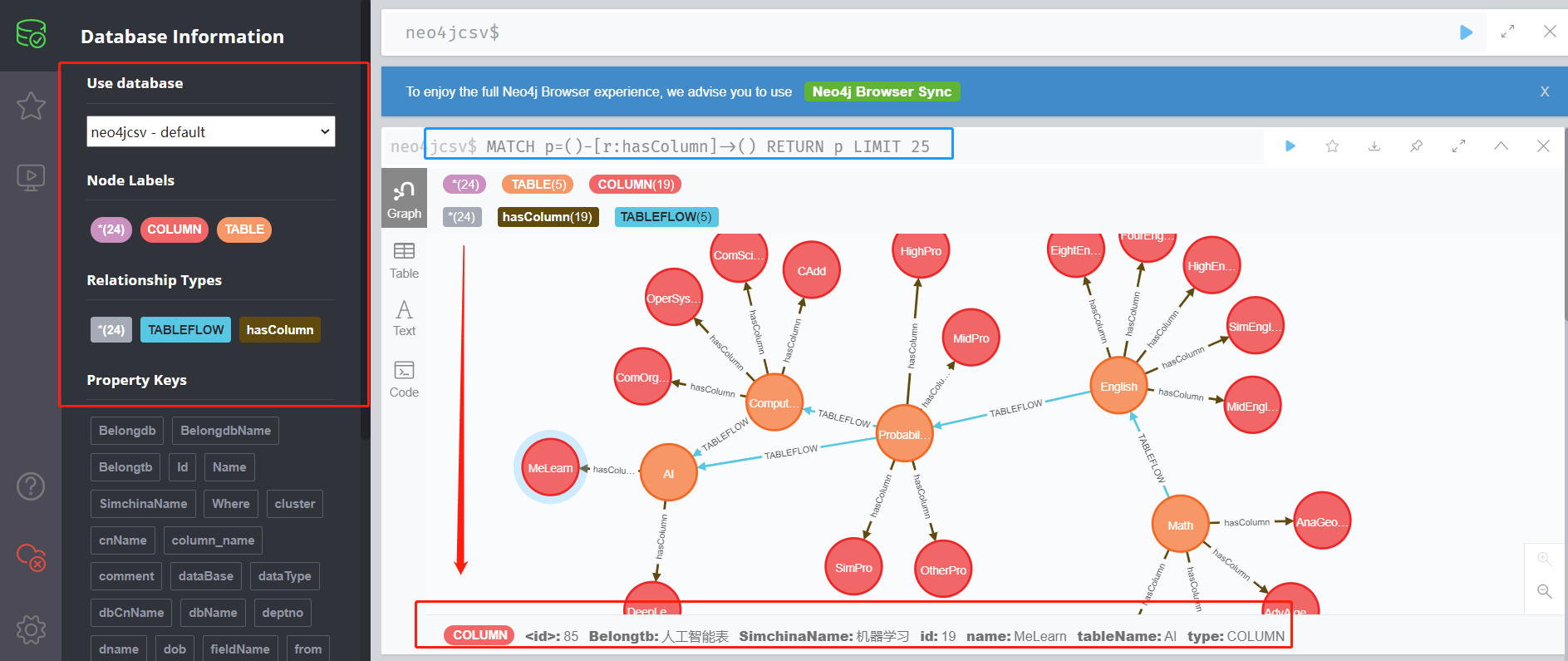
:auto USING PERIODIC COMMIT 500 LOAD CSV WITH HEADERS

FROM "file:///blood\_data/table\_and\_column\_relation.csv" AS line

match (from:TABLE{id:line.from\_table\_id}),(to:COLUMN {id:line.to\_column\_id})

merge (from)-[r:hasColumn]->(to)

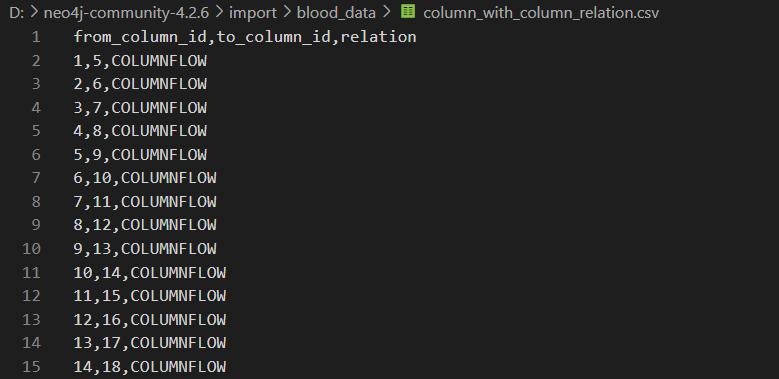
运行后我们查看一下结果，可以看出已经建立了表和字段之间的关系：



**2.5 字段与字段关系的导入**

最后我们导入字段与字段之间的关系

**假设column\_with\_column\_relation.csv的内容如下：**



在neo4j的web管理界面键入以下语句：

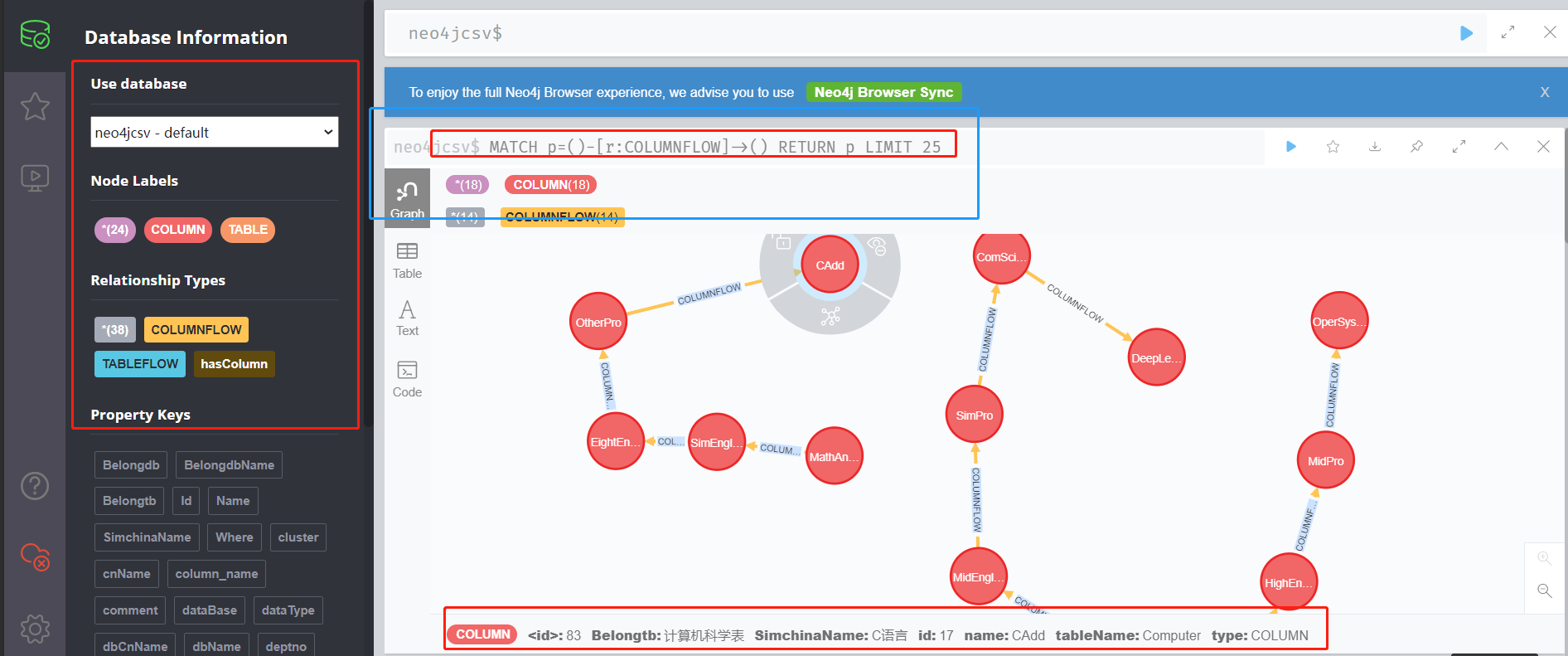
:auto USING PERIODIC COMMIT 500 LOAD CSV WITH HEADERS

FROM "file:///blood\_data/column\_with\_column\_relation.csv" AS line

match (from:COLUMN{id:line.from\_column\_id}),(to:COLUMN {id:line.to\_column\_id})

merge (from)-[r:COLUMNFLOW]->(to)

MATCH p=()-[r:COLUMNFLOW]->() RETURN p LIMIT 25



可以看出我们的表和字段的关系就完了，之后就可以进行相应的查询操作。

**情景三、查询等操作**

**备份数据**

在neo4j-community-3.4.11/bin路径下执行neo4j-admin命令：

1. ./neo4j-admin dump --database=graph.db(数据库名称，默认是这个) --to=xxx/xxxx.dump

MATCH (n) OPTIONAL MATCH (n)-[r]-() DELETE n,r

// CSV文件压缩为ZIP之后进行导入

using periodic commit 1000

load csv with headers from "file:///studentBatch.zip" as line with line

merge (n:Linkin {name:line.LinkedinID}) with \*

merge (m:学校 {name:line.学校}) with \*

merge (n)-[r:教育经历]->(m)

3333

3333

rm -rf data/databases/graph.db/;

./bin/neo4j-admin import --nodes /home/watson/crawl/company\_info/qichacha.com/nodes\_script/nodes.csv --relationships /home/watson/crawl/company\_info/qichacha.com/nodes\_script/relationships.csv --ignore-duplicate-nodes --ignore-extra-columns true;

./bin/neo4j restart

https://blog.csdn.net/Appleyk/article/details/80332911