How to use Neo4j to build and search a graph of knowledge points in JAVA

Author: Yunzhe Li @Shanghai University

Something to make clear first

Classes in this graph:

Four basic classes I suppose, namely *Course*, *Chapter*, *Content* and *Knowledge*.

Chart 1 Brief description of node labels

Node Labels	Description	Property	Examples
Course	The course in this project is JAVA, can be somehow replace by other courses.	Name	Java
		Description	/
Chapter	Main parts of the course.	Name	类和对象
		Description	/
Content	Some sub-divided parts of the content.	Name	类
		Feature	Key / common
		Description	/
Knowledge	Detailed points of content.	Name	类声明
		Feature	Key / Common
		Description	/

Chart 2 Brief description of relationship types

Relationship Types	Label(source)	Label(dest)	Property	Example
Chapter_Of	**Chapter**	**Course**	/	/
Content_Of	**Content**	**Chapter**	/	/
Knowledge_Of	**Knowledge**	**Content**	Name	第 xx 章 知识点 x.x.x

Commands to present the graph

To show all chapters of the **Course**:

MATCH (n:Chapter) return n

To show all content of one specific **Chapter**:

MATCH (n:Content)--(m:Chapter {name:"类和对象"}) return n.name

To match key knowledge of one specific **Content**:

MATCH (Content {name:'类'})<-[r:Knowledge_Of {feature:'key'}]-(k) return k.name

Manual

Manual hyper-link: [Clauses - Neo4j Cypher Manual]

Quick Create

Create multiple nodes and relations

```
MATCH (n {name:"类和对象"})
CREATE (section:Content {name:"访问权限", section:"12"})
CREATE (section)-[:Content_Of]->(n)

CREATE (sub1:Knowledge {name:"私有变量和私有方法", sub_section:"1"})
CREATE (sub2:Knowledge {name:"公有变量和公有方法", sub_section:"2"})
CREATE (sub3:Knowledge {name:"友好变量和友好方法", sub_section:"3"})

CREATE (sub1)-[:Knowledge_Of]->(section)
CREATE (sub2)-[:Knowledge_Of]->(section)
CREATE (sub3)-[:Knowledge_Of]->(section)
```

CREATE

Create node and add labels and properties

```
CREATE   (n:Class_name {name: 'xxx', title: 'xxx'})
```

Create node with relationship

```
CREATE   (JavaSwing:Content {name: 'JavaSwing', …..})
CREATE   (method_1:Knowledge {name: 'method_1, …..})
CREATE   (method_1)-[:Knowledge_Of {features:['key']}]->(JavaSwing)
```

Create relationships from existing nodes

```
MATCH
  (a:Content),
  (b:Chapter)
where a.name='构造方法与对象的创建' and b.name='类和对象'
CREATE (a)-[r:Content_Of]->(b)
```

SET

Set properties of any node. Match first, then SET.

```
MATCH   (n {name:"编程语言的几个发展阶段"})
SET n.section = 1
```

Match

Quick match

```
MATCH (n {name:’xxx’})
Return n
```

Match all key knowledge in the graph.

```
MATCH (Content) <- [:Knowledge_Of {feature:'key'}] - (k)
RETURN k.name
```

Match key knowledge of specific content

```
match (Content {name:'类'})<-[r:Knowledge_Of {feature:'key'}]-(k)
return k.name
```

Match knowledge of one specific content

```
match (Content {name:'类'})<-[:Knowledge_Of]-(k)
return k.name
```

Delete

Delete relationships

```
MATCH (n {name: 'Andy'})-[r:KNOWS]->()
DELETE r
```

Delete nodes and relationships with it

```
MATCH (n {name: 'Andy'})-[r:KNOWS]->(x)
DELETE r
DELETE x
```

Delete all

```
MATCH (n)
DETACH DELETE n
```

Remove

Remove one property

```
MATCH (a {name: 'Andy'})
REMOVE a.age
RETURN a.name, a.age
```

Export graph with APOC

Why APOC?

To gather all related data from our collaborators, apoc is a powerful tool for us to deal with this problem.

How to use?

My neo4j-community version is 4.1.12, and apoc-4.1.0.0-all.jar is need to download from the official website. Move it into /plugins directory. Revise the .conf file in /conf directory (for what to revise, just google or baidu).

To export data, just command `CALL apoc.export.csv.all("xxx.csv", {})`, and you will get xxx.csv in your /import directory where you install your neo4j.

Gather data

Re_id is necessary to create relationships, different node labels should be imported respectively.

Appendix

Lyz's importing code

```
CALL apoc.load.csv("./36/content_36.csv")
YIELD lineNo, map, list
CREATE (:Content{name:list[4], re_id:list[0], description:list[2], feature:list[3]})

CALL apoc.load.csv("./36/knowledge_36.csv")
YIELD lineNo, map, list
CREATE (:Knowledge{name:list[4], re_id:list[0], description:list[2], feature:list[3]})

CALL apoc.load.csv("./36/chapter_36.csv")
YIELD lineNo, map, list
CREATE (:Chapter {name:list[4], re_id:list[0]})

CALL apoc.load.csv("./36/re_36.csv")
YIELD lineNo, map, list
match
    (n:Knowledge),
    (m:Content)
where n.re_id = list[0] and m.re_id = list[1]
CREATE (n)-[:Knowledge_Of {name:list[3]}]->(m)

CALL apoc.load.csv("./36/re_36.csv")
YIELD lineNo, map, list
match
    (n:Content),
    (m:Chapter)
where n.re_id = list[0] and m.re_id = list[1]
CREATE (n)-[:Content_Of {name:list[3]}]->(m)
```

Sty's importing code

```
CALL apoc.load.csv("./147/content_147.csv")
YIELD lineNo, map, list
CREATE (:Content{name:list[4], re_id:list[9], description:list[2],   feature:list[3]})


CALL apoc.load.csv("./147/chapter_147.csv")
YIELD lineNo, map, list
CREATE (:Chapter {name:list[4], re_id:list[9], description:list[2]})

CALL apoc.load.csv("./147/knowledge_147.csv")
YIELD lineNo, map, list
CREATE (:Knowledge{name:list[4], re_id:list[9], description:list[2],   feature:list[3]})
```

```
CALL apoc.load.csv("./147/re_147.csv")
YIELD lineNo, map, list
match
   (n:Content),
   (m:Chapter)
where n.re_id = list[5] and m.re_id = list[6]
CREATE (n)-[:Content_Of{name:list[3]}]->(m)

CALL apoc.load.csv("./147/re_147.csv")
YIELD lineNo, map, list
match
   (n:Knowledge),
   (m:Content)
where n.re_id = list[5] and m.re_id = list[6]
CREATE (n)-[:Knowledge_Of {name:list[3]}]->(m)
```

lzy's importing code

```
CALL apoc.load.csv("./258/content_258.csv")
YIELD lineNo, map, list
CREATE (:Content{name:list[2], re_id:list[0], description:list[3],   feature:list[4]})


CALL apoc.load.csv("./258/chapter_258.csv")
YIELD lineNo, map, list
CREATE (:Chapter {name:list[2], re_id:list[0]})

CALL apoc.load.csv("./258/knowledge_258.csv")
YIELD lineNo, map, list
CREATE (:Knowledge{name:list[2], re_id:list[0], description:list[3],   feature:list[4]})


CALL apoc.load.csv("./258/re_258.csv")
YIELD lineNo, map, list
match
   (n:Content),
   (m:Chapter)
where n.re_id = list[0] and m.re_id = list[1]
CREATE (n)-[:Content_Of{name:list[3]}]->(m)

CALL apoc.load.csv("./258/re_258.csv")
YIELD lineNo, map, list
match
```

```
    (n:Knowledge),
    (m:Content)
where n.re_id = list[0] and m.re_id = list[1]
CREATE (n)-[:Knowledge_Of {name:list[3]}]->(m)
```