THE UNIVERSITY OF SYDNEY

School of Computer Science
Dr. Ying Zhou

COMP5338: Advanced Data Models **2.Sem./2020**

Week 6: Neo4j tutorial

29.09.2020

Learning Objectives

In this week, we will go through the build-in tutorials of Neo4j Community edition to learn basic components of Cypher query, such as patterns, clauses and sub-clauses. We assume you have Neo4j installed on your own computer and have properly configure the APOC library as plugin following the Neo4j setup guide released before the tutorial.

Question 1: Neo4j and Cypher query language basics

Start Neo4j server as described in the set up guide. Open a browser window and type `http://localhost:7474/` in the address bar to connect to the neo4j web interface of the local server.

At the very top of the browser window is a editor box for entering commands. All commands that have been executed along with their results will be displayed on the in their own frames. You can easily discard any unwanted history command by removing the frame. Detailed instruction on various components and shortcuts of neo4j browser window can be find from this official manual.

Figure 1 shows an example of browser window after connecting to the server. It shows three quick guides each with a clickable link.

- **Learn about Neo4j** introduces basic Neo4j concepts and Cypher query languge.
- **Jump into code** contains two comprehensive tutorials: **The Movie Graph** and **The Northwind Database**. **The Movie Graph** contains commands to set up a mini graph model of movies and actors; it also contains several query use-cases, including simple query, aggregation and shortest path queries; **The Northwind Database** demonstrates how to import data from CSV files representing nodes and how to build relationships using Cypher commands.
- **Monitor the System** shows basic system statistics

You are asked to follow the **movie graph tutorials** to get familiar with the graph database concept and the various constructs of Cypher query language. The database is needed for next question, so do not remove it after you complete the tutorial.

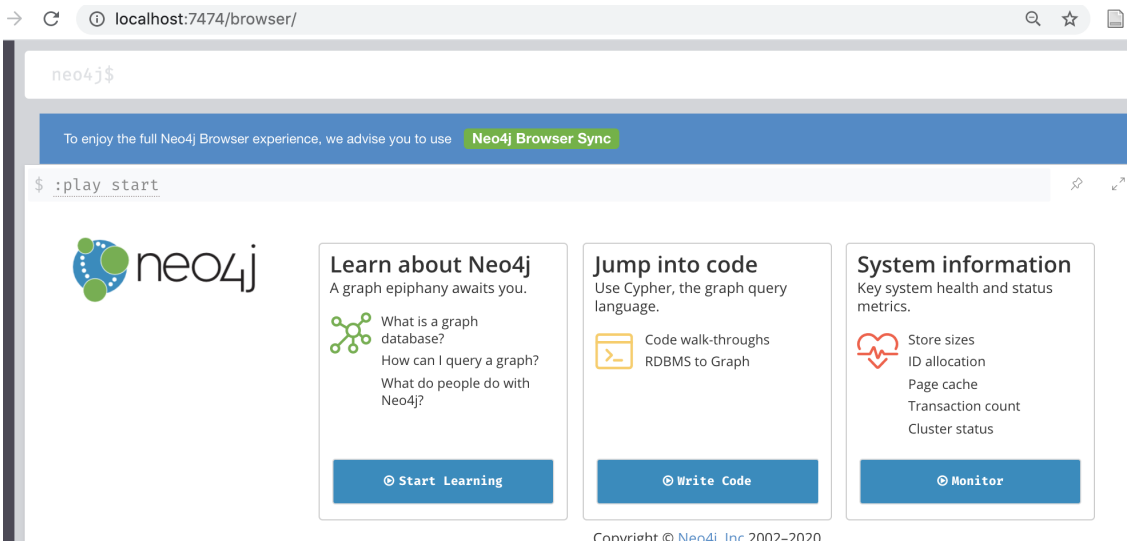

Figure 1: Neo4j Web Browser

Question 2: Write Your Own Queries

Now try to answer the following questions using Movie Graph:

- Find out how many movies Tom Hanks acts in.
- Find out how many actors in the movie 'Cloud Atlas'.
- Find out the average rate of the movie 'The Replacements'.
- Find out the average age of actors in the movie 'Cloud Atlas'.
- Find out which movie has the highest average rate.
- Find out the top 3 movies based on the number of rating it received.

Question 3: Loading JSON Data

CSV files can be loaded directly using Cypher command. The usage is demonstrated in the **The Northwind Database** tutorial. Loading data in other format needs to use procedures defined in APOC library.

A number of procedures can be used to load JSON files from different data sources. The complete list can be found from APOC manual on loading JSON. This exercise extends the simple example on the manual.

Neo4j server supports multiple databases in enterprise version. The community edition can only use the default database. You need to clear the Movies database to be able to see nodes and relationships created in this step.

Two files are used: `person.json` and `persons.json`. They can be downloaded from canvas. Once downloaded, save them in the `import` sub-folder of your Neo4j's home directory. This is the folder where you should put files to be imported into Neo4j.

The `person.json` file contains a single json document representing one person with his children's names saved in an array. The `persons.json` contains a list of json documents each representing a person, the documents have slightly different structure with certain fields like `children` missing in some documents.

Follow the APOC instruction to inspect the raw content being loaded using the following command:

```
CALL apoc.load.json("file:///person.json")
YIELD value
RETURN value;
```

Now create a graph from the json document using the following command

```
CALL apoc.load.json("file:///person.json")
YIELD value
MERGE (p:Person {name: value.name})
SET p.age = value.age
WITH p, value
UNWIND value.children AS child
MERGE (c:Person {name: child})
MERGE (c)-[:CHILD_OF]->(p);
```

If importing is successful, you should see a screen similar to Figure 2. You can also run the following query to visualize the graph.

```
MATCH (p:Person)
RETURN p;
```

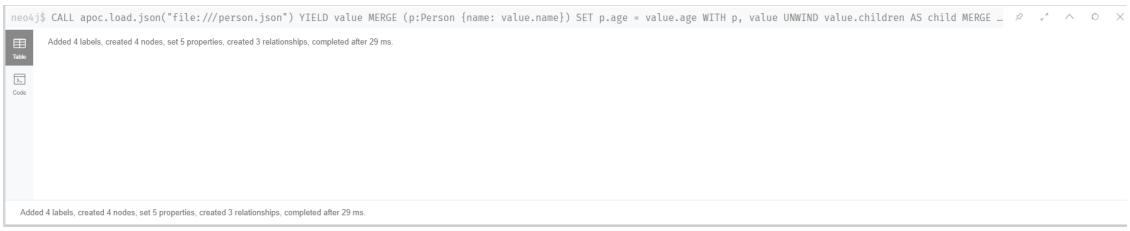

Figure 2: Import results for person.json

Run a similar command on `persons.json` to augment the graph with more details on some nodes and also a few additional nodes:

```
CALL apoc.load.json("file:///persons.json")
YIELD value
MERGE (p:Person {name: value.name})
SET p.age = value.age
```

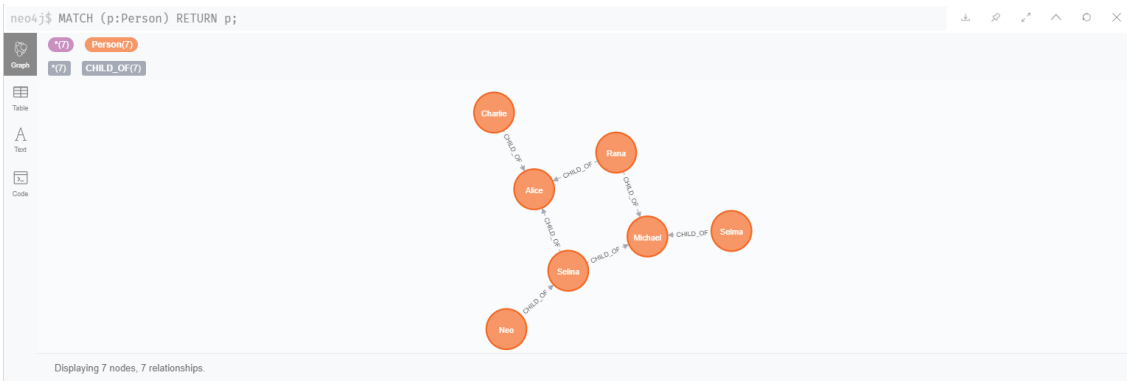

Figure 3: Visualization of Person

```
WITH p, value
UNWIND value.children AS child
MERGE (c:Person {name: child})
MERGE (c)-[:CHILD_OF]->(p);
```

This example also illustrates that MERGE can perform either a CREATE or a MATCH/SET. The resulting graph is shown in Figure 3.