Graph Databases: Neo4j

# Installation

Download the tarball and extract it.

$ wget -O neo4j.tar.gz https://neo4j.com/artifact.php?name=neo4j-community-2.3.12-unix.tar.gz  
$ tar -xvzf neo4j.tar.gz

Start the server:

$ cd neo4j-community-2.3.12

$ bin/neo4j console

View the Neo4J server on following default URL:

http://localhost:7474

Proceed with neo4j / neo4j username and password. Change the password on the next step.

# Basics

We’ll be taking advantage of built-in learning materials. Go through Graph Fundamentals by clicking “Start learning” button. After you finish, type these commands in console and read another documentation parts:

$ :play intro

$ :play cypher

Navigate to the “Movie Graph” section:

$ :play movie graph

We’ll be working through the examples provided in this section for working with a graph of data relations between actors, directors and movies, with some additional tasks.

*N. B. Next paragraph names are following names of the slides in “The Movie Graph” section, go ahead with reading the lab first and do not switch to other slide until you are done with respective lab paragraph. Press “>” button and find “Create” slide.*

# Create

The first step is to create the movie graph database itself. For this part of the lab we will be filling the data using a long single Cypher query statement composed of many CREATE statements. As prompted on the browser, click on the code block and it will be copied into the editor window along the top. Execute it with “Play” button to create the database. Last step of the query displays information about 10 movies in which Tom Hanks acted and directors of those movies.

Note following things in the result frame:

* multiple modes for displaying the information in the left part of the frame
* export button in the upper right part of the frame.
* a person nodes can have properties like “id”(built-in ID), “name”, “born”(year of birth).

# Find

After the data is available, you can work further through provided examples or perform your own experiments for better understanding. Click the “>” button and follow the code prompts to query for individual node data. Work through the four provided Cypher examples.

Paste here the query that returns name and year of birth of one person of your choosing.

MATCH (keanu {name: "Keanu Reeves"})

MATCH (keanu {born: 1964})

RETURN keanu

Paste here the exported as CSV results of the query executed.

keanu"{""born"":1964,""name"":""Keanu Reeves""}"

# Query

The MATCH statement is used to find patterns within the graph by relating different pieces of the data by a specified parameter. If no data for the relationship exists the MATCH statement will create the relation. Go through the prompts to return all Tom Hanks movies and various relations between Tom Hanks’ co-actors and his movies.

Now, execute this query to list names of all the actors in our database:

$ MATCH (p:Person) WHERE (p)-[:ACTED\_IN]->() RETURN p.name

Next, select one of them and create your own query to display following information about him/her: what other younger actors (later year of birth) starred together with that actor and what were the movies?

Copy here the query you used.

MATCH (Demi:Person {name:"Demi Moore"})-[:ACTED\_IN]->(m)<-[:ACTED\_IN]-(coActors) WHERE Demi.born > coActors.born RETURN coActors.name, m.title

# Solve

A Cypher statement can specify the number of nodes to ‘hop’ in returning data. The returned describes the actual nodes that are of a specified distance away from the starting node. The basic syntax is:

–[\*minHops..maxHops]

where minHops and maxHops are optional and minHops will default to 1.

The shortestPath command will find a single shortest path between two nodes. You need to define the starting node, the connecting relationship and the end node. The common syntax is:

p = shortestPath((start node) – [\*..10] – (end node))

where 10 is the maximum number of hops the path can take.

# Recommend

Using the relationships between nodes and neighboring nodes, Neo4j can be used to recommend new actions that would likely be favorable. Neo does this by finding connections outside of the immediate region of connected data which are connected to each other. Go through both examples to find out how to recommend new actors for Tom Hanks to work with.

Notice as the Cypher commands become more complicated the results take longer to be returned. Use the PROFILE command before a query statement to determine the path cypher takes to execute a given query. This can be used to compare different queries and determine which query takes the minimum number of steps the query takes to traverse the graph. The amount of work for a query execution step is measured in abstract units – db hits.

$ profile MATCH (tom:Person {name: "Tom Hanks"})-[:ACTED\_IN] ->(tomHanksMovies) RETURN tom, tomHanksMovies

# Delete

After the steps have been completed the movie data set can be deleted from the Neo4j server. The DELETE command is used to delete nodes and relationships that are identified within the MATCH clause. The data to be deleted can be further qualified by a WHERE statement. Following example will remove all nodes related to Tom Hanks:

$ MATCH (n) – [r] – () WHERE n.name = ‘Tom Hanks’ delete r

Go through the two code samples to delete the movie data and then use the RETURN command after to check that the data was truly deleted.

