## HU Extension Assignment 02 E63 Big Data Analytics

### Handed out: 02/03/2017 Due by 11:59PM EST on Friday, 02/10/2017

Please, describe every step of your work and present all intermediate and final results in a Word document. Please, copy past text version of all essential command and snippets of results into the Word document with explanations of the purpose of those commands. We cannot retype text that is in JPG images. Please, always submit a separate copy of the original, working scripts and/or class files you used. Sometimes we need to run your code and retyping is too costly. Please include in your MS Word document only relevant portions of the console output or output files. Sometime either console output or the result file is too long and including it into the MS Word document makes that document too hard to read. PLEASE DO NOT EMBED files into your MS Word document. For issues and comments visit the class Discussion Board on Piazza.

You can do this assignment in Python, Java, R, Scala or any other language of your convenience.

I will proceed with my exercise using Python 2.7 and curl for Windows. All my comments and code will be in blue color to make a difference with the original document code. It is stated in Piazza to add to this document snippets of code, not the complete code, but my scripts are not that large, and seem quite easy to go through them, so I have included the complete source code. In same way it is mentioned in Piazza not to upload CSV files resulting from the database export, so I would proceed according to it.

I upload together with this document complete source code for my python scripts and a bat file for windows with the curl sentence required for Problem 1. I have renamed the file from .bat to .txt.

As database I am using a local instance with user neo4j and password neo4j2.

I hope this will be enough detail to clarify what submitted as part of mi Homework 2.

**Problem 1.** The following is the content ofMovies database.Bring that database into Neo4J using curl.

CREATE (matrix1:Movie { title : 'The Matrix', year : '1999-03-31' }) return id(matrix1)

CREATE (matrix2:Movie { title : 'The Matrix Reloaded', year : '2003-05-07' }) return id(matrix2)

CREATE (matrix3:Movie { title : 'The Matrix Revolutions', year : '2003-10-27' }) return id(matrix3)

CREATE (keanu:Actor { name:'Keanu Reeves' }) return id(Keanu)

CREATE (laurence:Actor { name:'Laurence Fishburne' })

CREATE (carrieanne:Actor { name:'Carrie-Anne Moss' })

CREATE (keanu)-[:ACTS\_IN { role : 'Neo' }]->(matrix1)

CREATE (keanu)-[:ACTS\_IN { role : 'Neo' }]->(matrix2)

CREATE (keanu)-[:ACTS\_IN { role : 'Neo' }]->(matrix3)

CREATE (laurence)-[:ACTS\_IN { role : 'Morpheus' }]->(matrix1)

CREATE (laurence)-[:ACTS\_IN { role : 'Morpheus' }]->(matrix2)

CREATE (laurence)-[:ACTS\_IN { role : 'Morpheus' }]->(matrix3)

CREATE (carrieanne)-[:ACTS\_IN { role : 'Trinity' }]->(matrix1)

CREATE (carrieanne)-[:ACTS\_IN { role : 'Trinity' }]->(matrix2)

CREATE (carrieanne)-[:ACTS\_IN { role : 'Trinity' }]->(matrix3)

**Explanation:**

This first problem can be accomplished executing a curl sentence from Windows system prompt. Note that like it is mandatory to change neo4j password, in my case I changed it to neo4j2. Afterwards I run the below curl sentence that included all the nodes, labels and relationships as per the above list formatted as JSON. Neo4j instance in hosted in my localhost.

As a side note, I have seen in the slides, and also because it’s a normal practice in database design, that my creation sentence should have a unique ID, but like the problem does not included them I have created the JSON file as it is.

**Code Snippet:**

|  |
| --- |
| curl -i -H accept:application/json -H content-type:application/json -XPOST http://localhost:7474/db/data/transaction/commit -d '{"statements":[{"statement":"CREATE (matrix1:Movie { title : \"The Matrix\", year : \"1999-03-31\" }) CREATE (matrix2:Movie { title : \"The Matrix Reloaded\", year : \"2003-05-07\" }) CREATE (matrix3:Movie { title : \"The Matrix Revolutions\", year : \"2003-10-27\" }) CREATE (keanu:Actor { name:\"Keanu Reeves\" }) CREATE (laurence:Actor { name:\"Laurence Fishburne\" }) CREATE (carrieanne:Actor { name:\"Carrie-Anne Moss\" }) CREATE (keanu)-[:ACTS\_IN { role : \"Neo\" }]->(matrix1) CREATE (keanu)-[:ACTS\_IN { role : \"Neo\" }]->(matrix2) CREATE (keanu)-[:ACTS\_IN { role : \"Neo\" }]->(matrix3) CREATE (laurence)-[:ACTS\_IN { role : \"Morpheus\" }]->(matrix1) CREATE (laurence)-[:ACTS\_IN { role : \"Morpheus\" }]->(matrix2) CREATE (laurence)-[:ACTS\_IN { role : \"Morpheus\" }]->(matrix3) CREATE (carrieanne)-[:ACTS\_IN { role : \"Trinity\" }]->(matrix1) CREATE (carrieanne)-[:ACTS\_IN { role : \"Trinity\" }]->(matrix2) CREATE (carrieanne)-[:ACTS\_IN { role : \"Trinity\" }]->(matrix3)"}]}' -u neo4j:neo4j2 |

**Results:**

Aft**er** executing the code the below screenshot are taken from the neo4j UI:

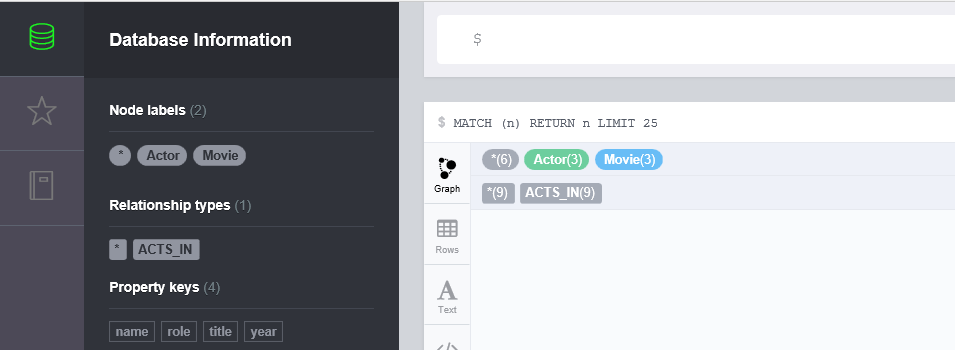


Figure - Database Labels, relationships and properties

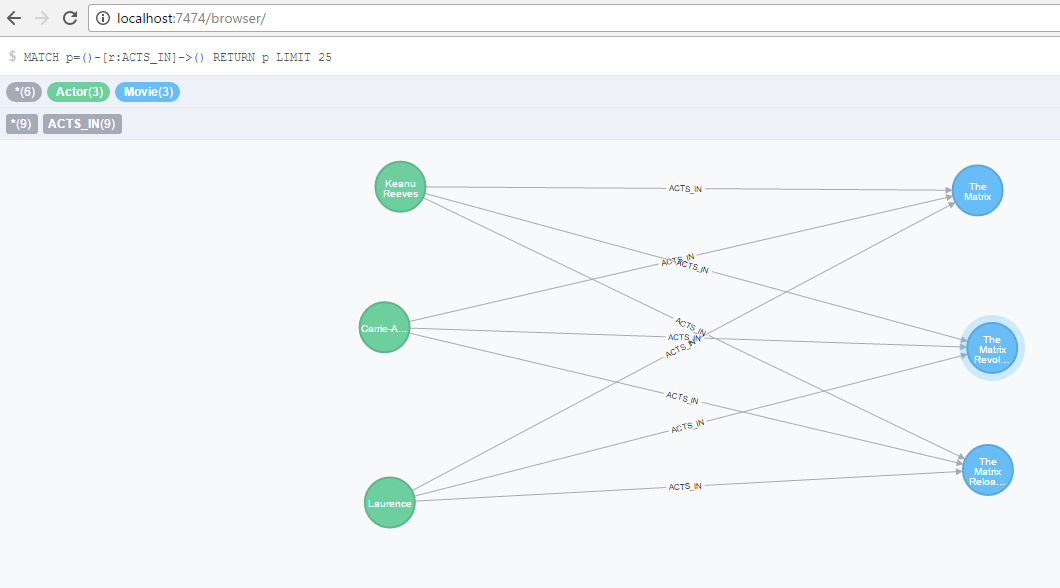


Figure - Graph Structure

**Problem 2**. Keanu Reeves acted in the movie “John Wick” which is not in the database. That movie was directed by Chad Stahelski and David Leitch. Cast of the movie included William Dafoe and Michael Nyquist. Add all of those people and the roles they played in this movie to the database using JAVA REST API or some other APIs for Neo4J in a language of your choice. Demonstrate that you have successfully brought data about John Wick movie into the database. You can use Cypher Browser or any other means.

**Explanation:**

The following queries Cypher queries a required to add the new movie with its actors and directors. I have created a new type on node for Directors, and added those 2 to it.

|  |
| --- |
| CREATE (johnwick:Movie {title:"John Wick",year:2014})  CREATE (william:Actor {name:"William Dafoe"})  CREATE (michael:Actor {name:"Michael Nyquist"})  CREATE (chad:Director {name:"Chad Stahelski"})  CREATE (david:Director {name:"David Leitch"})  CREATE (william)-[:ACTS\_IN {role:"Marcus"}]->(johnwick)  CREATE (michael)-[:ACTS\_IN {role:"Viggo Tarasov"}]->(johnwick)  CREATE (chad)-[:DIRECTOR\_OF]->(johnwick)  CREATE (david)-[:DIRECTOR\_OF]->(johnwick) |

Afterwards, since Keanu Reeves already exists in the database a relationship with the recently created film requires the below query:

|  |
| --- |
| MATCH (m:Movie) WHERE m.title = "John Wick"  MATCH (a:Actor) WHERE a.name = "Keanu Reeves"  CREATE (a)-[:ACTS\_IN {role:"John Wick"}]->(m) |

These are pure Cypher queries, but the problem asks for using a REST interface, so I need to serialize the queries to JSON:

I will doing it through 2 post queries. 1st one creates new film and the new actors and directors with the relationship. Since Keanu Reaves already exists I create the relationship in a 2nd statement

|  |
| --- |
| {      "**statements**":[         {            "**statement**":"CREATE (johnwick:Movie {title:\"John Wick\",year:2014}) CREATE (william:Actor {name:\"William Dafoe\"}) CREATE (michael:Actor {name:\"Michael Nyquist\"}) CREATE (chad:Director {name:\"Chad Stahelski\"}) CREATE (david:Director {name:\"David Leitch\"}) CREATE (william)-[:ACTS\_IN {role:\"Marcus\"}]->(johnwick) CREATE (michael)-[:ACTS\_IN {role:\"Viggo Tarasov\"}]->(johnwick) CREATE (chad)-[:DIRECTOR\_OF]->(johnwick) CREATE (david)-[:DIRECTOR\_OF]->(johnwick)"       }    ] } |

Second sentence that creates the relationships:

|  |
| --- |
| {      "**statements**":[         {            "**statement**":"MATCH (m:Movie) WHERE m.title = \"John Wick\" MATCH (a:Actor) WHERE a.name = \"Keanu Reeves\" CREATE (a)-[:ACTS\_IN {role:\"John Wick\"}]->(m)"       }    ] } |

And finally it is required to post the JSON statements to the database using any REST client. In my case I am using Chrome’s REST Advanced Client.

**Code Snippet:**

Now, using Chrome’s REST client I post the previous JSON sentences to my neo4j database:

URL:

|  |
| --- |
| <http://neo4j:neo4j2@localhost:7474/db/data/transaction/commit> |

Raw Headers:

|  |
| --- |
| Accept: application/json; charset=UTF-8  Content-Type: application/json |

Payload (1st sentence):

|  |
| --- |
| {  "statements":[  {  "statement":"CREATE (johnwick:Movie {title:\"John Wick\",year:2014}) CREATE (william:Actor {name:\"William Dafoe\"}) CREATE (michael:Actor {name:\"Michael Nyquist\"}) CREATE (chad:Director {name:\"Chad Stahelski\"}) CREATE (david:Director {name:\"David Leitch\"}) CREATE (william)-[:ACTS\_IN {role:\"Marcus\"}]->(johnwick) CREATE (michael)-[:ACTS\_IN {role:\"Viggo Tarasov\"}]->(johnwick) CREATE (chad)-[:DIRECTOR\_OF]->(johnwick) CREATE (david)-[:DIRECTOR\_OF]->(johnwick)"  }  ] |

PayLoad (2nd sentence):

|  |
| --- |
| {"statements":[  {"statement":  "MATCH (m:Movie) WHERE m.title = \"John Wick\" MATCH (a:Actor) WHERE a.name = \"Keanu Reeves\" CREATE (a)-[:ACTS\_IN {role:\"John Wick\"}]->(m)"  }  ]} |

Detail of the REST client UI:

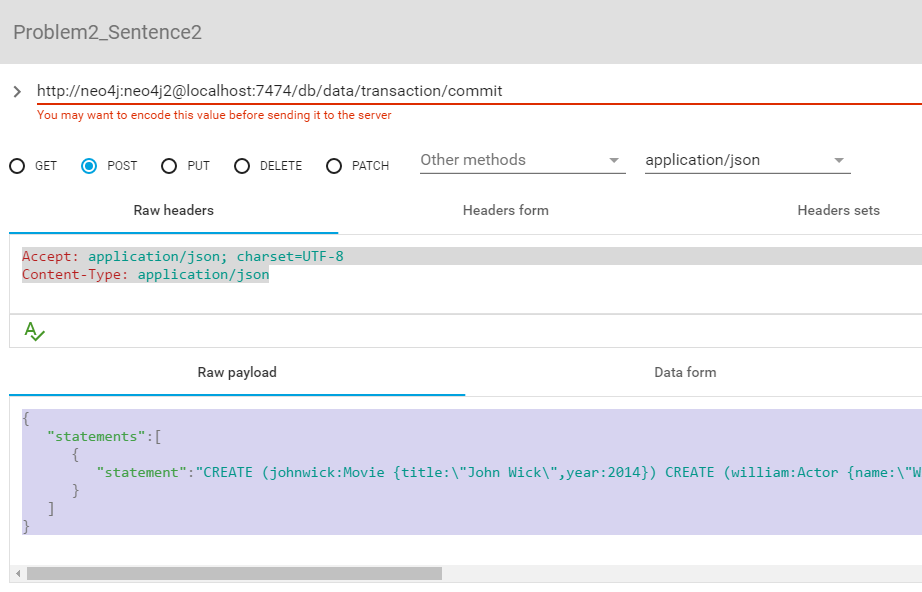


Figure - Detail or the REST client

**Results:**

In the below image the complete graph is depicted. It can be appreciated the new films with its actors and directors, including Keanu Reeves that has a relationship with the new film John Wick in addition to the previous 3 Matrix movies.

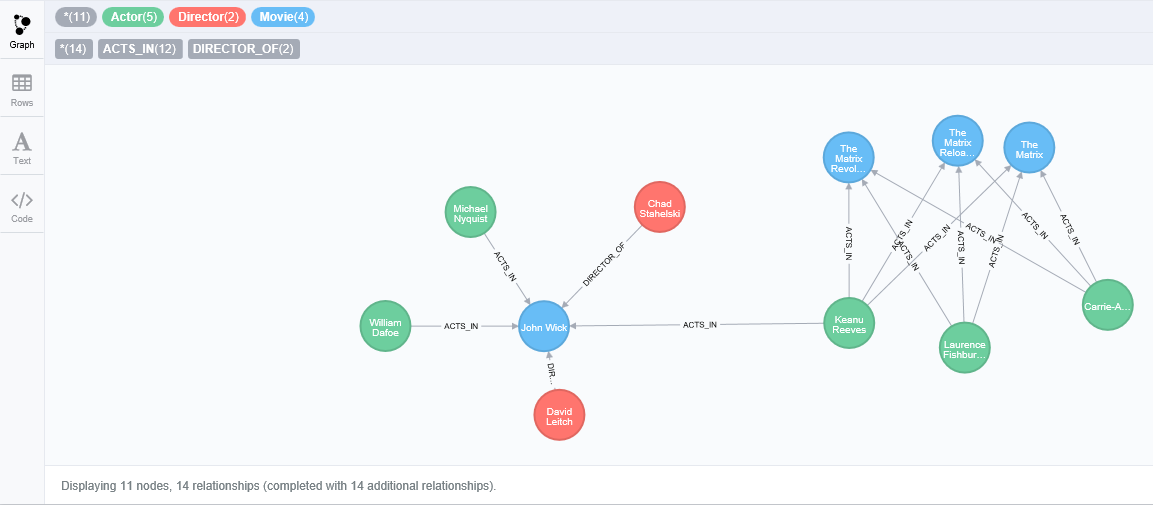


Figure - Graph Status after completing the Matrix saga with John Wick movie

**Problem 3**. Find a list of actors playing in movies in which Keanu Reeves played. Find directors of movies in which K. Reeves played. Please use any programming language of your convenience. Verify your results using Cypher queries in Cypher Browser

**Explanation:**

In my case I am using python 2.7- First thing to do is to install neo4j-driver library within my python environment. I usually do it in a straightforward way using pip.

**pip install neo4j-driver**

By using neo4j I can use native drive through bolt protocol, and execute the requested cypher sentences in this way:

**#import libraries**

from neo4j.v1 import GraphDatabase, basic\_auth

**#conn to neo4j instance**

driver = GraphDatabase.driver("bolt://localhost:7687",auth=basic\_auth("neo4j", "neo4j2"))

**#create session**

session = driver.session()

**#run Cypher queries**

result = session.run("CYPHER QUERIES")

**Code Snippet:**

|  |
| --- |
| #Problem 3. Find a list of actors playing in movies in which Keanu Reeves played.  #Find directors of movies in which K. Reeves played. Please use any programming  #language of your convenience. Verify your results using Cypher queries in Cypher Browser  #import libraries  from neo4j.v1 import GraphDatabase, basic\_auth  #conn to neo4j instance  driver = GraphDatabase.driver("bolt://localhost:7687",auth=basic\_auth("neo4j", "neo4j2"))  #run creation queries: actors in same films as Keanue Reeves  session = driver.session()  result = session.run("""  MATCH (k:Actor) WHERE k.name = "Keanu Reeves"  MATCH (a:Actor) WHERE [k:ACTED\_IN] and a<>k return a.name as name  """)  #print actors  print "List of actors that played in same movies as Keanue Reeves:"  for record in result:  print("%s" % (record["name"]))  #run creation queries: directors in same films as Keanue Reeves  result = session.run("""  MATCH (k:Actor) WHERE k.name = "Keanu Reeves"  MATCH (d:Director) WHERE [k:ACTED\_IN] return d.name as name  """)  #print directors  print "List of directors that directed Keanue Reeves:"  for record in result:  print("%s" % (record["name"]))    #close session  session.close() |

**Results:**

Previous script returns the following:

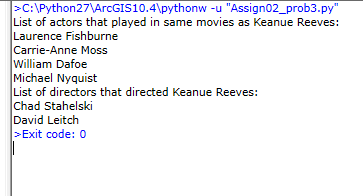


Figure - Actors and Director related to Keanu Reeves

Running Cypher queries from neo4J console return the same results:

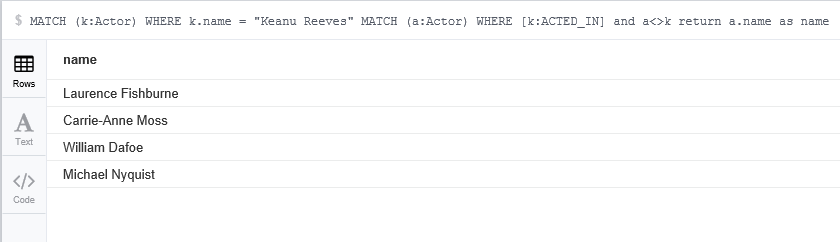


Figure - Actors is same films as Keanu Reeves (Cypher match query)

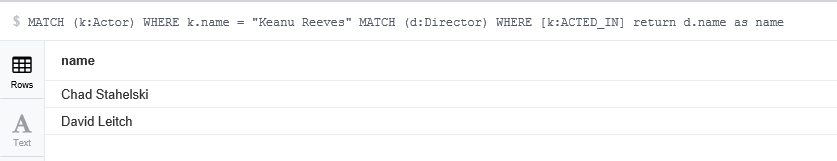


Figure - Directors in same films as Keanu Reeves (Cypher match query)

**Problem 4**. Find a way to export data from Neo4j into a set of CSV files. Delete your database and demonstrate that you can recreate the database by loading those CSV files. Please use any programming language of your convenience: Java, Python, R, or Scala.

Explanation:

I have created a python script to export each entity of the database to a csv file. The result are 5 csv files.

* Node labels:
  + Actors: actor.csv
  + Directors: director.csv
  + Movies: movie.csv
* Relationship types:
  + Act In: acst-in.csv
  + Director of: director-of.csv

**Code Snippet:**

Export database content to CSV

|  |
| --- |
| #Problem 4. Find a way to export data from Neo4j into a set of CSV files.  #Delete your database and demonstrate that you can recreate the database  #by loading those CSV files. Please use any programming language of your convenience: Java, Python, R, or Scala.  #import libraries  from neo4j.v1 import GraphDatabase, basic\_auth  #conn to neo4j instance  driver = GraphDatabase.driver("bolt://localhost:7687",auth=basic\_auth("neo4j", "neo4j2"))  session = driver.session()  #gets all actor node labels write to a csv file  result = session.run("""  MATCH (n:Actor)  RETURN n.name as name;  """)  f = open('d:\\temp\\actors.csv','w')  f.write("name\n")  for record in result:  f.write(record["name"] +"\n")  f.close()  #same fo director node lables  result = session.run("""  MATCH (d:Director)  RETURN d.name as name;  """)  f = open('d:\\temp\\directors.csv','w')  f.write("name\n")  for record in result:  f.write(record["name"] +"\n")  f.close()  #same for movie node lables  result = session.run("""  MATCH (m:Movie)  RETURN m.title as title, m.year as year  """)  f = open('d:\\temp\\movie.csv','w')  f.write("title,year\n")  for record in result:  f.write(record["title"] + "," + str(record["year"]) + "\n")  f.close()  #same for relationship types for actors  f = open('d:\\temp\\acts-in.csv','w')  f.write("name,role,year,title\n")  result = session.run("""  MATCH p=(a:Actor)-[r:ACTS\_IN]->(m:Movie)  RETURN a.name as name, r.role as role, m.year as year, m.title as title  """)  for record in result:  f.write(record["name"] + "," + record["role"] + "," + str(record["year"]) + "," + record["title"] + "\n")  f.close()  #same for relationship types for directors  f = open('d:\\temp\\director-of.csv','w')  f.write("name,year,title\n")  result = session.run("""  MATCH p=(d:Director)-[r:DIRECTOR\_OF]->(m:Movie)  RETURN d.name as name, m.year as year, m.title as title  """)  for record in result:  f.write(record["name"] + "," + str(record["year"]) + "," + record["title"] + "\n")  f.close()  #close session  session.close() |

Once created the export files, I proceed to delete all entities from the database and recreate database’s content from the CSV files.

MATCH (n) DETACH DELETE n;

Finally I will recreate the database from the CSV files- First step before importing the CSV files is to enable the import directory from neo4j.conf. According to my computer the line will be as the following:

dbms.directories.import=d:/temp

Once restarted the database instance, I execute the following script to recreate the database:

|  |
| --- |
| #Problem 4. Find a way to export data from Neo4j into a set of CSV files.  #Delete your database and demonstrate that you can recreate the database  #by loading those CSV files. Please use any programming language of your convenience: Java, Python, R, or Scala.  #import libraries  from neo4j.v1 import GraphDatabase, basic\_auth  #conn to neo4j instance  driver = GraphDatabase.driver("bolt://localhost:7687",auth=basic\_auth("neo4j", "neo4j2"))  session = driver.session()  #gets all actor node labels write to a csv file and load them into the databse  result = session.run("""  LOAD CSV WITH HEADERS FROM "file:///movies.csv" AS line  CREATE (m:Movie {title:line.title, year:line.year});  """)  #same for actors  result = session.run("""  LOAD CSV WITH HEADERS FROM "file:///actors.csv" AS line  CREATE (a:Actor {name:line.name});  """)  #same for directors  result = session.run("""  LOAD CSV WITH HEADERS FROM "file:///directors.csv" AS line  CREATE (d:Director {name:line.name});  """)  #finally relationships for actors  result = session.run("""  LOAD CSV WITH HEADERS FROM "file:///acts-in.csv" AS line  MATCH (m:Movie) WHERE m.title = line.title  MATCH (a:Actor) WHERE a.name = line.name  CREATE (a)-[:ACTS\_IN {role:line.role}]->(m)  """)  #and for directors  result = session.run("""  LOAD CSV WITH HEADERS FROM "file:///director-of.csv" AS line  MATCH (d:Director) WHERE d.name = line.name  MATCH (m:Movie) WHERE m.title = line.title and m.year=line.year  CREATE (d)-[:DIRECTOR\_OF]->(m)  """)  #close session  session.close() |

**Results:**

After successfully running the script and refreshing the web browser the database is back again as shown in the below image:

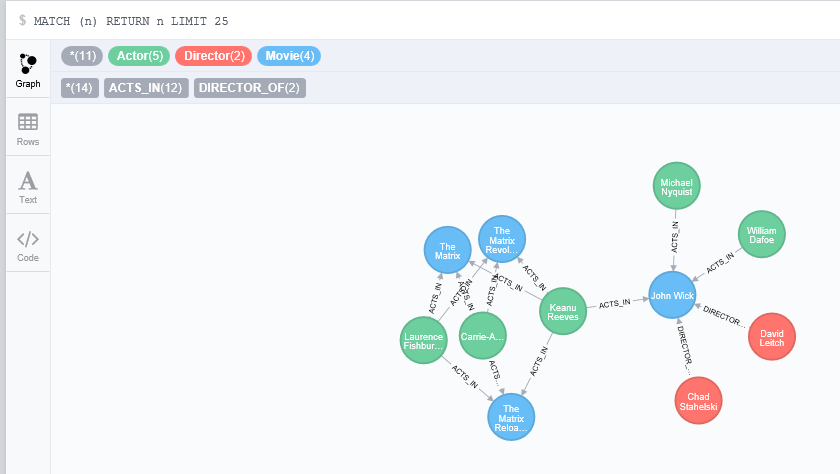


Figure 8 - Database recreated from CSV files