Neo4j Learning Report

Installing Neo4j Enterprise Server

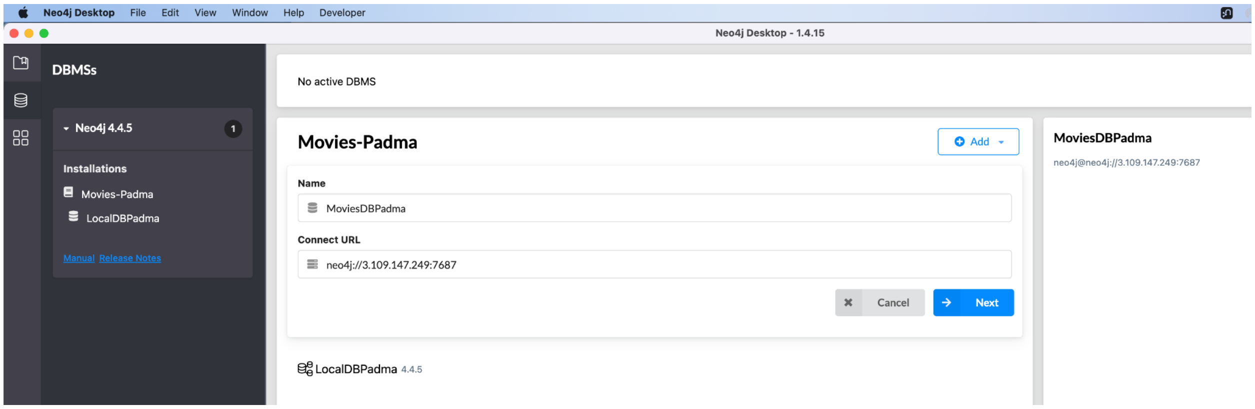
1. Launched a new EC2 instance on AWS using Neo4j AMI
   a. AMI ID: ami-08916596a0646ef66
   b. AMI name: neo4j-enterprise-1-4.4.3-2022-01-17T01_39_47Z

2. Created new security group for the EC2 instance by allowing ports 7474, 7473, 7687, 80, 443, 22

3. Server is up and running at public IP 3.109.147.249

Installing Neo4j Desktop

1. Downloaded from https://neo4j.com/download/ and installed on laptop
2. Created a new project - “Movies-Padma”
3. Added a remote DB - MoviesDBPadma - this DB is created in Enterprise server



No active DBMS

Movies-Padma

Name
MoviesDBPadma

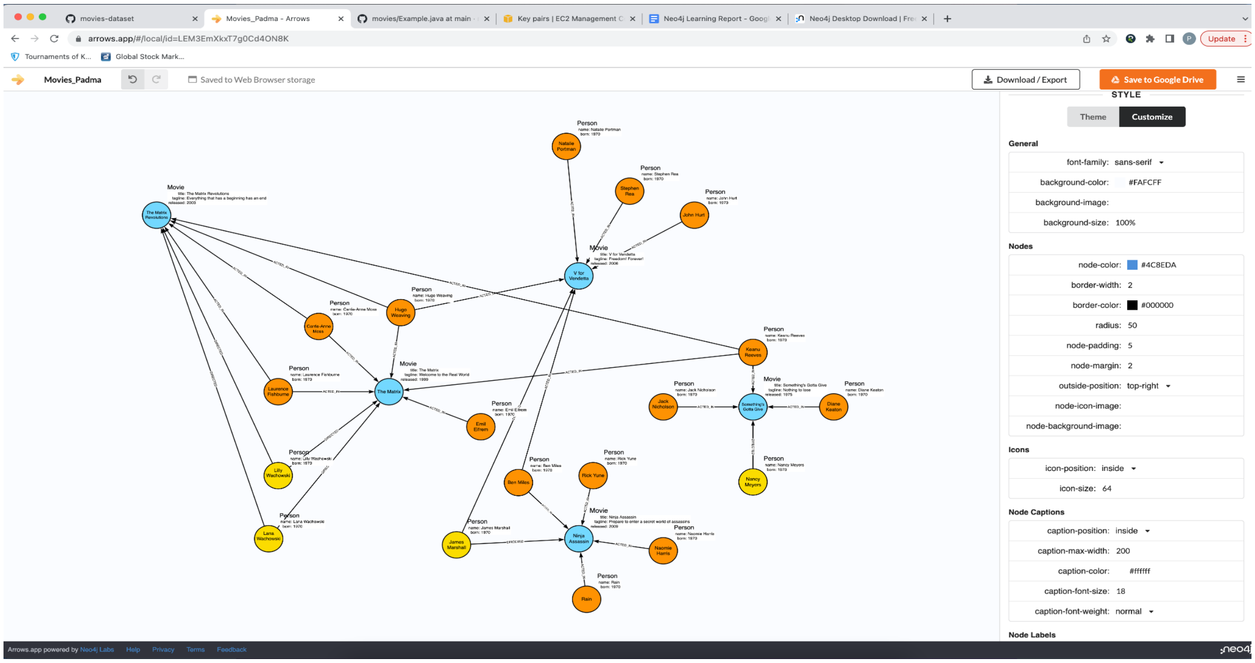
Connect URL
neo4j://3.109.147.249:7687

MoviesDBPadma
neo4j@neo4j://3.109.147.249:7687

LocalDBPadma 4.4.5

Modeling Graph DB for Movies using arrows.app

1. Created nodes for Movies and Persons
2. Created relationships - ACTED_IN and DIRECTED
3. Exported in JSON and SVG format, provided along with this document



STYLE

Theme | Customize

General

font-family: sans-serif
background-color: #FAFCFF
background-image:
background-size: 100%

Nodes

node-color: #4C8EDA
border-width: 2
border-color: #000000
radius: 50
node-padding: 5
node-margin: 2
outside-position: top-right
node-icon-image:
node-background-image:

Icons

icon-position: inside
icon-size: 64

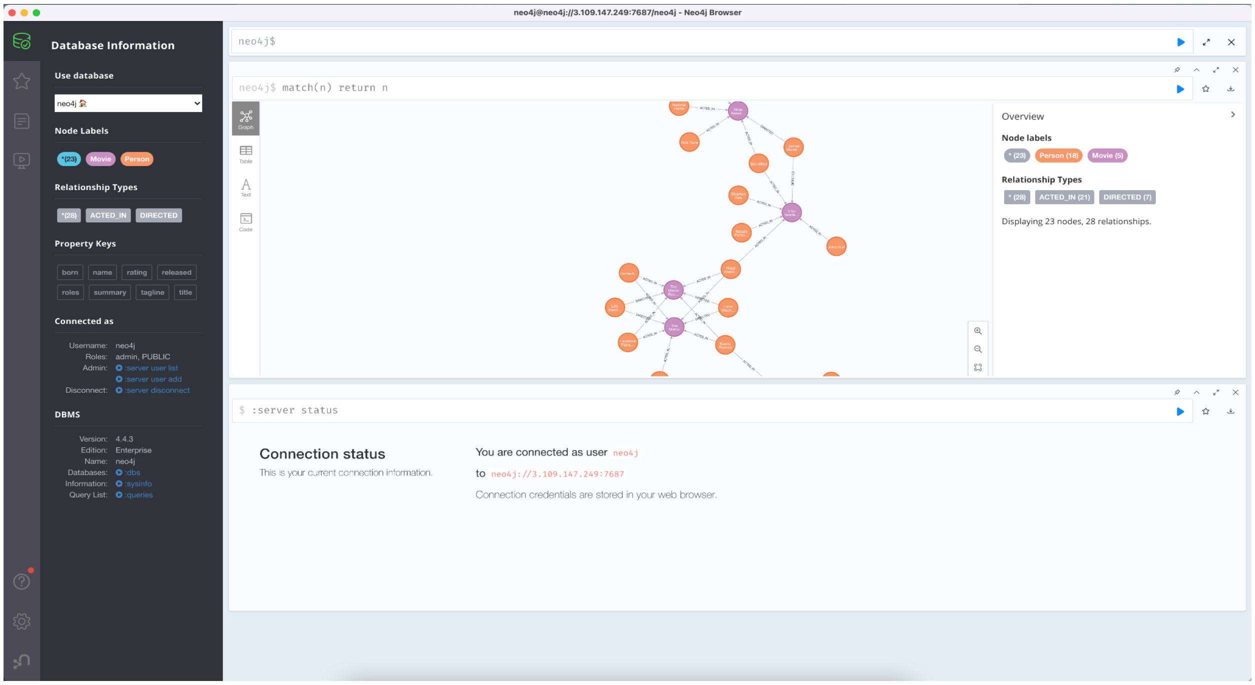
Node Captions

caption-position: inside
caption-max-width: 200
caption-color: #ffffff
caption-font-size: 18
caption-font-weight: normal

Node Labels

Exploring Neo4j Browser

1. Launch Browser from the Neo4j Desktop for the remote DB MoviesDBPadma
2. Ingest Movies data using the Cypher statements exported from arrows.app
3. Try various Cypher statement to retrieve nodes, relationships etc



Database Information

Use database

neo4j

Node Labels

*(23) Movie Person

Relationship Types

*(28) ACTED_IN DIRECTED

Property Keys

born name rating released roles summary tagline title

Connected as

Username: neo4j
Roles: admin, PUBLIC
Admin: :server user list
:server user add
Disconnect: :server disconnect

DBMS

Version: 4.4.3
Edition: Enterprise
Name: neo4j
Databases: :dbs
Information: :sysinfo
Query List: :queries

```
neo4j$
```

```
neo4j$ match(n) return n
```

Overview

Node labels

* (23) Person (18) Movie (5)

Relationship Types

* (28) ACTED_IN (21) DIRECTED (7)

Displaying 23 nodes, 28 relationships.

```
$ :server status
```

Connection status

This is your current connection information.

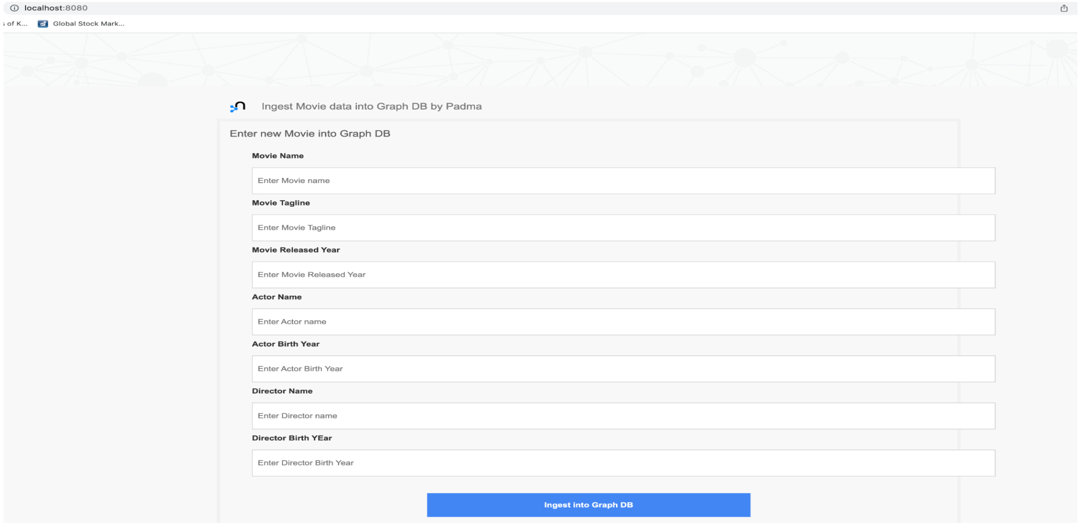
You are connected as user neo4j

to neo4j://3.109.147.249:7687

Connection credentials are stored in your web browser.

Graph enabled application to ingest data

1. Downloaded example Java application from https://github.com/neo4j-graph-examples/movies.git
2. Modified index.html to add a html form to collect movie information from user
3. Modified backend Java code to parse the parameters and execute Cypher queries to insert Movie node, Person nodes and relationships
4. Code is uploaded to https://github.com/padmashreemp/movies-graphdb-padma



Ingest Movie data into Graph DB by Padma

Enter new Movie into Graph DB

Movie Name

Enter Movie name

Movie Tagline

Enter Movie Tagline

Movie Released Year

Enter Movie Released Year

Actor Name

Enter Actor name

Actor Birth Year

Enter Actor Birth Year

Director Name

Enter Director name

Director Birth YEar

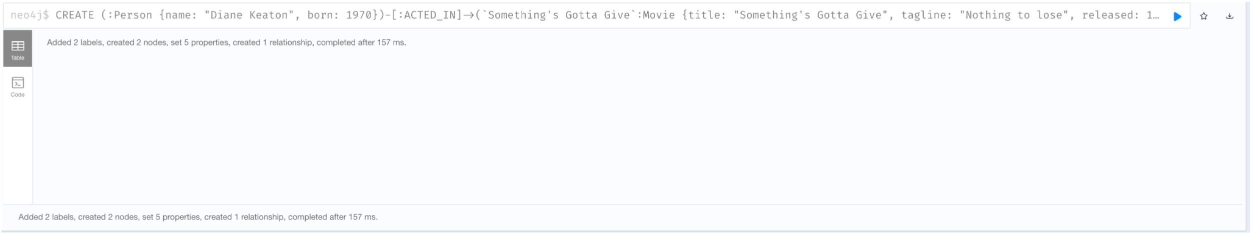
Enter Director Birth Year

Cypher Queries

Below are some of the queries i have written and executed as part of this assignment

1. Create a Person node and Movie node and add relationship ACTED_IN between the nodes

```
CREATE (:Person {name: "Diane Keaton", born: 1970})-[:ACTED_IN]->(`Something's Gotta Give`:Movie {title: "Something's Gotta Give", tagline: "Nothing to lose", released: 1975})
```



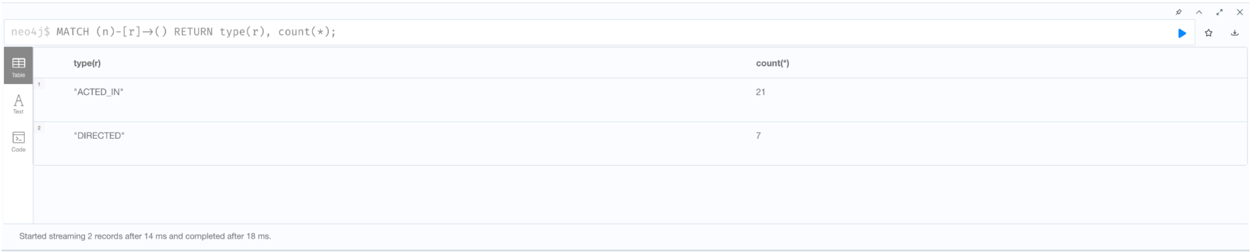
```
neo4j$ CREATE (:Person {name: "Diane Keaton", born: 1970})-[:ACTED_IN]→(`Something's Gotta Give`:Movie {title: "Something's Gotta Give", tagline: "Nothing to lose", released: 1…
```

Added 2 labels, created 2 nodes, set 5 properties, created 1 relationship, completed after 157 ms.

Added 2 labels, created 2 nodes, set 5 properties, created 1 relationship, completed after 157 ms.

2. Count number of relationships of each type

```
MATCH (n)-[r]->() RETURN type(r), count(*);
```



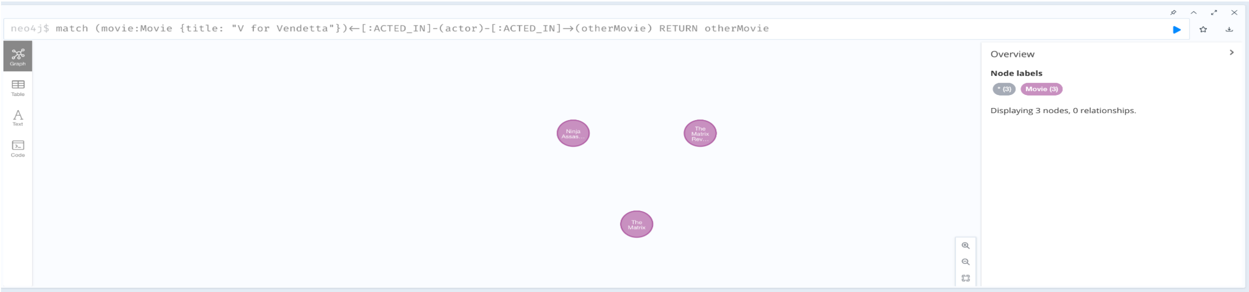
```
neo4j$ MATCH (n)-[r]→() RETURN type(r), count(*);
```

	type(r)	count(*)
1	"ACTED_IN"	21
2	"DIRECTED"	7

Started streaming 2 records after 14 ms and completed after 18 ms.

3. Find other movies of the actor who has acted in V for Vendetta

```
MATCH (movie:Movie {title: "V for
Vendetta"})<-[:ACTED_IN]-(actor)-[:ACTED_IN]->(otherMovie)
RETURN otherMovie
```



```
neo4j$ match (movie:Movie {title: "V for Vendetta"})←[:ACTED_IN]-(actor)-[:ACTED_IN]→(otherMovie) RETURN otherMovie
```

Overview

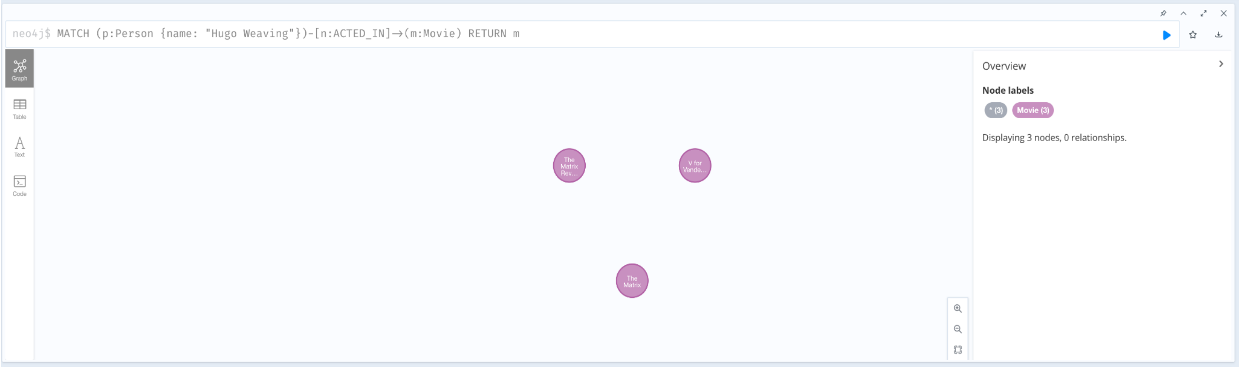
Node labels

* (3) Movie (3)

Displaying 3 nodes, 0 relationships.

4. Find out movies that has Hugo Weaving as an actor

```
MATCH (p:Person {name: "Hugo Weaving"})-[n:ACTED_IN]->(m:Movie) RETURN m
```



```
neo4j$ MATCH (p:Person {name: "Hugo Weaving"})-[n:ACTED_IN]→(m:Movie) RETURN m
```

Overview

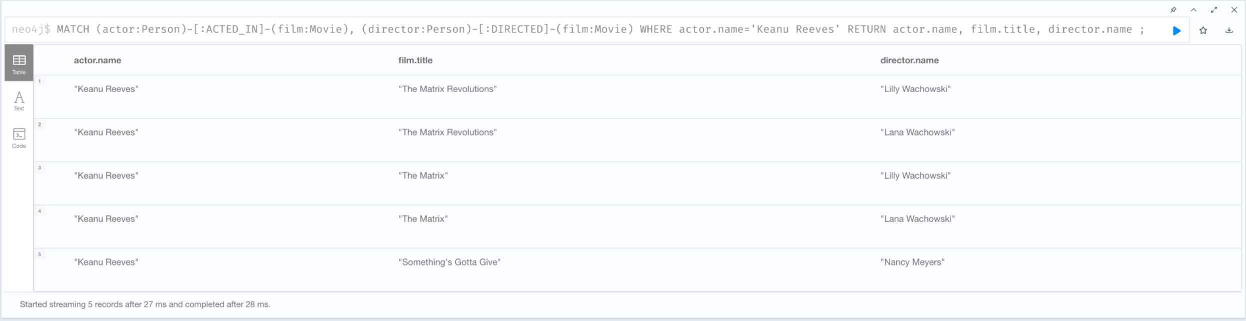
Node labels

* (3) Movie (3)

Displaying 3 nodes, 0 relationships.

5. Find the actors who have directed movies that has Keanu Reeves as Actor

```
MATCH (actor:Person)-[:ACTED_IN]-(film:Movie),
(director:Person)-[:DIRECTED]-(film:Movie)
WHERE actor.name='Keanu Reeves' RETURN actor.name, film.title, director.name ;
```



```
neo4j$ MATCH (actor:Person)-[:ACTED_IN]-(film:Movie), (director:Person)-[:DIRECTED]-(film:Movie) WHERE actor.name='Keanu Reeves' RETURN actor.name, film.title, director.name ;
```

	actor.name	film.title	director.name
1	"Keanu Reeves"	"The Matrix Revolutions"	"Lilly Wachowski"
2	"Keanu Reeves"	"The Matrix Revolutions"	"Lana Wachowski"
3	"Keanu Reeves"	"The Matrix"	"Lilly Wachowski"
4	"Keanu Reeves"	"The Matrix"	"Lana Wachowski"
5	"Keanu Reeves"	"Something's Gotta Give"	"Nancy Meyers"

Started streaming 5 records after 27 ms and completed after 28 ms.