Neo4j - Cypher exercise

Load db dump into Neo4j

Create useful folders

```
mkdir neo4j
mkdir neo4j/data
mkdir neo4j/import
cd neo4j
```

Create and run Neo4j on Docker

```
docker run \
        --publish=7474:7474 --publish=7687:7687 \
        --volume=./data:/data \
        --volume=./import:/import \
        --name=neo4j \
        --env=NEO4J_AUTH=none \
        -d \
        neo4j
```

Stop the Neo4j container

```
docker stop neo4j
```

Place data dump into your local import folder

Run the Neo4j-admin import tool

```
docker run --interactive --tty --rm \
        --volume=./data:/data \
        --volume=./import:/import \
    neo4j/neo4j-admin \
        neo4j-admin database load neo4j \
    --from-path=/import \
    --overwrite-destination=true
```

Start the Neo4j container

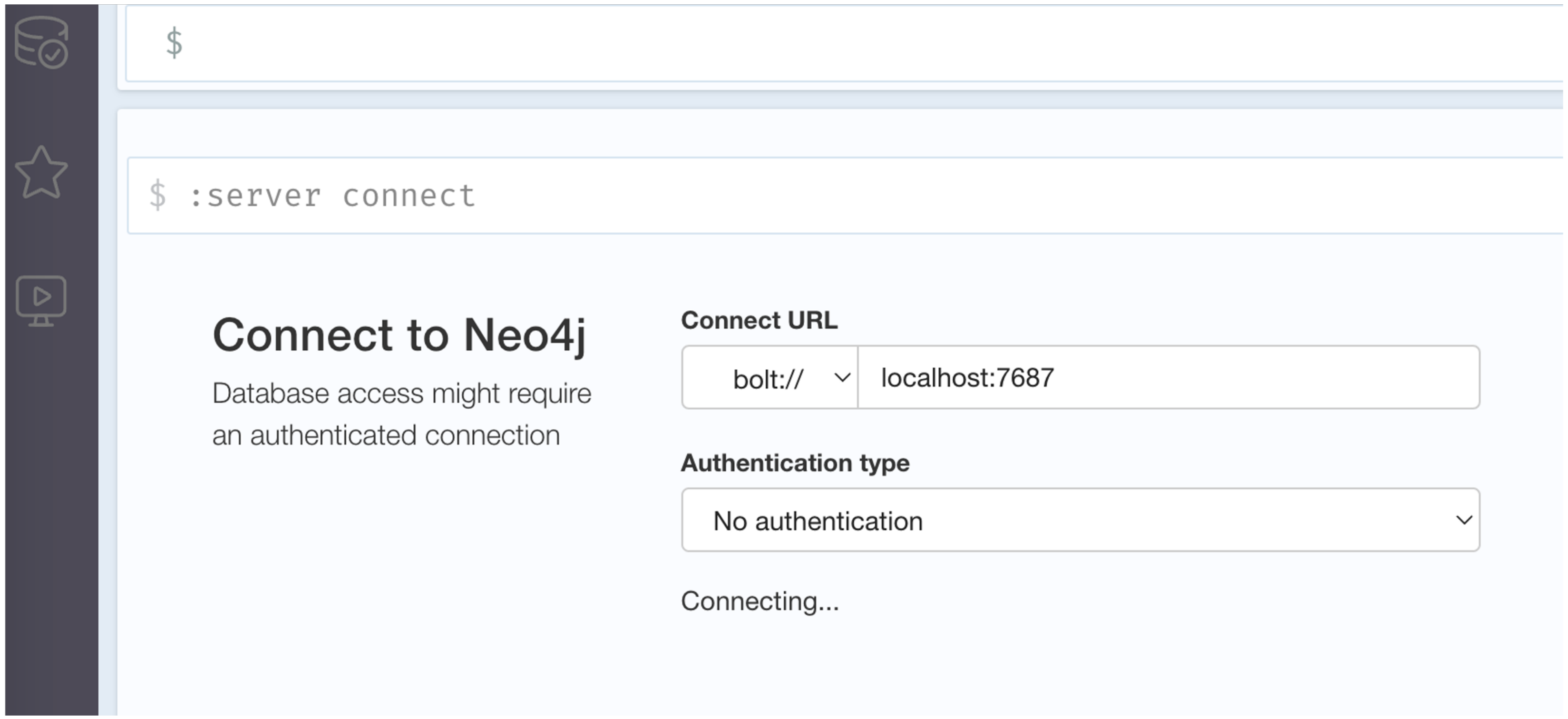
```
docker start neo4j
```

Querying the data

Connect to Neo4j browser

http://localhost:7474

Select the Authentication type



```
$
```

```
$ :server connect
```

Connect to Neo4j

Database access might require an authenticated connection

Connect URL

bolt:// localhost:7687

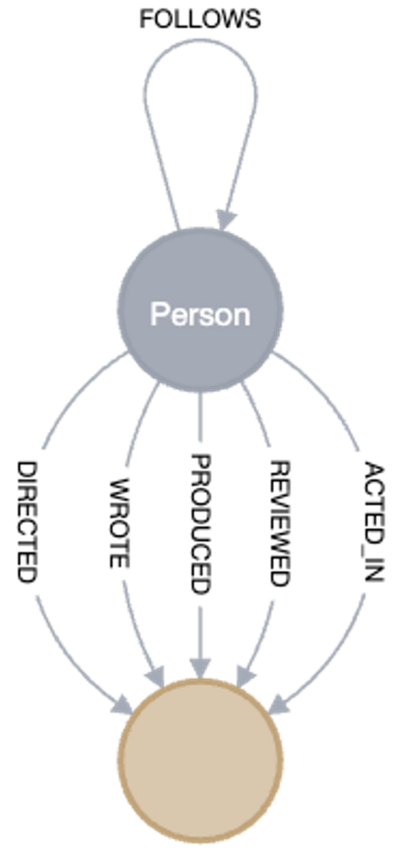
Authentication type

No authentication

Connecting...

Describe the database content

CALL db.schema.visualization



Select *

```
MATCH(ALL)
RETURN *
```

Find all Person(s) who acted in a Movie

```
MATCH(p:Person) -[a:ACTED_IN]-> (m:Movie)

RETURN *
```

Find Movie(s) where Keanu Reeves acted

```
MATCH(p:Person) -[a:ACTED_IN]-> (m:Movie)

WHERE p.name="Keanu Reeves"

RETURN *
```

Find Movie(s) where Keanu Reeves acted

```
MATCH(p:Person) -[a:ACTED_IN]-> (m:Movie)

WHERE p.name="Keanu Reeves"

RETURN *
```

OR...

Find Movie(s) where Keanu Reeves acted

```
MATCH(p:Person{name:"Keanu Reeves"}) -[a:ACTED_IN]-> (m:Movie)

RETURN *
```

Find Movie(s) released in year 2003

???

Find Movie(s) released in year 2003

```
MATCH (m:Movie)

WHERE  m.released = 2003

RETURN m
```

OR…

Find Movie(s) released in year 2003

```
MATCH (m:Movie {released:2003})

RETURN m
```

Find Movie(s) released the earliest

???

Find Movie(s) released the earliest

```
MATCH (m:Movie)

RETURN MIN(m.released)
```

Retrieve all Movie nodes from the database and return the title, released, and tagline values

???

Retrieve all Movie nodes from the database and return the title, released, and tagline values

```
MATCH (m:Movie)

RETURN m.title, m.released, m.tagline
```

Retrieve all Movie(s) connected with Tom Hanks

???

Retrieve all Movie(s) connected with Tom Hanks

```
MATCH (m:Movie) <-- (:Person {name: 'Tom Hanks'})

RETURN m.*
```

Retrieve all Movie(s) connected with Tom Hanks, and specify the relationship

???

Retrieve all Movie(s) connected with Tom Hanks, and specify the relationship

```
MATCH (m:Movie) -[rel]- (:Person {name: 'Tom Hanks'})

RETURN m.title, type(rel)
```

Retrieve all people that were born in the 70’s and return their names and year born

???

Retrieve all people that were born in the 70’s and return their names and year born

```
MATCH (a:Person)

WHERE a.born >= 1970 AND a.born < 1980

RETURN a.name as Name, a.born as `Year Born`
```

Retrieve the actors who acted in the movie The Matrix who were born after 1960, and return their names and year born

???

Retrieve the actors who acted in the movie The Matrix who were born after 1960, and return their names and year born

```
MATCH (a:Person) -[:ACTED_IN]-> (m:Movie)

WHERE a.born > 1960 AND m.title = 'The Matrix'

RETURN a.name as Name, a.born as `Year Born`
```

Retrieve all people that wrote movies by testing the relationship between two nodes

???

Retrieve all people that wrote movies by testing the relationship between two nodes

```
MATCH (a)-[rel]->(m)

WHERE a:Person

        AND type(rel) = 'WROTE'

        AND m:Movie

RETURN a.name as Name, m.title as Movie
```

Retrieve all people in the graph that do not have a born property, returning their names

???

Retrieve all people in the graph that do not have a born property, returning their names

```
MATCH (a:Person)
WHERE (a.born) is null
RETURN a.name
```

Retrieve all people related to movies where the relationship has the rating property, then return their name, movie title, and the rating

???

Retrieve all people related to movies where the relationship has the rating property, then return their name, movie title, and the rating

```
MATCH (a:Person)-[rel:REVIEWED]->(m:Movie)
WHERE (rel.rating) is not null
RETURN a.name as Name, m.title as Movie, rel.rating as Rating
```

Retrieve all actors whose name begins with James, returning their names

???

Retrieve all actors whose name begins with James, returning their names

```
MATCH (a:Person) -[:ACTED_IN]-> (:Movie)

WHERE toLower(a.name)  STARTS WITH 'James'

RETURN a.name
```

Retrieve the actors who have acted in exactly five movies

???

Retrieve the actors who have acted in exactly five movies

```
MATCH (a:Person)-[:ACTED_IN]->(m:Movie)

WITH a, count(m) AS numMovies

WHERE numMovies = 5

RETURN a.name
```

Retrieve the actors who have acted in exactly five movies, also returning the name of the actor, and the list of movies for that actor

???

Retrieve the actors who have acted in exactly five movies, also returning the name of the actor, and the list of movies for that actor

```
MATCH (a:Person)-[:ACTED_IN]->(m:Movie)

WITH a, count(m) AS numMovies, collect(m.title) AS movies

WHERE numMovies = 5

RETURN a.name, movies
```

Retrieve the movies that have at least 2 directors

???

Retrieve the movies that have at least 2 directors

```
MATCH p=(:Person)-[:DIRECTED]->(m)

WITH m, COUNT(p) AS directors

WHERE directors >= 2

RETURN  *
```

Retrieve the movies that have at least 2 directors with other optional data

???

Retrieve the movies that have at least 2 directors with other optional data

```
MATCH p=(:Person)-[:DIRECTED]->(m)
WITH m, COUNT(p) AS directors
WHERE directors >= 2
OPTIONAL MATCH (p:Person)-[:REVIEWED]->(m)
RETURN  *
```

Creating new nodes

Create a Movie node for the movie with the title « Forrest Gump»

Create a Movie node for the movie with the title « Forrest Gump»

```
CREATE (:Movie {title: 'Forrest Gump'})
```

Create a Person node for the person with the name «Robin Wright»

Create a Person node for the person with the name «Robin Wright»

```
CREATE (:Person {name: 'Robin Wright'})
```

Update existing nodes

Add the label OlderMovie to any Movie node that was released before 2010.

Add the label OlderMovie to any Movie node that was released before 2010.

```
MATCH (m:Movie)

WHERE m.released < 2010

SET m:OlderMovie

RETURN DISTINCT labels(m)
```

Retrieve all older movie nodes to test that the label was indeed added to these nodes

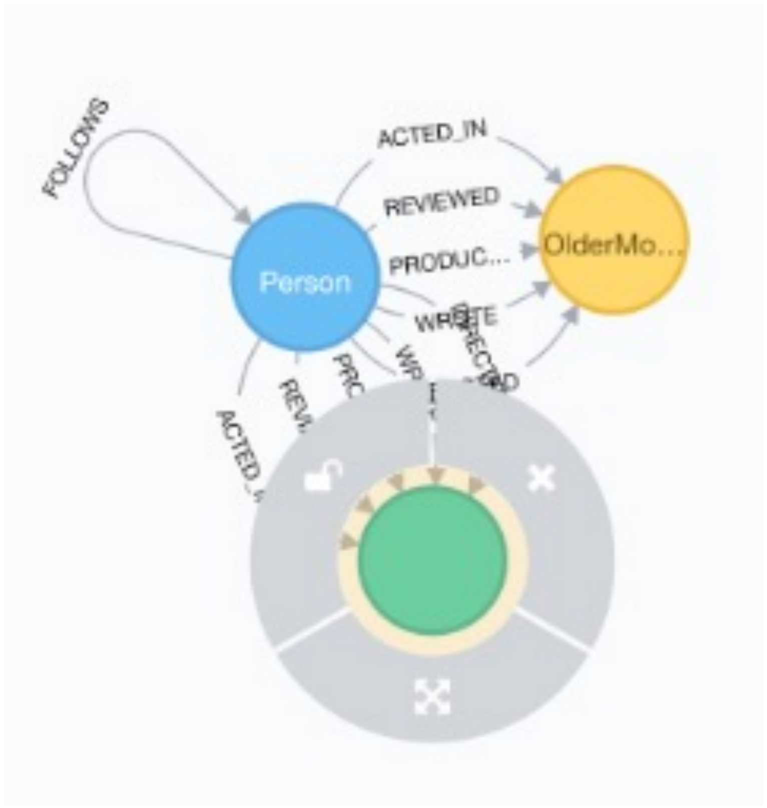
Retrieve all older movie nodes to test that the label was indeed added to these nodes

```
MATCH (m:OlderMovie)

RETURN m.title, m.released
```

Describe the database content

CALL db.schema.visualization



Add the following properties to the movie, Forrest Gump

released: 1994

tagline: Life is like a box of chocolates...you never know what you’re gonna get.

lengthInMinutes: 142

Add the following properties to the movie, Forrest Gump

released: 1994

tagline: Life is like a box of chocolates…you never know what you’re gonna get.

lengthInMinutes: 142

```
MATCH (m:Movie)

WHERE m.title = 'Forrest Gump'

SET m:OlderMovie,

m.released = 1994,

m.tagline = "Life is like a box of chocolates...you never
know what you're gonna get.",

m.lengthInMinutes = 142
```

Remove the lengthInMinutes property from the movie «Forrest Gump»

Remove the lengthInMinutes property from the movie «Forrest Gump»

```
MATCH (m:Movie)

WHERE m.title = 'Forrest Gump'

SET m.lengthInMinutes = null
```

Add relationships to nodes

Create the ACTED_IN relationship between the actors, Robin Wright, Tom Hanks, and Gary Sinise and the movie, Forrest Gump.

Create the ACTED_IN relationship between the actors, Robin Wright, Tom Hanks, and Gary Sinise and the movie, Forrest Gump.

```
MATCH (m:Movie)

WHERE m.title = 'Forrest Gump'

MATCH (p:Person)

WHERE p.name = 'Tom Hanks' OR p.name = 'Robin Wright' OR p.name = 'Gary Sinise'

CREATE (p)-[:ACTED_IN]->(m)
```

Delete a nodes

Delete the Forrest Gump node

Delete the Forrest Gump node

```
MATCH (m:Movie)

WHERE m.title = 'Forrest Gump'

DELETE m
```

Delete the Forrest Gump node

```
MATCH (m:Movie)

WHERE m.title = 'Forrest Gump'

DELETE m
```

This will not work because there are relationships attached

Delete the Forrest Gump node with its relationships

Delete the Forrest Gump node with its relationships

```
MATCH (m:Movie)

WHERE m.title = 'Forrest Gump'

DETACH DELETE m
```