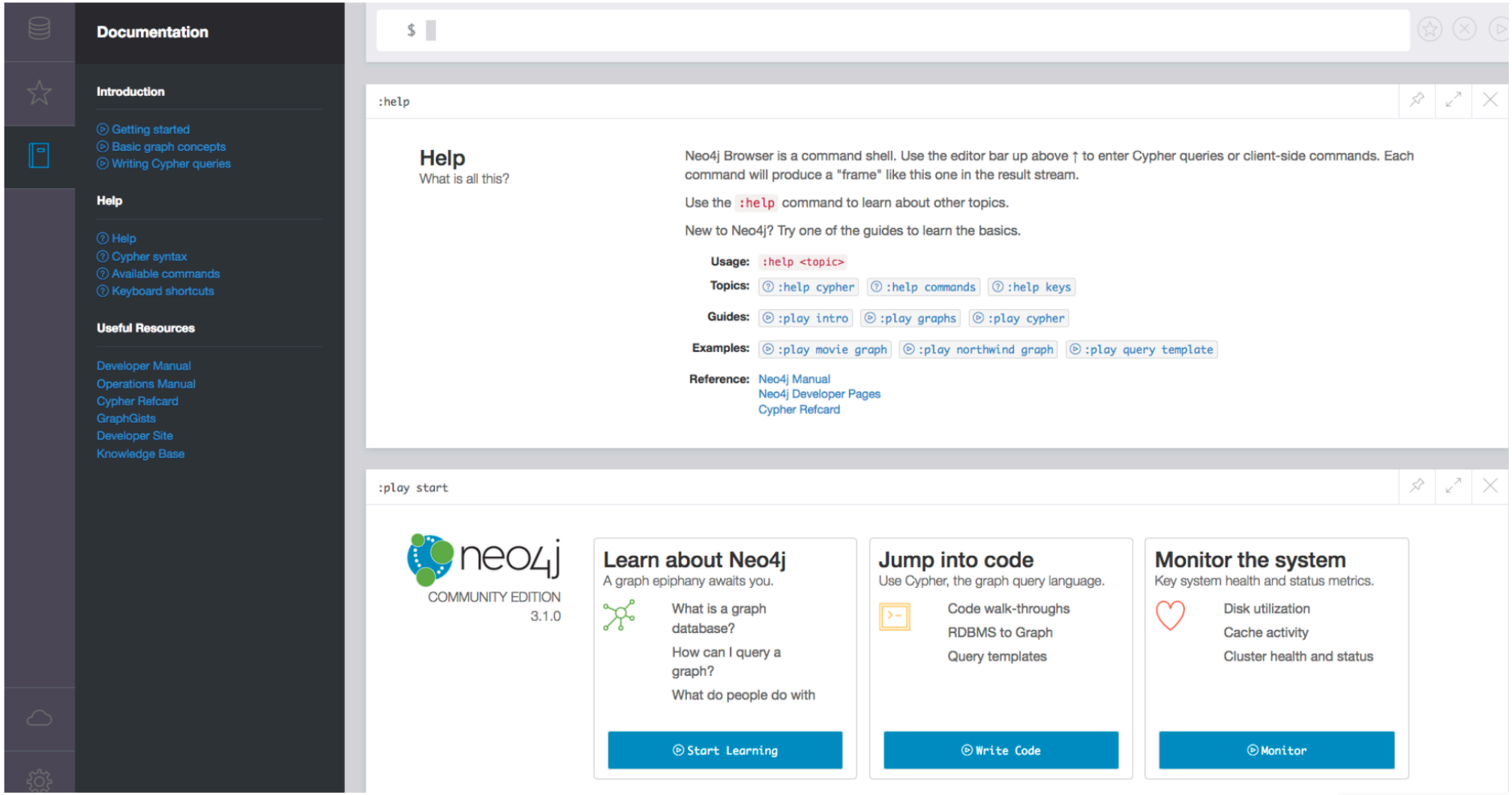
Lab3 – Neo4j

Data Structuring and NoSQL Databases

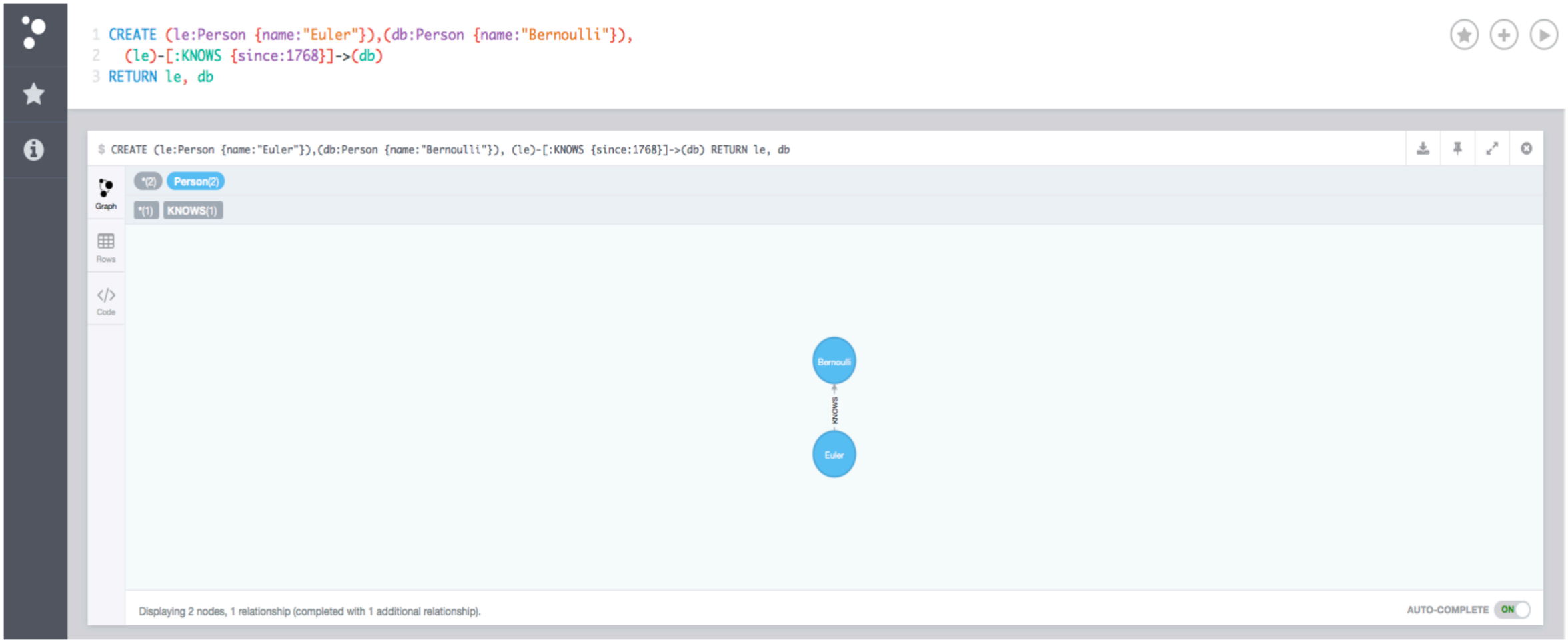
Larbi Boubchir

Neo4J browser

Help

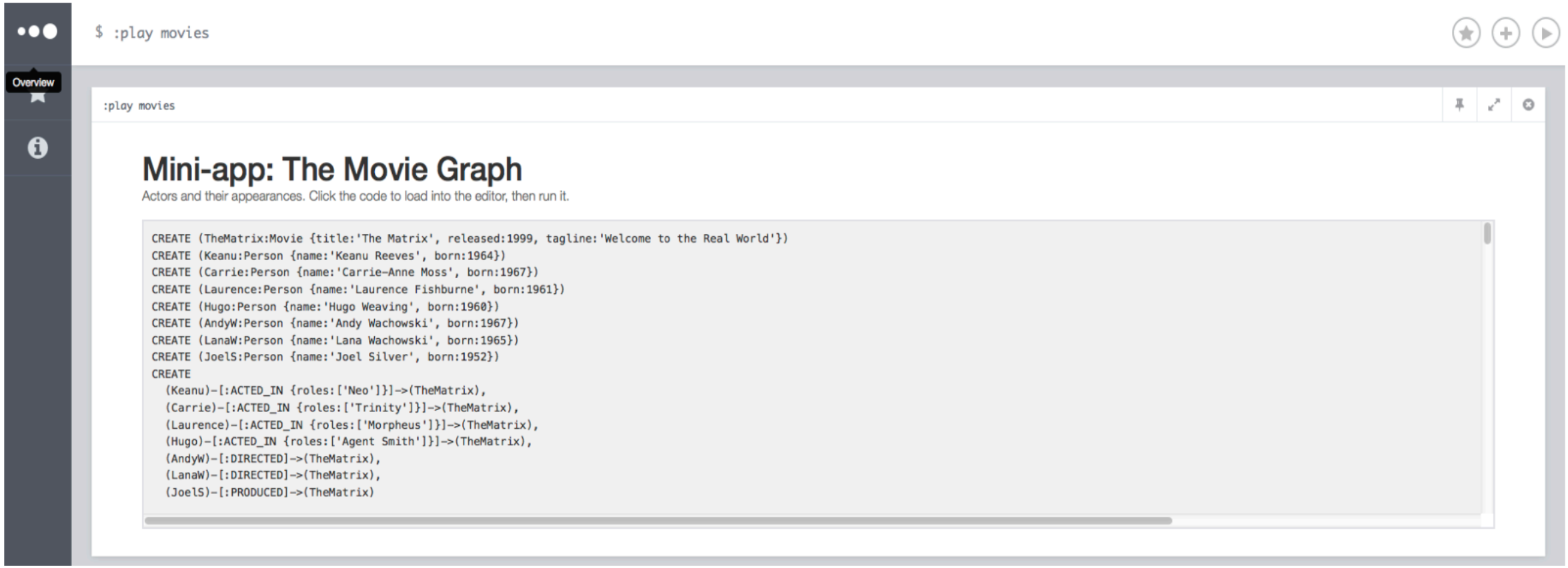


Queries and visualization



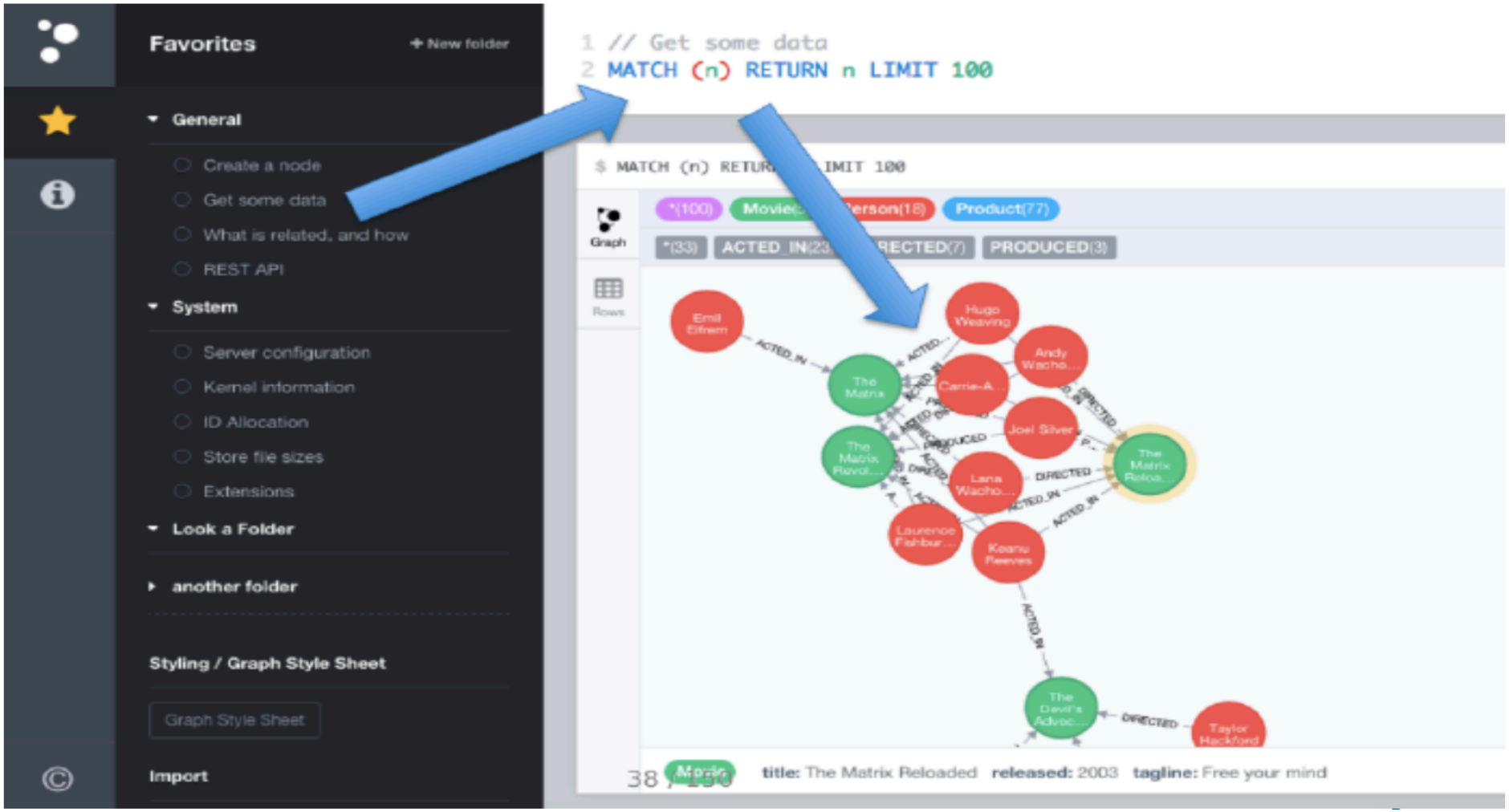
Import dataset

:play movies



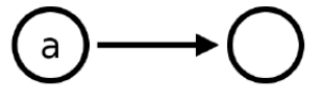
Neo4j browser – Display data

- Example: MATCH (n)-[r]->(n2) RETURN r, n1, n2 LIMIT 25



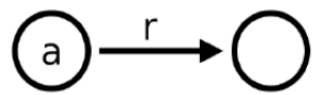
CYPHER
Neo4j Query Language

Two nodes, one relationship



(a)--> ()

```
MATCH (a) --> ()
RETURN a.name
```



(a)-[r]-> ()

```
MATCH (a) –[r]-> ()
RETURN a.name, type(r)
```

Optional match

- We look for the node a with its relationships if they exist

```
OPTIONAL MATCH (a) –[r]-> ()
RETURN a.name, type(r)
```

Two nodes, a known relationship

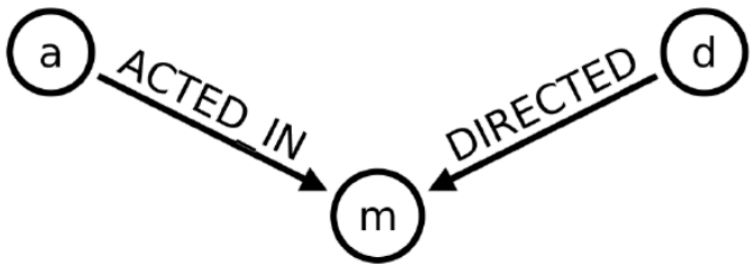
(a)-[:ACTED_IN]-> (m)

```
MATCH (a) –[:ACTED_IN]-> (m)
RETURN a.name, m.title
```

- **Returning the properties of the relations**

```
MATCH (a) –[r:ACTED_IN]-> (m)
RETURN a.name,r.roles, m.title
```

Paths



- **MATCH (a) –[:ACTED_IN]-> (m) <-[:DIRECTED] – (d)**
- **RETURN a.name, m.title, d.name**

Queries on « Movies » example (1/3)

- Display the actor « Tom Hanks »

```
MATCH (tom {name: "Tom Hanks"}) RETURN tom
```

- Display the movie which title is « Cloud Atlas »

```
MATCH (cloudAtlas {title: "Cloud Atlas"}) RETURN cloudAtlas
```

- Display 10 persons

```
MATCH (people:Person) RETURN people.name LIMIT 10
```

- Display movies released in the '90s

```
MATCH (nineties:Movie) WHERE nineties.released > 1990 AND nineties.released < 2000 RETURN nineties.title
```

- Which actors have played in the same movie as Tom Hanks?

```
MATCH (tom:Person {name:"Tom Hanks"})-[:ACTED_IN]->(m)<-[:ACTED_IN]-(coActors) RETURN coActors.name
```

Queries on « Movies »

- **Display Tom Hanks' movies**
- **Who directed the film "Cloud Atlas"?**
- **Which director also played in a movie?**

```
MATCH (tom:Person {name: "Tom Hanks"})-[:ACTED_IN]->(tomHanksMovies)
    RETURN tom,tomHanksMovies

MATCH (cloudAtlas {title: "Cloud Atlas"})<-[:DIRECTED]-(directors) RETURN
    directors.name

MATCH (tom:Person {name:"Tom Hanks"})-[:ACTED_IN]->(m)<-[:ACTED_IN]-
    (coActors) RETURN coActors.name

MATCH (a) –[:ACTED_IN]-> (m) <-[:DIRECTED] – (a)
RETURN a.name, m.title

How people are related to "Cloud Atlas"...
MATCH (people:Person)-[relatedTo]-(:Movie {title: "Cloud Atlas"}) RETURN
    people.name, Type(relatedTo), relatedTo
```

Queries on « Movies » example (2/3)

```
MATCH (a) –[:ACTED_IN]-> (m) <-[:DIRECTED] – (d)
RETURN a.name, m.title, d.name
```

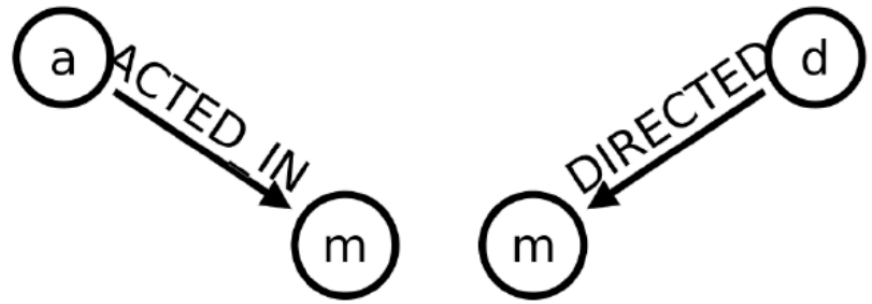
a.name	m.title	d.name
"Keanu Reeves"	"The Matrix"	"Andy Wachowski"
"Keanu Reeves"	"The Matrix Reloaded"	"Andy Wachowski"
"Noah Wyle"	"A Few Good Men"	"Rob Reiner"
"Tom Hanks"	"Cloud Atlas"	"Andy Wachowski"

Alias

```
MATCH (a) –[:ACTED_IN]-> (m) <-[:DIRECTED] – (d)
RETURN a.name AS actor , m.title AS movie, d.name
AS director
```

actor	movie	director
“Keanu Reeves”	“The Matrix”	“Andy Wachowski”
“Keanu Reeves”	“The Matrix Reloaded”	“Andy Wachowski”
“Noah Wyle”	“A Few Good Men”	“Rob Reiner”
“Tom Hanks”	“Cloud Atlas”	“Andy Wachowski”

More queries



1st way

```
MATCH (a) –[:ACTED_IN]-> (m), (m) <-[:DIRECTED] – (d)
RETURN a.name, m.title, d.name
```

2nd way:

```
MATCH (a) –[:ACTED_IN]-> (m), (d) -[:DIRECTED] –> (m)
RETURN a.name, m.title, d.name
```

Aggregation functions

- Count(x) – The number of occurrences
- Min(x) – minimum value
- Max(x) – maximum value
- Avg(x) – average
- Sum(x) – sum
- Collect(x) – Aggregates data in a table

Example – count(*)

```
MATCH (a) –[:ACTED_IN]-> (m) <-[:DIRECTED] – (d)
RETURN a.name, d.name, count(*)
```

a.name	d.name	count(*)
"Aaron Sorkin"	"Rob Reiner"	2
"Keanu Reeves"	"Andy Wachowski"	3
"Hugo Weaving"	"Tom Tykwer"	1

```
MATCH (a) –[:ACTED_IN]-> (m) <-[:DIRECTED] – (d)
RETURN a.name AS actor, d.name AS director , count(m)
AS count
```

SORT and limit

```
MATCH (a) –[:ACTED_IN]-> (m) <-[:DIRECTED] – (d)
RETURN a.name AS actor, d.name AS director ,
count(m) AS count
ORDER BY count DESC
LIMIT 5
```

Aggregation - collect

```
MATCH (a) –[:ACTED_IN]-> (m) <-[:DIRECTED] – (d)
RETURN a.name AS actor, d.name AS director ,
collect (m.title) AS list
```

Find all the nodes

```
MATCH (n)
RETURN n
```

Directors who directed movies with Tom Hanks as actor

```
MATCH (tom:Person) – [:ACTED_IN] ->
(movie:Movie), (director:Person) – [:DIRECTED] ->
(movie:Movie)
WHERE tom.name="Tom Hanks"
RETURN director.name
```

director.name
Mike Nichols
Robert Zemeckis
Penny Marshall
Robert Zemeckis
Ron Howard
Frank Darabont
Ron Howard

DISTINCT

```
MATCH (tom:Person) – [:ACTED_IN] ->
(movie:Movie), (director:Person) – [:DIRECTED] ->
(movie:Movie)
WHERE tom.name="Tom Hanks"
RETURN DISTINCT director.name
```

Index creation

- The 'Person' nodes, indexed by their 'name'

```
CREATE INDEX ON :Person(name)
```

- The nodes 'Movie', indexed by their 'title'

```
CREATE INDEX ON :Movie(title)
```

Conditions

- Find movies where Tom Hanks and Kevin Bacon played

- MATCH (tom:Person) –[:ACTED_IN] -> (movie), (kevin:Person)-[:ACTED_IN]->(movie)
- WHERE tom.name="Tom Hanks " AND kevin.name= "Kevin Bacon"
- RETURN movie.title

Conditions on the properties

- The films where Keanu Rives played the role of "Neo »

```
MATCH (actor:Person) –[r:ACTED_IN] -> (movie)
WHERE actor.name= ”Keanu Reeves " AND  ”Neo ”
   IN (r.roles)
RETURN movie.title
```

- 2nd solution:

```
MATCH (actor:Person) –[r:ACTED_IN] -> (movie)
WHERE actor.name= ”Keanu Reeves " AND ANY( x
   IN r.roles WHERE x=‘’Neo”)
RETURN movie.title
```

Conditions with comparison

- Find actors who have played with Tom Hanks and who are older than him

```
MATCH (tom:Person) –[r:ACTED_IN] -> (movie),
    (a:Person)-[:ACTED_IN]->(movie)
WHERE tom.name= ”Tom Hanks "
    AND a.born < tom.born
RETURN DISTINCT a.name, (tom.born-a.born) AS
    diff
```

Conditions on patterns (1/2)

- Actors who have worked with Gene Hackman and who have previously directed films (are also directors)

```
MATCH (gene:Person)-[:ACTED_IN]->(movie),
      (n)-[:ACTED_IN]->(movie)
WHERE gene.name=''Gene Hackman''
      AND (n)-[:DIRECTED]->()
RETURN DISTINCT n.name
```

CONDITIONS on patterns (2/2)

- Actors who worked with " Keanu Rives ", but not when he played with " Hugo Weaving ’’

```
MATCH (keanu:Person)-[:ACTED_IN]->(movie),
      (n)-[:ACTED_IN]->(movie),
      (hugo:Person)
WHERE keanu.name=« Keanu Reeves » AND
      hugo.name=« Hugo Weaving »
      AND NOT (hugo)-[:ACTED_IN]->(movie)
RETURN DISTINCT n.name
```

String Comparison

```
MATCH (a) –[:ACTED_IN]-> (matrix:Movie)
WHERE matrix.title='The Matrix' AND a.name
CONTAINS 'Emil'
RETURN a.name
```

- =~ "regexp »

CONTAINS

STARTS WITH

ENDS WITH

Exercise

- **Display 5 directors who have directed the largest number of films**

Update with CYPHER

Node creation

```
CREATE (p:Person {name: 'Me'})
```

```
MATCH (p:Person)
WHERE p.name='Me'
RETURN p
```

- Example with 2 properties:

```
CREATE (m:Movie {title: 'Mystic River', released: 1993})
```

Creation with MERGE

```
MERGE (p:Person {name: 'Me'})
RETURN p
```

- Guarantees unique creation

With some options:

```
MERGE (p:Person {name: 'Me'})
ON CREATE SET p.created=timestamp()
ON MATCH SET p.accessed= colaesce(p.accessed,0)+1
RETURN p
```

ON CREATE SET – Executed when creating

ON MATCH SET – Executed when Matching

Adding properties

```
MATCH (p:Person)
WHERE p.name='Me'
//Add property
SET p.born='1980'
RETURN p
```

Modifying properties

```
MATCH (p:Person)
WHERE p.name='Me'
//ajout de la propriété
SET p.born='1985'
RETURN p
```

Adding Relationships

```
MATCH (movie:Movie),(kevin:Person)
WHERE movie.title='Mystic River' AND kevin.name='Kevin Bacon'
//creation of relationship
MERGE (kevin) –[:ACTED_IN {roles:['Sean']}]-> (movie)
```

```
MATCH (kevin)-[:ACTED_IN] -> (movie)
WHERE kevin.name ='Kevin Bacon'
RETURN movie.title
```

Modifying a Relationship Property

- Change the role of Kevin Bacon in the movie Mystic River from "Sean" to "Sean Devine »

```
MATCH (kevin:Person)-[r:ACTED_IN] -> (movie:Movie)
WHERE kevin.name =’Kevin Bacon’ and movie.title=‘Mystic River’
SET r.roles=[‘Sean Devine’]
RETURN r.roles
```

Delete a Node

```
MATCH (emil:Person)
WHERE emil.name = 'Emil Eifrem'
DELETE emil
```

- The relationships still exist

Delete a Node

```
MATCH (emil:Person) –[r]-()
WHERE emil.name = ‘Emil Eifrem’
DELETE r
```

Deleting nodes and all relationships

```
OPTIONAL MATCH (emil) –[r]-()
where emil.name = "Emil Eifrem"
DELETE emil, r
```

NOT TO DO!!!

- **Deleting all content from the database**

```
MATCH (n)
OPTIONAL MATCH (n) –[r]-()
DELETE n, r
```

Exercise

- **Add the KNOWS relationship between all the actors in the same movie**

```
MATCH (a:Person)-[:ACTED_IN]->()<-[:ACTED_IN]-(b:Person)
MERGE (a)-[:KNOWS]-(b);
```

Recommendation

Exercise

- Recommend 3 actors with whom Keanu Reeves could work but this has never been the case

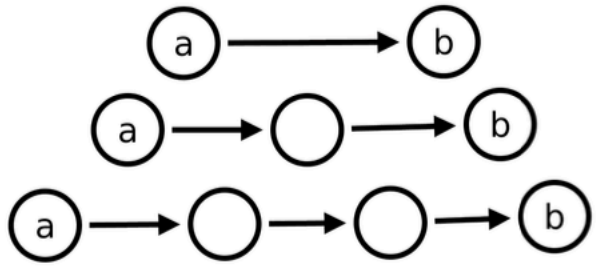
```
MATCH (keanu:Person)-[:ACTED_IN]->()<-[:ACTED_IN]-(c),
(c)-[:ACTED_IN]->()<-[:ACTED_IN]-(coc)
WHERE keanu.name="Keanu Reeves"
AND coc <> keanu
AND NOT ((keanu)-[:ACTED_IN]-()<-[:ACTED_IN]-(coc))
RETURN coc.name, count(coc)
ORDER BY count(coc) DESC
LIMIT 3
```

Matching many relationships

```
MATCH (a)-[:ACTED_IN|:DIRECTED]->()<-
[:ACTED_IN|:DIRECTED]-(b)
MERGE (a)-[:KNOWS]-(b);
```

(Creation of the KNOWS relationship between the actors and directors who worked together)

Path with variable length



(a)-[*n]->(b)

- **Friends of friends:**

```
MATCH (keanu:Person)-[:KNOWS*]->(fof)
WHERE keanu.name="Keanu Reeves" AND NOT
(keanu)-[:KNOWS]-(fof)
RETURN DISTINCT fof.name;
```

Length of the relationship

```
MATCH p=shortestpath((keanu:Person)-[:KNOWS*]->(demi:Person))
WHERE keanu.name="Keanu Reeves" AND NOT demi.name="Demi moore"
RETURN length(rels(p));
```