Andrea Canepa
Stefano Rebora

ADM – ASSIGNMENT 4

Exercise 1

1. Return a given person, retrieved by a condition on his/her name specified in the query.

```
MATCH(p:Person)WHERE p.name ='Hugo Weaving' return p;
```

2. Return the titles of the movies in which a given person, whose name is specified in the query, acted.

```
MATCH (p:Person{name:'Keanu Reeves'})-[:ACTED_IN]->(m:Movie) RETURN
m.title;
```

3. Return the persons interpreted a specific role, specified in the query, in any movie. Choose a role which is present in more than a movie or interpreted by more than one actor.

```
MATCH (p:Person)-[a:ACTED_IN]->(m:Movie) WHERE 'Neo' in a.roles RETURN
p.name,m.title;
```

4. Return the pairs of actors that acted together in any movie.

```
MATCH (p1:Person)-[a1:ACTED_IN]->(m:Movie)<-[a2:ACTED_IN]-(p2:Person)
WHERE p1.name < p2.name RETURN p1.name,p2.name,m.title;
```

5. Return the persons that are related somehow (by a relationship of any type) to Kevin Bacon, also displaying information about the relationship.

```
MATCH (p1:Person{name:'Kevin Bacon'})-[r]-(p2:Person) RETURN p2,r;
```

6. Return the persons that are related somehow (by a path of any length and of any type) to Kevin Bacon, limiting the result to the first 25 results. Rerun the query without the limit.

```
MATCH (p1:Person{name:'Kevin Bacon'})-[*]-(p2:Person) WHERE p1 <> p2
RETURN p2.name LIMIT 25;
```

Without the limit the query does not terminate because the system is not able to explore all the graph.

7. Return the pairs of nodes that are related by two different relationships.

```
MATCH (n1)-[r1]-(n2)-[r2]-(n1) where n1 <> n2 AND type(r1) <> type(r2)
return n1,n2;
```

8. Return the persons that directed a movie but did not wrote it.

```
MATCH (p:Person)-[:DIRECTED]->(m:Movie) WHERE not((p)-[:WROTE]-(m))
return p,m;
```

9. Return the shortest path between any pair of movies traversing at least an edge of type acted in.

```
MATCH p = shortestpath((m1:Movie)-[*]-(m2:Movie)) WHERE m1 <> m2 AND
ANY(r in relationships(p) where type(r) ="ACTED_IN" ) return m1,m2,p;
```

10. Return the three nodes with the highest number of relationships.

```
MATCH (n1)-[r]-() WITH n1,count(r) as numbers RETURN n1,numbers ORDER BY
numbers DESC LIMIT 3;
```

11. Return the length of the shortest path between two movies of your choice.

```
MATCH p = shortestpath((m1:Movie{title:'The Matrix'})-[*1..5]-(m2:Movie{title:'The Matrix Reloaded'}))
return p,length(p);
```

The system has suggested limiting the numbers of hops and we have chosen five.

12. Return the movies that are at most 4 hops according to any relationship from The Matrix.

```
MATCH (m1:Movie{title:'The Matrix'})-[*1..4]-(m2:Movie) WHERE m1<>m2
return m2;
```

Exercise 2

1. add a new movie of your choice and its actors

```
CREATE (SpiderMan:Movie {title:'Spiderman', released:2002, tagline:'An
Ordinary Man.An Extraordinary Power.'})
CREATE (Tobey:Person {name:'Tobey Maguire', born:1975})
CREATE (Willem:Person {name:'Willem Dafoe', born:1955})
CREATE (Kirsten:Person {name:'Kirsten Dunst', born:1982})
CREATE
  (Tobey)-[:ACTED_IN {roles:['Spiderman']}]->(SpiderMan),
  (Willem)-[:ACTED_IN {roles:['Green Goblin']}]->(SpiderMan),
  (Kirsten)-[:ACTED_IN {roles:['Mary Jane']}]->(SpiderMan) ;
```

IMPACT:
These statements add a node for the new film and a node for each actor with the related properties. Then, they add a relationship of type “ACTED_IN” with a property roles (the property is a pair KEY VALUE) from each actor to the film.

2. add a property to a movie or to an actor of your choice

```
MATCH (p:Person{name:'Kirsten Dunst'}) SET p.gender = 'female' RETURN p;
```

IMPACT:
This statement adds a pair key-value (the property gender = female) to the set of properties of the node with label “person” and name “Kirsten Dunst”.

3. add a property (with a default or computed value) to all the nodes with a certain characteristic of your choice (making used of the FOREACH statement).

```
MATCH (p:Person)-[:ACTED_IN]-> (m:Movie{title:'Spiderman'}) WITH
collect(p) as actors FOREACH (node in actors| SET node.myAttribute =
'something');
```

IMPACT:
This statement adds a pair key-value (the property myAttribute = something) to the set of properties of each node representing an actor in Spiderman.

Exercise 3

Extend the graph database with information about the books from which movies derive and corresponding author, publishing information, and significant differences between the book and the movie.

1. propose a way of representing this information in the property graph mode

We can represent the books as nodes with label 'Book' and with a set of properties representing the publishing information (publisher, date, edition etc).
Then we can indicate:

- the book author using a relationship of type 'IS_THE_BOOK_AUTHOR' from a node with label 'person' to a node with label 'book'.
- the book from which a film derives using a relationship of type 'DERIVES_FROM' from a node with label 'film' to a node with label 'book'.
- the differences between the book and the related film using a relationship of type 'DIFFERENCES' (with a set of properties indicating the main differences) from a node with label 'book' to a node with label 'film'**.**

2. add at least an instance for each introduced concept

```
CREATE (DaVinci:Book {title:'The Da Vinci Code',
 publ_date:'02-06-2003',num_of_editions:6, genre:'thriller' })
CREATE(Dan:Person{name:'Dan Brown', born:1964})
CREATE (Dan)-[:IS_THE_BOOK_AUTHOR]->(DaVinci);


MATCH (m:Movie{title:"The Da Vinci Code"}),(b:Book{title:'The Da Vinci
Code'})
CREATE (m)-[:DERIVES_FROM]->(b)
CREATE (b)-[:DIFFERENCES{some_type_of_diff:'diff description'}]->(m);
```

3. formulate in natural language and in Cypher a query involving the information you added.

Find all films that derive from a book of Dan Brown:

```
MATCH (m:Movie)-[:DERIVES_FROM]->(b:Book)-[:IS_THE_BOOK_AUTHOR]-
(p:Person) WHERE p.name='Dan Brown' RETURN m;
```