*UNIVERSIDADE NOVA DE LISBOA*
*Faculdade de Ciências e Tecnologia*
***Departamento de Informática*** **MIEI**

# Data Modeling
# 2017 / 2018

27, October 2017
1:45 hours

**1 - QUESTION [6]**
Consider the Movies Database from Neo4J tutorial, that includes: **node labels** Movie, Person; **relationships types** ACTED_IN, DIRECTED, FOLLOWS, PRODUCED, REVIEWED, WROTE; and **properties**: born, name, rating, released, roles, summary, tagline.

a) Write a cypher query to find all the actors that acted with Tom Hanks in more than one movie. Present two variants, one allowing repetitions and another avoiding repetitions .
b) Write a cypher query to find all movies where the director also acted.
c) Write a cypher query to return the list of movies with an average rate larger than 80.
d) Explain what will be returned by the following queries and indicate what you can say about the number of returned results

(i) `MATCH (x) -- () RETURN x`

(ii) `MATCH (x) -- () -- () -- (x) RETURN x`

## 2 - QUESTION [7]

Consider the following N3 data of the RDF graph **O**:

```
@prefix : <http://foo.ex/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

:p1 :name "John"; :drives :c1, :c2 .
:p2 :name "Mary"; :rides_in :c1; :drives :c1, :c2, :c3 .
:c3 rdfs:label "AA-00-00" .
:c2 rdfs:label [] .
:p1 :likes :p2 .
:p2 :likes :p1 .
[]  :likes :p1; :rides_in :c2 .
```

a) Draw a graph draw the corresponding RDF graph.

Consider the following 4 RDF graphs where the default prefix **:** is associated to IRI `< http://foo.ex/>`

| | |
|---|---|
| 1. `_:x :likes _:y .`<br>`_:x :drives _:z.`<br>`_:y :rides_in _:z .` | 2. `_:x :rides_in _:y .`<br>`_:x :drives _:y .`<br>`_:y rdfs:label _:z .` |
| 3. `_:x :likes _:y .`<br>`_:y :likes _:z .`<br>`_:z :name "John" .` | 4. `_:x :likes _:x .`<br>`_:x :drives _:y .`<br>`_:y rdfs:label _:z .` |

b) From the graphs specified previously indicate the ones which are simply entailed by graph **O**. Justify in detail by using any of the methods studied for simple entailment.

c) The initial data in **O** is extended with the following schema information where the rdfs prefix is associated with the usual URI identifying RDF Schema.

```
1. :likes rdfs:domain :Person .
2. :drives rdfs:subPropertyOf :transported_in .
3. :rides_in rdfs:subPropertyOf :transported_in .
4. :transported_in rdfs:domain :Entity .
5. :transported_in rdfs:range :Vehicle .
6. :Person rdfs:subClassOf :Agent .
7. :Vehicle rdfs:subClassOf :Entity .
```

Using the inference rules for RDF Schema entailment, check whether the following graph is entailed by the initial RDF graph **O** when extended by the schema information. Justify your answer.

```
[ a :Agent, :Entity ] :transported_in [ a :Entity ] .
```

**3 - QUESTION [7]**

Consider now the RDF graph of previous group, extended with the RDF Schema.

a) What is the result of the SPARQL query below, in case the entailment regime of the endpoint is: (i) Simple Entailment? (ii) RDF Entailment?

```
SELECT ?person
WHERE {?person rdf:type :Person; :age ?age. Filter (?age > 30)}
```

In the next questions assume RDFS Entailment.

b) Specify a SPARQL query for obtaining: the names of the persons who drive the vehicle AA-00-00 (i.e. the vehicle whose rdfs:label is "AA-00-00") and are more than 30 years old.

c) Specify a SPARQL query for obtaining: the persons that like "John" and, when available, the vehicles that each of those persons drive.

d) Specify a SPARQL query for obtaining: the number of vehicles that transport each of the persons. i.e. for each person, on how many vehicles is that person transported in.

# Annex

### RDFS entailment patterns

Let aaa, bbb, ... be IRIs; uuu, vvv, ... be IRIs or bnodes; and xxx, yyy, ... be IRIs, bnodes or literals.

| | **If S contains:** | **then S RDFS entails recognizing D:** |
|---|---|---|
| ***rdfs1*** | any IRI aaa in D | aaa `rdf:type rdfs:Datatype .` |
| ***rdfs2*** | aaa `rdfs:domain` xxx `.`<br>uuu aaa yyy `.` | uuu `rdf:type` xxx `.` |
| ***rdfs3*** | aaa `rdfs:range` xxx `.`<br>uuu aaa vvv `.` | vvv `rdf:type` xxx `.` |
| ***rdfs4a*** | uuu aaa xxx `.` | uuu `rdf:type rdfs:Resource .` |
| ***rdfs4b*** | uuu aaa vvv`.` | vvv `rdf:type rdfs:Resource .` |
| ***rdfs5*** | uuu `rdfs:subPropertyOf` vvv `.`<br>vvv `rdfs:subPropertyOf` xxx `.` | uuu `rdfs:subPropertyOf` xxx `.` |
| ***rdfs6*** | uuu `rdf:type rdf:Property .` | uuu `rdfs:subPropertyOf` uuu `.` |
| ***rdfs7*** | aaa `rdfs:subPropertyOf` bbb `.`<br>uuu aaa xxx `.` | uuu bbb xxx `.` |
| ***rdfs8*** | uuu `rdf:type rdfs:Class .` | uuu `rdfs:subClassOf rdfs:Resource .` |
| ***rdfs9*** | uuu `rdfs:subClassOf` xxx `.`<br>vvv `rdf:type` uuu `.` | vvv `rdf:type` xxx`.` |
| ***rdfs10*** | uuu `rdf:type rdfs:Class .` | uuu `rdfs:subClassOf` uuu `.` |
| ***rdfs11*** | uuu `rdfs:subClassOf` vvv `.`<br>vvv `rdfs:subClassOf` xxx `.` | uuu `rdfs:subClassOf` xxx `.` |
| ***rdfs12*** | uuu `rdf:type`<br>`rdfs:ContainerMembershipProperty .` | uuu `rdfs:subPropertyOf rdfs:member .` |
| ***rdfs13*** | uuu `rdf:type rdfs:Datatype .` | uuu `rdfs:subClassOf rdfs:Literal .` |

### Simple entailment patterns

1. If S contains uuu aaa vvv. then one can add _:nnn aaa vvv. where _:nnn identifies a blank node allocated to IRI uuu.
2. If S contains uuu aaa lll. then one can add uuu aaa _:nnn. where _:nnn identifies a blank node allocated to literal or IRI lll.

3. If S contains _:nnn aaa vvv. then one can add uuu aaa vvv. where _:nnn identifies a blank node allocated to IRI uuu by rule 1.

4. If S contains uuu aaa _:nnn. then one can add uuu aaa lll. where _:nnn identifies a blank node allocated to literal or IRI lll by rule 2.