Universidade de Nova de Lisboa **MIEI**

Faculdade de Ciências e Tecnologia **November 9th, 2019**

Data Modelling
Modelação de Dados

Test
Duration: 2 hours

Group 1 [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: **title, born**, **name**, **rating**, **released**, **roles**, **summary**, **tagline**.

a) **[2]** Write a cypher query to check in which movies "Angelina Jolie" and "Brad Pitt" acted or produced together (note that a movie where one of them acted and that the other produced should also be returned).
b) **[2]** Write a cypher query to list the names of pairs of actors who have acted together in at least 4 different movies.
c) **[2]** Write a cypher query to present the list of all actors' names as well as the list of all producers' names (separate columns with the lists of names), whenever they exist. Moreover, the titles of ALL movies should also appear in the result.

Group 2 [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#> .
@prefix fam: <http://www.family.org/onto#> .

 [] fam:is-mother-of :c, :m.
 :c rdfs:label "Charles" ; fam:is-faher-of :r ;
    fam:has-wife [ fam:has-husband  :c; fam:is-mother-of :r ] .
 :m rdfs:label "Manuel" ; fam:is-faher-of _:i,_:f .
 :r a fam:Female .
_:i fam:brother-of _:f .
_:f fam:brother-of _:i .
```

2a) [1] Draw the corresponding RDF graph.

2b) [3] Consider the next four RDF graphs where the default prefix : is associated to IRI < http://foo.ex/>

1. _:x fam:has-wife _:y . _:x fam:is-mother-of _:z . _:y fam:is-mother-of _:z .	2. _:x rdfs:label _:y . _:x fam:is-father-of _:z . _:z fam:is-brother-of _:z .
3. _:x fam:is-mother-of _:y . _:x fam:is-mother-of _:z . _:y rdfs:label _:w . _:z rdfs:label _:w .	4. _:x fam:is-mother-of _:y . _:x fam:is-mother-of _:z . _:y fam:is-mother-of _:t . _:z fam:is-father-of _:u .

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 RDF simple entailment.

2c) [3] 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.  fam:is-mother-of  rdfs:domain        fam:Female .
2.  fam:is-father-of  rdfs:domain        fam:Male .
3.  fam:is-sister-of  rdfs:domain        fam:Female .
4.  fam:is-brother-of rdfs:domain        fam:Male .
5.  fam:has-wife      rdfs:range         fam:Female .
6.  fam:has-husband   rdfs:range         fam:Male .
7.  fam:is-mother-of  rdfs:subPropertyOf fam:parent-of .
8.  fam:is-father-of  rdfs:subPropertyOf fam:parent-of .
9.  fam:has-husband   rdfs:subPropertyOf fam:married-to .
10. fam:has-wife      rdfs:subPropertyOf fam:married-to .
11. fam:Male          rdfs:subClassOf    fam:Person .
12. fam:Female        rdfs:subClassOf    fam:Person .
```

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 fam:Person ] fam:is-parent-of [ fam:is-parent-of [ a fam:Male ] ].
```

Group 3 [7]

3a) [2] Present the solutions to the following SPARQL query when applied to graph **O of the previous group**, justifying the obtained solutions using the studied SPARQL algebra. Treat blank nodes in **O** as if they were distinct and different IRIs.

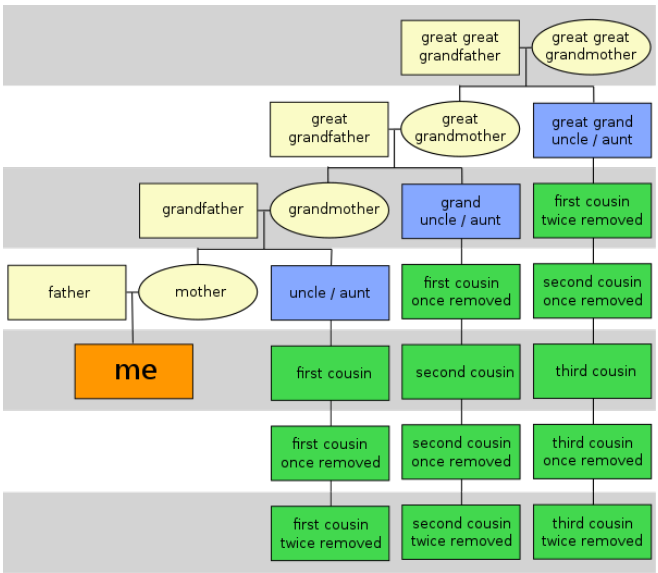
```
PREFIX : <http://foo.ex/> .
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
PREFIX fam: <http://www.family.org/onto#> .

SELECT ?x ?y ?z ?n
WHERE { { { ?x fam:is-father-of ?y } UNION { ?x fam:is-mother-of ?y } }
        { { ?x fam:is-father-of ?z } UNION { ?x fam:is-mother-of ?z } }
        FILTER ( ?y != ?z )
        OPTIONAL
        { ?x rdfs:label ?n }
} .
```

In the questions below you should produce queries that would produce results for RDF graphs of the form of the one in Group 2. Assume that entailment mode is simple but the graph has been previously closed with all the triples obtained with RDFS closure rules.

3b) [2] Construct a SPARQL query that retrieves all persons (and their sex if it is known) that have common parents.

3c) [2] Present a SPARQL query to obtain the persons without two parents.

3d) [1] Construct a SPARQL query to obtain all cousins of all persons, according to the figure on the left (note that the figure only shows the mother side, but the same applies for the father's ancestors). Use property path expressions to construct the query. Note that according to the English terminology, these do not include the ones related by marriage of ancestors.



THE END

Annex

Let **aaa**, **bbb**,... be IRIs, **uuu**, **vvv**, ... be blank nodes or IRIs, and **xxx**, **yyy**, ... be blank nodes, IRI references or literals.

Simple Entailment rules

Rule Name	If S contains	then add
se1	uuu aaa xxx .	uuu aaa _:nnn . where _:nnn designates a blank node allocated to xxx by rules se1 or se2.
se2	uuu aaa xxx .	_:nnn aaa xxx . where _:nnn designates a blank node allocated to uuu by rules se1 or se2.

RDF Entailment rules

Name	If S contains	then add
GrdfD1	xxx aaa "sss"^^ddd . for ddd in D	"sss"^^ddd rdf:type ddd .
rdfD2	uuu aaa yyy .	aaa rdf:type rdf:Property .

RDFS Entailment rules

	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 . uuu aaa yyy .	uuu `rdf:type` xxx .
rdfs3	aaa `rdfs:range` xxx . 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 . vvv `rdfs:subPropertyOf` xxx .	uuu `rdfs:subPropertyOf` xxx .
rdfs6	uuu `rdf:type rdf:Property .`	uuu `rdfs:subPropertyOf` uuu .
rdfs7	aaa `rdfs:subPropertyOf` bbb . uuu aaa xxx .	uuu bbb xxx .
rdfs8	uuu `rdf:type rdfs:Class .`	uuu `rdfs:subClassOf rdfs:Resource .`
rdfs9	uuu `rdfs:subClassOf` xxx . vvv `rdf:type` uuu .	vvv `rdf:type` xxx.
rdfs10	uuu `rdf:type rdfs:Class .`	uuu `rdfs:subClassOf` uuu .
rdfs11	uuu `rdfs:subClassOf` vvv . vvv `rdfs:subClassOf` xxx .	uuu `rdfs:subClassOf` xxx .
rdfs12	uuu `rdf:type` `rdfs:ContainerMembershipProperty .`	uuu `rdfs:subPropertyOf rdfs:member .`
rdfs13	uuu `rdf:type rdfs:Datatype .`	uuu `rdfs:subClassOf rdfs:Literal .`

If needed you can use the rdfs7x rule:

rdfs7x	aaa rdfs:subPropertyOf vvv . uuu aaa yyy .	uuu vvv yyy .