Lab1: Semantic Web

Exercise 1 Model the dataset below with an RDF graph :

Name	Director	Release Date	Actor
Pulp Fiction	Quentin Tarantino	1994	John Travolta
Taste of Cherry	Abbas Kiarostami	1997	Homayoun Ershadi
Saturday Night Fever	John Badham	1977	John Travolta

Festival Name	Year	Name	Country
Cannes Film Festival	1994	Pulp Fiction	USA
Cannes Film Festival	1997	Taste of Cherry	Iran

Exercise 2 Consider the following RDF document in Turtle format with information about celestial bodies.

```
@prefix  ex: <http://example.org/> .
ex:sun          ex:radius      "1.392e6"^^xsd:double ;
                ex:satellite   ex:mercury , ex:venus , ex:earth , ex:mars  .
ex:mercury      ex:radius      "2439.7"^^xsd:double  .
ex:venus        ex:radius      "6051.9"^^xsd:double  .
ex:earth        ex:radius      "6372.8"^^xsd:double  ;
                ex:satellite    ex:moon .
ex:mars         ex:radius       "3402.5"^^xsd:double  ;
                ex:satellite    ex:phobos, ex:deimos  .
ex:moon         ex:name         "Lune@fr", "Moon@en"  ;
                ex:radius       "1737.1"^^xsd:double  .
ex:phobos       ex:name         "Phobos" .
ex:deimos       ex:name         "Deimos" .
```

Q1. Please give the Graph-based Data Model [1] representation for the RDF triples above.

Q2. Consider the following SPARQL query for "Object which orbits around the sun or around a satellite of the sun".

```
PREFIX ex: <http://example.org/> .
SELECT ?object
WHERE {
        { ex:Sun ex:satellite ?object . } UNION
        { ex:Sun ex:satellite ?object_tmp .
          ?object_tmp ex:satellite ?object . }
      }
```

1. See `https://www.w3.org/TR/rdf11-concepts/` and `https://www.w3.org/TR/rdf-schema/`

Query 1	`PREFIX ex: <http://example.org/> .`
	`SELECT ?object ?center`
	`WHERE {`
	`{ ?object ex:radius ?rad . }`
	`OPTIONAL { ?center ex:satellite ?object . }`
	`FILTER (?rad > 3000)`
	`}`

Query 2	`PREFIX ex: <http://example.org/> .`
	`SELECT ?object`
	`WHERE {`
	`?object ex:satellite ?satellite .`
	`?satellite ex:name ?name .`
	`?center ex:satellite ?object .`
	`?center ex:radius ?rad .`
	`FILTER (langMATCHES(LANG(?name), "en"))`
	`FILTER (?rad > 3000)`
	`}`

Query 3	`PREFIX ex: <http://example.org/> .`
	`SELECT ?object`
	`WHERE {`
	`?object ex:satellite ?satellite1 .`
	`?object ex:satellite ?satellite2 .`
	`FILTER (!sameTerm(?satellite1, ?satellite2))`
	`}`

TABLE 1 – Sparql queries

Please give the answer to this query based on the RDF document above.

Q3. Consider the three SPARQL queries given in Table 1 and the following three questions in natural language.

(a) Objects with a satellite for which an English name is given, and which furthermore are satellites of an object with radius greater than 3000 (km).

(b) Objects with two or more satellites. Assume for this that different URIs denote different objects.

(c) Objects with a radius greater than 3000 (km) together with the object – if it exists – of which they are a satellite.

For each question above, which is its corresponding SPARQL query from Table 1 ?

Exercise 3 Write SPARQL queries [2] to answer the movie related questions below using the displayed Turtle data.

- Names of all movies.
- Names of movies and directors sorted descending by the year the movie appeared.
- Names and directors of all movies before1996.
- Names all movies whose genre is Crime.
- Names of all actors who are above 50 (at 2016).
- Names of all movies whose directors are above 70 (at 2016).

2. Refer to `https://www.w3.org/TR/sparql11-query/` for details of the W3C standard.

```
@prefix ex: <http://example.org/movies/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

ex:m1   rdf:type ex:Movie;
        ex:genre ex:Drama;
        ex:year "2006"^^xsd:gYear;
        rdfs:label "Marie Antoinette";
        ex:country ex:USA;
        ex:director ex:p1;
        ex:actor ex:p2 .
ex:p1   rdf:type ex:Director;
        foaf:familyName "Coppola";
        foaf:givenName  "Sofia";
        ex:birthYear "1971"^^xsd:gYear .
ex:p2   rdf:type ex:Actor;
        foaf:familyName "Dunst";
        foaf:givenName  "Kirsten";
        ex:birthYear "1982"^^xsd:gYear .
ex:p5   rdf:type ex:Actor;
        foaf:familyName "De Niro";
        foaf:givenName  "Robert";
        ex:birthYear "1943"^^xsd:gYear .
ex:m2 rdf:type ex:Movie;
        ex:genre ex:Crime;
        ex:year "1995"^^xsd:gYear;
        rdfs:label "Heat";
        ex:country ex:USA;
        ex:director ex:p3;
        ex:actor ex:p4 , ex:p5.
ex:p3   rdf:type ex:Director;
        foaf:familyName "Mann";
        foaf:givenName  "Michael";
        ex:birthYear "1943"^^xsd:gYear .
ex:p4   rdf:type ex:Actor;
        foaf:familyName "Pacino";
        foaf:givenName  "Al";
        ex:birthYear "1940"^^xsd:gYear .
```

Exercise 4 Install Fuseki (go to https ://jena.apache.org/download to download, then start "fuseki-server" on the command line and go to http ://localhost :3030). Load the data from Exercise 2 (in .ttl format) and try the corresponding SPARQL queries. Experiment with leaving out and/or adding further triple patterns.

Exercise 5 Create SPARQL queries for DBpedia.

You can test the queries on https ://dbpedia.org/sparql or https ://live.dbpedia.org/sparql, or in code (see an example `https://www.lri.fr/~ma/APP5_SW/ex.py`). Please note that DBpedia data in this endpoint is subject to change (in particular the live version will be frequently updated). In some cases, this can result in the queries below no longer working and/or the changes in properties or classes. For each question, we provide you with the DBpedia entities, which are not in the RDF, RDFS or OWL namespace (e.g. rdf :type is not listed). The following prefixes should be used (note that the DBpedia endpoint already has those predefined as listed in https ://dbpedia.org/sparql ?nsdecl so you do not need to manually add them in your SPARQL queries) :

```
PREFIX rdf : <http://www.w3. org/1999/02/22?rdf?syntax?ns#>
```

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf?schema#>
PREFIX res: <http://dbpedia.org/resource/>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX yago: <http://dbpedia.org/class/yago/>
```

We distinguish between individuals (specific entities e.g. Angela Merkel), classes (sets of individuals) and properties (connecting individuals), so you can more easily define your queries.

- How tall is Claudia Schiffer ?
  - Classes : –
  - Properties : dbo :height
  - Individuals : res :Claudia_Schiffer
- Give me all female Russian astronauts.
  - Classes : yago :RussianCosmonauts, yago :FemaleAstronauts
  - Properties : –
  - Individuals : –
- How many monarchical countries are there in Europe ?
  - Classes : yago :EuropeanCountries
  - Properties : dbo :governmentType
  - Individuals : –
- Which states of Germany are governed by the Social Democratic Party ?
  - Classes : yago :StatesOfGermany
  - Properties : dbp :rulingParty
  - Individuals : res :Social_Democratic_Party_of_Germany
- Which monarchs of the United Kingdom were married to a German ?
  - Classes : yago :MonarchsOfTheUnitedKingdom
  - Properties : dbo :spouse, dbo :birthPlace
  - Individuals : res :Germany
- Which countries have places with more than two caves ?
  - Classes : dbo :Cave, dbo :Country
  - Properties : dbo :location
  - Individuals :
- Give me all cities in New Jersey with more than 100000 inhabitants.
  - Classes : dbo :City
  - Properties : dbo :isPartOf, dbp :populationTotal
  - Individuals : res :New_Jersey
- Is proinsulin a protein ?
  - Classes : dbo :Protein
  - Properties : –
  - Individuals : res :Proinsulin
- Is Frank Herbert still alive ?
  - Classes : –
  - Properties : dbo :deathDate
  - Individuals : res :Frank_Herbert
- Which mountain is the highest after the Annapurna ?
  - Classes : dbo :Mountain
  - Properties : dbo :elevation
  - Individuals : res :Annapurna