

Examen - web sémantique

Vendredi 7 janvier 2022

(durée : 1h30 – barème : 20 points. Les TP seront notés chacun sur 10. La note de l'examen aura un coefficient de 0,7 et celle des TP de 0,3)

Les documents supports de cours sont autorisés.

1. Questions de cours (répondre de manière synthétique. 5 phrases maximum par réponse) (5pts)

1.1 Les graphes de connaissance RDF et RDFS (1,5 + 1,5 pts)

1.1.1 Que permettent de représenter les classes `rdf:Property` en RDF, `owl:ObjectProperty` et `owl:DatatypeProperty` en OWL ? préciser la signification de chacune de ces classes.

1.1.2 Expliquer la sémantique (donner les inférences possibles sur des instances de C1 et C2) de `rdfs:domain` et `rdfs:range` à partir de
`p rdf:type rdf:Property, p rdfs:domain C1 et p rdfs:range C2`

1.2 **DBPedia** : Donner au moins deux manières dont des connaissances sont extraites de Wikipedia pour construire DBPedia (2pts)

2. Exercice (15 pts) : Représentation et interrogation de données géolocalisées

GeoSPARQL est un vocabulaire minimal en RDF/OWL/SPARQL pour représenter et interroger des données géolocalisées. Il peut être facilement associé à toute ontologie dans laquelle on a besoin de représenter des informations spatialisées. Ce vocabulaire permet de représenter des géométries (forme spatiale d'une entité) grâce aux concepts `geo:Geometry` et `geo:Feature` (toute chose ayant une géométrie) et à la propriété `geo:hasGeometry`. Ce vocabulaire permet aussi d'exprimer des relations spatiales entre entités de type `geo:Feature` à travers des relations calculées entre leurs géométries.

Le cadre ci-dessous présente une petite ontologie préfixée `ex` : . Elle va vous servir pour répondre aux questions 2.1 à 2.6.

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix geo: <http://www.opengis.net/ont/geosparql#> .
@prefix ex: <http://www.example.org/POI#> .
@prefix sf: <http://www.opengis.net/ont/sf#> .

ex:Massif a owl:Class ;
          rdfs:subClassOf ex: PointOfInterest .
ex:Peak   a owl:Class ;
          rdfs:subClassOf ex: PointOfInterest .
ex:Country a owl:Class ;
          rdfs:subClassOf ex: PointOfInterest .
ex:PointOfInterest a owl:Class ;
                  rdfs:subClassOf geo:Feature .
```

```

ex:isIn a owl:ObjectProperty ;
    rdfs:domain ex:Massif ;
    rdfs:range ex:Country .

ex:hasCulminatingPoint a owl:FunctionalProperty ;
    rdfs:domain ex:Massif ;
    rdfs:range ex:Peak ;
    .
ex:isCulminatingPointOf a owl:ObjectProperty ;
    owl:inversePropertyOf ex:hasCulminatingPoint .

ex:hasHeight a owl:DatatypeProperty ;
    rdfs:domain ex:Peak ;
    rdfs:range xsd:integer .

ex:CulminatingPeak a owl:Class ;
    owl:equivalentClass [ a owl:Class ;
                            owl:intersectionOf ( ex:Peak
                                [ a owl:Restriction ;
                                  owl:onProperty ex:isCulminatingPointOf ;
                                  owl:onClass ex:Massif ;
                                  owl:minQualifiedCardinality 1"^^xsd:nonNegativeInteger
                                ]
                            )
    ] .

```

2.1. Représenter des instances (2pts) : Définir, en utilisant la syntaxe Turtle , deux entités :

Ex:GrandPicDeBelledonne, un pic de hauteur 2977 m et de nom « grand pic de Belledonne » situé dans le « massif de Belledonne », et ex:MassifdeBelledonne de type owl:Class avec son étiquette.

Représenter que « Grand Pic de Belledonne » est point culminant du Massif de Belledonne.

2.2. Inférences sur les instances (2pts) : on lance un raisonneur sur cette base de connaissance. Quelle est la nouvelle classe dont ex :MassifdeBelledonne devient une instance ? expliquer pourquoi.

Quelle est la nouvelle classe dont ex :GrandPicDeBelledonne devient une instance ? expliquer pourquoi.

2.3. Requête SPARQL sur ces données (2 pts). On suppose que plusieurs autres pics et massifs sont décrits dans le graphe situé dans l'espace de nom ex : On suppose que pour exprimer qu'un pic se trouve dans un massif, on utilise la propriété ex:isIn entre 2 lieux (des geo:Feature). Ecrire une requête SPARQL qui liste tous les pics du massif de Belledonne, classés par altitude, et affiche leur altitude et leur nom, en considérant que ces 2 informations ne sont peut-être pas toujours renseignées.

2.4. Raisonnement géométrique. (3pts)

On peut exprimer qu'une entité A a une géométrie, et que cette géométrie est représentée par un ensemble de coordonnées : un couple (latitude, longitude) pour un point, une liste de couples pour le polygone dessinant une surface. Ces coordonnées sont au format WKT (un standard en géographie).

A geo:Geometry gA

gA geo:asWkt Awkt

Une fonction `GeoSPARQL`, `geo:sfWithin`, permet d'indiquer si une géométrie est incluse dans une autre : `ex:Awkt geo:sfWithin ex:BWKT` est vraie si le polygone ou le point représentant la géométrie de A est incluse dans celle de B.

Ecrire la même requête qu'en 2.3 (retourner tous les pics se trouvant dans un massif, classés par altitude, avec leur altitude et leur label) en utilisant la fonction `geoSPARQL:geo:sfWithin`.

- 2.5. **Utilisation de COUNT et GROUP-BY (3pts)** : Ecrire une requête qui compte le nombre de pics du massif de Belledonne présents dans la base et dont l'altitude est supérieure à 2000 m. (adapter la requête du 2.4)
- 2.6. (3 pts) Ecrire une requête qui liste tous les massifs de la base et pour chacun, le nombre de pics de ce massif présents dans la base et son pic culminant dans cette base s'il est connu. (adapter la requête du 2.5)

Rappel de la syntaxe de COUNT et GROUP BY pour compter le nombre d'entités en lien avec une autre entité : la requête suivante affiche, pour chaque film, le nombre d'acteurs connus dans la base `tp1` comme jouant dans ce film

```
SELECT ?film (COUNT (distinct ?acteur) as ?count)
WHERE { ?acteur tp1:joueDansFilm ?film }
GROUP BY ?acteur
```