Examen - web sémantique

Mardi 13 décembre 2022

(durée : 1h00 – barême : 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)

1. Questions de cours (4 phrases maximum par réponse) **(5pts)**

1.1 Les graphes de connaissance RDF et RDFS

1.1.1 Que permet de représenter la propriété `rdf:type` ? que peut-on dire de e1 et e2 lorsque e1~rdf:type e2 ? (1pt)

Elle associe 2 ressources du web et exprime que l'une e1 est le 'type' de l'autre e2. On peut regrouper plusieurs ressources comme e1 si on les considère comme similaires selon certains critères, et le type e2 est le nom de cet ensemble.

1.1.2 Que permet de représenter la propriété `rdfs:subClassOf` ? quels sont le domaine et le co-domaine (range) de cette propriété ? que peut-on dire de e1 et e2 lorsque e1~`rdfs:subClassOf` e2 ? que peut-on inférer sur les instances d'e1 et e2 dans ce cas ? (2pts)

Elle associe 2 ressources qui ont été déclarées comme des rdfs:Class, et que l'une est plus générale que l'autre. E1 est un ensemble d'entités ayant des propriétés communes qui est inclus dans e2, ensemble plus vaste d'entités. Si une entité e est instance de e1 (e rdf:type e1) alors elle est aussi instance de e2 (e rdf:type e2).

1.2 Citer 3 des avantages d'associer une ontologie à un graphe de connaissances, que ce soit en matière de représentation ou d'interrogation du graphe (2pts)

Une ontologie contient un ensemble de classes et de propriétés. L'avance de l'associer à un graphe est

1. de donner des types aux entités (nœuds mais aussi arrêtes) du graphe. Les types leur confèrent une sorte de « sémantique » dans la mesure où on peut traiter formellement de manière identique toutes les entités du graphe rattachées au même type.
2. De permettre d'inférer de nouveaux triplets et enrichir le graphe avant de l'interroger
3. De se conformer à des standards et de faciliter la réutilisation du graphe

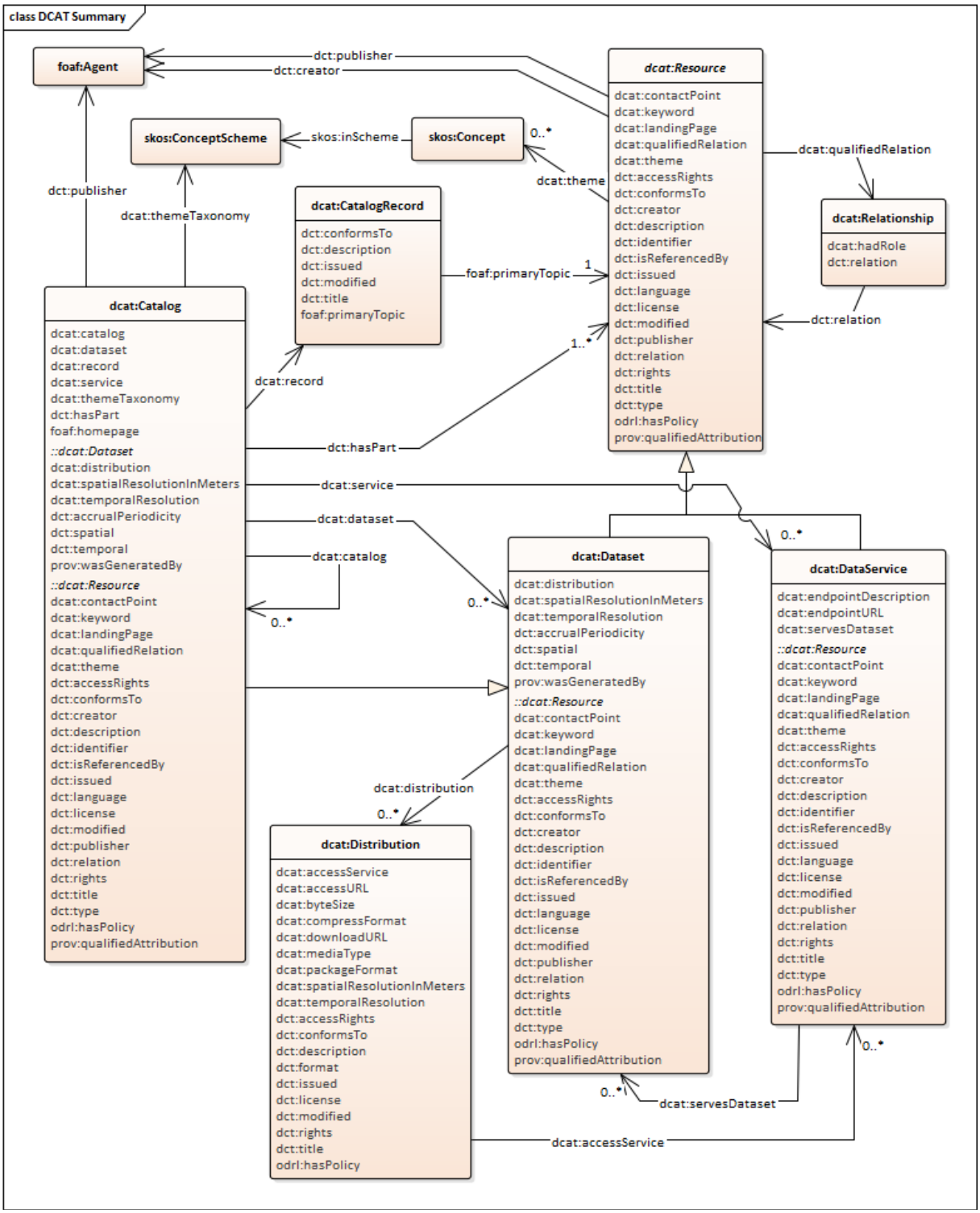
4. Exercice (15 pts) : Représentation et interrogation de métadonnées sémantiques

DCAT est un vocabulaire minimal du W3C préconisé pour représenter et interroger des métadonnées de jeux de données (dataset) présents dans des catalogues. Il contribue à représenter le schéma de ces méta-données. Pour cela, il peut être étendu par d'autres vocabulaires et contraintes pour définir des « profils d'applications », et par des ontologies de domaine pour préciser les entités présentes dans ces jeux de données.
A partir de DCAT, un profil d'application appelé DCAT-AP a été spécifié pour décrire les ensembles de données du secteur public en Europe sur les portails de données. DCAT-AP permet la recherche d'ensembles de données à travers ces portails et facilite la recherche de données publiques au-delà des frontières et des secteurs. .

La figure ci-après est une représentation « type UML » proposée sur le site de la spécification du W3C pour présenter les classes (`owl:Class`) (cadres) et leur dataProperty (propriétés listées

dans les cadres) ainsi que les propriétés reliant les classes (`owl:ObjectProperties`) (arcs étiquetés) de ce vocabulaire en indiquant leur domaine et leur co-domaine (range).

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
Prefix rdf: < http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix geo: <http://www.opengis.net/ont/geosparql#> .
@prefix dc: http://purl.org/dc/elements/1.1 .
@prefix dcat: <http://www.w3c.org/ns/dcat#> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix dctype: <http://purl.org/dc/dcmitype/> .
@prefix foaf: <http:// xmlns.com/foaf/0.1> .
@prefix skos: < http://www.w3.org/2004/02/skos/core#> .
```



2.1. **Représenter des classes et des propriétés (3pts) e**n utilisant la syntaxe Turtle :

. les classes `dcat:Dataset` et `dcat:DataService` comme sous-classes de `dcat:Resource` (ne pas lister toutes les propriétés, juste la relation hiérarchique)

. les objectProperties `dcat:distribution`, `dcat:catalog` et `dcat:dataset` avec leur domaine et co-domaine

```
dcat:Dataset a dcat:Resource
dcat:DataService a dcat:Resource

dcat:distribution rdfs:domain dcat:Dataset;
```

```
                    rdfs:range dcat:Distribution.
dcat:catalog rdfs:domain dcat:Catalog; rdfs:range dcat:Catalog .
dcat:dataset rdfs:domain dcat:Catalog; rdfs:range dcat:Dataset.
```

2.2. **Ecrire des contraintes SHACL en turtle (3pts) :** En utilisant la syntaxe Turtle, écrire un shape SHACL concernant la classe `dcat:Catalog` et indiquant
. qu'un catalogue valide doit avoir un `dct:publisher` déclaré de type `foaf:Agent` ;
. qu'il est composé (`dct:hasPart`) d'au moins une entité de type `dcat:Resource` ;
. qu'un `dcat:Dataset` doit avoir au moins une `dcat:Distribution` (bien que le schéma indique que la valeur min soit 0).

```
ex:CatalogShape
   a sh:NodeShape;
   sh:targetClass dcat:Catalog ;
   sh:property [
     sh:path dct:publisher ;
     sh:minCount  1 ;
     sh:class foaf:Agent  ;
   ] ;
   sh:property [
     sh:path dct:hasPart ;
     sh:minCount  1 ;
     sh:class dcat:Resource  ;
   ]
.
ex:DatasetShape
   a sh:NodeShape;
   sh:targetClass dcat:Dataset ;
   sh:property [
     sh:path dct:distribution ;
     sh:minCount  1 ;
     sh:class dcat:Distribution ;
   ]
  .
```

2.3. **Application des contraintes SHACL à des données (2pts) :** Considérons les instances des classes `dcat:Catalog` et `dcat:Dataset` ci-dessous (adaptées du site de spécification de DCAT). Pour chaque instance de `dcat:Catalog` et de `dcat:Dataset`, indiquer si elle respecte ou non toutes les contraintes écrites en 2.2. Expliquer pourquoi (donner toutes les contraintes violées).

:catalog001

Est de type dcat:Catalog donc doit verifier le ex:CatalogShape.

un catalogue valide doit avoir un `dct:publisher` déclaré de type `foaf:Agent` : il y a bien un dct :publisher mais l'entité n'est pas de type foaf:Person mais de type foaf:Organization ;

un catalogue valide doit est composé (`dct:hasPart`) d'au moins une entité de type `dcat:Resource` : il n'y en aucune

donc non, catalog001 n'est pas valide

:catalog002

Est de type dcat:Catalog donc doit verifier le ex:CatalogShape.

un catalogue valide doit est composé (`dct:hasPart`) d'au moins une entité de type `dcat:Resource` : il n'y en aucune : ce catalogue est composé de 3 datasets de type dcat:Dataset. Donc la contrainte n'est pas respectée

une catalogue valide doit avoir une dcat:publisher de type foaf:Agent. Il a bien un dcat :publisher mais de type foaf :Organization. Cette contrainte n'est pas respectée

donc catalog002 n'est pas valide non plus.

Un `dcat:Dataset` doit avoir au moins une `dcat:Distribution`. On a 3 entités de type `dcat:Dataset`

`:dataset-001 a bien une dcat:distribution donc valide`
`:dataset-002 et :dataset-003 n'ont pas de distribution donc non valides`

2.4. **Contraintes et raisonnement (2pts) :** le vocabulaire FOAF contient la connaissance suivante : `foaf:Organization rdfs:subClassOf foaf:Agent`
Si on lance un raisonneur, quelle nouvelle connaissance concernera l'entité `:transparency-office` ? et quel nouveau type sera associé à `:dataset-001`, `:dataset-002` et `:dataset-003` ?

`:transparency-office a foaf:Agent`
:dataset-001 a dcat:Resource .
:dataset-002 a dcat:Resource .
:dataset-003 a dcat:Resource .

Après inférence, si on lance une vérification SHACL sur le graphe, quelles entités sont conformes aux contraintes et lesquelles ne le sont pas ? pourquoi ?

On aura :catalog002 `dct:publisher :transparency-office avec :transparency-office de type foaf:Agent donc la contrainte ex:CatalogShape`
`Et :catalog002 dcat:hasPart dataset-001 ; :dataset-002 ; :dataset-003 avec les 3 entités de type dcat :Resource`
`donc :catalog002 vérifie ex:CatalogShape`
`:catalog001 n'a pas de dcat:hasPart donc ne vérifie toujours pas le ex:CatalogShape`

2.5. **Requêtes SPARQL sur des datasets (3pts)**. On suppose que plusieurs autres datasets sont décrits dans le graphe dont les exemples ci-dessus sont un extrait.
Ecrire une requête SPARQL qui retourne les 10 premiers datasets de ce catalogue publiés par le ministère des finances, par ordre alphabétique de leur titre, qui affiche leur label (s'ils en ont un), leur titre, leur date de création et la liste des titres des distributions associées avec, pour chacune le titre du service qui permet d'y accéder. (chercher les propriétés à utiliser sur le diagramme pseudo-UML).

```
SELECT ?datasetTitre ?label ?dateCreation ?titreDistrib ?titreService
WHERE {
    :catalog001 dcat :hasPart ?dataset .
    ?dataset a dcat:Dataset .
    ?dataset dct:title ?datasetTitre .
    ?dataset dcat:distribution ?distrib .
    ?distrib dct:title ?titreDistrib .
    ?distrib dcat:accessService ?service .
    ?service dct:title ?titreService .
    ?dataset dct:issued ?dateCreation .
    OPTIONAL (?dataset rdfs:label ?label .)
```

```
}
ORDER BY ?datasetTitre
LIMIT 10
```

2.6. **Utilisation de COUNT et GROUP-BY (2pts)** : Ecrire une requête qui compte le nombre de distributions de jeux de données de ce catalogue créés avant 2020 et dont un des mots clés est « accountability ».

```
SELECT ?nbDistrib ?dataset (COUNT(?distrib) AS ?nbDistrib)
WHERE {
    :catalog001 dcat:hasPart ?dataset .
    ?dataset a dcat:Dataset .
    ?dataset dcat:distribution ?distrib .
    ?dataset dct:issued ?dateCreation .
    ?dataset dct:keyword “accountability” .
    FILTER ( ?dateCreation < 2020^^xsd:date ) .
}
GROUP BY ?dataset
```

```
:catalog001
  a dcat:Catalog ;
  dct:title "Empty Catalog"@en , “Catalogue vide”@fr ;
  rdfs:label "Empty Catalog"@en, “Catalogue vide”@fr ;
  foaf:homepage <http://example.org/catalog1> ;
  dct:publisher :transparency-office ;
  dct:language <http://id.loc.gov/vocabulary/iso639-1/en>  ;

:catalog002
  a dcat:Catalog ;
  dct:title "Imaginary Catalog"@en , “Catalogue imaginaire”@fr ;
  rdfs:label "Imaginary Catalog"@en, “Catalogue imaginaire”@fr ;
  foaf:homepage <http://example.org/catalog2> ;
  dct:publisher :transparency-office ;
  dct:language <http://id.loc.gov/vocabulary/iso639-1/en>  ;
  dcat:dataset :dataset-001  , :dataset-002 , :dataset-003 ;
  dct:hasPart :dataset-001, :dataset-002 , :dataset-003 ;

:transparency-office
  a foaf:Organization ;
  rdfs:label "Transparency Office"@en ;

:dataset-001
  a dcat:Dataset ;
  dct:title "Dataset1"@en ;
  dcat:keyword "accountability"@en, "transparency"@en, "payments"@en ;
  dct:creator :finance-employee-001 ;
  dct:issued "2011-12-05"^^xsd:date ;
  dct:modified "2011-12-15"^^xsd:date ;
  dcat:contactPoint <http://example.org/transparency-office/contact> ;
  dct:temporal <http://reference.data.gov.uk/id/quarter/2006-Q1> ;
  dcat:temporalResolution "P1D"^^xsd:duration ;
  dct:spatial <http://sws.geonames.org/6695072/> ;
  dcat:spatialResolutionInMeters "30.0"^^xsd:decimal ;
  dct:publisher :finance-ministry ;
  dct:language <http://id.loc.gov/vocabulary/iso639-1/en>  ;
```

```
  dct:accrualPeriodicity <http://purl.org/linked-data/sdmx/2009/code#freq-
W>  ;
  dcat:distribution :dataset-001-csv ;

:dataset-002
  a dcat:Dataset ;

:dataset-003
  a dcat:Dataset ;
```

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 ?film