

## **Les graphes de connaissance RDF et RDFS**

### **- Comment la ressource `rdfs:Class` permet-elle de distinguer les différents types de ressources d'un graphe RDF ?**

→ Par la classe `rdfs:Class`, on peut distinguer certaines ressources et les déclarer comme des classes : `Ci rdf:type rdfs:Class`. Cela veut dire que ces ressources `Ci` représentent des ensembles de ressources `r` représentant des entités, et qui sont du type de ces classes : `r rdf:type Ci`

### **- Quelle est la différence entre `rdf:type` et `rdfs:subClassOf` ?**

→ **`rdf:type`** associe un type à une ressource (la ressource appartient à l'ensemble du type) ; **`rdfs:subClassOf`** relie 2 entités de type `rdfs:Class`, l'une étant plus spécifique que l'autre (relation d'inclusion entre 2 ensembles)

### **- Que permet de représenter la propriété `rdf:type` ? Que peut-on dire de `e1` et `e2` lorsque `e1 rdf:type e2` ?**

→ 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.

### **- 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 de `e1` et `e2` dans ce cas ?**

→ 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`).

### **- 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.**

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

1. de donner des types aux entités (nœuds mais aussi arêtes) 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

### **- 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.**

→ **`rdf:Property`** : Représente une relation ou un prédicat entre deux ressources dans un graphe RDF.

→ **`owl:ObjectProperty`** : Représente une relation entre deux individus (ressources) dans OWL, utilisée pour relier des entités complexes.

→ **`owl:DatatypeProperty`** : Représente une relation entre un individu et une donnée littérale (exemple : une chaîne, un nombre) en OWL.

### **- Expliquer la sémantique de `rdf:type` et `owl:subClassOf` ? Que peut-on déduire pour `c` si `c rdf:type C2` et `C2 owl:subClassOf C1`**

→ **`rdf:type`** : Indique qu'une ressource `c` est une instance de la classe `C2` (appartient à l'ensemble défini par `C2`).

→ **`owl:subClassOf`** : Indique une relation hiérarchique entre deux classes (`C2` est un sous-ensemble de `C1`), signifiant que toutes les instances de `C2` sont aussi des instances de `C1`.

→ **Déduction** : Si `c rdf:type C2` et `C2 owl:subClassOf C1`, alors `c rdf:type C1` par transitivité.

### **- Comment représenter en RDFS que `C` est une classe et `i` est une instance de `C` ?**

→ `C rdf:type rdfs:Class`

→ `i rdf:type C`

### **- Comment un raisonnement produit-il de nouvelles connaissances lorsqu'il existe des instances de classes reliées par la propriété `rdfs:subClassOf` ?**

→ Si une instance `i` est de type `C2` (`i rdf:type C2`) et que `C2 rdfs:subClassOf C1`, le raisonneur déduit que `i rdf:type C1`.

## **Utilisation des données liées ouvertes**

**- Quel est l'intérêt d'utiliser l'URI représentant Paris de DBPedia (dbr:Paris) dans une application plutôt que la chaîne de caractère « Paris » pour indiquer une adresse par exemple ?**

→ Grâce à la ressource dbr:Paris on a moins d'ambiguïté qu'avec la chaîne de caractère Paris (qui pourrait être un nom commun ou une ville d'un autre pays de la France). Un autre avantage est de pouvoir collecter des propriétés de Paris, des connaissances grâce aux propriétés associées à dbr:Paris dans DBPedia.

**- Expliquez 2 manières dont un moteur de recherche peut utiliser les données liées ou les bases de connaissances disponibles sous forme de données liées.**

Un ensemble de données liées peut servir à un moteur de recherche à traiter les requêtes de manière plus précise en reconnaissant des entités d'une base de connaissances dans chaque requête et dans les textes. Ainsi, au lieu de chercher des similarités entre des mots pouvant avoir plusieurs sens, on cherche des similarités entre concepts. Une autre manière d'utiliser les données liées est de fournir toutes les connaissances relatives à une entité en réponse à une requête portant sur cette entité (cf encadrés synthétiques fournis par Google en réponse).

**- Comment peut-on utiliser l'URI représentant Paris de DBPedia (dbr:Paris) dans une application annotant un texte écrit en Russe à l'aide d'entités issues de DBPedia ? Mentionner 2 problèmes qui peuvent se poser.**

→ Associer l'URI dbr:Paris aux mentions de Paris dans le texte en identifiant les entités pertinentes via des correspondances multilingues (par exemple, rdfs:label en russe)

### **Problèmes :**

→ **Ambiguïté linguistique** : "Париж" peut désigner Paris ou d'autres entités, nécessitant une désambiguïsation contextuelle.

→ **Différences de granularité** : Les informations dans DBPedia peuvent ne pas correspondre directement aux besoins de l'application ou être incomplètes en russe.

→ **Problèmes d'encodage ou de normalisation des données** : Les caractères cyrilliques du texte russe pourraient ne pas correspondre correctement aux libellés disponibles dans DBPedia.

→ **Absence d'une correspondance exacte** : Certaines mentions dans le texte (comme des synonymes ou des variations locales) pourraient ne pas être directement liées à dbr:Paris dans DBPedia.

→ **Différences culturelles ou sémantiques** : La manière dont Paris est perçu ou référencé en russe peut différer, compliquant l'identification correcte de l'entité.

**- DBPedia : Comment un graphe de connaissances comme DBPedia ou Schema.org permettent-ils d'améliorer la recherche d'information dans Google ?**

→ Désambiguïsation : Ils permettent d'identifier précisément les entités mentionnées dans une requête grâce à leurs URI, réduisant ainsi les ambiguïtés.

→ Récupération contextuelle : Ils fournissent des relations et des propriétés sur les entités pour afficher des réponses synthétiques et contextuelles (exemple : Knowledge Panel).

→ Recherche multilingue : Ils facilitent l'interprétation et la restitution des informations dans différentes langues en reliant les entités entre elles.

**- Que sont les données liées ouvertes ?**

Ce sont des données structurées accessibles à tous sur des serveurs web (donc ouvertes), identifiées chacune par un identifiant unique (une URI), reliées entre elles par des relations sémantiques. Chaque ensemble de données est représenté par un graphe de triplets RDF, et ces graphes sont reliés entre eux par des alignements, à savoir des relations d'identité ou de similarité entre des nœuds de ces graphes.

**- Donner 3 manières dont un moteur de recherche comme Google peut utiliser un graphe de connaissances générales pour fournir de meilleures réponses à des requêtes.**

→ trouver des mots synonymes à ceux de la requête pour la reformuler ou l'élargir à ces mots . désambiguïser les mots de la requête pour éviter de fournir des réponses non pertinentes . reconnaître des entités de la base de connaissances dans la question ou dans la réponse, et fournir des informations complémentaires disponibles dans la BC et concernant ces entités

**- Définir ce qu'est base de connaissances du web sémantique ; donner des exemples de bases de connaissances disponibles sur le web des données**

→ Une base de connaissance est une ressource dans laquelle des connaissances sont représentées selon un format permettant leur manipulation informatique, et surtout la production de raisonnement ou d'inférence. Dans le contexte du web sémantique, les bases de connaissances sont représentées sous forme de graphes de concepts reliés par des relations sémantiques. Ces graphes sont représentés en RDF. Exemples de BC du LOD : DBPedia, Yago, BabelNet, WikiData

#### **- Quel est l'objectif général du web sémantique ?**

Associer des représentations formelles décrivant le contenu des pages et des données du web afin de pouvoir faire traiter ces contenus par des programmes. Cela suppose de définir des vocabulaires structurés (ou mieux des bases de connaissances) dans des formats standards et d'associer des éléments de ces vocabulaires aux pages ou données du web.

→ la standardisation des formats de représentation des connaissances

→ la constitution d'ontologies, de vocabulaires formels et de bases de connaissances partagées pour décrire les contenus du web

→ l'ajout d'annotations sémantiques aux pages web pour représenter les métadonnées et les contenus sous la forme de graphes de connaissances

#### **- En quoi le web des données est-il une évolution du web sémantique ?**

→ Le web des données applique les mêmes objectifs aux données ouvertes du web. Ces données sont décrites par des vocabulaires plus souvent que par des ontologies. La priorité est mise sur la réutilisabilité et la simplicité des vocabulaires, et ensuite sur le partage des sources ainsi représentées sous forme de données ouvertes et liées. Cette approche favorise l'interopérabilité, la découvrabilité et la réutilisation des données entre différentes applications et domaines, en facilitant le croisement et l'enrichissement des informations grâce à des relations sémantiques standardisées.

#### **- Expliquer comment des services et des ressources du web sémantique et du web des données permettraient de répondre de façon « intelligente » à la requête « organiser un week-end de 3 jours à Londres » avec des visites des musées » en comparant des offres de transport, d'hébergement et de visites à l'utilisateur.**

Pour faire simple, on propose de faire un service spécialisé pour la ville de Londres.

1. identifier des sites fournissant des informations sur Londres, les transports à Londres et les musées ou la culture à Londres.
2. récupérer les données de ces sites, et le schéma de la BD s'il y a lieu
3. Identifier pour chacune des sources un vocabulaire adapté, pris sur le LOD, ou une ontologie du tourisme par exemple
4. représenter toutes ces données selon ces vocabulaires ou ontologies, ou alors de faire une interface qui traduise des requêtes SPARQL exprimées selon ce vocabulaire en requêtes vers ces bases de données
5. définir un service qui permette à l'utilisateur de choisir une date, une durée, des monuments à visiter etc parmi ceux de la base RDF ou des différentes BD.

#### **- Quelle est la nature de la ressource DBPedia ? à partir de quelle source est-elle construite et comment ?**

→ DBPedia est une base de connaissances ouverte et publiée sur le web. Elle fait partie du LOD. Elle comporte plusieurs ontologies. Elle a été construite par extraction de concepts et de relations à partir des catégories Wikipédia, de la structure des pages, ainsi que des résumés encadrés (info-boxes) se trouvant sur les pages. Plusieurs dizaines de logiciels (les extracteurs) ont été définis pour extraire ces informations. Une nouvelle version de Wikipedia est générée chaque année pour intégrer le nouveau contenu de Wikipedia.

#### **- Donner une définition d'ontologie. De quoi une ontologie est-elle composée ?**

→ Une ontologie est un modèle formel (en général logique) de représentation des connaissances d'un domaine. On la définit comme la conceptualisation des notions et objets retenus comme importants dans ce domaine. On les représente le plus souvent sous la forme d'un ensemble de formules logiques ou d'un graphe de concepts (classes), de relations entre ces concepts et d'axiomes qui permettent de générer de nouvelles classes, instances ou faits à partir des faits connus. Une ontologie est organisée autour d'une hiérarchie de classes reliées par des relations est-un (inclusion de classes) qui peuvent être reliées par d'autres types de relations.

#### **- Quelle est la différence avec une base de connaissances ?**

→ Une base de connaissances englobe une ontologie, les entités appartenant aux classes des ontologies (appelées instances) et leurs relations entre ces entités.

**- Quelle est la place des ontologies dans le Web Sémantique ?**

→ Les ontologies jouent un rôle clé dans le Web Sémantique. En effet, le web sémantique a pour but d'associer des méta-données formelles et structurées décrivant le contenu des pages web de manière à ce que des applications informatiques puissent traiter ces contenus. Les ontologies sont des vocabulaires formalisés, respectant des règles de structuration et de bonne construction. Elles définissent les concepts d'un domaine, leurs propriétés et les relations entre eux sous forme d'un graphe. Elles servent à définir des vocabulaires partagés par des communautés de développeurs et d'utilisateurs afin qu'ils annotent ou enrichissent les pages web. Elles servent aussi à décrire des schémas de bases de données à intégrer et partager, pour que plus de données ouvertes soient accessibles dans des formats standards sur le web. Elles contribuent donc au web sémantique en fournissant des vocabulaires pour partager les données, annoter des données ou des pages web, en facilitant la réutilisation et l'interopérabilité.

**- D'après vous, dans quel contexte une entreprise a-t-elle intérêt à publier une ontologie et des données comme des Linked Open Data ?**

→ Une entreprise a intérêt à partager ses données lorsque celles-ci ne sont pas révélatrice de secrets et fabrication, lorsqu'elle respectent les critères exigés pour publier des données ouvertes et liées. L'avantage pour l'entreprise est de se faire connaître en mettant à disposition des données de qualité, mais aussi de pouvoir en retour utiliser d'autres données du LDO. C'est un moyen de faire connaître ses activités et ses compétences, de se rendre visible. En liant ses données à d'autres données, elles deviennent exploitables pour plus d'applications et prennent de la valeur en qq sorte.

**- DBpedia serait-elle une ressource adaptée pour annoter un corpus d'articles scientifiques du domaine de la médecine ? Pourquoi ?**

Non. DBpedia est un vocabulaire général, alors que les articles scientifiques contiennent un vocabulaire technique et spécialisé. Utiliser plutôt MESH ou UMLS

**- Bien que vous n'ayez pas la partie du fichier qui indique les espaces de nom, donnez 3 préfixes vus en cours et indiquez à quel espace de nom ils font appel, et ce à quoi ils font référence**

→ Dcterms : Dublin Core terms : vocabulaire défini pour décrire les méta-données de documents numériques

→ Foaf : friend of a friend, une petite ontologie pour représenter les personnes et leurs relations dans les réseaux sociaux numériques.

→ Rdfs : le schéma de RDF défini par le W3C pour représenter des ontologies légères

**- Que représente (de manière générale) une suite d'URI séparées par une virgule après un nom de propriété (comme dans la propriété foaf:depiction) dans la définition d'un concept ? (pour simplifier, vous pouvez appeler a, b, c, d. les 4 entités mentionnées)**

→ L'entité décrite est <http://data.bnf.fr/ark:/12148/cb119751881#frbr:Work>. foaf:depiction est le nom d'une propriété qui relie cette entité à a, b, c d. Donc cette ligne définit 4 triples : (entité, foaf:depiction, a), (entité, foaf:depiction, b) (entité, foaf:depiction, c) et (entité, foaf:depiction, d)

**Règle du raisonneur :**

inférer la classe la plus précise possible.

**FUNCTIONAL :**

Functional indique qu'une propriété relie une ressource à AU PLUS une autre ressource (et non exactement 1).

→ **Comment différencier entre deux entités** : les 2 entités tp1:Masculin et tp1:Feminin sont différentes avec la property **owl:differentFrom**

**Syntaxe Turtle :**

- La classe Dataset sous classe de Resource

→ dcat:Dataset a dcat:Resource

- l'objectProperties dataset a pour domaine Catalog et co-domaine Dataset

→ dcat:dataset rdfs:domain dcat:Catalog; rdfs:range dcat:Dataset.

## SHACL


- 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 ;
  ]
.
```

Voici un exemple simple de SHACL, où nous définissons une **shape** pour une ressource `Person` qui doit avoir un nom (`foaf:name`) de type chaîne et un âge (`ex:age`) de type entier, et l'âge doit être supérieur ou égal à 18 :

turtle

 Copier le code

```
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix ex: <http://example.org/> .

# Définition d'une shape pour "Person"
ex:PersonShape
  a sh:NodeShape ;
  sh:targetClass foaf:Person ; # Cibler les ressources de type foaf:Person
  sh:property [
    sh:path foaf:name ;          # La propriété "foaf:name"
    sh:datatype xsd:string ;     # Le type de "foaf:name" est une chaîne de caractères
  ] ;
  sh:property [
    sh:path ex:age ;             # La propriété "ex:age"
    sh:datatype xsd:int ;        # Le type de "ex:age" est un entier
    sh:minInclusive 18 ;         # L'âge doit être supérieur ou égal à 18
  ] .
```

## SPARQL

- 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
```

---

- 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
```

---

```
SELECT ?filmLabel ?date ?prop ?valPropLabel WHERE {
  ?film (wdt:P31/wdt:P279*) wd:Q11424.
  ?film ?prop ?valProp.
  ?valProp rdfs:label ?valPropLabel.
  ?valProp rdfs:label ?valPropLabel.
  OPTIONAL { ?film wdt:P583 ?date. } # date
  OPTIONAL { ?film wdt:P571 ?date. } #date début
  ?film rdfs:label ?filmLabel.
  FILTER((BOUND(?Date)) && ((DATATYPE(?Date)) = xsd:dateTime)) )
```

**BOUND** : Le mot-clé BOUND vérifie si une variable est liée à une valeur.

---

### *filtrer la langue*

```
SELECT ?filmLabel ?date ?prop ?valPropLabel WHERE {
  ?film (wdt:P31/wdt:P279*) wd:Q11424.
  ?film ?prop ?valProp.
  ?valProp rdfs:label ?valPropLabel.
  ?film wdt:P17 wd:Q30.
  ?valProp rdfs:label ?valPropLabel.
  OPTIONAL { ?film wdt:P583 ?date. } # date
  OPTIONAL { ?film wdt:P571 ?date. } #date début
  ?film rdfs:label ?filmLabel.
  FILTER((LANG(?filmLabel)) = "en")
  FILTER((LANG(?valPropLabel)) = "en")
  FILTER((BOUND(?Date)) && ((DATATYPE(?Date)) = xsd:dateTime)) )
```

```

SELECT DISTINCT ?name ?piece
WHERE {
  # Requête fédérée vers l'endpoint SPARQL de DBpedia
  SERVICE <https://dbpedia.org/sparql> {
    # Trouver l'IRI de Tim Berners-Lee en utilisant son alias "TBL"
    ?tblIRI dbo:alias "TBL"@en .
    # Récupérer le parent de TBL
    ?tblIRI dbo:parent ?parent .
    # Récupérer les enfants du parent
    ?parent dbo:child ?child .
    # Exclure Tim Berners-Lee de la liste des enfants
    FILTER(?child != ?tblIRI)
    # Récupérer le label du child (le nom complet)
    ?child rdfs:label ?label .
    # Extraire le premier mot du label (son prénom)
    BIND(STRBEFORE(STR(?label), " ") AS ?firstName)
  }

  # Requête fédérée vers l'endpoint SPARQL de Wikidata pour récupérer le titre honorifique
  SERVICE <https://query.wikidata.org/sparql> {
    # Identifier Tim Berners-Lee
    wd:Q80 wdt:P511 ?honorTitleIRI .
    # Récupérer le label du titre honorifique en anglais
    ?honorTitleIRI rdfs:label ?honorTitle .
    FILTER(LANG(?honorTitle) = "en")
  }

  # Pièce toujours dans la maison
  <https://w3id.org/cluedo4KG/KG#LaMaisonDuMeurtre> :maisonContientPiece ?piece .
  ?piece rdf:type :Piece .
  # Les personnes dans la pièce
  ?person rdf:type :Personne .
  ?person rdfs:label ?name .
  ?piece :pieceContientPersonne ?person .

  # Filtrer les noms contenant soit le prénom soit le titre honorifique
  FILTER(CONTAINS(LCASE(?name), LCASE(?firstName)))
  FILTER(CONTAINS(LCASE(?name), LCASE(?honorTitle)))
}

```

```

SELECT ?piece (COUNT(?person) AS ?numberOfPerson)
WHERE {
  # Piece toujours dans la maison
  <https://w3id.org/cluedo4KG/KG#LaMaisonDuMeurtre> :maisonContientPiece ?piece .
  ?piece rdf:type :Piece .
  # Personnes dans la pièce
  ?person rdf:type :Personne .
  ?piece :pieceContientPersonne ?person .
}
GROUP BY ?piece
HAVING (COUNT(?person) > 0)
ORDER BY ?piece
LIMIT 100

```

```

SELECT DISTINCT ?pieceVide
WHERE {
  # Piece toujours dans la maison
  <https://w3id.org/cluedo4KG/KG#LaMaisonDuMeurtre> :maisonContientPiece ?piecevide .
  ?pieceVide rdf:type :Piece .
  FILTER NOT EXISTS {?pieceVide :pieceContientPersonne ?person}
}
LIMIT 100

```