

# TP 4 – Représentation des connaissances

**NOM : GHODBANI**

**PRÉNOM : Fares**

**Groupe : Miage**

**Réalisé avec Canva, Corese, W3C, EasyRdf, DBPedia, WebOWL, yEd Graph Editor et Visual studio code**

## **1. Objectif du TP**

Ce TP visait à relier la modélisation (RDF/TTL), la visualisation (graphes) et l'interrogation sémantique (SPARQL) afin de mieux comprendre le fonctionnement d'un système de représentation des connaissances.

## **2. Fichiers fournis et complétés**

- **ONTOLOGY.TTL** : fichier Turtle complété avec les éléments manquants ([???] remplacés).
- **“ONTOLOGY.RDF”** : fichier RDF généré à partir de ontology.ttl via EasyRDF pour être exploitable dans Corese.

## **3. Représentation graphique de la base de connaissance**

### **1. GRAPHIQUE W3C :**

La première représentation obtenue via le validateur RDF du W3C était très simple, avec un seul nœud et un triple minimal. Cependant, après avoir reconvertis et corrigé mon fichier, j'ai pu obtenir une visualisation bien plus détaillée, montrant clairement les différents individus et leurs relations.

### **2. GRAPHIQUE WEBOWL :**

Pour simuler une représentation plus logique, j'ai chargé ontology.rdf dans WebOWL. Cela a généré un graphe montrant toutes les ressources et relations, ce qui facilite la compréhension des liens entre individus, propriétés et types.

### **3. GRAPHIQUE MANUEL YED GRAPH EDITOR :**

Enfin, je me suis amusé à créer un graphe détaillé avec yEd, montrant explicitement les liens de parenté, le mariage, les enfants, les relations

professionnelles (foaf:knows), ainsi que les propriétés comme l'âge, la taille, la taille de chaussures ou de pantalon.

**\*\*Ces images sont jointes dans l'archive de rendu\*\***

## **4. Exploitation des connaissances avec Corese**

### **3.1 CHARGEMENT DU FICHIER**

- Le fichier ontology.rdf a été chargé via : File → Load RDF → Open.
- La section “Loaded files” affiche maintenant le chemin complet du fichier.

### **3.2 PRÉFIXES UTILISÉS POUR LES REQUÊTES**

```
prefix family: <http://uca.fr/family/member> .  
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .  
prefix foaf: <http://xmlns.com/foaf/0.1/> .  
prefix uca: <http://uca.fr/staff/> .  
prefix job: <http://uca.fr/professions/> .  
prefix type: <http://cestbieng.fr/types/> .  
prefix prop: <http://uca.fr/property/> .
```

### **3.3 REQUÊTES ET RÉSULTATS**

#### 1)- Liste de tous les triples

**OBJECTIF :** Visualiser la structure de la base de connaissance.

```
SELECT ?subject ?property ?predicate  
WHERE {  
    ?subject ?property ?predicate  
}
```

**REMARQUE :** Les triples respectent la forme <Sujet ; Prédicat ; Objet>.

Résultat :

Graph	XML-RDF	Table	Validate
num	?subject	?property	?predicate
35	<http://uca.fr/family/member/Jessica>	<http://uca.fr/property/size>	170
36	<http://uca.fr/family/member/Jean>	<http://uca.fr/property/size>	160
37	<http://uca.fr/family/member/Caroline>	<http://uca.fr/property/size>	160
38	<http://uca.fr/family/member/Edouard>	<http://uca.fr/property/spouse>	<http://uca.fr/family/member/Jessica>
39	<http://uca.fr/family/member/Jessica>	<http://uca.fr/property/spouse>	<http://uca.fr/family/member/Edouard>
40	foaf:type	rdf:type	rdf:Property
41	<http://uca.fr/family/member/Edouard>	foaf:type	<http://cestbeing.fr/types/Person>
42	<http://uca.fr/family/member/Edouard>	foaf:type	<http://uca.fr/professions/teacher>
43	<http://uca.fr/property/gender>	rdf:type	rdf:Property
44	<http://uca.fr/property/name>	rdf:type	rdf:Property
45	<http://uca.fr/property/shoeSize>	rdf:type	rdf:Property
46	<http://uca.fr/property/age>	rdf:type	rdf:Property
47	<http://uca.fr/property/size>	rdf:type	rdf:Property
48	<http://uca.fr/property/spouse>	rdf:type	rdf:Property
49	<http://uca.fr/family/member/Jessica>	rdf:type	<http://cestbeing.fr/types/Person>
50	foaf:knows	rdf:type	rdf:Property
51	<http://uca.fr/staff/Alice>	rdf:type	<http://uca.fr/professions/teacher>
52	<http://uca.fr/staff/Pierre>	rdf:type	<http://cestbeing.fr/types/person>
53	<http://uca.fr/staff/Pierre>	rdf:type	<http://cestbeing.fr/types/person>
54	<http://uca.fr/property/parent>	rdf:type	<http://uca.fr/professions/teacher>
55	<http://uca.fr/family/member/Jean>	rdf:type	rdf:Property
56	<http://uca.fr/family/member/Caroline>	rdf:type	<http://cestbeing.fr/types/person>
57	<http://uca.fr/property/child>	rdf:type	<http://cestbeing.fr/types/person>
58	<http://uca.fr/family/member/Yannick>	rdf:type	<http://cestbeing.fr/types/person>
59	<http://uca.fr/family/member/Victoria>	rdf:type	<http://cestbeing.fr/types/person>
60	foaf:knows	rdf:type	rdf:Property
61	<http://uca.fr/property/size>	rdf:type	rdf:Property
62	<http://uca.fr/family/member/Edouard>	foaf:knows	<http://uca.fr/staff/Alice>
63	<http://uca.fr/family/member/Edouard>	foaf:knows	<http://uca.fr/staff/Pierre>
64	<http://uca.fr/staff/Alice>	foaf:knows	<http://uca.fr/family/member/Caroline>
65	<http://uca.fr/staff/Pierre>	foaf:knows	<http://uca.fr/family/member/Edouard>
66	<http://uca.fr/staff/Pierre>	foaf:knows	<http://uca.fr/family/member/Edouard>
67	<http://uca.fr/staff/Pierre>	foaf:knows	<http://uca.fr/staff/Alice>

## 2)- Toutes les femmes de la base

```
SELECT ?women
WHERE {
  ?women prop:gender type:woman
}
```

**Remarque :** Permet de filtrer les individus selon la propriété prop:gender

**Résultat :**

num	?women
1	<http://uca.fr/family/member/Jessica>
2	<http://uca.fr/staff/Alice>
3	<http://uca.fr/family/member/Caroline>
4	<http://uca.fr/family/member/Victoria>

## 3)- Professeurs et leur âge

```
SELECT ?teacher ?age ?name
WHERE {
  ?teacher a job:teacher ;
    prop:age ?age ;
    prop:name ?name
}
```

**Résultat :** Seul Edouard apparaît car c'est le seul professeur ayant défini la propriété prop:age.

num	?teacher	?age	?name
1	<http://uca.fr/family/member/Edouard>	40	Edouard

**Remarque :** Il est important de noter que toutes les ressources doivent posséder la propriété interrogée pour apparaître.

## 5)- Tous les parents distincts

```

SELECT DISTINCT ?parent
WHERE {
  ?parent prop:child ?child
}

```

**Résultat :**

- Jean
- Caroline
- Edouard
- Jessica

	num	?parent
1		<http://uca.fr/family/member/Edouard>
2		<http://uca.fr/family/member/Jean>
3		<http://uca.fr/family/member/Caroline>

**Remarque :** La clause DISTINCT supprime les doublons causés par plusieurs enfants ayant les mêmes parents.

## 6)- Individus avec taille $\leq 170$ cm

```

SELECT ?person ?size
WHERE {
  ?person prop:size ?size
  FILTER(?size <= 170)
}

```

**Objectif :** Filtrer les individus selon une valeur numérique.

**Résultat :** Jessica, Caroline, Jean.

	num	?person	?size
1		<http://uca.fr/family/member/Jessica>	170
2		<http://uca.fr/family/member/Jean>	160
3		<http://uca.fr/family/member/Caroline>	160

Liens de parenté avec CONSTRUCT

```

CONSTRUCT {
  ?parent prop:parent ?child
}
WHERE {
  ?child prop:parent ?parent
}

```

**Objectif** : Visualiser graphiquement les liens de parenté dans Corese.

**Remarque/Résultat** : Le graphique généré dans Corese montre clairement les relations parent-enfant.

	num	?child	?parent
1		< <a href="http://uca.fr/family/member/Edouard">http://uca.fr/family/member/Edouard</a> >	< <a href="http://uca.fr/family/member/Jean">http://uca.fr/family/member/Jean</a> >
2		< <a href="http://uca.fr/family/member/Edouard">http://uca.fr/family/member/Edouard</a> >	< <a href="http://uca.fr/family/member/Caroline">http://uca.fr/family/member/Caroline</a> >
3		< <a href="http://uca.fr/family/member/Yannick">http://uca.fr/family/member/Yannick</a> >	< <a href="http://uca.fr/family/member/Edouard">http://uca.fr/family/member/Edouard</a> >
4		< <a href="http://uca.fr/family/member/Yannick">http://uca.fr/family/member/Yannick</a> >	< <a href="http://uca.fr/family/member/Jessica">http://uca.fr/family/member/Jessica</a> >
5		< <a href="http://uca.fr/family/member/Victoria">http://uca.fr/family/member/Victoria</a> >	< <a href="http://uca.fr/family/member/Edouard">http://uca.fr/family/member/Edouard</a> >
6		< <a href="http://uca.fr/family/member/Victoria">http://uca.fr/family/member/Victoria</a> >	< <a href="http://uca.fr/family/member/Jessica">http://uca.fr/family/member/Jessica</a> >

## 7)- Vérification du grand-père de Victoria avec ASK

```
ASK {  
    dbr:Victoria prop:parent <Edouard> .  
    <Edouard> prop:parent <Jean> .  
}
```

Résultat : true

Remarque : Jean est bien le grand-père de Victoria.

## 5. REQUÊTES SPARQL CRÉATIVES SUR DBPEDIA (POUR L'ILLUSTRATION)

On remarque que ces requêtes sont uniquement à titre illustratif, elles ne peuvent pas être exécutées sur Corese avec notre fichier ontology.rdf.

- Date de naissance de Frank Ocean

```
PREFIX dbpedia: <http://dbpedia.org/resource/>  
PREFIX prop: <http://dbpedia.org/ontology/>
```

```
SELECT ?birthDate  
WHERE {  
    dbpedia:Frank_Ocean prop:birthDate ?birthDate .  
}
```

Résultat attendu : 1987-10-28

- Capitales de 10 pays

PREFIX dbpedia: <<http://dbpedia.org/resource/>>

PREFIX prop: <<http://dbpedia.org/ontology/>>

```
SELECT ?country ?capital  
WHERE {  
    ?country prop:capital ?capital .  
}  
LIMIT 10
```

Exemple de résultats : Paris, Berlin, Rome, Madrid, Washington D.C., etc.

- [Documentation sur Octobre Rose](#)

PREFIX dbpedia: <<http://dbpedia.org/resource/>>

PREFIX prop: <<http://dbpedia.org/ontology/>>

```
SELECT ?abstract  
WHERE {  
    dbpedia:Breast_cancer_awareness_month prop:abstract ?abstract .  
    FILTER (lang(?abstract) = "en")  
}  
LIMIT 1
```

Résultat attendu : Bref résumé de la campagne de sensibilisation sur le cancer du sein.

## **6- Conclusion:**

Ce TP m'a permis de comprendre comment représenter et interroger une base de connaissances à l'aide des langages RDF et SPARQL, tout en découvrant le principe de représentation des connaissances.

J'ai exploré différents outils comme EasyRDF, CoReSe, WebVOWL et yEd Graph Editor, qui m'ont aidé à visualiser et manipuler les relations entre les données.

Cette approche m'a donné une meilleure compréhension du Web sémantique et de la structuration logique des informations.

