TD Web sémantique 4-IF-WS (2020–2021)

Sylvie Calabretto, **Mehdi Kaytoue**,
Aimene Belfodil, Adnene Belfodil,
Mohamed Maouche

1 Description (RDF) et Interrogation (SPARQL) des Données (1h45)

1.1 Syntaxes

On considère le jeu de données etudiants.n3. Ouvrir et comprendre ce fichier. Dessiner le graphe RDF correspondant. Convertir dans d'autres syntaxes (RDF/XML, RDF/JSON, triples, ...) avec des outils du Web donnés en cours (www.w3.org/RDF/Validator/, www.easyrdf.org/converter, ...).

Graphe RDF : Le graphe RDF est représenté dans la figure 1

1.2 Trouver les réponses à la requête

On considère le vocabulaire insa avec les prédicats suivants et leur signification intuitive :

- `http://insa-lyon.fr#inscrit` *le sujet est inscrit à l'objet*
- `http://insa-lyon.fr#binome` *le sujet a comme binôme l'objet*
- `http://insa-lyon.fr#nom` *le sujet a pour nom l'objet*
- `http://insa-lyon.fr#formation` *le sujet est dans la formation objet*
- `http://insa-lyon.fr#departement` *le sujet est dans le département objet*

Pour chaque requête, trouver la(les) réponse(s) possible(s). On dessinera le graphe requête que l'on cherchera à appareiller avec le graphe données.

1.
```
PREFIX insa: <http://insa-lyon.fr/insa#>
PREFIX ue: <http://insa-lyon.fr/ue#>
SELECT * WHERE
{
  ?e insa:inscrit ue:alia .
}
```

2.
```
PREFIX insa: <http://insa-lyon.fr/insa#>
PREFIX ue: <http://insa-lyon.fr/ue#>
SELECT ?e WHERE
{
  ?e insa:binome ?e2 .
  ?e2 insa:inscrit ue:alia .
}
```

3.
```
PREFIX insa: <http://insa-lyon.fr/insa#>
PREFIX ue: <http://insa-lyon.fr/ue#>
SELECT ?e WHERE {
  { { ?e insa:binome ?e2 . }
  UNION
  { ?e2 insa:binome ?e .} }
  ?e2 insa:inscrit ue:alia .
}
```

4.
```
PREFIX insa: <http://insa-lyon.fr/insa#>
PREFIX ue: <http://insa-lyon.fr/ue#>
PREFIX foaf: <http://xmlns.com/foaf/spec/>
SELECT ?p ?e2 WHERE {
  ?e ?p ?e2; foaf:name ?n .
  FILTER(?n = "Basile")
}
```

Remarque :

1. **Question 3 (Union).** Expliquer aux étudiants qu'il faut qu'ils transforment leurs requête en forme disjonctive. Comme ça il sauront que le moteur de recherche doit apareiller soit le premier graphe soit le deuxième mais pas les deux au même temps. Le **ou** dans le diagramme est important !.
2. **Question 4 (Filter).** Expliquer aux étudiants que même si de manière déclarative, enelever le Filter et remplacer la variable **?n** par "Basile" est équivalent. L'aspect procédural est différent. En effet, essayer d'appareiller directement avec "Basile" ou d'appareiller une variable puis tester si elle est égale à "Basile" n'est pas exactement le même chose !

Graphes de requêtes : La figure 2 représente les graphes des requêtes ci-dessus.
Résultats :

e
<http ://insa-lyon.fr/etudiant#4567890>
<http ://insa-lyon.fr/etudiant#1234567>

Requête 1

e
<http ://insa-lyon.fr/etudiant#4567890>

Requête 2

e
<http ://insa-lyon.fr/etudiant#4567890>
<http ://insa-lyon.fr/etudiant#1234567>

Requête 3

p	e2
foaf:name	"Basile"
insa:binome	etu:3456789
insa:inscrit	ue:datamining
insa:inscrit	ue:se

Requête 4

1.3 Utilisation de Twinkle

Twinkle fournit une interface simple et un moteur SPARQL. Il permet de charger un fichier RDF ou encore de se connecter à un point d'accès comme celui de DBPEDIA. Tester les requêtes des deux questions précédentes et vérifier vos réponses !

Observation : Lorsque vous écrivez une requête SPARQL, l'ordre des clauses est important pour la performance d'exécution (l'ordre ne change pas le résultat obtenu). Allez toujours de la clause la plus restrictive à la plus générale.

1.4 Trouver la requête

Écrire en SPARQL les requêtes suivantes :

- *Quel(s) étudiants ont Alice pour binôme ?*
  Réponse :

```
PREFIX insa: <http://insa-lyon.fr/insa#>
PREFIX foaf: <http://xmlns.com/foaf/spec/>
SELECT ?e WHERE
{
  ?e insa:binome ?e2 .
  ?e2 foaf:name ?n  .
  FILTER(?n = "Alice")
}
```

- *Quels étudiants sont inscrits à une UE de la formation* http://insa-lyon.fr/formation#4if ?
  Réponse :

```
PREFIX insa: <http://insa-lyon.fr/insa#>
PREFIX form: <http://insa-lyon.fr/formation#>
SELECT DISTINCT ?e WHERE
{
  ?u insa:formation form:4if  .
  ?e insa:inscrit ?u .
}
```

- *Quels sont les étudiants en binôme inscrit à la même UE ?*
  Réponse :

```
PREFIX insa: <http://insa-lyon.fr/insa#>
PREFIX ue: <http://insa-lyon.fr/ue#>
PREFIX foaf: <http://xmlns.com/foaf/spec/>
SELECT ?n ?n2 ?ue WHERE {
  ?e insa:binome ?e2 .
  ?e insa:inscrit ?ue; foaf:name ?n.
  ?e2 insa:inscrit ?ue; foaf:name ?n2.
}
```

- *Quelles sont les UE d'une même formation qui ont des départements différents ?*

```
PREFIX insa: <http://insa-lyon.fr/insa#>
SELECT ?u1 ?d1 ?u2 ?d2 ?f WHERE {
  ?u1 insa:formation ?f; insa:departement ?d1 .
  ?u2 insa:formation ?f; insa:departement ?d2 .
  FILTER(?d1 != ?d2)
}
```

– *Quels étudiants partagent un même binôme ?*
Réponse :

```
PREFIX insa: <http://insa-lyon.fr/insa#>
SELECT ?e1 ?e2 ?b WHERE {
  { { ?e1 insa:binome ?b .
      ?e2 insa:binome ?b . }
  UNION
  { ?b insa:binome ?e1 .
    ?b insa:binome ?e2 . }
  UNION
  {  ?e1 insa:binome ?b .
     ?b insa:binome ?e2 .} }
  FILTER(?e1 != ?e2)
}
```

Remarquons que la réponse de la requête précédente contient des redondances. Afin de palier à cela, nous utilisons la requête suivante (qui se base sur la comparaison des noms des étudiants).

```
PREFIX insa: <http://insa-lyon.fr/insa#>
PREFIX foaf: <http://xmlns.com/foaf/spec/>
SELECT ?n1 ?n2 ?nb WHERE {
  { { ?e1 insa:binome ?b .
      ?e2 insa:binome ?b . }
  UNION
  { ?b insa:binome ?e1 .
    ?b insa:binome ?e2 . }
  UNION
  {  ?e1 insa:binome ?b .
     ?b insa:binome ?e2 .} }
  ?e1 foaf:name ?n1 .
  ?e2 foaf:name ?n2 .
  ?b foaf:name ?nb .
  FILTER(?n1 < ?n2)
}
```

– *Même question que la précédente si on considère qu'il faut rendre symétrique la relation binôme.*
Réponse :

```
PREFIX insa: <http://insa-lyon.fr/insa#>
SELECT ?e1 ?e2 ?b WHERE {
  ?e1 insa:binome ?b .
  ?e2 insa:binome ?b .
  FILTER(?e1 != ?e2)
}
```

Afin que la relation binôme devienne symétrique (dans nos données), une façon de faire est d'exécuter la requête suivante.

```
PREFIX insa: <http://insa-lyon.fr/insa#>
INSERT {?e1 insa:binome ?e2}
WHERE {?e2 insa:binome ?e1}
```

2 Injection des données sémantiques (RDFa et Microdata) (15mn)

Le plus important à retenir ici c'est d'apprendre à repérer des balises RDFa dans un code HTML. Et comprendre le fait que le web sémantique n'est pas si développé que ça!!! Ne passez pas trop de temps dans cette exercice!!!

3 DBpedia (1h30)

3.1 Exploration

– *Chercher des personnes, des pays, des villes, ... Que peut-on dire de ces concepts ? Les descriptions sont-elles toujours cohérentes ?*
– *Pour cela, taper un URI dans le navigateur Web, et naviguer.*
– *On pourra aussi chercher des vocabulaires* http://prefix.cc/

Observations :

– Afin de récupérer le nom d'une ressource **DBpedia** à partir d'une page wikipedia il suffit d'extraire le nom de la page de l'URL **"https ://fr.wikipedia.org/wiki/PAGE_NAME"** est de construire l'URL de la ressource dans DBpedia **"http ://dbpedia.org/resource/PAGE_NAME"**. **Exemple :** Pour la ville de Lyon, la page wikipedia de la ville est **"https ://fr.wikipedia.org/wiki/Lyon"**, le nom de la ressource est ainsi **"http ://dbpedia.org/resource/Lyon"**.
– La requête suivante est un exemple de comment parcourir le graphe RDF de DBpedia (pour découvrir de nouveau concepts, le vocabulaire utilisé, ...) :

```
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT * WHERE {
  <http://dbpedia.org/resource/Lyon> ?p ?v .
}
LIMIT 200
```

3.2 Requêtes

1. *Quels sont les pays de plus de 15 millions d'habitants dont on peut avoir le nom en anglais ?*

```
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT * WHERE {
  ?country a dbo:Country; dbo:populationTotal ?population; rdfs:label ?label.
  FILTER (?population > 15000000 && langMatches(lang(?label), "EN"))
}
ORDER BY DESC(?population)
LIMIT 50
```

2. *Quels sont les pays qui ont 'Republic' dans leur nom anglais et établit avant 1920 ?*

```
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT * WHERE {
  ?country a dbo:Country; dbo:longName ?longname; dbo:foundingDate ?foundingdate .
  FILTER( ?foundingdate<"1920-01-01"^^xsd:date && regex(?longname , "[Rr]epublic"))
}
ORDER BY DESC(?foundingdate)
LIMIT 50
```

3. *Quels sont les pays qui utilisent l'euro et où l'on parle français ? Et de moins de 5000 habitants ?*

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX dbr: <http://dbpedia.org/resource/>

SELECT * WHERE {
  ?country a dbo:Country; dbo:populationTotal ?population;
           dbo:currency dbr:Euro;  dbo:language dbr:French_language .
  FILTER (?population<=5000)
}
ORDER BY ASC(?population)
```

4. *Le fleuve Amazone est-il plus grand que le Nil ?*

```
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX dbr: <http://dbpedia.org/resource/>

SELECT (?lamazon>?lnile as ?answer) WHERE {
  dbr:Amazon_River dbp:length ?lamazon.
  dbr:Nile dbp:length ?lnile.
}
```

Remarque : Néanmoins, on ne va pas retrouver les longueurs des deux fleuves sur DBpedia (qui étaient présentes l'année passée).

5. *Donner le film qui a le second revenu le plus élevé donné en dollars*
(<http ://dbpedia.org/datatype/usDollar>)

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dtype: <http://dbpedia.org/datatype/>

SELECT ?o ?gross
WHERE
{
   ?o dbo:gross ?gross .
   ?o a  dbo:Film .
   FILTER ( DATATYPE(?gross) = dtype:usDollar )
}
```

```
ORDER BY DESC(xsd:integer(?gross))
OFFSET 2
LIMIT 1
```

6.

7. *Donner pour chaque film dirigé (director) par Steven Spielberg la liste des acteurs et producteurs si possible. (on pour utiliser un* GROUP BY *et la fonction d'agrégation* GROUP_CONCAT(...)*. On filtrera le résultat pour ne garder que les 5 films les plus anciens.*

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?film
GROUP_CONCAT(DISTINCT ?screenPlayerName; SEPARATOR="|") as ?actors
GROUP_CONCAT(DISTINCT ?producerName; SEPARATOR="|") as ?productors
SAMPLE(?releasedate) as ?release
WHERE
{
?film dbo:director dbr:Steven_Spielberg; rdfs:label ?filmName; dbo:releaseDate ?releasedate.
optional {
  ?film dbo:starring ?screenplayer .
  ?screenplayer foaf:name ?screenPlayerName .
}
optional {
  ?film dbo:producer ?producer .
  ?producer foaf:name ?producerName .
}
FILTER(lang(?filmName)="en")
}
GROUP BY ?film
ORDER BY ASC(?release)
LIMIT 5
```

8. *Lister tous les couples de livres de la série Harry Potter tels que le second est un successeur (pas forcément direct) du premier. Pour chaque couple, on donnera la distance du chemin qui les sépare (Utilisation de* `dbo:subsequentWork*` *et* `dbo:subsequentWork+`*)*

```
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?start, ?end, count(?mid) AS ?distance
WHERE {
  ?start a dbo:Book; dbo:series dbr:Harry_Potter; dbo:subsequentWork* ?mid.
  ?mid a dbo:Book; dbo:series dbr:Harry_Potter; dbo:subsequentWork+ ?end.
  ?end a dbo:Book; dbo:series dbr:Harry_Potter.
}
GROUP BY ?start ?end
ORDER BY ASC(?start) ASC(?distance)
```

Divers

- TWINKLE : outil graphique multi-fonctions http://www.ldodds.com/projects/twinkle/
- JENA : Outil multi-fonctions http://jena.apache.org/download
- Tutorial SPARQL http://jena.apache.org/tutorials/sparql.html
- DBpedia access point http://dbpedia.org/snorql/
- DBpedia access point http://dbpedia.org/sparql/

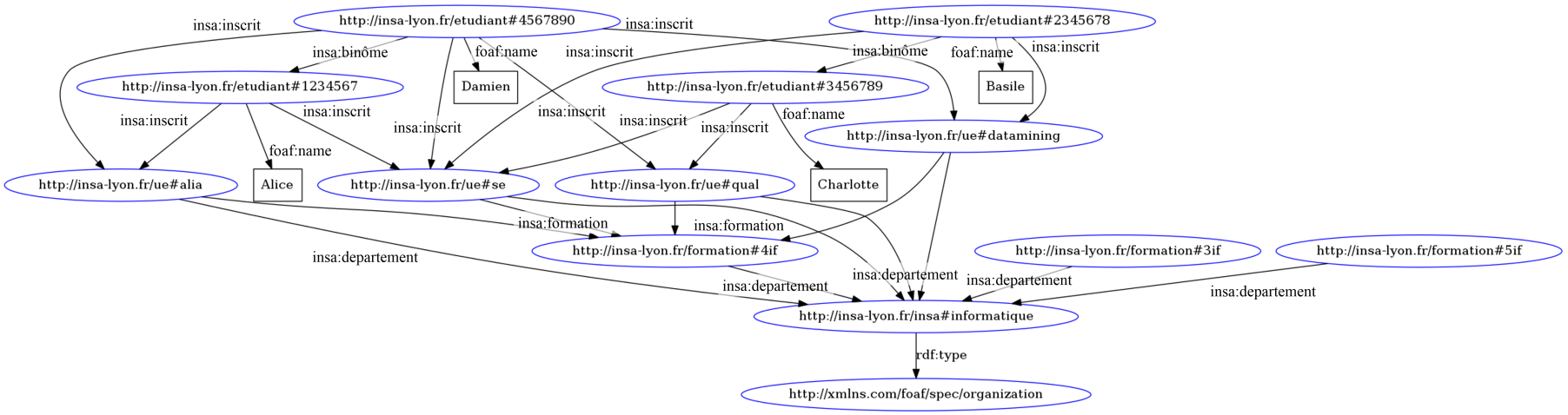

FIGURE 1 – Graphe RDF de etudiants.n3

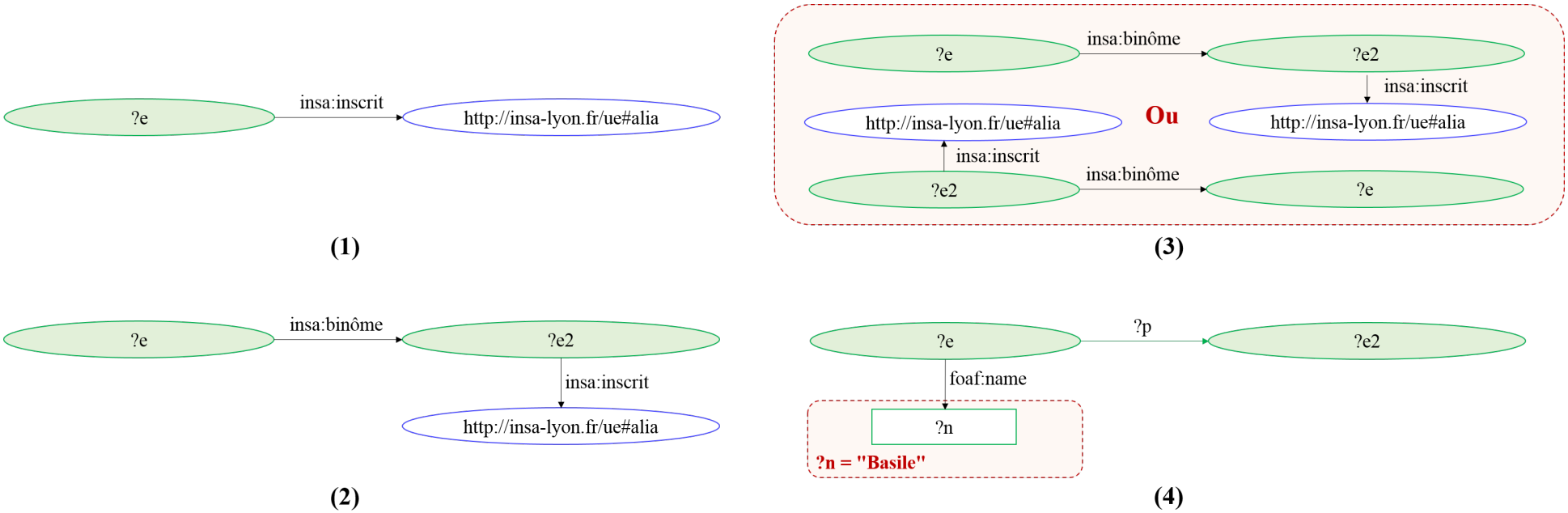

FIGURE 2 – Graphes des requêtes