TD et TME : RDF et SPARQL

L'objectif de ce TD/TME est de se familiariser avec la syntaxe de représentation des données RDF Turtle et de comprendre le mécanisme des motifs de graphes qui sont à la base du langage d'interrogation Sparql.

Exercice 1 : Dépliage des données factorisées

Représenter les données suivantes sous format Turtle en des triplets n'utilisant pas de factorisation.

```
@base <http://example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rel: <http://www.perceive.net/schemas/relationship/> .

<#green-goblin>
   rel:enemyOf <#spiderman> ;
   a foaf:Person ;    # in the context of the Marvel universe
   foaf:name "Green Goblin" .
<#spiderman>
   rel:enemyOf <#green-goblin> ;
   a foaf:Person ;
   foaf:name "Spiderman", "Человек-паук"@ru .
```

Exercice 2 : Motifs de graphes

Le but de cet exercice est d'illustrer les principaux motifs de graphes utilisés dans le langage SPARQL. Rappelons qu'une requête SPARQL est générée à partir de la syntaxe suivante

SELECT [distinct] |CONSTRUCT | ASK | DESCRIBE

FROM source

WHERE { **MOTIFS** }

ORDER | LIMIT | OFFSET

Pour cet exercice, nous nous focalisons sur la partie **MOTIFS** des requêtes et considérons que leur évaluation retourne toutes variables qui y sont exprimées.

Par exemple : l'évaluation du motif ?X rel:enemyOf ?y sur les triplets de l'exercice précédent retourne les liaisons indiquées dans le tableau suivant

?X	?Y
#green-goblin	#spiderman
#spiderman	#green-goblin

Les données de l'encadré ci-dessous sont celles qu'on utilisera dans cet exercice. Pour simplifier, les espaces de noms ainsi que le symbole ':' en préfixe des IRI sont omis.

P1	name	"john"
P2	name	"rick"
P3	name	"lars"
P4	name	"liz"
P1	phone	11-345-89
P2	email	rick@org
P3	webpage	www.lars.com
P4	email	liz@aut.gov
P4	phone	09-134-21
P4	webpage	www.liz.pers

Remarques :
1- Les motifs doivent toujours être séparés par des points (cf point a ci-dessous)
2- Dans Sparql, un groupe de motifs {M1. M2. M3} signifie la conjonction de ces motifs. Ceci explique l'absence d'un opérateur spécifique pour exprimer la conjonction.
3- L'opérateur UNION prend comme opérandes deux groupes de motifs qui doivent **obligatoirement** être entourés d'accolades { } même si la cardinalité des groups est 1 (cf point b ci-dessous)
4- l'UNION est plus prioritaire par rapport à la conjonction des groupes de motifs.

Evaluer sur ces triplets les motifs suivants et expliquer en langage naturel ce qu'il permettent de calculer. On supposera une sémantique ensembliste.

Remarque : *Pour tester ces requêtes en TME, il suffit de rajouter la clause select avec le symbole * et d'imbriquer les motifs dans la clause where.*

1.
    a. ?p email ?e . ?p webpage ?w
    b. {?p email ?e} UNION {?p webpage ?w}
    c. {?p email ?e} OPTIONAL {?p webpage ?w}
2. ?p name ?n OPTIONAL {?p email ?e} OPTIONAL {?p webpage ?w}
3. ?p name ?n OPTIONAL { ?p email ?e OPTIONAL {?p webpage ?w} }

Y-a-t-il une différence entre les motif des questions 2 et 3 ? Si oui laquelle?

Discussion : Que remarquez-vous en comparant les réponses des motifs des questions 2 et 3? Est-ce qu'en général on peut dire que l'opérateur OPTIONAL est associatif, i.e

{M1 OPTIONAL M2} OPTIONAL M3 = M1 OPTIONAL { M2 OPTIONAL M3} où M1, M2 et M3 sont des motifs?

En cas de réponse négative, quelle modification mineure (ajout ou suppression de quelques triplets) permettrait d'avoir pour les données de l'exercice un résultat identique en évaluant les motifs des questions 2 et 3?

4. ?p name ?n . { {?p email ?e} UNION {?p phone ?l } }
5.
    a. ?p webpage ?w OPTIONAL {?p phone ?l} FILTER (! bound(?l))
    b. ?p webpage ?w minus {?p phone ?l}

Remarque : Bien que Sparql n'oblige pas que les variables utilisées par la fonction bound() aient été utilisées dans le groupe de motif qui précède le filter, il est plus sûr de respecter cette contrainte.
Exemple : le motif *?p webpage ?w OPTIONAL {?p phone ?l} FILTER (! bound(?**k**))*
retourne les liaisons décrites comme suit

?p	?w	?l
P4	"www.liz.pers"	"09-134-21"
P3	"www.lars.com"	

Exercice 3 : Requêtes

Considérer les données suivantes qui portent sur des personnes.

```
@base <http://example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rel: <http://www.perceive.net/schemas/relationship/> .
@prefix : <http://example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix univ: <http://www.faculty.ac> .

#rdf types
:john       a              :Person .
:liz        a              :Person .
:robert     a              :Person .
:dan        a              :Person .
:J.Horrow   a              :Professor .
:larry      a              :Artist .
:cmu        a              :university .
:mit        a              :institute .
:mi         a              :state .
:nyc        a              :City .
:pittsburgh a              :City .

#family and friendships
:john       :hasMother     :suzan .
:john       :hasFather     :richard .
```

```
:richard    :hasBrother     :luke .
:john       :hasBrother     :richard .
:larry      :hasFather      :john .
:liz        :hasBrother     :robert .
:liz        :hasFather      :luke .
:liz        :hasMother      :monica .
:dan        rel:ChildOf     univ:J.Horrow .
:Jim        rel:ChildOf     univ:L.Yang .
:liz        :friend         :dan .
:liz        :friend         :larry .
:larry      :friend         :dan .
:dan        :friend         :john .
:larry      :friend         :liz .

#POI
:robert     :livesIn        :nyc .
:liz        :livesIn        :westLafayette .

#degree and university
:luke       :hasDegree      "Eng" .
:luke       :studiedAt      :cmu .
:liz        :studiedAt      :cmu .
:richard    :studiedAt      :mit .
:suzan      :studiedAt      :ucsd .
:monica     :supervisedBy   univ:L.Yang .
:monica     :supervisedBy   univ:J.Horrow .
:Jim        :studiedAt      :ucsb .
:monica     :studiedAt      :cmu .
:john       :studiedAt      :ucsd .
:larry      :studiedAt      :ucsd .
:J.Horrow   :studiedAt      :ucla .
#misc descriptions
:ucsb       :locatedAt      :santaBarbara .
:cmu        :locatedAt      :pittsburgh .
```

Quel est le format utilisé? Représentez les trois premiers triplets sous leur forme *étendue*.
Exprimer les requêtes suivantes :

1.
    a. Extraire l'ensemble des IRI des sujets
    b. idem en retournant les noms locaux

Forme du résultat attendu :

a)	b)
| x | ========== | <http://example.org/john> | | <http://example.org/mi> | | <http://example.org/cmu> | | <http://example.org/richard> | |.....	---------- | x | ========== | :john | | :dan | | :mi | | :cmu | …...

2. Même question avec les prédicats.

3. Extraire les villes citées dans des triples de cette base.

4. Extraire les personnes qui ont étudié dans la même université que l'un de leur parents. (Ne pas considerer les triplets avec la prédicat childOf).

5. Extraire les personnes qui ont étudié dans une université où leur deux parents ont étudié.

6. Extraire les personnes qui ont étudié dans une université différente de celle de leur père et leur mère.

7. Extraire les personnes qui ont un étudié dans la même université qu'un frère ou une soeur.

(Question subsidiarie) Donnez les noms et les universités des personnes qui ont au moins un frère ou une soeur et qui n'ont pas de frère ou de soeur qui ont étudié à la même université qu'elles.

8. Extraire les personnes qui étudient dans une ville différente de celle où ils habitent.

9. Extraire les personnes qui sont ami(e)s d'un(e) ami(e) de Liz.
   Remarque : ne pas retourner les ami(e)s direct(e)s de Liz!

Exercice 4 : Requêtes

Considérons les données RDF dans l'encadré.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
```

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix f: <http://www.cems.uwe.ac.uk/empdept/concept/> .
@prefix : <http://www.abc.org/> .

#rdf types
:john       a              f:emp .
:liz        a              f:emp .
:dan        a              f:emp .
:larry      a              f:emp .
:jim        a              f:emp .
:clark      a              f:emp .
:robert     a              f:emp .
#job
:john   f:Job    "architect" .
:liz    f:Job    "doctor" .
:dan    f:Job    "engineer" .
:larry  f:Job    "singer" .

#surnames
:john       foaf:surname    "john" .
:liz        foaf:surname    "liz" .
:dan        foaf:surname    "dan" .
:larry      foaf:surname    "larry" .
```

Exprimer les requêtes retournant les informations suivantes:

1. Noms des employés avec leur job suivant l'ordre alphabétique de leur nom
2. Les 3 premiers employés suivant l'ordre alphabétique
3. Les 3 premiers employés, ayant un surname, les mieux payés
4. Les employés qui ne sont ni médecin ni chanteur
5. Les employés dont le nom commence par 'l' et termine par 'ry'.
6. Le plus grand salaire
    a. sans utiliser ORDER BY et LIMIT
    b. en utilisant ORDER BY et LIMIT
7. Les couples d'employés qui gagnent le même salaire (chaque couple d'employés doit apparaître une seule fois).
8. Les couples d'employés tels que la différence entre leurs salaires est supérieure à 5.
   Indice : utiliser la function PREFIX xs: <http://www.w3.org/2001/XMLSchema#>
9. Le nombre de départements
10. Le nombre d'employés par département