Examen MIF18 - Bases de données avancées - 17 décembre 2013
Durée : 1h30
Documents autorisés

Numéro de copie :

Il faut rendre ces feuilles en les glissant dans votre copie anonyme. Ne pas l'utiliser comme brouillon. Les réponses sont à donner sur ces feuilles, pas dans la copie. Remplir le champ ci-dessus avec le *numéro de la copie* dans laquelle vous allez la glisser. Remplir la partie d'anonymat de la copie, puis coller le coin.

Exercice 1: XPath - DTD

On considère la DTD suivante :

```
<!ELEMENT livres (livre+)>
<!ELEMENT livre (titre,auteur+,resume?)>
<!ATTLIST livre id ID #REQUIRED>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT auteur (#PCDATA)>
<!ATTLIST auteur num ID #IMPLIED>
<!ELEMENT resume (#PCDATA|ref|em)*>
<!ELEMENT em (#PCDATA|ref|em)*>
<!ELEMENT ref (#PCDATA|ref|em)*>
<!ATTLIST ref idr IDREF #REQUIRED>
```

Pour chacune des requêtes XPath suivantes dire si la requête renvoie forcément un résultat vide lorsqu'elle est exécutée sur un document valide vis-à-vis de la DTD précédente.On justifiera la réponse dans les cas où le résultat de la requête est forcément vide.

1. `//auteur[@num = //livre/@id]`

2. `//auteur[@num]/../livre[id = 3]`

3. `//auteur/../[em/resume = "Hugo"]`

Exercice 2: XPath : *forward axis*

Réécrire la requête XPath suivante en utilisant uniquement des axes de type *forward* : `/documentation//h3/../preceding-sibling::h2[1]/em/..`

Exercice 3: RDF Schema

On considère les déclarations RDF Schema suivantes :

```
1  PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2  PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
3  PREFIX ucbl:<http://www.univ-lyon1.fr/>
4
5  ucbl:x12345             rdf:type            ucbl:etudiant   .
6  ucbl:x12345             ucbl:formation      ucbl:m1info   .
7  ucbl:x12345             ucbl:user           ucbl:forge   .
8
9  ucbl:emmanuel.coquery   rdf:type            ucbl:enseignant.
10 ucbl:emmanuel.coquery   ucbl:batiment       ucbl:nautibus.
11 ucbl:emmanuel.coquery   ucbl:admin          ucbl:forge   .
12
13 ucbl:etudiant           rdfs:subClassOf     ucbl:membre   .
14 ucbl:enseignant         rdfs:subClassOf     ucbl:personnel   .
15 ucbl:personnel          rdfs:subClassOf     ucbl:membre   .
16
17 ucbl:admin              rdfs:subPropertyOf  ucbl:user   .
18 ucbl:user               rdfs:range          ucbl:logiciel   .
19 ucbl:user               rdfs:domain         ucbl:compte-cas   .
```

Question 1: En prenant en compte les mécanismes d'inférence RDF Schema dire pour chacun des triplets suivants s'il peut être déduit. Dans le cas où il peut être déduit, donner les numéros de ligne des triplets utilisés de la déclaration précédente pour faire cette déduction.

1. `ucbl:x12345 rdf:type ucbl:membre .`

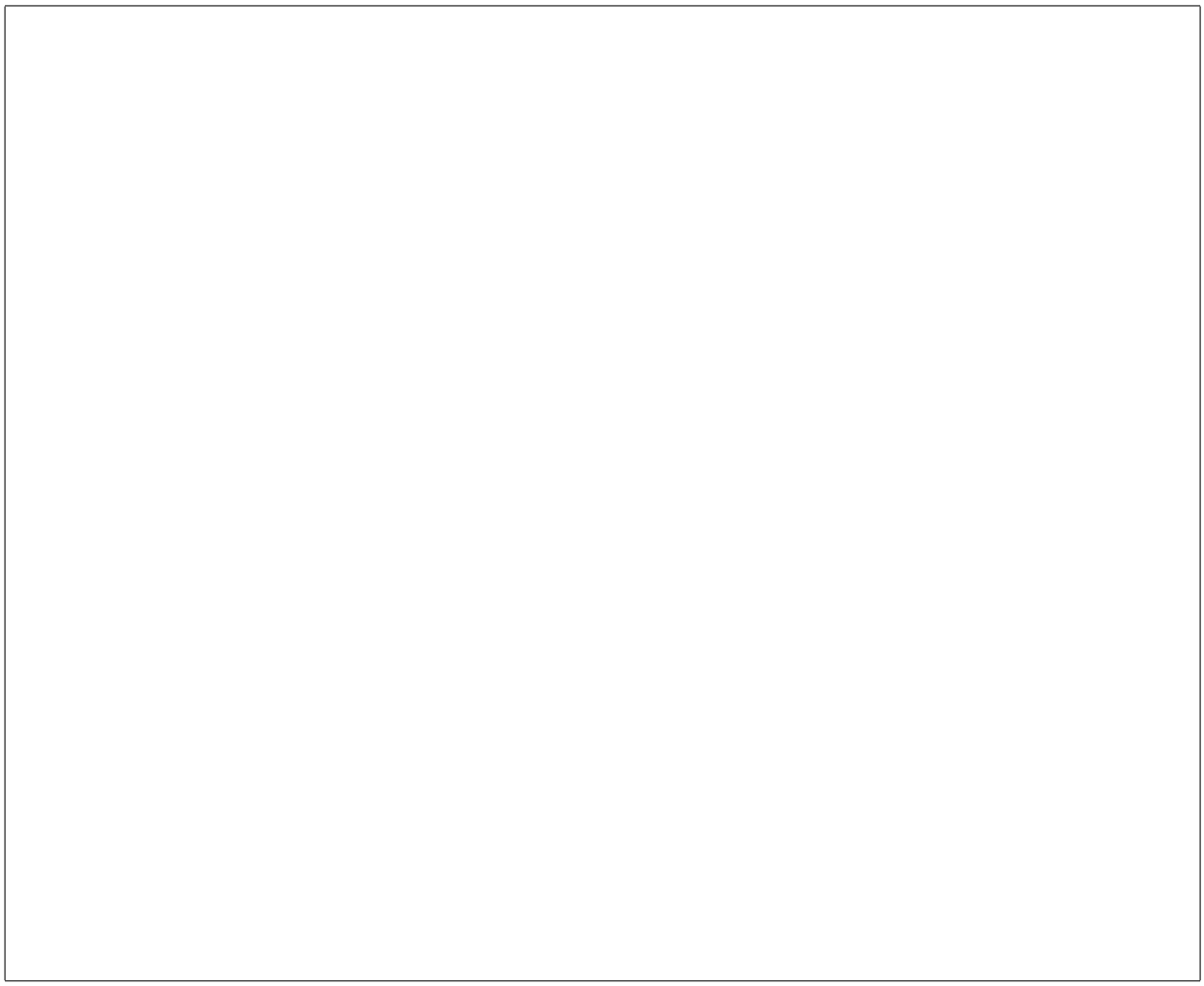
2. `ucbl:emmanuel.coquery rdf:type ucbl:compte-cas .`

3. `ucbl:m1info rdf:type ucbl:formation .`

Question 2: Réécrire la requête SPARQL suivante pour prendre en compte les triplets inférables par les déclarations RDF Schema ci-dessus.

```
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
PREFIX ucbl:<http://www.univ-lyon1.fr/>

SELECT ?u WHERE {
?u ucbl:user ucbl:tomuss .
?u rdf:type ucbl:personnel .
?u ucbl:batiment ucbl:nautibus .
}
```



Exercice 4: SPARQL en XQuery

On considère des données RDF représentées en XML de la manière suivante :

- étant donnée une URI de la forme *url*`#`*fragment*, on lui associe la paire (*url*`#`, *fragment*) où *url*`#` est vu comme un espace de nommage et *fragment* est vu comme un nom local.
- chaque noeud n du graphe qui est sujet d'un prédicat est représenté par un élément e. Le nom de cet élément est `rdf:Description` et il possède un attribut `rdf:about` dont la valeur est n
- pour chaque triplet dont n est sujet, e possède un élément enfant p. Cet élément a pour nom la paire associée à l'URI du prédicat. Si le sujet est un littéral, ce dernier est représenté par un noeud texte fils de cet élément. Sinon cet élément possède un attribut `rdf:resource` dont la valeur est l'URI du sujet.

Voici un exemple d'un tel document :

```
<rdf:RDF
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:autos="http://www.mesautos.com/autos#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
  <rdf:Description
       rdf:about="http://www.mesautos.com/voiture/123456">
    <autos:marque
         rdf:resource="http://www.mesautos.com/marque/2"/>
    <autos:annee>2013</autos:annee>
  </rdf:Description>
  <rdf:Description rdf:about="http://www.mesautos.com/marque/2">
    <rdfs:label>Renault</rdfs:label>
  </rdf:Description>
</rdf:RDF>
```

et sa version Turtle :

```
PREFIX autos:<http://www.mesautos.com/autos#>
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>

<http://www.mesautos.com/voiture/123456> autos:marque
                                <http://www.mesautos.com/marque/2> .
<http://www.mesautos.com/voiture/123456> autos:annee "2013" .
<http://www.mesautos.com/marque/2> rdfs:label "Renault" .
```

Soit la requête SPARQL suivante

```
PREFIX autos:<http://www.mesautos.com/autos#>
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
CONSTRUCT {
  ?v autos:prix ?pr .
}
WHERE {
  ?v autos:nbportes ?p .
  ?v autos:marque ?m .
  ?v autos:prix ?pr
  ?m rdfs:label "Peugeot" .
  FILTER(?p >= 4).
}
```

Ecrire une requête XQuery permettant de calculer la même chose que la requête SPARQL. Les formats d'entrée et de résultat doivent suivre la description ci-dessus, les espaces de nommage utiles ont été déclarés.

```
declare namespace autos="http://www.mesautos.com/autos#";
declare namespace rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#";
declare namespace rdfs="http://www.w3.org/2000/01/rdf-schema#";
```

Exercice 5: SPARQL en Map/Reduce

On considère la requête SPARQL suivante :

```
PREFIX ucbl:<http://www.univ-lyon1.fr/>

SELECT ?e,?p WHERE {
  ?e ucbl:inscrit ?ue.
  ?p ucbl:enseigne ?ue.
}
```

Soit t un triplet. On suppose que son sujet (respectivement son prédicat et son objet) est accessible via $t.sujet$ (respectivement $t.predicat$ et $t.objet$). Ecrire un *mapper* et un *reducer* permettant de calculer le résultat de la requête SPARQL précédente. On considère que chaque triplet est stocké indépendamment des autres (i.e. le *mapper* traite un triplet à la fois).

Remarques :

- Il est possible d'utiliser des URI comme clés.
- Il est possible d'utiliser des structures ad-hoc. Dans ce cas ont utilisera une notation type JSON, par exemple `{a:"toto", b:3}` est la structure dont le champ `a` est la chaîne `"toto"` et le champ `b` est l'entier 3.

Mapper :

Reducer :