MLBDA – 4I801- Examen réparti du 17 Janvier 2018

Seuls les documents de cours et de TD sont autorisés – Durée : 2h.

Répondre aux questions sur la feuille du sujet dans les cadres appropriés. Utiliser le dos de la feuille précédente si la réponse déborde du cadre. Le barème est donné à titre indicatif. La qualité de la rédaction sera prise en compte. Ecrire à l'encre bleue ou noire. Ne pas dégrafer le sujet. Eteindre et **ranger** tout **téléphone** et autre appareil électronique.

EX1	EX2	EX3	EX4	Total

Exercice 1. XSchema — 5 pts

Question 1. (3 pts)
On considère le fichier *test.xsd* suivant :

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:element name="A">
          <xsd:complexType mixed="true">
             <xsd:choice minOccurs="2" maxOccurs="3">
                 <xsd:sequence>
                     <xsd:element name="B" minOccurs="0"type="xsd:integer"/>
                 </xsd:sequence>
                 <xsd:sequence>
                     <xsd:element name="C" maxOccurs="3" type="myint"/>
                     <xsd:element name="D" minOccurs="0" type="xsd:integer"/>
                 </xsd:sequence>
             </xsd:choice>
          </xsd:complexType>
       </xsd:element>

<xsd:simpleType name="myint">
          <xsd:restriction base="xsd:integer">
           <xsd:minExclusive value="1"/>
           <xsd:maxInclusive value="6"/>
          </xsd:restriction>
</xsd:simpleType>
</xsd:schema>
```

On considère également les 6 fichiers XML suivants, bien formés. Pour chacun de ces fichiers, indiquez s'il est valide par rapport au schéma *test.xsd.*

1. `<A><C>1</C><D>2</D></A>`

Entourez la bonne réponse : valide non valide

2. `<A>abc<C>6</C>def<C>2</C><B>0</B></A>`

Entourez la bonne réponse : valide non valide

3. `<A><C>6</C><C>2</C><C>5</C><C>4</C>abc<D>3</D></A>`

Entourez la bonne réponse : valide non valide

4. `<A><B>10</B></A>`

Entourez la bonne réponse : valide non valide

5. `<A><B>1</B><D>1</D>abc<C>2</C></A>`

Entourez la bonne réponse : valide non valide

6. `<A><B>abc</B><C>2</C><D>2</D></A>`

Entourez la bonne réponse : valide non valide

Question 2. (2 pts)

On considère le fichier *magasin.xsd* suivant, décrivant des clients et leurs commandes dans ce magasin.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
3 <xs:element name='magasin'>
4  <xs:complexType>
5   <xs:sequence>

6     <xs:element name='clients'>
7     <xs:complexType>
8       <xs:sequence>
9         <xs:element name='client' type='ClType' minOccurs='0' maxOccurs='unbounded' />
10      </xs:sequence>
11    </xs:complexType>
12    </xs:element>

13    <xs:element name='commandes'>
14    <xs:complexType>
15      <xs:sequence>
16        <xs:element name='commande' type='CdeType'  minOccurs='0' maxOccurs='unbounded' />
17      </xs:sequence>
18    </xs:complexType>
19   </xs:element>

20  </xs:sequence>
21 </xs:complexType>
22 </xs:element>

23   <xs:complexType name='ClType'>
24      <xs:sequence>
25          <xs:element name='nom' type='xs:string'/>
26          <xs:element name='prenom' type='xs:string'/>
```

```
27          <xs:element name='dateNaissance' type='xs:string'/>
32       </xs:sequence>
33       <xs:attribute name='clientID' type='xs:integer'/>
34    </xs:complexType>

35   <xs:complexType name='CdeType'>
36      <xs:sequence>
37         <xs:element name='clientID' type='xs:integer'/>
38         <xs:element name='dateCommande' type='xs:date'/>
39         <xs:element name='dateLivraison' type='xs:date'/>
40         <xs:element name='article' type='xs:string'/>
41        <xs:element name='cout' type='xs:integer'/>
42       </xs:sequence>
43    </xs:complexType>

44 </xs:schema>
```

Complétez ou modifiez le schéma *magasin.xsd* lorsque c'est possible pour qu'il vérifie les contraintes d'intégrité suivantes. Vous utiliserez les numéros de ligne pour indiquer la ou les lignes que vous modifiez, ou bien l'endroit où vous insérez des instructions.

a) Le coût d'une commande doit être supérieur ou égal à 10.

b) Un document ne peut pas contenir deux fois le même client

c) La date de livraison doit être postérieure à la date de commande.

d) Pour qu'une commande soit valide, elle doit concerner un client existant.

Exercice 2. XPath — 4 pts

On considère la DTD ci-dessous qui décrit les planètes du système solaire. Les données considérées renseignent des informations de base sur les planètes : leur nom, leur diamètre, leurs satellites naturels ainsi que leurs composants chimiques. L'ordre des planètes dans le document reflète leur ordre dans la réalité (la Terre, Mars, etc). On connaît pour chaque satellite naturel son nom et la distance qui le sépare de son astre.

```
<!ELEMENT solaire (planete)*>
<!ELEMENT planete (nom, diametre, satellite*, composant*)
<!ELEMENT nom (#PCDATA)>
<!ELEMENT diametre (#PCDATA)>
<!ELEMENT satellite (nom, distance)>
<!ELEMENT distance (#PCDATA)>
<!ELEMENT composant (#PCDATA)>
```

Remarque : A toutes fins utiles, un document, *solar.xml*, conforme pour cette DTD est fourni en annexe.

Exprimer en XPath les requêtes suivantes.

1- Retourner les planètes ayant un diamètre inférieur au diamètre des planètes qui lui sont immédiatement proches, i.e. la planète précédente et la suivante.

Par exemple, si on applique cette requête sur *solar.xml*, elle doit retourner la planète 'Mars'.

2- Retourner les planètes du système interne qui ont un même composant qu'une des planètes du système externe. Le système interne est composé des quatre premières planètes (allant de Mercure jusqu'à Mars inclus), le reste des planètes sont dans le système externe.

Par exemple, si on applique cette requête sur *solar.xml*, elle doit retourner les planètes 'Mercure' et 'Venus'.

3- Retourner les planètes dont on connait au moins un composant et dont aucun des composants ne porte le nom d'un satellite naturel.

Par exemple, si on applique cette requête sur *solar.xml*, elle doit retourner les planètes 'Mercure', 'Terre' et 'Mars'.

4- Retourner les planètes qui ont deux satellites co-orbitaux. Des satellites sont co-orbitaux lorsqu'ils sont à la même distance de leur planète.

Par exemple, si on applique cette requête sur *solar.xml*, elle doit retourner la planète 'Jupiter'.

Exercice 3. XQuery — 5 pts

On considère la DTD ci-dessous qui décrit des données sur des publications scientifiques. On distingue trois types de publications : *article*, *these* et *master*. Tout article doit comporter au moins un auteur, et exactement : un éditeur, un titre, une année de publication (annep). Il doit en outre citer au moins une publication de la base. Une thèse doit contenir exactement : un auteur, un titre, une année de publication (annep), une université de rattachement (univ). Une thèse peut citer un nombre arbitraire de publications de la base. La structure d'un master est identique à celle d'une thèse.
Chaque publication est identifiée par son attribut *cle* et on se sert du mécanisme ID-IDREF pour citer les articles. On connaît pour chaque article le rang de la conférence dans laquelle il est publié.
Remarque : aucun document n'est fourni en annexe pour cet exercice.

Exprimer en XQuery les requêtes suivantes.

```
<!ELEMENT publiScience (article|these|master)*>
<!ELEMENT article (auteur+, editeur, titre, anneep, citation+)>
<!ATTLIST article cle ID #REQUIRED
        rang CDATA #REQUIRED >
<!ELEMENT auteur (#PCDATA)>
<!ELEMENT editeur (#PCDATA)>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT anneep (#PCDATA)>
<!ELEMENT citation EMPTY>
<!ATTLIST citation
        ref IDREF #REQUIRED>

<!ELEMENT these (auteur, titre, anneep, univ, citation*)>
<!ELEMENT datep (#PCDATA)>
<!ELEMENT univ (#PCDATA)>
<!ATTLIST these
        cle ID #REQUIRED>

<!ELEMENT master (auteur, titre, anneep, univ, citation*)>
<!ATTLIST master
    cle ID #REQUIRED>
```

1- **(1,5 pt)** Retourner pour chaque article le nombre des références vers lui, i.e. le nombre d'articles qui le citent. Le résultat doit être trié par le rang de l'article et avoir le format suivant :

```
<resultat>
    <article cle= '…'  rang ='…' nbRef ='...'/>
    …
</resultat>
```

2- **(1,5 pt)** Retourner les auteurs qui s'auto-référencent, i.e. les auteurs qui citent, dans au moins une de leurs publications, une autre publication dans laquelle ils sont auteur. Le résultat final ne doit pas comporter de doublons et doit avoir la forme suivante :

```
<resultat>
    <auteur> …</auteur>
    …
</resultat>
```

3- **(0,5 pt)** Décrire ce que permet de calculer la requête ci-dessous.

```
for $a in //auteur, $b in //auteur
where $a/following-sibling::auteur = $b and  $b/preceding-sibling::auteur = $a
return <res>{$a,$b}</res>
```

4- **(1,5 pt)** Retourner les articles où tous les auteurs sont soit auteur d'une thèse soit auteur d'un master. Le résultat final doit avoir la forme suivante :

<resultat>

<article cle= '…'/>

…

</resultat>

Exercice 4. RDF et SPARQL 6 pts

Question 1 (1,5 pt). Donnez le graphe RDF qui correspond à l'expression Turtle suivante. On suppose que les nœuds et les propriétés appartiennent à l'espace de noms `http://example.org/.` Les nœuds du graphe doivent être étiquetés par leur identifiants ou par une lettre précédée par '_:' pour les nœuds blancs. Les étiquettes des arcs représentent des propriétés.

```
@prefix : <http://example.org> .
:n1 :p :n2,[:p :n3; :r :n4] ; :r :n4 .
:n2 :r :n3; :p :n3, _:y, :n4 .
:n3 :r _:y .
```

Réponse:

Considérons les triplets du document *amis.ttl* donnés sous forme factorisée (cf. annexe). Ces données peuvent être représentées par un graphe ***g1*** *(différent de celui de l'exercice précédent).* Exprimez les requêtes **SPARQL** qui retournent les informations suivantes. Vous n'êtes pas obligés de préciser les espaces de noms utilisés ni les données interrogées (expressions SELECT...WHERE)

Question 2 (0,75 pt). Donnez la liste des âges inférieurs à 21 et des noms des personnes qui commencent par la lettre "T". Le résultat est le suivant:

liste
15
20
"Tom"
"Tim"

SELECT

WHERE

Question 3 (0.75pt). Les couples de personnes qui ont le même ami. **Chaque couple doit apparaître une seule fois**. Le résultat est le suivant:

nom1	nom2
"Tim"	"Mike"

SELECT

WHERE

Question 4 (1 pt). Les noms des personnes qui sont amies de quelqu'un qui n'est pas lui-même ami de quelqu'un. Le résultat est le suivant ("Mike" est l'ami de "Mary", qui n'est l'amie de personne):

nom
"Mike"

SELECT

WHERE

Question 5 (1 pt). Pour chaque personne, l'âge moyen des personnes qui l'ont désignée comme ami (:u5 doit faire partie du résultat). On utilisera un group by. Le résultat est le suivant:

user	agemoy
:u4	35
:u5	
:u2	25
:u3	20
:u1	27,5

SELECT

WHERE

Question 6 (1 pt). Pour chaque personne (:u5 **ne doit pas** faire partie du résultat), l'âge de la personne la plus âgée l'ayant désignée comme ami. **NE PAS** utiliser une fonction d'agrégation (par exemple max). Le résultat est le suivant (par exemple, pour :u1, :u2 est plus âgé que :u4):

user	ami	age
:u2	:u5	25

:u1	**:u2**	**40**
:u4	:u3	35
:u3	:u1	20

SELECT

WHERE