Nom :	Prénom :	*page 1*

MLBDA – 4I801- Examen réparti du 4 Janvier 2017

Version avec SOLUTIONS

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 — 4 pts

On considère la DTD suivante représentant une base de personnes :

```
< !ELEMENT Base(personne )* >
< !ELEMENT personne (nom, prenom, datenais, enfant *)>
< !ATTLIST Personne pseudo ID #REQUIRED
                    pere IDREF #IMPLIED
                    mere IDREF #IMPLIED >
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT datenais (#PCDATA)>
<!ELEMENT enfant  (#PCDATA)>
```

Chaque personne a un pseudo, utilisé par exemple pour indiquer les enfants d'une personne. Le fragment XML suivant décrit une personne Pierre et son fils Luc.

```
<personne pseudo='pierrot'>
  <nom >Durand</nom>
  <prenom>Pierre</prenom>
  <datenais>1970-01-01</datenais>
  <enfant>lulu</enfant>
</personne>

<personne pseudo='lulu' pere='pierrot'>
  <nom>Durand</nom>
  <prenom>Luc</prenom>
  <datenais>2000-01-20</datenais>
</personne>
```

Question 1 (1pt). On souhaite modéliser ces informations en XSchema. Compléter le schéma suivant en définissant le type PersonType.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE xs:schema SYSTEM "XMLSchema.dtd">
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" >

<xs:complexType name='PersonType' >
...
...
...
...
...
```

```
<xs:element name='base'>
  <xs:complexType>
    <xs:sequence maxOccurs='unbounded'>
      <xs:element name='personne' type='PersonType' />
    </xs:sequence>
    </xs:complexType>
</xs:element>
```

1 pt

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE xs:schema SYSTEM "XMLSchema.dtd">
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" >

<xs:complexType name='PersonType'>
      <xs:sequence>
        <xs:element name='nom' type='xs:string' />
        <xs:element name='prenom' type='xs:string' />
        <xs:element name='datenais' type='DatePossible' />
        <xs:element name='enfant' type='xs:string' minOccurs='0'
maxOccurs='unbounded' />
      </xs:sequence>
      <xs:attribute name='ident' type='xs:string' use='required'/>
      <xs:attribute name='pere' type='xs:string' use='optional'/>
      <xs:attribute name='mere' type='xs:string' use='optional'/>
 </xs:complexType>
 ICI, on peut mettre DateNaissance et persMajType

<xs:element name='base17'>
  <xs:complexType>
    <xs:sequence maxOccurs='unbounded'>
      <xs:element name='personne' type='PersonType' />
    </xs:sequence>
    </xs:complexType>
 ICI ajouter la clef(question b), puis la keyref (question c))

</xs:element>
```

Question 2 (2 pts). Complétez ou modifiez le schéma pour prendre en compte les contraintes suivantes, en précisant à quel endroit du schéma les ajouts doivent être insérés (ou en réécrivant l'élément que vous modifiez) :

a) La date de naissance doit être inférieure au 1er janvier 2017.

0,5 pt

```
<xs:simpleType name='Datenaissance'>
<xs:restriction base='xs:date'>
  <xs:maxExclusive value="2017-01-01"/>
  </xs:restriction>
</xs:simpleType>
```

Peut être ajouté à plusieurs endroits, par ex. après la définition de PersonType.

b) Une personne est identifiée de façon unique par son pseudo.

```
0,75 pt
     <xs:key name='clePers'>
      <xs:selector xpath='personne'/>
      <xs:field xpath='@ident'/>
    </xs:key>
```

Doit être ajouté dans l'élément base, juste avant la balise fermante (ou juste après le </xscomplexType>

c) Un enfant est une personne

```
0,75pt
<xs:keyref name='assocEnfantPersonne' refer ='clePers'>
      <xs:selector xpath ='personne/enfant'/>
      <xs:field xpath='.'/>
    </xs:keyref>
A mettre juste après la clef (juste avant la balise fermante de l'élément base).
0,5pt pour l'emplacement de b et c.
```

Question 3 (1pt). On veut pouvoir distinguer dans cette base les personnes majeures. Les personnes majeures peuvent avoir un conjoint, une profession et un employeur. Le conjoint et l'employeur sont modélisés sous forme d'éléments, la profession sous forme d'attribut. Définir un type PersMajeureType à partir du type PersonType décrivant les personnes majeures.

```
<xs:complexType name='PersMajeureType'>
...
...
...
...
...
...
...
```

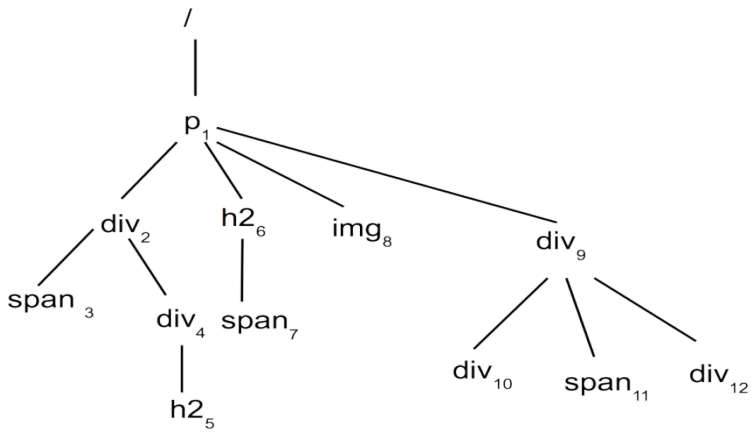
```
...
```

1,5 pt

```
<xs:complexType name='PersMajType'>
  <xs:complexContent>
    <xs:extension base='PersonType'>
      <xs:sequence>
     <xs:element name="employeur" type='xs:string' minOccurs='0'/>
     <xs:element name=”conjoint” type=”xs:string” />
      </xs:sequence>
      <xs:attribute name='profession' type='xs:string' use='optional'/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Exercice 2. XPath 5 pts

On considère l'arbre XML suivant qui modélise un extrait d'un document HTML. Dans cet arbre chaque nœud est identifié par un attribut @id (l'identifiant de la racine est égal à 0, son élément fils de type p a l'identifiant 1, etc):



Le résultat d'une expression XPath est une liste de nœuds DOM triés dans l'ordre du document, sans doublon. Par exemple, l'expression `/descendant::div/following-sibling::*/@id` retourne les identifiants `6,8,9,11,12`.

Question 1 (3 pts). Donnez pour chaque expression XPath la liste des identifiants des nœuds qui sont retournés.
1. */descendant::div[child::*]/@id*

Réponse:

```
2, 4, 9
```

2. */descendant::div/descendant::span/@id*

Réponse:

```
3, 11
```

3. */descendant::*/child::*[1]/@id*

Réponse:

```
2, 3, 5, 7, 10
```

4. */descendant::*/following-sibling::div[1]/@id*

Réponse:

```
4, 9, 12
```

5. */descendant::span[not(following-sibling::*)]/@id*

Réponse:

```
7
```

6. */descendant::*[child::div[not(following-sibling::*)]]/@id*

Réponse:

```
1, 2, 9
```

Question 2 (2 pts). Donnez pour chaque séquence de nœuds ci-dessous une expression XPath (syntaxe étendue ou abrégée) qui la calcule:

1. *3, 7, 11*

Réponse:

```
//span/@id
```

2. *4, 9, 12*

Réponse:

```
//div[preceding-sibling::*]/@id
```

3. *4, 6, 11*

Réponse:

```
//*[2]/@id
```

4. *1, 4*

Réponse:

```
//h2/../@id
```

Exercice 3. XQuery 5 pts

Considérons le document *projets.xml* suivant, qui contient trois projets avec leurs employés, leur location et leur budget.

```
<projects>
    <project name='Accounting'>
        <employee><ln> Jenkins</ln><fn> David</fn></employee>
        <employee><ln>Williams </ln><fn> Jessica</fn> </employee>
        <employee><ln>Smith </ln><fn>Alex </fn> </employee>
        <budget> 400 </budget>
        <location>New York </location>
            <duration>24 months</duration>
    </project>
    <project name='Quality'>
        <employee><ln>Crosby </ln><fn>Julian </fn></employee>
        <employee><ln> Williams</ln><fn>Jessica </fn> </employee>
        <employee><ln> Perry</ln><fn>Karl </fn> </employee>
        <budget> 200 </budget>
        <location> San Francisco</location>
    </project>
    <project name='Design'>
        <employee><ln>Crosby </ln><fn>Williams </fn></employee>
        <employee><ln> Thomas</ln><fn>Martin </fn> </employee>
        <location>San Francisco</location>
            <budget>100</budget>
            <duration>12 months </duration>
    </project>
</projects>
```

Question 1 (1 pt). Donner le nombre et la liste d'éléments *<projects>* qui seront retournés par la requête Xquery suivante:

```
<resultat>
  {  for $p in document("projects.xml")/projects, $p1 in $p/project, $p2 in $p/project
     where $p1//ln=$p2//ln
     return
              <projects p1= "{$p1/name}"  p2= "{$p2/name}"/>
  }
</resultat>
```

Nombre de résultats:

Liste de résultats:

<projects p1=..... p2=.....> .....

7

<projects p1="Accounting" p2="Accounting"/> <projects p1="Accounting" p2="Quality"/>
<projects p1="Quality" p2="Accounting"/> <projects p1="Quality" p2="Quality"/>
<projects p1="Quality" p2="Design"/> <projects p1="Design" p2="Quality"/> <projects p1="Design" p2="Design"/>

Question 2 (1 pt). Exprimez en français la requête suivante et donnez son résultat:

```
<résultat>
{ for $l in distinct-values(document("projects.xml")//project/location)
  return
     {   for $p in document("projects.xml")//project[location=$l]
         where every $x in document("projects.xml")//project satisfies $x/budget <= $p/budget
         return $l
     }
}
</résultat>
```

Description en français:

Résultat:

La requête donne le nom de la location où se trouve le projet avec le budget le plus élevé

```
<resultat>
  <location> New York </location>
</resultat>
```

Question 3 (1 pt). Exprimez en XQuery la requête qui retourne pour le troisième projet son nombre d'employés ainsi que leur nom. Le résultat attendu est le suivant:

<project nb_emp='2'> <ln> Crosby </ln> <ln> Thomas </ln> </project>

Réponse:

```
for  $p in  document(" projects.xml")//project[3]
let  $e := $p/employe
return <project nb_emp="{count ($e)}">{ $ e/l n } </project>
```

Question 4 (1 pt). Exprimez en XQuery la requête qui retourne les employés qui travaillent uniquement dans des projets situés à San Francisco. Le résultat attendu est le suivant (*Williams* n'est pas retournée car elle travaille aussi à New York):

<résultat><ln>Crosby</ln> <ln>Perry</ln><ln>Thomas</ln></résultat>

Réponse:

```
<résultat>
for $e  i n distinct-values(document ( "projects. xml " ) / / employe)
    let  $p  := document ( " projects . xml " )//project[employe/ln = $e /l n ]
    where every $pp  in  $p satisfies $pp/location = "San Francisco"
     return  {$e/ln}
</résultat>
```

Question 5 (1 pt). Exprimez en XQuery la requête qui retourne la durée pour les projets dont la durée est connue, et une balise <unknown-duration> pour les projets dont la durée n'est pas connue. Le résultat attendu est le suivant:

<résultat>
<duration project="Accounting">24 months </duration>
<unknown-duration project="Quality"/>
<duration project="Design">12 months </duration>
</résultat>

Réponse:

```
<résultat>
for $p  in  document ("projects.xml")//project
let $d = $p/duration
return
{if exists($p) then  <duration project='{$p/@name}' >{$d}</duration>
else <unknown-duration project='{$p/@name}'/>}
</résultat>
```

Exercice 4. SPARQL — 6 pts

Considérons les triplets du document *biblio.ttl* donnés sous forme factorisée (cf. annexe)
Exprimer les requêtes **SPARQL** qui retournent les informations suivantes.

Question 1 (0.5 pt). Nombre de chercheurs.

Le résultat de la requête est :

?res
4

Q1

```
select (count(?a) as ?nb_aut)
{?a a :researcher}
```

Question 2 (0.5 pt). Nombre d'institutions.

Le résultat de la requête est :

?res
2

Q2

```
select (count(distinct ?u) as ?nb_univ)
{?a :institution ?u}
```

Question 3 (0.5 pt). Nombre d'étudiants appartenant à deux institutions différentes.

Le résultat de la requête est :

?res
1

Q3

```
select (count(distinct ?e) as ?nb_student)
{?e :institution ?u . ?e :institution ?v Filter(?v!=?u) }
```

Question 4 (0.5 pt). Nombre d'étudiants qui n'ont aucune publication.

Le résultat de la requête est :

?res
2

Q4

```
select (count(distinct ?e) as ?nb_nopubli)
{?e a :etudiant minus {?e :auteur ?p}}
```

Question 5 (1 pt). Chercheurs qui encadrent au moins un étudiant avec qui ils n'ont pas eu d'article en commun

Le résultat de la requête est :

?res
:fafati
:jalman
:dascu

Q5

```
select distinct ?a
{?a :encadre ?e optional {?a :auteur ?p. ?e :auteur ?p} filter(!bound(?p))}
#{?a :encadre ?e minus {?a :auteur ?p. ?e :auteur ?p} }
```

Question 6 (1 pt). Paires de chercheurs qui encadrent au moins deux étudiants en commun. Le résultat de la requête ne doit comporter ni doublons ni de redondance d'information. Il doit être comme indiqué ci-dessous **.**

?c1	?c2
:dascu	:jalman

Q6

```
select distinct ?r ?p
{?r :encadre ?e1. ?p :encadre ?e1.  ?r :encadre ?e2. ?p :encadre ?e2 Filter
(str(?r)<str(?p) && ?e1!=?e2)}
```

Question 7 (1.5 pt). Les publications co-rédigées par exactement deux chercheurs et dont aucun co-auteur n'est étudiant. Le résultat doit être trié par ordre alphabétique**.**

Le résultat de la requête est :

?p
:mapred
:views

Q7

```
select distinct ?p
{?a1 :auteur ?p. ?a2 :auteur ?p. ?a1 a :chercheur. ?a2 a :chercheur optional {?e
:auteur ?p. ?e a :etudiant} Filter(!bound(?e) && ?a1!=?a2) }
order by ?p
```

Question 8 (0.5 pt). Indiquer ce que retourne cette requête en français.

ask {?a1 :auteur ?p. ?a2 :auteur ?p filter(str(?a1)<str(?a2))**}**

```
Y a t il des publications avec au moins deux auteurs distincts ?
La réponse de la requête est le booleen vrai
```