TD : XSCHEMA

L'objectif de ce TD/TME est de maîtriser le langage de définition de schéma XSchema.

Notions : définir un élément, un attribut, un type simple ou complexe. Définir le contenu d'un élément : (sequence, choix, all) et les occurrences. Définir une restriction de type. Définir un espace de noms, manipuler un espace de nom prédéfini. etc…

Exercice 1. Conformité d'un document XML

On considère le fichier *test.xsd* suivant :

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

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

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

1. `<A><B/><B/></A>`
2. `<A><C>1</C><D>2</D></A>`
3. `<A>abc<C>6</C>def<C>2</C><B>0</B></A>`
4. `<A><C>6</C><C>7</C><C>5</C><C>4</C>abc<D>3</D></A>`
5. `<A><B>7</B></A>`
6. `<A><B>1</B><D>1</D>hello<C>2</C>bonjour</A>`
7. `<A><B>abc</B><C>2</C><D>2</D></A>`
8. `<A><B>2</B><C>2</C></A>`

Exercice 2. Contraintes : Cas des commandes dans un magasin

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 3. Schéma pour des personnes

On veut compléter le fichier *Personnes.xsd* suivant :

```
1<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
2  <xs:element name="personnes">
3                        <xs:complexType>
4                              <xs:sequence maxOccurs="unbounded" >
5                                      <xs:element ref="personne"/>
6                              </xs:sequence>
7                      </xs:complexType>
8  </xs:element>
9
10  <xs:complexType name="T-Personne">
11                     <xs:sequence>
12                             <xs:element name="nom"            type="xs:string"/>
13                             <xs:element name="prénom"         type="xs:string"/>
14                             <xs:element name="dateN"          type="xs:date"/>
15                             <xs:element name="genre"          type="T-genre" />
16                             <xs:element name="enfants"        type="T-enfants"/>
17                     </xs:sequence>
18                     <xs:attribute name="idP"  type="xs:string"            use="required" />
19   </xs:complexType>
20
21    <xs:element name="personne"          type="T-Personne"/>
22</xs:schema>
```

Question 1. Définissez le type *T-genre*, sachant que l'élément *genre* peut prendre les valeurs 'masculin' ou 'féminin'.

Question 2. Définissez le type *T-enfants*, qui décrit l'ensemble des enfants d'une personne. Pour chaque enfant, on donnera le prénom, l'âge et une référence à la personne représentant sa mère. Toutes ces informations doivent être représentées par des attributs et doivent toujours être renseignées.

Question 3. Complétez ou modifiez le schéma *personnes.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, et l'endroit où vous insérez des instructions.

a) Les personnes sont identifiées de façon unique.

b) La mère d'un enfant est une personne

c) L'âge des enfants est inférieur à 18 ans.

d) Les prénoms des enfants d'une même fratrie sont tous différents.

Question 4. Définir un sous-type T-PersonneSansEnfantt du type T-Personne pour représenter les personnes n'ayant pas d'enfant.

Question 5. Définir un sous-type *T-PersonneAvecProfession* du type *T-Personne* qui contient la profession d'une personne et son employeur s'il en a un. La profession et l'employeur sont de type *xs:string*.

Question 6. Complétez le type *T-Personne* pour pouvoir représenter le conjoint d'une personne (indiquez clairement à l'aide des numéros de ligne les endroits modifiés).

Exercice 4 : Jeu de cartes

Soit un jeu de 32 cartes composé de 4 as, 4 rois, 4 dames, 4 valets, et 16 petites cartes (valant 7, 8, 9 ou 10). Une application de jeu utilise le format XML pour représenter les mains de cartes.

Question 1 : Unicité

Pour prendre en compte la couleur des cartes, et les mains distribuées aux joueurs, on considère le schéma suivant :

```
 1  <?xml version="1.0" encoding="ISO-8859-1"?>
 2  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 3
 4    <xs:element name="jeu">
 5      <xs:complexType>
 6        <xs:sequence maxOccurs="unbounded">
 7            <xs:element ref="main"/>
 8        </xs:sequence>
 9      </xs:complexType>
10    </xs:element>
11
12    <xs:element name="main">
13      <xs:complexType>
14        <xs:sequence>
15            <xs:element name="C" maxOccurs="unbounded">
16              <xs:complexType>
17                <xs:attribute name="nom" type="nom_carte" use="required"/>
18                <xs:attribute name="coul" type="coul_carte" use="required"/>
19              </xs:complexType>
20            </xs:element>
21        </xs:sequence>
22      </xs:complexType>
23    </xs:element>
24
25    <xs:simpleType name="nom_carte">
26      <xs:restriction base="xs:string">
27        <xs:enumeration value="A"/><xs:enumeration value="R"/>
28        <xs:enumeration value="D"/><xs:enumeration value="V"/>
29        <xs:enumeration value="10"/><xs:enumeration value="9"/>
30        <xs:enumeration value="8"/><xs:enumeration value="7"/>
31      </xs:restriction>
32    </xs:simpleType>
33    <xs:simpleType name="coul_carte">
34      <xs:restriction base="xs:string">
35        <xs:enumeration value="Pi"/><xs:enumeration value="Co"/>
36        <xs:enumeration value="Ca"/><xs:enumeration value="Tr"/>
37      </xs:restriction>
38    </xs:simpleType>
39  </xs:schema>
```

1) On souhaite modifier le schéma afin que les contraintes suivantes soient vérifiées :

C1: Une main a exactement 5 cartes,

C2: il y a entre 2 et 4 joueurs (avec une main par joueur).

C3: Chaque carte est unique, il ne peut y avoir deux fois la même carte de même couleur dans le jeu.

Pour chaque contrainte, indiquer le numéro des lignes à modifier, puis écrivez seulement le morceau modifié du schéma.

Question 2 : élement *all*

L'objectif de cette question est d'approfondir la notion d'ensemble (non ordonné) défini avec l'élément all. On représente les cartes par des éléments : A (as), R (roi), D (dame), V (valet) et N pour les petites cartes 7, 8, 9 et 10.

```
 1 <xs:complexType name =  "T_carte">
 2       <xs:attribute name="coul" type="coul_carte" use="required"/>
 3  </xs:complexType>
 4
 5 <xs:element name="A" type = "T_carte"/>
 6 <xs:element name="R" type = "T_carte"/>
 7 <xs:element name="D" type = "T_carte"/>
 8 <xs:element name="V" type = "T_carte"/>
 9 <xs:element name="N" type = "T_carte"/>
10
```

On suppose qu'un joueur ne **trie pas** ses cartes donc elles peuvent être rangées dans un ordre quelconque dans la main d'un joueur. Proposer un schéma pour décrire une main qui possède :

a) un carré d'as et une autre carte quelconque.

b) Une *suite* (sans ordre) valet dame roi et deux autres petites cartes quelconques

c) Est ce possible de décrire un schéma validant une main avec 2 paires ?

Exercice 5 : Conversion d'un XSchema en DTD : cas de juicers

On sait que la DTD d'un document XML peut être exprimée dans le formalisme Xschema. Mais l'inverse n'est pas toujours possible. L'objectif est d'étudier les cas où la conversion vers une DTD fait perdre des informations.

On considère le XSchema *juicer.xsd*. Ecrire la DTD *juicer.dtd* corespondante.

juicer.xsd

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.juicers.org"
            xmlns="http://www.juicers.org"
            elementFormDefault="qualified">

    <xsd:element name="juicers">
```

```
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="juicer" minOccurs="0"
                                          maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="juicer">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="name"/>
                <xsd:element ref="image"/>
                <xsd:element ref="description"/>
                <xsd:element ref="warranty" minOccurs="0"/>
                <xsd:element ref="weight" minOccurs="0"/>
                <xsd:element ref="cost" maxOccurs="unbounded"/>
                <xsd:element ref="retailer"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="name"              type="xsd:string"/>
    <xsd:element name="image"             type="xsd:string"/>
    <xsd:element name="description"       type="xsd:string"/>
    <xsd:element name="warranty"          type="xsd:string"/>
    <xsd:element name="weight"            type="xsd:string"/>
    <xsd:element name="cost"              type="xsd:string"/>
    <xsd:element name="retailer"          type="xsd:string"/>
</xsd:schema>
```

juicer.xml

```
<?xml version="1.0"?>

<juicers xmlns ="http://www.juicers.org"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.juicers.org juicers.xsd">

      <juicer>
          <name>OJ Home Juicer</name>
          <image>images\mighty_oj.gif</image>
          <description> no substitute for ... </description>
          <warranty>lifetime warranty</warranty>
          <cost>41.95</cost>
          <retailer>http://www.thewhitewhale.com/oj.htm</retailer>
      </juicer>


      <juicer>
          <name>Wheateena Wheatgrass Juicer</name>
          <image>images\wheateena.jpg</image>
          <description>Wheatgrass juice </description>
          <warranty> (6) months.</warranty>
          <weight>46</weight>
          <cost>639.99</cost>
           <retailer>http://www.rawfoods.com </retailer>
      </juicer>

</juicers>
```

Exercice 6 : Schémas imbriqués et types . Cas de juicer.xml

Question 1 Modifier le schéma juicers.xsd de l'exercice précédent afin de le rendre plus compact en imbriquant la déclaration de tous les éléments dans l'élément *juicers*.

Question 2: Modifier le schéma juicers.xsd afin d'utiliser des types plus significatifs que le type String pour la déclaration des éléments *weight*, *cost*, et *retailer*.

Question 3 Modifier le schéma de l'exercice 6.4 en définissant un nouveau type, *money*, afin de l'utiliser dans la déclaration de l'élément *cost*.

Question 4 Modifier le schéma de telle sorte que le namespace par défaut (i.e., sans préfixe) soit celui de XMLSchema et que le namespace des éléments définis (juicers, juicer, etc) ait un préfixe.