TME XSLT (2)

Exercice 1 : Examen Juin 2008 Session 2

On considère des fichiers XML qui contiennent l'arbre généalogique descendant d'une personne (arbre qui contient les descendants (enfants, petits-enfants, arrière-...) d'une personne). Pour chaque personne de l'arbre, on connaît son nom, sa date de naissance et, éventuellement, ses enfants représentés eux aussi comme des personnes. Ces fichiers suivent la DTD genea.dtd suivante :

```
<!ELEMENT personne (nom,date,personne*) >
<!ELEMENT nom (#PCDATA) >
<!ELEMENT date (#PCDATA) >
```

Par exemple, le fichier exemple.xml ci-dessous contient l'arbre généalogique descendant du compositeur Johann Strauss I (il a trois fils connus, et son troisième fils à deux fils connus) :

```
<personne>
   <nom>Johann Strauss I</nom>
   <date>1804</date>
   <personne><nom>Johann Strauss II</nom><date>1825</date></personne>
   <personne><nom>Josef Strauss</nom><date>1827</date></personne>
   <personne>
      <nom>Eduard Strauss</nom>
      <date>1835</date>
      <personne><nom>Johann Strauss III</nom><date>1866</date></personne>
      <personne><nom>Josef Eduard
Strauss</nom><date>1868?</date></personne>
   </personne>
</personne>
```

Remarque : vous répondrez aux questions suivantes afin que vos feuilles XSL puissent traiter tous fichiers XML suivant la DTD genea.dtd.

Question 1. Ecrire une feuille XSL qui transforme un document XML (qui suit la DTD genea.dtd) en un autre document XML qui contient la liste des noms des personnes. Ce nouveau document suit la DTD :

```
<!ELEMENT liste (info*) >
<!ELEMENT info (#PCDATA)>
```

Par exemple, le résultat obtenu pour le fichier exemple.xml est :

```
<liste>
   <info>Johann Strauss I</info>
   <info>Johann Strauss II</info>
   <info>Josef Strauss</info>
   <info>Eduard Strauss</info>
   <info>Johann Strauss III</info>
   <info>Josef Eduard Strauss</info>
</liste>
```

Question 2. Ecrire une feuille XSL qui transforme un document XML (qui suit la DTD genea.dtd) en un nouveau document XML indiquant (dans une balise <nbd>) pour chaque personne le nombre de descendants de cette personne. Ce nouveau document suit la DTD :

```
<!ELEMENT personne (nom,personne*,nbd) >
<!ELEMENT nom (#PCDATA) >
<!ELEMENT nbd (#PCDATA) >
```

Par exemple, le résultat obtenu pour le fichier exemple.xml est :

```
<personne>
   <nom>Johann Strauss I</nom>
   <personne><nom>Johann Strauss II</nom><nbd>0</nbd></personne>
   <personne><nom>Josef Strauss</nom><nbd>0</nbd></personne>
   <personne>
      <nom>Eduard Strauss</nom>
      <personne><nom>Johann Strauss III</nom><nbd>0</nbd></personne>
      <personne><nom>Josef Eduard Strauss</nom><nbd>0</nbd></personne>
      <nbd>2</nbd>
```

```
      </personne>
      <nbd>5</nbd>
   </personne>
```

Question 3 : Ecrire une feuille XSL qui transforme le document XML (qui suit la DTD genea.dtd) en une page HTML affichant d'abord le nom de la personne de l'élément racine, puis les noms de ses fils et le nom de ses petits-fils. Par exemple, le résultat obtenu pour le fichier exemple.xml est :

```
<HTML>
   <H2>Johann Strauss I</H2>
   <p>Ses fils sont :
      <UL>
         <LI>Johann Strauss II</LI>
         <LI>Josef Strauss</LI>
         <LI>Eduard Strauss</LI>
      </UL>
   </p>
   <p>Ses petits-fils sont :
      <UL>
         <LI>Johann Strauss III</LI>
         <LI>Josef Eduard Strauss</LI>
      </UL>
   </p>
</HTML>
```

Question 4. Ecrire une feuille XSL qui transforme un document XML (qui suit la DTD genea.dtd) en un autre document XML qui suit la DTD suivante :

```
<!ELEMENT liste (personne*) >
<!ELEMENT personne (nom,liste?)
<!ATTLIST personne date CDATA #REQUIRED>
<!ELEMENT nom (#PCDATA) >
```

L'élément racine est maintenant un élément <liste>. L'élément <date> a été remplacé par un attribut date dans l'élément <personne>. Lorsque qu'une personne a au moins un enfant, alors la liste de ses enfants se trouve dans un élément <liste>. Par exemple, le résultat obtenu pour le fichier exemple.xml est :

```
<liste>
   <personne date="1804">
      <nom>Johann Strauss I</nom>
      <liste>
         <personne date="1825"><nom>Johann Strauss II</nom></personne>
         <personne date="1827"><nom>Josef Strauss</nom></personne>
         <personne date="1835">
            <nom>Eduard Strauss</nom>
            <liste>
               <personne date="1866">
                  <nom>Johann Strauss III</nom>
               </personne>
               <personne date="1868?">
                  <nom>Josef Eduard Strauss</nom>
               </personne>
            </liste>
         </personne>
       </liste>
   </personne>
</liste>
```

Exercice 2 : Transformation du document XML

Nous utiliserons dans ce TME le fichier « baseProjet.xml ». Il contient une base de films et d'abonnés, stockés de la façon suivante :

1. La racine de l'arbre XML est la balise <videoexpress>,
2. Pour chaque film, il y a un sous-arbre de <videoexpress> qui commence à la balise <film>, et qui contient les informations relatives à ce film,
3. Pour chaque cassette d'un film, il y a un sous-arbre <cassette> du nœud <film> correspondant,
4. Pour chaque cassette d'un film, si elle est actuellement empruntée, il y a un sous-arbre <empres> qui contient un nœud <dateempres> contenant la date d'emprunt ou de réservation, et un sous-arbre

<abonne> qui contient l'ensemble des informations de l'abonné qui a emprunté ou réservé la cassette,

5. Ensuite, pour chaque abonné qui n'a pas de réservation ou d'emprunt en cours, il y a un sous-arbre <abonne> de <videoexpress> qui contient les informations relatives à cet abonné.

Ainsi, la base « baseProjet.xml » stocke d'abord les films. Si un abonné a réservé (ou emprunté) une cassette, il est stocké dans un sous-arbre de la cassette réservée (resp. empruntée).

L'objectif de ce TME est de créer un fichier XSLT qui change le mode de stockage de la base : dans le fichier XML résultant, chaque abonné sera un fils de la racine, et chaque film réservé (ou emprunté) par cet abonné sera stocké dans une balise <empres> de l'abonné. Si un film n'est ni emprunté ni réservé, il est un sous-arbre <film> de la racine. Pour plus de clarté, on fournit le fichier résultant de la transformation attendue (« baseProjet_par_abonne.xml »). L'objectif est donc de créer un fichier XSLT qui permet de transformer « baseProjet.xml » en « baseProjet_par_abonne.xml ». Note : Afin de vérifier vos réponses, pour chaque question, le fichier XML résultant de la transformation demandée est donné dans un fichier : « resultat_q[numéroExercice]_[numéroQuestion].xml »

Partie 1 : Récupérer les films sans les informations d'emprunt/réservation.

La 1ère étape à effectuer est de récupérer les informations relatives à un film sans les informations relatives aux emprunts/réservation de ce film (qui seront stockées, dans le fichier résultant dans une sous-balise <empres> de <abonne>). Cela revient à récupérer les sous-arbres <films> sans les sous-arbres <empres>.

Question 1.1 : Créer un fichier XSLT qui permet de générer un fichier XML dont la racine est <videoexpress>, et qui contient, pour chaque film de « baseProjet.xml », une balise <film> avec les attributs correspondants, sans le sous-arbre.

Question 1.2 : Modifiez votre fichier XSLT pour qu'il y ait, pour chaque film, une balise par cassette de ce film (avec les attributs « support » et « statut » correspondants).

Question 1.3 : Modifier votre fichier XSLT pour qu'il y ait, pour chaque film, exactement le même sous-arbre que dans « baseProjet.xml », avec les balises <empres> supprimées. Note 1 : vous pourrez utiliser l'élément XSLT xsl:copy-of. Note 2 : on se rappellera que, lorsque plusieurs templates correspondent à une clause select d'un élément xsl:apply-templates, le plus spécifique est exécuté.

Partie 2 : Récupérer l'ensemble des abonnés et les films qu'ils ont réservé/emprunté.

La seconde étape est de récupérer l'ensemble des abonnés, et les informations correspondantes. Une des difficultés est liée aux doublons dans la base « baseProjet.xml » : un abonné ayant emprunté ou réservé plus d'une cassette apparaît plusieurs fois. Dans notre base résultat, il faudra donc supprimer les doublons.

Question 2.1 : Ecrire un fichier XSLT qui produit un fichier XML ayant comme racine <videoexpress>, et un sous-arbre <abonne> pour chaque sous-arbre <abonne> existant dans la base « baseProjet.xml ».

Question 2.2 : On suppose que dans la base, les abonnés ont tous des noms différents (balise <noma>). Modifier la requête Xpath du fichier précédent pour ne pas avoir de doublons dans les abonnés. Note : on pourra utiliser l'axe *preceding* dans une requête Xpath.

Question 2.3 : Modifiez votre fichier XSLT pour rajouter un nœud <empres/> sous un nœud <abonne> pour chaque cassette empruntée par l'abonné considéré. Note 1 : on pourra utiliser, dans le template associé aux balises <abonne>, une boucle xsl:for-each, où la clause select va rechercher la liste des cassettes qui possèdent un sous-arbre <abonne> pour lequel la balise <noma> a la même valeur que l'abonné courant. Note 2 : on pourra utiliser la fonction Xpath « current() » dans la clause select de l'élément xsl:for-each.

Question 2.4 : Modifiez votre fichier précédent pour que, sous chaque sous-arbre <empres>, il y a un sous-arbre <dateempres> contenant la date d'emprunt ou de réservation, et le sous-arbre correspondant au film emprunté. Note : utiliser les templates déjà écrits pour écrire les sous-arbres correspondants aux films.

Partie 3 : Ecrire la base complète

Question 3.1 : Ecrivez un fichier XSLT permettant de générer le fichier « baseProjet_par_abonne.xml ». Par rapport à la Question 2.4, vous devez rajouter un sous-arbre <film> de <videoexpress> pour chaque film qui n'est ni emprunté ni réservé.