Ecole d'ingénieurs et d'architectes de Fribourg
Hochschule für Technik und Architektur Freiburg
University of Applied Sciences of Fribourg

Travail pratique
XQuery - 2

Houda CHABBI DRISSI

1 Solutions

1.1 SQL vs XQuery

```
<VenteMono>
{
    for
          $i in document("invoices.xml")//invoice
    where
          $i//productID = "MONO"
    return
          <Commande>
                <Date>{string($i/@date)}</Date>
                <Nom>{$i/customer/name/text()}</Nom>
          </Commande>
}
</VenteMono>
```

Le tableau ci-dessous illustre la correspondance entre les différents composants de la requête XQuery et ceux de la requête SQL :

XQuery	SQL
`FOR $i` `IN document("invoices.xml")//invoice`	`FROM Customers, Invoices,` `InvoiceItems`

`WHERE $i//productID = "MONO"`	`WHERE Customers.customerID =`
	`Invoices.customerID AND`
	`Invoices.invoiceNumber =`
	`InvoiceItems.invoiceNumber AND`
	`InvoiceItems.productID = 'MONO'`

`RETURN`
`<Commande>`
`  <Date>{string($i/@date)}</Date>`
`  <Nom>{$i/customer/name/text()}</Nom>`
`</Commande>`
	`SELECT Invoices.date, Customers.name`

1.2 XSLT vs XQuery

```
<html>
{
    for
        $col in document("bd.xml")/collection
    return
        <body>
           {
             for
                  $bd in $col//bd
             return
                  (
                     <h3>{$bd/titre/text()} ({$bd/sortie/text()})</h3>,
                     <ul>
                         {
                           for
                                $p in document("personnes.xml")//personne
                           where
                                $p/@id = $bd/auteur/@ref
                           return
                                <li>Auteur:{$p/nom/text()}, {$p/prenom/text()}</li>
                         }
                         {
                           for
                                $p in document("personnes.xml")//personne
                           where
                                $p/@id = $bd/illustrateur/@ref
                           return
                             <li>Illustrateur:{$p/nom/text()}, {$p/prenom/text()}</li>
                         }
                     </ul>,
                     <b>Resume :</b>,
                     <br/>,
                     if (exists($bd/resume))
                     then $bd/resume
                     else "-"
                  )
           }
        </body>
}
</html>
```

Pour cette requête, la complexité se trouve surtout dans la clause « return ». C'est en effet là que le résultat est transformé en format HTML pouvant être affiché dans un navigateur.

- Le premier « for » sélectionne l'élément « collection » et se contente ensuite de générer un élément « body » qui représentera le corps de la page contenant les DB de cette collection.
- Le deuxième « for » s'occupe d'afficher les informations de chaque BD de la collection. Afin de pouvoir générer plusieurs éléments XML au même niveau dans la clause « return ».

Deux solutions sont envisageables. La première consiste à les encapsuler dans un seul et unique élément racine. Par exemple :

```
return
   <racine>
      <elt1>…</elt1>
      <elt2>…</elt2>
   </racine>
```

Cette solution ne correspondant par vraiment à nos besoins, c'est donc la deuxième qui à été implémentée. Celle-ci consiste à placer tous les éléments entre parenthèse et à les séparer par des virgules. Voici un exemple :

```
return
   (
      <elt1>…</elt1>,
      <elt2>…</elt2>
   )
```

Ensuite, la balise HTML « UL » permettant d'afficher une liste contient deux « for ». Le premier permet d'aller rechercher les nom et prénom de l'auteur et le deuxième permet d'aller chercher les nom et prénom de l'illustrateur.
Finalement, comme l'élément « resume » de chaque DB est facultatif, nous avons utilisé un « if » afin de test son existence. Si cet élément est présent, il est affiché sinon, un « - » est affiché.

1.3 XQuery et les namespaces

L'erreur provoquée par la requête provient du fait que l'interpréteur XQuery n'arrive pas à trouver à quoi correspond le prefix « o: »

```
Undefined: Cannot resolve namespace prefix o
```

Pour corriger ce problème, il faut utiliser la directive « declare namespace » dans le prolog de la requête. Le prolog est la partie de la requête se trouvant juste avant la clause « for ». Cette partie contient toutes les déclarations nécessaires à la bonne exécution de la requête (ex : déclaration de namespaces).
Voici donc la requête correcte :

```
declare namespace o = "http://www.eif.ch/order"
for
    $i in document("namespace.xml")//o:item
return
    $i/o:title
```