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

Travail pratique
XQuery

Houda CHABBI DRISSI

1 Solutions

Adaptations par rapport à l'utilisation de BaseX

- Si vous voulez travailler sur une collection (base de donnée) chargée dans votre BaseX, il suffit de l'ouvrir et de lancer la requête. Attention la requête sera interprétée par rapport à tous les fichiers de la collection. Donc si l'élément « bd » se retrouve dans plusieurs fichiers xml de la collection, ils seront tous traités.
  Ainsi pour la première solution sera la suivante. Le résultat est différent de l'attendu !puisque l'élément « bd » est également dans « distributeurs.xml »

```
for   $bd in //bd
return
      <titre>{$bd/titre/text()}</titre>
```

- Si vous voulez attaquez qu'un des fichiers de la collection il faut le spécifier.
  Ainsi pour la première solution sera la suivante avec une solution correcte.

```
for  $bd in doc("MasColTP/bd.xml")//bd
return
      <titre>{$bd/titre/text()}</titre>
```

- Si vous voulez travailler sur des fichiers externes à votre BaseX et l'utiliser pour lancer les requêtes Xquery, vous pouvez utiliser les formulations données dans les solutions si dessous mais en mettant le chemin complet par rapport à votre système de fichier pour trouver le fichier en question.

```
for $bd in doc("D:\monchemin\bd.xml")//bd
return
    <titre>{$bd/titre/text()}</titre>
```

1.1 Requêtes simples

1. Tous les titres des DB

```
for
    $bd in doc("bd.xml")//bd
return
    <titre>{$bd/titre/text()}</titre>
```

Cette requête permet de passer tous les éléments « bd » du document « bd.xml » et d'en retourner leur contenu (grâce à la fonction « text() ») dans un nouvel élément « titre ».
Les « { } » dans la clause « return » permettent de forcer l'évaluation de leur contenu. Sans ces crochets, l'affichage généré par le return aurait chaque fois eu la forme suivante : <titre>$bd/titre/text()</titre>

Variante XPath :

```
//bd/titre
```

2. Le contenu des attributs « numéro » des BDs dans ce fichier

```
doc("bd.xml")//bd/data(@numero)
```

On ne peut pas sortir directement les informations des attributs il faut utiliser les fonctions « string » ou « data ».

3. Tous les titres des BD triés par ordre alphabétique

```
for
    $bd in doc("bd.xml")//bd
order by $bd/titre
return
    <titre>{$bd/titre/text()}</titre>
```

Pour cette requête, nous avons simplement ajouté la clause « order by » permettant de trier les résultats d'après le titre des BD.

Version XPath : Pas possible car la clause « order by » n'existe pas avec XPath.

4. Tous les titres des DB triés par ordre alphabétique le tout dans un élément <TitreOrd>

```
<TitreOrd>{
for
    $bd in doc("bd.xml")//bd
order by $bd/titre
return

    <titre>{$bd/titre/text()}</titre>
    }
</TitreOrd>
```

5. Le résumé de la DB « Retour au college »

```
for
    $bd in doc("bd.xml")//bd
where
    $bd/titre="Retour au college"
return
    $bd/resume
```

Pour cette requête, nous limitons les résultats en filtrant les titres de DB.
Dans la clause « return », nous retournons directement le sous-élément « resume » de l'élément « bd » sans l'encapsuler dans un nouvel élément XML comme nous l'avions fait dans les requêtes précédentes. Ceci permet d'afficher l'élément en question avec ses balises ainsi que son contenu (ex : <resume>….</resume>).

Variante XPath :

```
//bd[titre="Retour au college"]/resume
```

6. Les titres de la série « Lanfeust de Troy »

```
for
    $bd in doc("bd.xml")//bd
where
    $bd/../titreserie="Lanfeust de Troy"
return
    <titre>{$bd/titre/text()}</titre>
```

Pour cette requête, nous avons utilisé la clause « where » afin de filtrer les résultats. Nous avons également utilisé les « .. » afin de remonter d'un niveau dans le document XML. Comme la clause « for » nous place au niveau des éléments « bd » et que nous devons filtrer les résultats d'après le titre de série, les « .. » nous permettent de passer de l'élément « bd » à son parent : « serie ».

Version XPath :

```
//serie[titreserie="Lanfeust de Troy"]/bd/titre
```

7. Les titres des DB sorties après 2000 avec leur date de sortie.

```
for
    $bd in doc("bd.xml")//bd
where
    $bd/sortie >= 2000
return
    <BD>
          <Titre>{$bd/titre/text()}</Titre>
          <Sortie>{$bd/sortie/text()}</Sortie>
    </BD>
```

Ici, nous utilisons l'opérateur de comparaison « >= » dans la clause « where » afin de trouver les DB dont la date de sortie est plus grande ou égale à 2000.
La clause « return » construit un élément « DB » comprenant deux fils contenant respectivement le titre de la DB ainsi que sa date de sortie.

Variante XPath : Pas possible car il n'est pas possible de construire un résultat personnalisé en XPath.

8. Combien de DB compte la série « Le retour a la terre » (fonction « count() »)

```
for
    $s in doc("bd.xml")//serie
where
    $s/titreserie="Le retour a la terre"
return
```

```
    count($s/bd)
```

Ici, nous utilisons la fonction « Count() » dans la clause « return », ce qui nous permet de compter le nombre d'élément « bd » retourné.

Variante XPath :

```
count(//serie[titreserie="Le retour a la terre"]/bd)
```

9. Les titres des DB n'ayant pas de résumé

```
for
    $bd in doc("bd.xml")//bd
where
    not($bd/resume)
return
    <Titre>{$bd/titre/text()}</Titre>
```

Dans le cas de cette requête, nous utilisons la fonction « not() » dans la clause « where » afin d'obtenir uniquement les BD ne possédant pas de sous-élément « resume ».

Variante XPath :

```
//bd[not(resume)]/titre
```

10. Le titre de la série dont au moins un des albums contient le mot « monde » dans sa description (fonction « contains() »).

```
for
    $res in doc("bd.xml")//resume
where contains($res,"monde")
return
    <Titre>{$res/../titre/text()}</Titre>
```

Pour cette requête, nous utilisons la fonction « contains() » dans la clause « where » afin de filtrer uniquement les résumés contenant le mot « monde ».
Afin de remonter de l'élément « resume » vers son parent « bd », nous utilisons à nouveau les « .. » dans le chemin XPath utilisé dans la clause « return ».

Variante XPath :

```
//bd[contains(resume,"monde")]/titre
```

11. Les titres des DB dont l'auteur et également l'illustrateur

```
for
    $bd in doc("bd.xml")//bd
where
    $bd/auteur/@ref = $bd/illustrateur/@ref
return
    <Titre>{$bd/titre/text()}</Titre>
```

Ici, nous testons simplement si le contenu de l'attribut « ref » de l'élément « auteur » est égal à celui de l'élément « illustrateur ».

Variante XPath :

```
//bd[auteur/@ref=illustrateur/@ref]/titre
```

12. Le titre de la DB avec « Jack Palmer » et « Ange Leoni »

Variante 1 :

```
for
    $bd in doc("bd.xml")//bd
where
    $bd/resume/perso="Jack Palmer" and $bd/resume/perso="Ange Leoni"
return
    <Titre>{$bd/titre/text()}</Titre>
```

Dans cette première solution, nous combinons simplement deux conditions dans la clause « where » à l'aide de l'opérateur logique « AND ».

Variante 2 :

```
for
    $bd1 in (
          for
                $bd in doc("bd.xml")//bd
          where
                $bd/resume/perso="Jack Palmer"
          return
                $bd
    ),
    $bd2 in (
          for
                $bd in $bd1
          where
                $bd/resume/perso="Ange Leoni"
          return
                $bd
    )
return
    <Titre>{$bd2/titre/text()}</Titre>
```

Pour cette deuxième solution, nous avons imbriqué deux « for » dans notre requête de base.
Le premier « for » imbriqué permet de trouver toutes les DB ayant un personnage nommé « Jack Palmer ».
Le deuxième « for » imbriqué reprend les résultats trouvés par le premier « for » imbriqué (« `for $bd in $bd1` ») et trouve parmi ceux-ci, toutes les DB ayant en plus un personnage nommé « Ange Léoni ».
La requête de base récupère les DB retournées par le deuxième « for » imbriqué (`$bd2`) et en affiche leur titre.

Variante XPath :

```
//bd[resume/perso="Jack Palmer" and resume/perso="Ange Leoni"]/titre
```

1.2 Requêtes complexes

Ecrivez les requêtes répondant aux questions suivantes :

1. Qui est l'auteur (nom et prénom) de la BD « Thanos l'incongru »

```
for
    $bd in doc("bd.xml")//bd,
    $a in doc("personnes.xml")//personne
where
    $bd/auteur/@ref = $a/@id and $bd/titre = "Thanos l'incongru"
return
    <Auteur>
          <Nom>{$a/nom/text()}</Nom>
          <Prenom>{$a/prenom/text()}</Prenom>
    </Auteur>
```

Pour cette requête, nous avons du effectuer une jointure entre le fichier « bd.xml » et le fichier « personnes.xml ».
La clause « for » récupère toutes les DB ainsi que toutes les personnes et la clause « where » s'occupe de faire la jointure entre l'attribut « ref » de l'élément « auteur » dans le fichier « bd.xml » et l'attribut « id » de l'élément « personne » dans le fichier « personnes.xml ».
Ensuite, grâce à l'opérateur « AND », nous avons pu ajouter une deuxième condition permettant de traiter uniquement les DB dont le titre est « Thanos l'incongru ».

2. Le titre des DB dont l'illustrateur se prénomme « Didier »

Variante 1 :

```
for
    $bd in doc("bd.xml")//bd,
    $a in doc("personnes.xml")//personne
where
    $bd/illustrateur/@ref = $a/@id and
    $a/prenom = "Didier"
return
    <Titre>{$bd/titre/text()}</Titre>
```

Pour cette requête, nous avons utilisé le même principe que pour la requête précédente. Cette fois-ci, la jointure est effectuée entre l'attibut « ref » de l'élément « illustrateur » et l'attribut « id » de l'élément « personne ».
La deuxième condition de la clause « where » permet de limiter les résultats à la personne prénommé « Didier » uniquement.

Variante 2 :

```
let $a := doc("personnes.xml")//personne[prenom = "Didier"]
for $bd in doc("bd.xml")//bd[illustrateur/@ref = $a/@id]
return
    <Titre>{$bd/titre/text()}</Titre>
```

Cette variante est plus directe. On récupère les éléments qui sont relatif à « Didier » et je recherche les éléments de bd qui ont comme illustrateur cette personne. On regarde donc @id.

3. La liste des personnages triés par ordre alphabétique avec la liste de DB dans lesquelles ils apparaissent.

```
<Personnages>
{
    for
          $persoval in distinct-values(doc("bd.xml")//bd/resume/perso)
    order by
          $persoval
    return
          <Perso>
                <Nom>{$persoval}</Nom>
                <BD>
                      {
                            for
                                  $bd in doc("bd.xml")//bd
                            where
                                  $bd/resume/perso = $persoval
                            return
                                  $bd/titre
                      }
                </BD>
          </Perso>
}
</Personnages>
```

Pour cette requête, nous avons utilisé deux « for » imbriqués.
Le premier « for » nous permet de trouver tous les personnages présents dans les résumés des DB. Nous utilisons la fonction « distinct-values() » afin de pouvoir éliminer les doublons qui pourraient apparaître lorsqu'un même personnage est cité dans plusieurs résumés. Attention : « distinct-values() » ne rend pas des nœuds mais des valeurs textuelles.
Nous utilisons ensuite également la clause « order by » afin de trier les personnages par ordre alphabétique.
Le deuxième « for » permet de trouver les titres des DB dans lesquelles chaque personnage apparaît. Le lien entre les deux « for » se fait au niveau de la clause « where » du deuxième « for » (« `where $bd/resume/perso = $perso` »)

4. Les personnes qui ne sont auteurs que des séries (pas de DB seules)

Variante 1 :

```
let $listeauteurSerie:= doc("bd.xml")/collection/serie/bd/auteur,
     $listeauteurBD := doc("bd.xml")/collection/bd/auteur,
     $listeauteurQueSerie := $listeauteurSerie  except $listeauteurBD
for $auteurQueSerieVal in distinct-values($listeauteurQueSerie/@ref)
let $personneQueSerie :=
           doc("personnes.xml")//personne[@id=$auteurQueSerieVal]
```

```
order by $auteurQueSerieVal
return
     $personneQueSerie
```

Pour cette requête, nous avons tout d'abord sélectionné, via la clause « let », tous les auteurs de série, tous les auteurs de BD et n'avons conservé que ceux de série via une différence d'ensemble. Attention la fonction « except() » ne travaille que sur des « node » et non des chaines de caractères. Puis, on élimine les doublons à l'aide de la fonction « distinct-values() » et récupérons dans le fichiers « personnes.xml » la personne qui correspond aux auteurs qui nous intéressent et dont nous avons la valeur de l'id. Nous avons ensuite récupéré toutes les personnes dans le fichier « personnes.xml ».

Variante 2 : utilisation le plus tôt que possible du distinct values

```
let $listeauteurSerieVal:=
            distinct-values(doc("bd.xml")/collection/serie/bd/auteur/@ref)
for $auteurSerieVal in $listeauteurSerieVal
where count(doc("bd.xml")/collection/bd/auteur[@ref = $auteurSerieVal ]) = 0
order by $auteurSerieVal
return
     doc("personnes.xml")//personne[@id=$auteurSerieVal]
```

Pour cette requête, nous avons tout d'abord sélectionné tous les auteurs de série en éliminant les doublons à l'aide de la fonction « distinct-values() ». Attention il faut prendre la chaine de l'attribut « @ref » sinon « distinct-values » renvoi la séquence vide.
La clause « where » nous permet de sélectionner uniquement les auteurs ne faisant pas partie du groupe des auteurs de DB. Pour cela, nous avons utilisé la fonction « count() » en testant si son résultat était égale à zéro. Puis on recherche l'information de l'auteur de pure série dans le fichier « personnes.xml ».

5. **La liste des auteurs avec la liste des DB qu'ils ont écrits triée dans l'ordre de leur sortie. Les personnes n'étant « que » des illustrateurs ne doivent pas être listées !**

```
<Auteurs>
{
    for   $p in doc("personnes.xml")//personne
     let    $a := doc("bd.xml")//bd[auteur/@ref = $p/@id]
    return
    if ($a) then
          <Auteur>
                <Nom>{$p/nom/text()}</Nom>
                <Prenom>{$p/prenom/text()}</Prenom>
                <BD>
                      {
                            for $bd in $a
                            order by $bd/sortie descending
                            return
                                  $bd/titre
                      }
```

```
                </BD>
          </Auteur>

    else ()
}
</Auteurs>
```

Pour cette requête, nous avons à nouveau utilisé deux « for » imbriqués.
Le premier « for » permet de récupérer toutes les personnes étant des auteurs. Pour cela, tous les éléments « personne » du le fichier « personne.xml » sont parcourus est placé dans la variable $p. La clause « let » permet de stocker dans une variable ($a) tous les éventuelles « BD » dont $p serait auteur « auteur ».
Le « if » permet de n'avoir une sortie que pour les personnes de type « auteurs » recherchés.
Le deuxième « for » permet de retourner tous les titres des BD de chaque auteur stocké dans la variable $a. La clause « order by » combinée avec le mot clé « descending » permet ensuite de trier les titres par order alphabétique inverse.

Question : Que se passe t-il si on enlève le « if » ?
Déjà on ne répond plus exactement à la requête !

6. Le nom de chaque auteur avec les titres de ses DB ainsi que les noms des distributeurs chez qui on peut les trouver.

```
for $a in distinct-values(doc("bd.xml")//auteur/@ref)
let $p := doc("personnes.xml")//personne[$a = @id]
return
<Auteur>
    <Nom>{$p/nom/text()}</Nom>
    {
          for   $bd in doc("bd.xml")//bd
          where $bd/auteur/@ref = $p/@id
          return
                <BD>
                      <Titre>{$bd/titre/text()}</Titre>
                      <Distributeurs>
                      {
                            for $dist in
                                    doc("distributeurs.xml")//distributeur
                            where $dist/catalogue/bd/@ref = $bd/@id
                            return
                                  <Nom>{$dist/nom/text()},
                                        {$dist/ville/text()}</Nom>
                      }
                      </Distributeurs>
                </BD>
          }
    </Auteur>
```

On commence par rechercher les « identifiants » des auteurs des « BD ». Puis on recherche pour chacun les informations « personne ». Une fois que l'on a un auteur on recherche les BD de cet auteur puis pour chaque titre on retrouve les distributeurs.

7. Le nom et le prénom de l'auteur ainsi que le nom du distributeur et le titre de la plus chère des BD.

```
let $PrixMax := max(doc("distributeurs.xml")/distributeurs//@prix)
for $refbdPrixMax in
doc("distributeurs.xml")/distributeurs/distributeur/catalogue/bd[@prix =
$PrixMax]
let  $bdPrixMax := doc("bd.xml")//bd[@id =  $refbdPrixMax/@ref],
      $AuteurPrixMax := doc("personnes.xml")//personne[@id =
$bdPrixMax/auteur/@ref]
return
<BdLaPlusChere>
    <Distributeur>{$refbdPrixMax/../../nom/text()},
{$refbdPrixMax/../../ville/text()}</Distributeur>
    <Titre>{$bdPrixMax/titre/text()}</Titre>
    <Auteur>{$AuteurPrixMax/nom/text()},
{$AuteurPrixMax/prenom/text()}</Auteur>
</BdLaPlusChere>
```

On commence par chercher le « prix » maximum. Pour chaque « bd » dans le « catalogue » d'un « distributeur » ayant ce prix max on retrouve la « bd » correspondante et ainsi l' « auteur » correspondant et on fini par afficher les informations demandées.

8. Le titre des BD qui sont vendues partout

```
for
    $bd in doc("bd.xml")//bd
let
    $nbVendeur :=
count(doc("distributeurs.xml")//distributeur//bd[@ref=$bd/@id]),
    $nbDist := count(doc("distributeurs.xml")//distributeur)
where
    $nbVendeur = $nbDist
return
     <Titre>{$bd/titre/text()}</Titre>
```

Pour cette requête, nous avons également utilisé un seul « for ». Ce dernier permet de parcourir toute les BD. La clause « let » nous permet de stocker temporairement le nombre de distributeurs proposant la BD courante (`$nbVendeur`) ainsi que le nombre total de distributeurs (`$nbDist`). La clause « where » permet ensuite de tester si le nombre de vendeurs et égale au nombre total de distributeur, auquel cas le titre de la BD courante est affiché.