Devoir

Zoubir hiba
Groupe : 04

Exercice 1 – Requêtes XQuery

1. Lister tous les albums en ordre alphabétique

```
for $a in doc("albums.xml")//album
order by $a/titre
return $a/titre
```

Je récupère tous les albums, je les trie par titre alphabétique, puis j'affiche chaque titre.

2. Albums publiés après 1970

```
for $a in doc("albums.xml")//album
where xs:integer($a/date/annee) > 1970
return $a/titre
```

Je parcours tous les albums, je garde ceux publiés après 1970, puis j'affiche leur titre.

3. Auteurs ayant participé à plus d'un album

```
for $auteur in distinct-values(doc("albums.xml")//auteur)
where count(doc("albums.xml")//album[auteur = $auteur]) > 1
return $auteur
```

Je récupère les auteurs distincts, je compte combien de fois chaque auteur apparaît, et je garde ceux qui ont plus d'un album.

4. Album le plus récent de chaque série

```
for $serie in distinct-values(doc("albums.xml")//album/@serie)
let $albums := doc("albums.xml")//album[@serie = $serie]
let $max := max(for $a in $albums return xs:integer($a/date/annee)
   )
return $albums[date/annee = $max]/titre
```

Je récupère chaque série, je cherche les albums de cette série, je trouve l'année maximale, puis je retourne le titre de l'album correspondant.

5. Regrouper les albums par série et compter

```
for $serie in distinct-values(doc("albums.xml")//album/@serie)
let $count := count(doc("albums.xml")//album[@serie = $serie])
return <serie nom="{$serie}" nb="{$count}"/>
```

Je prends chaque série unique, je compte combien d'albums elle a, puis je crée une balise avec le nom et le nombre.

6. Trouver la série avec le plus d'albums

```
let $groupes :=
  for $serie in distinct-values(doc("albums.xml")//album/@serie)
  let $nb := count(doc("albums.xml")//album[@serie = $serie])
  return <serie nom="{$serie}" nb="{$nb}"/>
return $groupes[@nb = max($groupes/@nb)]
```

Je construis une liste des séries avec le nombre d'albums, puis je garde celle(s) avec le maximum.

7. Années où le plus d'albums ont été publiés

```
let $annees := distinct-values(doc("albums.xml")//annee)
let $groupes :=
  for $a in $annees
  let $nb := count(doc("albums.xml")//annee[. = $a])
  return <annee val="{$a}" nb="{$nb}"/>
return $groupes[@nb = max($groupes/@nb)]
```

Je récupère les années uniques, je compte combien d'albums pour chaque, puis je garde celle(s) avec le plus grand nombre.

8. Albums avec plus de 10 ans d'écart

```
for $serie in distinct-values(doc("albums.xml")//album/@serie)
let $albums :=
  for $a in doc("albums.xml")//album[@serie = $serie]
  order by xs:integer($a/date/annee)
  return $a
for $i in 2 to count($albums)
let $prev := xs:integer($albums[$i - 1]/date/annee)
let $curr := xs:integer($albums[$i]/date/annee)
where $curr - $prev > 10
return $albums[$i]/titre
```

Pour chaque série, je trie les albums par année, puis je cherche un écart de plus de 10 ans entre deux albums successifs.

9. Auteurs ayant participé à plusieurs séries

```
for $auteur in distinct-values(doc("albums.xml")//auteur)
let $series := distinct-values(doc("albums.xml")//album[auteur =
    $auteur]/@serie)
where count($series) > 1
return $auteur
```

Je prends chaque auteur, je récupère les séries où il a participé, et je garde ceux qui ont travaillé sur plusieurs séries.

10. Auteur ayant écrit le plus d'albums

```
let $groupes :=
  for $a in distinct-values(doc("albums.xml")//auteur)
  let $nb := count(doc("albums.xml")//album[auteur = $a])
  return <auteur nom="{$a}" nb="{$nb}"/>
return $groupes[@nb = max($groupes/@nb)]
```

Je compte le nombre d'albums par auteur, puis je garde celui qui en a écrit le plus.

11. Albums même titre, série différente

```
for $t in distinct-values(doc("albums.xml")//titre)
let $s := distinct-values(doc("albums.xml")//album[titre = $t]/
    @serie)
where count($s) > 1
return $t
```

Je récupère tous les titres, puis je regarde s'ils sont utilisés dans plusieurs séries différentes.

12. Fonction pour albums les plus anciens d'un auteur

```
declare function local:plusAnciens($nom as xs:string) {
  let $albums := doc("albums.xml")//album[auteur = $nom]
  let $min := min(for $a in $albums return xs:integer($a/date/
      annee))
  return $albums[date/annee = $min]/titre
};
local:plusAnciens("Herg ")
```

Je crée une fonction qui prend un nom d'auteur, puis retourne le ou les titres les plus anciens qu'il a publiés.

13. Ajouter "Uderzo" à album 1 de Tintin

```
copy $doc := doc("albums.xml")
modify (
  insert node <auteur>Uderzo</auteur> into $doc//album[@serie="
      Tintin" and @numero="1"]
```

```
)
return $doc
```

14. Ajouter attribut "editeur"

```
copy $doc := doc("albums.xml")
modify (
  insert node attribute editeur {"La plume"} into $doc//album[
     @serie="Ast rix"][3]
)
return $doc
```

15. Ajouter "Hergé" si pas d'auteur

```
copy $doc := doc("albums.xml")
modify (
  for $a in $doc//album[@serie="Tintin"][not(auteur)]
  return insert node <auteur>Herg </auteur> into $a
)
return $doc
```

Pour chaque album de Tintin sans auteur, j'ajoute un auteur "Hergé".

16. Modifier nom de série

```
copy $doc := doc("albums.xml")
modify (
  for $a in $doc//album[@serie="Ast rix"]
  return replace value of node $a/@serie with "Ast rix et Ob lix
     "
)
return $doc
```

Je change le nom de la série "Astérix" en "Astérix et Obélix" pour tous les albums.

17. Supprimer albums Tintin avant 1950

```
copy $doc := doc("albums.xml")
modify (
  for $a in $doc//album[@serie="Tintin"]
  where xs:integer($a/date/annee) < 1950
  return delete node $a
)
return $doc
```

Je supprime les albums de Tintin publiés avant 1950.

18. Augmenter l'année après 1980 (Astérix)

```
copy $doc := doc("albums.xml")
modify (
  for $a in $doc//album[@serie="Ast rix"]
  where xs:integer($a/date/annee) > 1980
  return replace value of node $a/date/annee with xs:integer($a/
     date/annee) + 1
)
return $doc
```

J'augmente de 1 l'année de tous les albums Astérix publiés après 1980.

19. Renommer balise album n°1

```
copy $doc := doc("albums.xml")
modify (
  for $a in $doc//album[@numero="1"]
  return rename node $a as "Premier_album"
)
return $doc
```

Je renomme la balise <album> ayant le numéro 1 en <Premier$_a$*lbum* > .

20. Ajouter un nouvel album

```
copy $doc := doc("album.xml")
modify (
  insert node
    <album numero="25" serie="Tintin">
      <titre>Le Nouveau Myst re </titre>
      <auteur>Herg  </auteur>
      <date>
        <mois>mars</mois>
        <annee>2025</annee>
      </date>
    </album>
  as last into $doc/albums/album[@serie="Tintin"]
)
return $doc
```

Exercice 2 – Requêtes XQuery sur Films.xml et Artistes.xml

Je déclare les deux documents :

```
let $F := doc("Films.xml")/FILMS
let $A := doc("Artistes.xml")/ARTISTES
```

1. Le titre, genre et pays pour tous les films avant 1970

Je retourne le titre, le genre et le pays de tous les films dont l'année est inférieure à 1970.

```
for $f in $F/FILM[xs:integer(ANNEE) < 1970]
return <film titre="{ $f/@titre }" genre="{ $f/GENRE }" pays="{ $f
    /PAYS }"/>
```

2. Les rôles joués par Bruce Willis

Je récupère d'abord l'ID de Bruce Willis dans Artistes.xml, puis je liste tous les rôles dont l'attribut acteur correspond à cet ID.

```
let $idBW := $A/ARTISTE[NOM = "Willis" and PRENOM = "Bruce"]/@ID
for $r in $F//ROLE[@acteur = $idBW]
return data($r)
```

3. Les rôles de Bruce Willis sous forme d'élément `<role>`

Je crée pour chaque rôle un élément `<role>` contenant le titre du film et le nom du personnage.

```
let $idBW := $A/ARTISTE[NOM = "Willis" and PRENOM = "Bruce"]/@ID
for $r in $F//ROLE[@acteur = $idBW]
let $titre := $r/ancestor::FILM/@titre
return
  <role>
    <film>{ $titre }</film>
    <personnage>{ data($r) }</personnage>
  </role>
```

4. Le nom du metteur en scène du film *Vertigo*

Je sélectionne le film :

```
let $v := $F/FILM[@titre = "Vertigo"]
```

Puis je récupère le metteur en scène en joignant avec Artistes.xml et j'affiche son nom complet.

```
let $m := $A/ARTISTE[NOM = $v/MES]
return concat($m/PRENOM, " ", $m/NOM)
```

ou bien directement

```
let $v := $F/FILM[@titre = "Vertigo"]
return $v/MES
```

5. Pour chaque artiste, son nom et les titres des films qu'il a dirigés

Je parcours chaque artiste et je collecte les titres des films où il est metteur en scène.

```
for $art in $A/ARTISTE
let $filmsDir :=
  for $f in $F/FILM[MES = $art/NOM]
  return $f/@titre
return
  <artiste nom="{ concat($art/PRENOM, ' ', $art/NOM) }">
    {
      for $t in $filmsDir
      return <film>{$t}</film>
    }
  </artiste>
```

6. Pour chaque film, l'âge de son metteur en scène lors de la sortie

Je calcule l'âge du metteur en scène en soustrayant son année de naissance de l'année du film.

```
for $f in $F/FILM
let $m := $A/ARTISTE[NOM = $f/MES]
let $naiss := xs:integer($m/ANNEENAISS)
let $anneeF := xs:integer($f/ANNEE)
let $age := $anneeF - $naiss
return
  <film titre="{ $f/@titre }" ageMetteurEnScene="{ $age }"/>
```

7. Pour chaque genre, un élément <genre> contenant les titres

Je liste chaque genre unique et j'affiche les titres des films correspondants.

```
for $g in distinct-values($F/FILM/GENRE)
let $filmsG := for $f in $F/FILM[GENRE = $g] return $f/@titre
return
  <genre nom="{ $g }">
    { for $t in $filmsG return <film titre="{ $t }"/> }
  </genre>
```

8. Les artistes ayant joué dans un film qu'ils ont mis en scène

Je vérifie pour chaque artiste s'il est à la fois metteur en scène et acteur dans un même film.

```
for $art in $A/ARTISTE
let $filmsCommuns :=
  for $f in $F/FILM[MES = $art/NOM]
  where some $r in $f/ROLES/ROLE satisfies ($r/@acteur = $art/@ID)
  return $f
where exists($filmsCommuns)
return
  <artiste fullname="{ concat($art/PRENOM, ' ', $art/NOM) }">
    { for $f in $filmsCommuns return <film titre="{ $f/@titre }"
       annee="{ $f/ANNEE }"/> }
  </artiste>
```

Résumé sur template, apply-template, call-template et fonctions

- **template** : Un `template` est une règle de transformation utilisée en XSLT pour traiter des nœuds XML spécifiques.
- **apply-templates** : Utilisé pour appliquer automatiquement les templates définis sur les nœuds enfants.
- **call-template** : Permet d'appeler un template nommé explicitement (comme une fonction).
- **fonctions** : En XQuery, on peut définir des fonctions avec `declare function` pour réutiliser du code. Elles peuvent prendre des paramètres et retourner des résultats complexes.