

# Le langage de requêtes XQuery

L3Pro BDISE – XML

Mathieu Sassolas

IUT de Sénart Fontainebleau  
Département Informatique

Année 2016-2017  
Cours 5



XQuery

M. Sassolas

L3Pro

Cours 5

Motivation

Syntaxe

Exemples

Utilisation

Mise en  
application

- 1 Pourquoi un langage de requêtes ?
- 2 Syntaxe de XQuery
- 3 Quelques exemples
- 4 Question philosophique : quand utiliser XQuery ?
- 5 Mise en application

- ▶ On peut voir un document XML comme une **base de donnée** :

...

```
<person id="jcd00001">  
  <prenom>Jean-Claude</prenom>  
  <nom>Dusse</nom>  
  <age>55</age>  
  <hobby>Conclure</hobby>  
</person>
```

...

- ▶ On veut pouvoir **extraire** des informations de cette base :  
↪ équivalent de

```
SELECT * FROM personne WHERE hobby="Conclure";
```

- ▶ On peut le faire par **XSLT** :

**Exercice** : écrire la template XSLT idoine.

(Mais ce n'est pas très beau.)

- ▶ Ça ne devient utile que si on fait vraiment des traitements dessus.

- ▶ On peut le faire par **XSLT** :

```
<xsl:template match="personne">
  <xsl:if test="hobby/text() = 'Conclure'">
    <xsl:copy-of select="."/>
  </xsl:if>
</xsl:template>
```

(Mais ce n'est pas très beau.)

- ▶ Ça ne devient utile que si on fait vraiment des traitements dessus.

- ▶ On peut le faire par **XSLT** :

```
<xsl:template match="personne">  
  <xsl:if test="hobby/text() = 'Conclure'">  
    <xsl:copy-of select="."/>  
  </xsl:if>  
</xsl:template>
```

(Mais ce n'est pas très beau.)

- ▶ Ça ne devient utile que si on fait vraiment des traitements dessus.

On veut de vraies requêtes comme en SQL.

XQuery

M. Sassolas

L3Pro

Cours 5

Motivation

Syntaxe

Exemples

Utilisation

Mise en  
application

for	\$variable in expression
let	\$variable := expression
where	condition
order by	expression
return	expression

## Remarques

- ▶ Seul le `return` est obligatoire.
- ▶ On peut imbriquer des expressions dans d'autres.

- ▶ La variable `$variable` prendra tour à tour la valeur des éléments sélectionnés par l'`expression`.
- ▶ L'`expression` est en général une requête XPATH (simple).

### Attention !

L'arbre XML dans lequel on se place (notre « base de données ») n'est pas défini a priori (comme c'est le cas en XSLT puisque c'est le XML qui appelle la feuille XSLT) .

### Récupération de la racine d'un arbre XML

```
doc("chemin/vers/le/fichier.xml")
```

### Exemple

```
For $p in doc("../films/lesBronzes.xml")/personne
```

- ▶ Itérations numériques : `for $i in (1 to 42)`.
- ▶ Comptage de l'itération (sur un ensemble de nœuds) :  
`for $noeud at $i in doc("arbre.xml")/truc`.
- ▶ Produits cartésiens :  
`for $x in (1 to 5), $y in (4 to 21)`.

### Remarque

*Les nœuds sélectionnés pourraient l'être finement avec une requête XPath adaptée. La philosophie de XQuery est de plutôt laisser cela aux conditions du *where*.*

# let \$variable := expression

XQuery

M. Sassolas

L3Pro

Cours 5

Motivation

Syntaxe

Exemples

Utilisation

Mise en  
application

- ▶ Assignment de variable.
- ▶ Exemple : `let $persId := $p/@id`
- ▶ La variable peut aussi avoir comme valeur un (bout d')arbre XML.

- ▶ C'est ici que se fait la majorité du « filtrage ».
- ▶ Les conditions sont les mêmes que les tests XPath.
- ▶ Exemple : `where number($p/age) > 42`
- ▶ Remarque : la requête n'est pas du XML  $\rightsquigarrow$  on utilise les chevrons et non les entités.

## Condition quantifiée

- ▶ `where every $x in $ensemble satisfies cond_x`
- ▶ `where some $x in $ensemble satisfies cond_x`

XQuery

M. Sassolas

L3Pro

Cours 5

Motivation

Syntaxe

Exemples

Utilisation

Mise en  
application

- ▶ Définit le critère pour **ordonner** le parcours d'une boucle `for`.
- ▶ On peut inverser l'ordre avec **descending**; le cas **ascending** est implicite.
- ▶ On peut donner plusieurs critères en les mettant à la suite.
- ▶ Exemple : `order by number($p/age) descending, $p/nom`.

- ▶ C'est là qu'est réelement produit quelque-chose.
- ▶ Ce qui est produit peut être a priori n'importe quel code XML.
- ▶ Pour évaluer des expressions, on les met entre accolades.

## Exemple

```
return <personnage>{concat($p/prenom, " ", $p/nom, "  
(", $p/age, " ans)")}</personnage>
```

On manipule toujours des bouts d'arbre XML.

XQuery

M. Sassolas

L3Pro

Cours 5

Motivation

Syntaxe

Exemples

Utilisation

Mise en  
application

- ▶ On peut insérer la requête dans du XML (ou du XHTML...).

## Exemple

```
<nouvel_arbre>{  
  for $x in doc("vieil_arbre.xml")/noeud  
  return <truc>{$x/contenu}</truc>  
}</nouvel_arbre>
```

- ▶ Commentaires XQuery : (: Du commentaire :).

↪ On peut utiliser toutes les **fonctions XPath**.

- ▶ Fonctions sur les chaînes de caractères : **concat**, **contains**...
- ▶ Fonctions sur les nombres : **div**, **+**...
- ▶ **number(...)** traite les données comme des nombre. Très utile dans les **order by** ou dans des comparaisons dans un **where**.
- ▶ **data(...)** récupère les données textuelles (« #PCDATA ») du nœud (à comparer avec **text()** de XPath, **sauf qu'il descend dans les fils**). Exemples : `data($p/nom)` ; `data($var/@attribut)`.
- ▶ **distinct-values(...)** retire les doublons ; très utile dans les boucles **for**.
- ▶ **exists(...)** teste l'existence de nœuds ; très utile dans les tests **where**.

XQuery

M. Sassolas

L3Pro

Cours 5

Motivation

Syntaxe

Exemples

Utilisation

Mise en  
application

```
<cadeaux>
{
for $boite in
    doc("RoisMages.xml")/rois_mages/chameau/boite
order by number($boite/objet/@poids)
return <objet dest="{ $boite/destinataire}">
    { $boite/@id } { data($boite/objet) } </objet>
}
</cadeaux>
```

<?xml version="1.0" encoding="UTF-8"?>

<cadeaux>

```
<objet dest="Anthony" id="r492">Une brosse à dents</objet>
<objet dest="Benjamin" id="l636">Un camembert Président</objet>
<objet dest="Benjamin" id="j325">Un stylo</objet>
<objet dest="Billal" id="p562">Un cahier</objet>
<objet dest="Samuel" id="r362">Un bel aérateur pour bouffer les odeurs</objet>
<objet dest="Billal" id="w913">Des boîtes en plastique</objet>
<objet dest="Anthony" id="g721">Des draps qui chauffent</objet>
<objet dest="Samuel" id="j629">Un cire-godasse</objet>
<objet dest="Mohamed" id="g720">Un barbecue</objet>
<objet dest="Anthony" id="k430">Un allume barbecue</objet>
<objet dest="Mohamed" id="y626">Des tas de couverts</objet>
<objet dest="Benjamin" id="a256">Des pelles à gâteaux</objet>
<objet dest="Didier" id="a316">Un pistolet à gaufres</objet>
<objet dest="Billal" id="r312">Un coupe-friture</objet>
<objet dest="Samuel" id="f920">Un téléphone</objet>
<objet dest="Anthony" id="z245">Un écran plat</objet>
<objet dest="Luc" id="e162">Un atomixer</objet>
<objet dest="Luc" id="k257">Un tabouret à glace</objet>
<objet dest="Mohamed" id="k731">Une tente Quechua</objet>
<objet dest="Selma" id="m692">Une tourniquette pour faire la vinaigrette</objet>
<objet dest="Mohamed" id="m426">Un évier en fer</objet>
<objet dest="Mathieu" id="z362">Du jus d'ananas</objet>
<objet dest="Samuel" id="z481">Un repasse-limaces</objet>
<objet dest="Mohamed" id="e361">Du lait UHT</objet>
<objet dest="Selma" id="l352">Plein d'objets sans fil</objet>
<objet dest="Benjamin" id="p962">Un chasse filou</objet>
<objet dest=" Régine" id="h252">Une armoire à cuillère</objet>
<objet dest="Anthony" id="d073">Du Dunlopillo</objet>
<objet dest="Mathieu" id="j212">Un ratatine-ordures</objet>
<objet dest="Mohamed" id="n362">Un rottweiler</objet>
<objet dest=" Régine" id="g362">Un frigidaire</objet>
<objet dest="Didier" id="q729">Un poêle à mazout</objet>
<objet dest="Billal" id="e622">Une cuisinière</objet>
<objet dest="Benjamin" id="w425">Un four en verre</objet>
<objet dest="Benjamin" id="x526">Un joli scooter</objet>
<objet dest="Anthony" id="t092">Un avion pour deux</objet>
```

</cadeaux>

boite

objet>

<cadeaux>

<objet dest="Anthony" id="r492">Une brosse à dents</objet>  
<objet dest="Benjamin" id="l636">Un camembert Président</objet>  
<objet dest="Benjamin" id="j325">Un stylo</objet>  
<objet dest="Billal" id="p562">Un cahier</objet>  
<objet dest="Samuel" id="r362">Un bel aérateur pour bouffer les odeurs</objet>  
<objet dest="Billal" id="w913">Des boîtes en plastique</objet>  
<objet dest="Anthony" id="g721">Des draps qui chauffent</objet>  
<objet dest="Samuel" id="j629">Un cire-godasse</objet>  
<objet dest="Mohamed" id="g720">Un barbecue</objet>  
<objet dest="Anthony" id="k430">Un allume barbecue</objet>  
<objet dest="Mohamed" id="y626">Des tas de couverts</objet>  
<objet dest="Benjamin" id="a256">Des pelles à gâteaux</objet>  
<objet dest="Didier" id="a316">Un pistolet à gaufres</objet>  
<objet dest="Billal" id="r312">Un coupe-friture</objet>  
<objet dest="Samuel" id="f920">Un téléphone</objet>  
<objet dest="Anthony" id="z245">Un écran plat</objet>

<objet dest="Mathieu" id="z362">Du jus d'ananas</objet>

<objet dest="Mathieu" id="z362">Du jus d'ananas</objet>  
<objet dest="Selma" id="m692">Une tourniquette pour faire la vinaigrette</objet>  
<objet dest="Mohamed" id="m426">Un évier en fer</objet>  
<objet dest="Mathieu" id="z362">Du jus d'ananas</objet>  
<objet dest="Samuel" id="z481">Un repasse-limaces</objet>  
<objet dest="Mohamed" id="e361">Du lait UHT</objet>  
<objet dest="Selma" id="l352">Plein d'objets sans fil</objet>  
<objet dest="Benjamin" id="p962">Un chasse filou</objet>  
<objet dest=" Régine" id="h252">Une armoire à cuillère</objet>  
<objet dest="Anthony" id="d073">Du Dunlopillo</objet>  
<objet dest="Mathieu" id="j212">Un ratatine-ordures</objet>  
<objet dest="Mohamed" id="n362">Un rottweiler</objet>  
<objet dest=" Régine" id="g362">Un frigidaire</objet>  
<objet dest="Didier" id="q729">Un poêle à mazout</objet>  
<objet dest="Billal" id="e622">Une cuisinière</objet>  
<objet dest="Benjamin" id="w425">Un four en verre</objet>  
<objet dest="Benjamin" id="x526">Un joli scooter</objet>  
<objet dest="Anthony" id="t092">Un avion pour deux</objet>

</cadeaux>

XQuery

M. Sassolas

L3Pro

Cours 5

Motivation

Syntaxe

Exemples

Utilisation

Mise en  
application

```
<liste_references>
{
let $boitsage :=
  for $pers in
    doc("RoisMages.xml")/rois_mages/enfants/sage,
    $cad in doc("RoisMages.xml")/rois_mages/chameau/boite
  where $cad/destinataire = $pers
  return $cad
for $cado in $boitsage
order by $cado/destinataire
return <reference>{data($cado/@id)}</reference>
}
</liste_references>
```

```

<liste_ref
{
let $boitsa
  for $per
    doc("R
      $cad i
        where $c
          return $
for $cado
order by $
return <re
}
</liste_r
  
```

```

<?xml version="1.0" encoding="UTF-8"?>
<liste_references>
  <reference>n362</reference>
  <reference>m426</reference>
  <reference>z245</reference>
  <reference>d073</reference>
  <reference>g720</reference>
  <reference>r362</reference>
  <reference>e361</reference>
  <reference>t092</reference>
  <reference>l636</reference>
  <reference>p962</reference>
  <reference>e622</reference>
  <reference>r312</reference>
  <reference>w913</reference>
  <reference>g721</reference>
  <reference>k731</reference>
  <reference>y626</reference>
  <reference>f920</reference>
  <reference>z481</reference>
  <reference>j325</reference>
  <reference>a256</reference>
  <reference>p562</reference>
  <reference>x526</reference>
  <reference>r492</reference>
  <reference>j629</reference>
</liste_references>
  
```

s/sage,

s/chameau/boite

ence>

```

<taux_reussite>{
let $profs := for $c in doc("courses-noID.xml")//Course/Instructors/*
              order by $c/Last_Name, $c/First_Name
              return $c
for $sens in distinct-values($profs)
let $cours :=
for $catalog in doc("courses-noID.xml")//Course
where $catalog/Instructors/* = $sens
return data($catalog/@Number)
let $taux :=
for $catalog in doc("courses-noID.xml")//Course,
    $stat in doc ("courses-noID-stats.xml")//Course
where $catalog/@Number = $stat/@Number (: jointure :)
and $catalog/Instructors/* = $sens
return <cours nom="{data($catalog/@Number)}">{
    number(data($stat/Enrolled))
    div number(data($stat/Passed))
}
    </cours> (: du XML pour pouvoir le relire ensuite :)
return <enseignant><identite>{data($sens)}</identite>
    {for $tx in $taux
     return <taux Course="{data($tx/@nom)}">{data($tx)}</taux>}
    </enseignant>
}</taux_reussite>

```

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<Course_Statistics>
```

```
  <Course Number="CS106A">
    <Enrolled>147</Enrolled>
    <Passed>68</Passed>
    <Credits>3</Credits>
  </Course>
```

```
<taux_reu
let $prof
```

```
for $en
```

```
let $cc
```

```
for
```

```
when
```

```
retu
```

```
let $sta
```

```
for
```

```
when
```

```
and
```

```
retu
```

```
return
```

```
{for
```

```
retu
```

```
}</taux_r
```

```
  <Course Number="CS106B">
    <Enrolled>347</Enrolled>
    <Passed>274</Passed>
    <Credits>6</Credits>
  </Course>
```

```
  <Course Number="CS107">
    <Enrolled>47</Enrolled>
    <Passed>32</Passed>
    <Credits>6</Credits>
  </Course>
```

```
  <Course Number="CS109">
    <Enrolled>124</Enrolled>
    <Passed>107</Passed>
    <Credits>3</Credits>
  </Course>
```

```
...
```

```
Course/Instructors/*
```

```
e
```

```
course
```

```
course,
```

```
//Course
```

```
ointure :)
```

```
">{
```

```
ssed))}
```

```
elire ensuite :)
```

```
tite>
```

```
a($tx)}</taux>}
```

```

<taux_reussite>{
let $profs := for $c in doc("courses-noID.xml")//Course/Instructors/*
              order by $c/Last_Name, $c/First_Name
              return $c
for $ens in distinct-values($profs)
let $cours :=
for $catalog in doc("courses-noID.xml")//Course
where $catalog/Instructors/* = $ens
return data($catalog/@Number)
let $taux :=
for $catalog in doc("courses-noID.xml")//Course,
    $stat in doc ("courses-noID-stats.xml")//Course
where $catalog/@Number = $stat/@Number (: jointure :)
and $catalog/Instructors/* = $ens
return <cours nom="{data($catalog/@Number)}">{
    number(data($stat/Enrolled))
    div number(data($stat/Passed))
}
    </cours> (: du XML pour pouvoir le relire ensuite :)
return <enseignant><identite>{data($ens)}</identite>
    {for $tx in $taux
     return <taux Course="{data($tx/@nom)}">{data($tx)}</taux>}
    </enseignant>
}</taux_reussite>

```

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<taux_reussite>
```

```
<enseignant>
```

```
<identite>
```

```
Alex
```

```
S.
```

```
Aiken
```

```
</identite>
```

```
<taux Course="CS143">1.1873198847262247</taux>
```

```
</enseignant>
```

```
<enseignant>
```

```
<identite>
```

```
Jerry
```

```
R.
```

```
Cain
```

```
</identite>
```

```
<taux Course="CS106A">2.161764705882353</taux>
```

```
<taux Course="CS106B">1.2664233576642336</taux>
```

```
</enseignant>
```

```
<enseignant>
```

```
<identite>
```

```
William
```

```
J.
```

```
Dally
```

```
</identite>
```

```
<taux Course="EE108B">1.437956204379562</taux>
```

```
</enseignant>
```

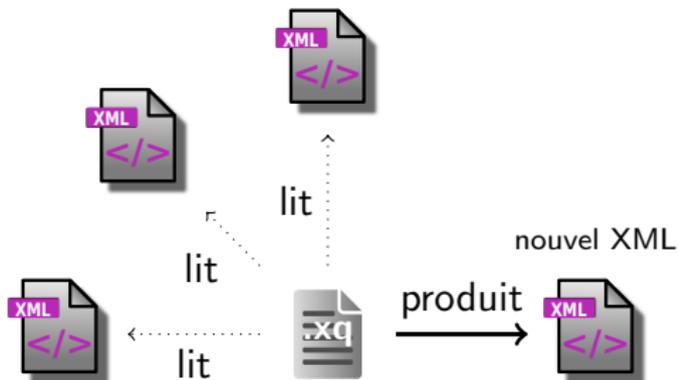
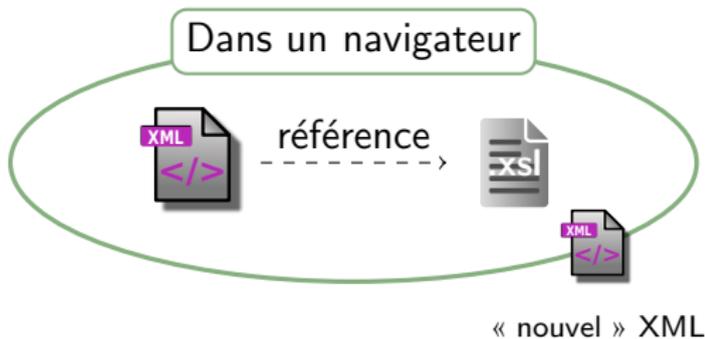
```
...
```

>>

nstructors/\*

e :)

nsuite :)



XQuery

M. Sassolas

L3Pro

Cours 5

Motivation

Syntaxe

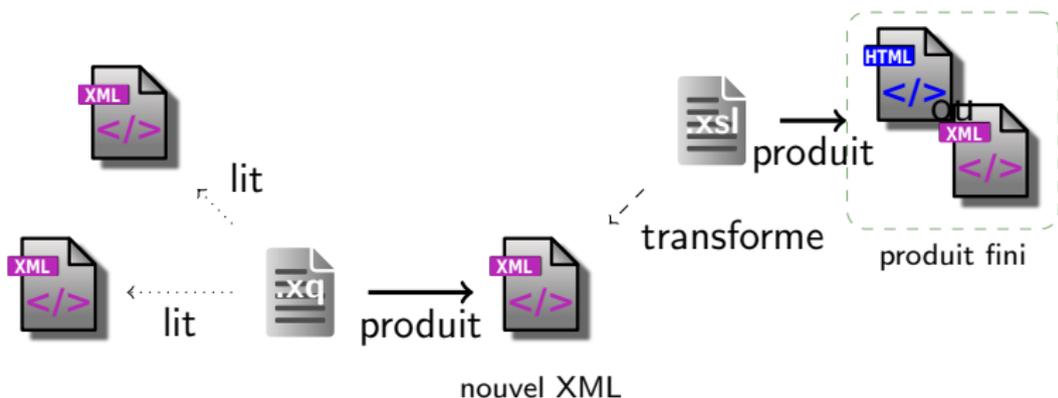
Exemples

Utilisation

Mise en  
application

- ▶ XQuery est fait pour extraire les données plus que pour les mettre en forme.
- ▶ L'exemple de requête sur `courses-noID[-stats].xml` est en fait très mauvais : on fait de la mise en forme et des calculs qui seraient plus propres en XSLT.

- ▶ XQuery est fait pour extraire les données plus que pour les mettre en forme.
- ▶ L'exemple de requête sur `courses-noID[-stats].xml` est en fait très mauvais : on fait de la mise en forme et des calculs qui seraient plus propres en XSLT.
- ▶ L'idéal est d'utiliser les deux en séquence : XQuery fait la jointure puis XSLT met en forme :



XQuery

M. Sassolas

L3Pro

Cours 5

Motivation

Syntaxe

Exemples

Utilisation

Mise en  
application

```
<Course_Catalog_With_Stats>{  
  for $d in doc("courses-noID.xml")//Department  
  return  
    <Department>{$d/@Code}  
      {$d/Title}  
      {$d/Chair}  
      {  
        for $c in $d/Course,  
          $stat in doc ("courses-noID-stats.xml")//Course  
        where $c/@Number = $stat/@Number (: jointure :)  
        return  
          <Course>{$c/@Number}{$c/*}{$stat/*}</Course>  
      }  
    </Department>  
}</Course_Catalog_With_Stats>
```

&lt;Co

fo

re

&lt;D

&lt;/I

}&lt;/C

```
...
<Course Number="CS106A">
  <Title>Programming Methodology</Title>
  <Description>
    Introduction to the engineering of computer
    applications emphasizing modern software
    engineering principles.
  </Description>
  <Instructors>
    <Lecturer>
      <First_Name>Jerry</First_Name>
      <Middle_Initial>R.</Middle_Initial>
      <Last_Name>Cain</Last_Name>
    </Lecturer>
    <Professor>
      <First_Name>Eric</First_Name>
      <Last_Name>Roberts</Last_Name>
    </Professor>
    <Professor>
      <First_Name>Mehran</First_Name>
      <Last_Name>Sahami</Last_Name>
    </Professor>
  </Instructors>
  <Enrolled>147</Enrolled>
  <Passed>68</Passed>
  <Credits>3</Credits>
</Course>
...

```

rtment

ml"))//Course

ointure :)

Course&gt;

XQuery

M. Sassolas

L3Pro

Cours 5

Motivation

Syntaxe

Exemples

Utilisation

Mise en  
application

## Syntaxe

```
saxon-xquery -o:fichierProduit.xml requete.xq
```

## Rappel

Les fichiers XML servant de bases de données sont référencés dans le .xq, donc pas passés comme arguments de la ligne de commande.

XQuery

M. Sassolas

L3Pro

Cours 5

Motivation

Syntaxe

Exemples

Utilisation

Mise en  
application

↳ C'est l'heure du TP ←