

XQuery

Une extension de XPath à la mode SQL
Un concurrent d'XSLT ?

Yves Bekkers

Mise à jour : 16 octobre 2007

Introduction

XQuery - Y. Bekkers

2

Un langage d'expression typé

Entier

Expression : `2+3`

Résultat : 5

Chaîne

Expression : `concat('bon', 'jour')`

Résultat : Bonjour

Élément xml

Expression : `<a>2`

Résultat : `<a>2`

XQuery - Y. Bekkers

3

Un langage d'expression typé - bis

• Booléen

– Expression : `not(or(true(),false()))`

– Résultat : false

• Éléments xml : expression XPath

– Expression : `//p`

– Résultat :
`<p> ... </p>`

XQuery - Y. Bekkers

4

Type "éléments XML"

• Contenu statique

Expression

`<résultat>5*10</résultat>`

Résultat

`<résultat>5*10</résultat>`

• Calcul dynamique de contenu

Expression

`<résultat>{5*10}</résultat>`

Résultat

`<résultat>50</résultat>`

Expression
XQuery

XQuery - Y. Bekkers

5

Variables

• Expression XQuery

let

`$x := 3+2,`

`$y := 6`

return

`10*$x+$y`

Expression
XQuery

Déclaration

Expression
XQuery

Utilisation

• Résultat

56

XQuery - Y. Bekkers

6

Séquences (listes, tuples)

- Expression
`(1,2,2)`
- Résultat
122
- Expression
`(1,"",2)`
- Résultat
1,2
- Expression
`1,2,5`
- Résultat
125
- Expression
`(1,"",2)`
- Résultat
1
2

XQuery - Y. Bekkers

7

Séquences de valeurs

Expression

```
let $a := (3,4),
    $b := ($a, $a),
    $c := 99,
    $d := ()
return (count($a),",",count($b),",",
        count($c),",",count($d))
```

Résultat

2, 4, 1, 0

XQuery - Y. Bekkers

8

Expression xpath

- Expression

```
<results> {  
  //epicerie/listeFournisseurs/fournisseur  
}</results>
```
- Résultat

```
<results>  
  <fournisseur>  
    <F>f1</F>  
    <Nom>Barnibus</Nom>  
    <Remise>0,05</Remise>  
    <Ville>Paris</Ville>  
  </fournisseur>  
  <fournisseur>  
    <F>f2</F>  
  ...
```

Contexte implicite

XQuery - Y. Bekkers

9

Valeur de type "nœud xpath"

Expression

```
let  
  $e := <p>This is <em>very</em>  
  cool.</p>  
return  
  $e/node()
```

Résultat

```
This is  
<?xml version="1.0" encoding="UTF-8"?>  
<em>very</em>  
cool.
```

XQuery - Y. Bekkers

10

Commenter les programmes XQuery

Expression

```
(: Ceci est un commentaire :)  
2*5
```

Résultat

10

XQuery - Y. Bekkers

11

Générer un commentaire ou une instruction de traitement

Expression

```
<!-- résulte du calcul 5*10 -->  
<résultat>{5*10}</résultat>
```

Résultat

```
<!-- résulte du calcul 5*10 -->  
<résultat>50</résultat>
```

Idem pour les instructions de traitement

XQuery - Y. Bekkers

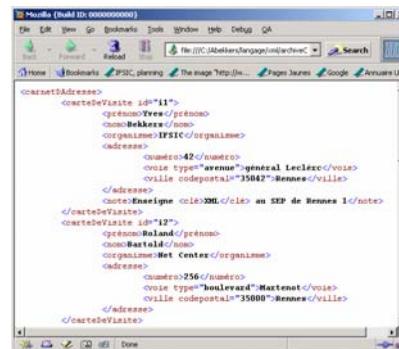
12

Expressions xpath

XQuery - Y. Bekkers

13

Exemple 1 - Le carnet d'adresse



[Voir le doc](#)

14

Expression xpath

- Les éléments <ville> de toutes les cartes de visite

```
doc("carnetDAdresse.xml")  
//carteDeVisite/adresse/ville
```

Résultat

```
<ville codepostal="35042">Rennes</ville>  
<ville codepostal="35042">Rennes</ville>  
...
```

- Le nom des villes de toutes les cartes de visite

```
doc("carnetDAdresse.xml")  
//carteDeVisite/adresse/ville/text()
```

Résultat

```
RennesRennesRennesRedonVannesBrest
```

XQuery - Y. Bekkers

15

Extension/restriction de xpath

- Extension :**

- Construction d'éléments xml (comme XSLT)
- Variables (comme XSLT)
- ...

- Restriction :**

- XQuery supporte seulement un sous ensemble des 13 axes de xpath (dépendant de l'implémentation)
- Sont obligatoires, les axes dont l'écriture abrégée est autorisée, c'est-à-dire :
Child::, parent::, attribut::, self::,
descendant::, descendant-or-self::

XQuery - Y. Bekkers

16

Expression FLWR

For-Let-Where-Return
se lit "flower"

XQuery - Y. Bekkers

17

Expression FLWR

- for** \$v in *expr*

- Génère un flux composé de variables liées aux valeurs prises dans l'ensemble résultant de l'évaluation de *expr*

- let** \$v := *expr*

- Génère un flux composé d'une variable liée à une seule valeur (qui peut être une séquence)

- where** *condition*

- Filtre le flux

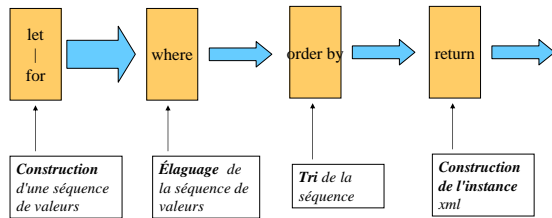
- return** *expr*

- Construit le résultat en évaluant *expr* une fois par élément du flux

XQuery - Y. Bekkers

18

Flux de données dans une expression FLWR



XQuery - Y. Bekkers

19

Variables

- Déclaration/initiaisation de variables
 - Instruction let
 - Exemple (valeurs de type élément)

```
let
  $a := <a><b>bonjour</b><c>monsieur</c></a>,
  $c := $a/c
return $c
```

Résultat
<c>monsieur</c>

XQuery - Y. Bekkers

20

Écritures équivalentes

```
let
  $a := <a><b>bonjour</b><c>monsieur</c></a>,
  $c := $a/c
return $c
```

Équivalent à

```
let
  $a := <a><b>bonjour</b><c>monsieur</c></a>
let $c := $a/c
return $c
```

XQuery - Y. Bekkers

21

Séquence en sortie

- Construction d'une séquence en sortie

```
let
  $a := <a><b>bonjour</b><c>monsieur</c></a>,
  $b := $a/b,
  $c := $a/c
return
  ($b, $c)
```

Résultat
bonjour
<c>monsieur</c>

XQuery - Y. Bekkers

22

Affectation d'une séquence

- Expression

```
let
  $a := (<one/>, <two/>, <three/>)
return
  <out>{$a}</out>
```

- Résultat

```
<out>
  <one/>
  <two/>
  <three/>
</out>
```

XQuery - Y. Bekkers

23

Itération sur une séquence

Expression

```
for
  $a in (<one/>, <two/>, <three/>)
return
  <out>{$a}</out>
```

Résultat

```
<out>  <one/>  </out>
<out>  <two/>  </out>
<out>  <three/> </out>
```

XQuery - Y. Bekkers

24

Séquence d'entiers consécutifs

- Séquence d'entiers consécutifs (pas d'équivalent dans xslt)

```
for $i in (1 to 3, 7)
return <out>{$i}</out>
```

Résultat

```
<out>1</out>
<out>2</out>
<out>3</out>
<out>7</out>
```

XQuery - Y. Bekkers

25

Affectation : **let** Itération : **for**

- Affectation d'une séquence à une variable

```
let $i := (1 to 4) return ($i,"")
```

\$i est de type liste d'entiers == (1,2,3,4)

– Résultat : 1234,

- Itération sur les valeurs d'une séquence

```
for $i in (1 to 4) return ($i,"")
```

\$i est de type entier \$i==1 puis \$i==2 ...

– Résultat : 1,2,3,4,

XQuery - Y. Bekkers

26

Séquence en entrée et en sortie

Expression

```
for $i in (1 to 3, 7)
return <out>{$i,'',$i+1}</out>
```

Résultat

```
<out>1,2</out>
<out>2,3</out>
<out>3,4</out>
<out>7,8</out>
```

XQuery - Y. Bekkers

27

Séquence en entrée et en sortie (bis)

Expression

```
let $i := (1 to 3, 7)
return <out>{$i}</out>
```

Résultat

```
<out>1237</out>
```

XQuery - Y. Bekkers

28

Ordre de génération des tuples

- La plus rapide à changer de valeur est celle qui arrive en dernier

– Expression

```
for $i in (1, 2), $j in (3, 4)
Return concat('$i = ',string($i),
'$j = ',string($j))
```

– Résultat

```
$i = 1, $j =3
$i = 1, $j =4
$i = 2, $j =3
$i = 2, $j =4
```

XQuery - Y. Bekkers

29

Itération sur des éléments

Question :

pour toutes les cartes de visite, créer un élément
<name> avec le nom de la carte comme contenu

```
for $c in doc("carnetDAdresse.xml")
/carnetDAdresse/carteDeVisite
return
<name>{$c/nom/text()}</name>
```

Réponse :

```
<name>Bekkers</name>
<name>Bartold</name>
```

...

XQuery - Y. Bekkers

30

Encore un exemple

- Pour chaque carte de visite créer un élément `<personne>` avec son nom et prénom

```
for $c in doc('carnetDAdresse.xml')
  /carnetDAdresse/carteDeVisite
return
  <personne>{
    $c/(prénom | nom)
  }</personne>
```

Résultat

```
<personne>
  <prénom>Yves</prénom>
  <nom>Bekkers</nom>
</personne>
```

...

XQuery - Y. Bekkers

31

Sélection "à la sql"

- Instruction "FLWR" partie "where"

```
for $carte in doc("carnetDAdresse.xml")
  //carteDeVisite
where $carte/adresse/ville = "Rennes"
return $carte/nom
```

Résultat

```
<nom>Bekkers</nom>
<nom>Bartold</nom>
<nom>Grosjean</nom>
```

XQuery - Y. Bekkers

32

Sélection écriture simplifiée

- Instruction "FLWR" partie "where"

```
for $carte in doc("carnetDAdresse.xml")
  /carnetDAdresse /carteDeVisite
where $carte/adresse/ville = "Rennes"
return $carte/nom
```

- Écriture simplifiée équivalente

```
for $carte in doc("carnetDAdresse.xml")
  /carnetDAdresse /carteDeVisite
[adresse/ville = "Rennes"]
return $carte/nom
```

XQuery - Y. Bekkers

33

Valeurs d'attributs

Évaluation à la manière d'XSLT

```
for $v in doc("carnetDAdresse.xml")
  //carteDeVisite/adresse/ville
[. = 'Rennes']
return
  <town code="{ $v/@codepostal }">
    Rennes</town>
```

Réponse :

```
<town code="35042">Rennes</town>
<town code="35000">Rennes</town>
...
```

XQuery - Y. Bekkers

34

Construire un résultat avec une itération

- Question

```
<result> {
  for $e in (<e>a</e>,
    <e>a</e>, <e>b</e>)
  return $e
} </result>
```

- Résultat

```
<result>
  <e>a</e>
  <e>a</e>
  <e>b</e>
</result>
```

- Question

```
for $e in (<e>a</e>,
  <e>a</e>, <e>b</e>)
return <res> {
  $e
} </res>
```

- Résultat

```
<res><e>a</e></res>
<res><e>a</e></res>
<res><e>b</e></res>
```

XQuery - Y. Bekkers

35

Conditionnelle

```
for $i in (1 to 4)
  return <out>
  {if ($i < 3) then <a/> else <b/>}</out>
```

Résultat

```
<out><a/></out>
<out><a/></out>
<out><b/></out>
<out><b/></out>
```

XQuery - Y. Bekkers

36

If : Comparaison XQuery/xslt

XQuery

```
if ($i < 3) then <a/> else <b/>
```

XSLT

```
<xsl:choose>
  <xsl:when select="$i < 3">
    <a/>
  </xsl:when>
  <xsl:otherwise><b/></xsl:otherwise>
</xsl:choose>
```

XQuery - Y. Bekkers

37

Composition d'expressions

- XQuery (xpath)
doc("carnetDAdresse.xml")/
carnetDAdresse/carteDeVisite/nom
- Est équivalent à
let \$c := doc("file.xml")/carnetDAdresse
return
for \$carte in \$c/carteDeVisite
return
\$carte/nom

XQuery - Y. Bekkers

38

Associativité

- Xpath
(carnetDAdresse/carteDeVisite)/nom
= carnetDAdresse/(carteDeVisite/nom)
- XQuery
for \$carnet in doc("file.xml")/carnetDAdresse
return
for \$carte in \$carnet/carteDeVisite
return \$carte/nom
- Est équivalent à
for \$carte in
for \$carnet in doc("file.xml")/carnetDAdresse
return \$carnet/carteDeVisite
return \$carte/nom

XQuery - Y. Bekkers

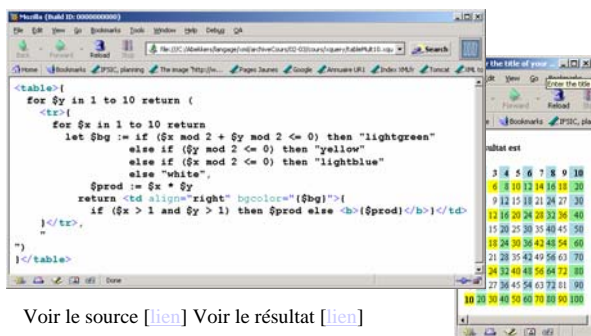
39

Jointures

XQuery - Y. Bekkers

40

Générer une table de multiplication

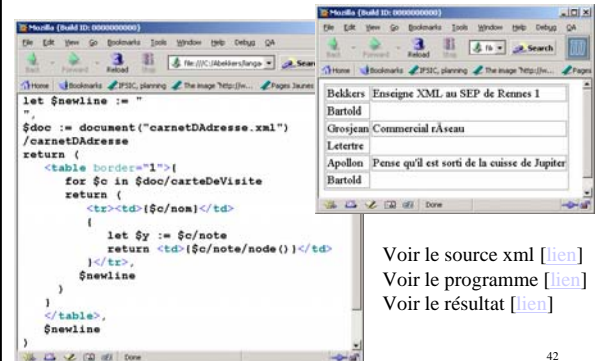


Voir le source [\[lien\]](#) Voir le résultat [\[lien\]](#)

XQuery - Y. Bekkers

41

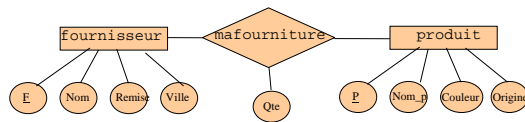
Carnet d'adresse – tableau des "notes"



Voir le source xml [\[lien\]](#)
Voir le programme [\[lien\]](#)
Voir le résultat [\[lien\]](#)

42

Exemple 2 – une base de donnée



fournisseur(F:chaîne, Nom:chaîne, Remise:réel, Ville:chaîne)
 mafourniture(F:chaîne, P:chaîne, Qte:réel)
 produit(P:chaîne, Nom_P:chaîne, Couleur:chaîne, Origine:chaîne)

XQuery - Y. Bekkers

43

Exemple 2 – représentation en xml

```
<fournisseur>
  <Row>
    <F>f1</F>
    <Nom>Barnibus</Nom>
    <Remise>0,05</Remise>
    <Ville>Paris</Ville>
  </Row>
  ...
</fournisseur>

<maFourniture>
  <Row>
    <F>f1</F>
    <P>p1</P>
    <Qte>1</Qte>
  </Row>
  ...
</maFourniture>

<produit>
  <Row>
    <P>p1</P>
    <Nom_p>cassis</Nom_p>
    <Couleur>rouge</Couleur>
    <Origine>Dijon</Origine>
  </Row>
  ...
</produit>
```

44

Jointure interne

Question : code des produits fournis par chaque fournisseur

```
for $fournisseur in
  doc("fournisseur.xml")/fournisseur/Row,
  $maFourniture in
  doc("maFourniture.xml")/maFourniture/Row
where
  $maFourniture/F = $fournisseur/F
return
<prod>{$fournisseur/Nom,$maFourniture/P}</prod>
```

Réponse :

```
<prod>
  <nom>Barnibus</nom>
  <P>p1</P>
</prod>
...
```

XQuery - Y. Bekkers

45

Jointure – écritures équivalentes

- Utilisation de la clause Where d'XQuery

```
for $fournisseur in
  doc("fournisseur.xml")/fournisseur/Row,
  $maFourniture in
  doc("maFourniture.xml")/maFourniture/Row
where
  $maFourniture/F = $fournisseur/F
```

- Utilisation d'un prédicat [...] d'XPath

Remarque: la différence

```
for $fournisseur in
  doc("fournisseur.xml")/fournisseur/Row,
  $maFourniture in
  doc("maFourniture.xml")/maFourniture/Row
[ $maFourniture/F = $fournisseur/F ]
```

XQuery - Y. Bekkers

46

Recherche dans une collection de docs

- Adresser un doc

doc("fournisseur.xml") ← Nom de fichier

- Adresser une collection de docs

collection("xml") ← Nom de répertoire

XQuery - Y. Bekkers

47

Jointure interne (Question Q1)

Question : donner la liste des couples <fournisseur, produit>

```
for $fournisseur in collection("xml")/fournisseur/Row,
  $maFourniture in collection("xml")/maFourniture/Row,
  $produit in collection("xml")/produit/Row
where
  $maFourniture/F = $fournisseur/F and
  $maFourniture/P = $produit/P
return
<prod>{$fournisseur/Nom, $produit/Nom_p}</prod>
```

Réponse :

```
<prod>
  <Nom>Barnibus</Nom>
  <Nom_p>moutarde</Nom_p>
</prod>
...
```

XQuery - Y. Bekkers

48

Jointure interne (Question Q2)

Question : donner la liste des tripets < fournisseur, produit, couleur>

```
for
  $fournisseur in collection("xml")/fournisseur/Row,
  $maFourniture in collection("xml")/maFourniture/Row,
  $produit in collection("xml")/produit/Row
where
  $maFourniture/F = $fournisseur/F and
  $maFourniture/P = $produit/P
return <prod>{
  $fournisseur/Nom, $produit/Nom_p, $produit/Couleur
} </prod>
```

Réponse :

```
<prod>
  <Nom>Barnibus</Nom>
  <Nom_p>moutarde</Nom_p>
  <Couleur>jaune</Couleur>
</prod>
```

49

Jointure externe

Question pour chaque fournisseur donner pour chaque couleur la listes des produits (on veut tous les fournisseurs)

Expression

```
for
  $fournisseur in collection("xml")/fournisseur/Row
return
  <fournisseur>{$fournisseur/Nom,
    for
      $couleur in distinct-values(
        collection("xml")/produit/Row/Couleur)
    return
      <prod couleur="{ $couleur }"> {
        ... composition page suivante ...
      } </prod>
  } </fournisseur>
```

Utilisation
de la fonction
distinct-values()

XQuery - Y. Bekkers

50

Jointure externe (suite)

Expression à inclure page précédente

```
...
for
  $produit in collection("xml")/produit/Row,
  $maFourniture in collection("xml")/maFourniture/Row
where
  $maFourniture/F = $fournisseur/F and
  $maFourniture/P = $produit/P and
  (:sélection de la couleur :)
  $produit/Couleur = $couleur
return
  $produit/Nom_p
...
```

XQuery - Y. Bekkers

51

Jointure externe (Réponse)

```
<fournisseur>
  <Nom>Barnibus</Nom>
  <prod couleur="rouge">
    <Nom_p>cassis</Nom_p>
  </prod>
  <prod couleur="blanc"/>
  <prod couleur="vert">
    <Nom_p>salade</Nom_p>
    <Nom_p>cornichon</Nom_p>
  </prod>
  <prod couleur="jaune">
    <Nom_p>moutarde</Nom_p>
  </prod>
</fournisseur>
```

Voir la question [\[lien\]](#) pour Galax
Voir le résultat [\[lien\]](#)

XQuery - Y. Bekkers

52

Jointure externe (Réponse)

```
<fournisseur>
  <Nom>Barnibus</Nom>
  <prod couleur="rouge">
    <Nom_p>cassis</Nom_p>
  </prod>
  <prod couleur="blanc"/>
  <prod couleur="vert">
    <Nom_p>salade</Nom_p>
    <Nom_p>cornichon</Nom_p>
  </prod>
  <prod couleur="jaune">
    <Nom_p>moutarde</Nom_p>
  </prod>
</fournisseur>
```

Voir la question [\[lien\]](#) pour Galax
Voir le résultat [\[lien\]](#)

Présence d'éléments
vides

XQuery - Y. Bekkers

53

Enlever les éléments vides (1)

Question pour chaque fournisseur donner pour chaque couleur la liste des produits fournis

Expression

```
for
  $fournisseur in collection("xml")/fournisseur/Row
return
  <fournisseur>{$fournisseur/Nom,
    for
      $couleur in distinct-values(
        collection("xml")/produit/Row/Couleur)
    return
      ... composition page suivante ...
  } </fournisseur>
```

XQuery - Y. Bekkers

54

Enlever les éléments vides (2)

```
let $result :=
  for
    $produit in doc("produit.xml")/produit/Row,
    $maFourniture in doc("maFourniture.xml")/
      maFourniture/Row
  where
    $maFourniture/F = $fournisseur/F and
    $maFourniture/P = $produit/P and
    $produit/Couleur = $couleur
  return
    $produit/Nom_p
return
  (: test du resultat non vide :)
  if ($result)
  then
    <prod couleur="{ $couleur }"> {
      $result
    } </prod>
  else ()
```

XQuery - Y. Bekkers

55

Réponse sans les éléments vides

```
<fournisseur>
  <Nom>Barnibus</Nom>
  <prod couleur="rouge">
    <Nom_p>cassis</Nom_p>
  </prod>
  <prod couleur="vert">
    <Nom_p>salade</Nom_p>
    <Nom_p>cornichon</Nom_p>
  </prod>
  <prod couleur="jaune">
    <Nom_p>moutarde</Nom_p>
  </prod>
</fournisseur>
```

...
Voir la question [\[lien\]](#) pour Galax
Voir le résultat [\[lien\]](#)

XQuery - Y. Bekkers

56

Attention à la combinatoire !

```
for
  $fournisseur in $fournisseurs/Row
return
  <fournisseur>{$fournisseur/Nom,
    for
      $couleur in distinct-values($produits/Row/Couleur)
    return
      let $result :=
        for $produit in $produits/Row,
        ...
        S'exécute en 17s, alors que
        ...
        for $produit in $produits/Row[Couleur=$couleur]
        ...
        S'exécute en 3s
```

XQuery - Y. Bekkers

57

Position dans une séquence - problème

Obtenir la position d'un élément au sein de sa fraternité

Premier essai

```
for $p in doc("produit.xml")/Import/Row
return $p/position()
```

Résultat

1111111

Second essai

```
for $p in doc("produit.xml")/Import/Row/position()
return $p
```

Résultat

1234567

XQuery - Y. Bekkers

58

Position dans une séquence - exemple

Question : sortir la position d'un élément au sein de sa fraternité

```
let $s := doc("produit.xml")/Import/Row
return
  for $i in 1 to count($s)
  return
    <produit no="{ $i }">
      { $s[ $i ]/Nom_p/text() }
    </produit>
```

Résultat

```
<produit no="1">cassis</produit>
<produit no="2">champagne</produit>
...
```

XQuery - Y. Bekkers

59

Fonctionnalités complémentaires

XQuery - Y. Bekkers

60

Calcul d'agrégats - comptage

- Nombre de produits verts

```
let
  $p := doc("produit.xml")/produit
return
  count($p/Row[Couleur='vert'])
```

Résultat

3

Expression rendant une séquence

XQuery - Y. Bekkers

61

Calcul d'agrégats - moyenne

Expression

```
let $f :=
  doc("fournisseur.xml")/fournisseur
return
  avg($f/Row/Remise)
```

Résultat

0.050714285714285714

Attention

Les contenus d'éléments doivent respecter la signature de la fonction

0,5 se note <Remise>0.5</Remise>

XQuery - Y. Bekkers

62

Recherche par le contenu fonction contains()

Expression : recherche de la chaîne "XML" dans le contenu d'élément <clé>

```
for $carte in doc("file.xml")/
  carnetDAdresse/carteDeVisite
where
  contains($carte/note/clé/text(), "XML")
return
  $carte/nom
```

Résultat

<nom>Bekkers</nom>

*Équivalent de
Clé LIKE %XML%
En SQL*

XQuery - Y. Bekkers

63

fonction contains() (bis)

Question : recherche de la chaîne "XML" dans les tous textes contenus en feuille des éléments <note>

```
for $carte in doc("file.xml")/
  carnetDAdresse/carteDeVisite
where
  contains(string($carte/note//text()), "XML")
return
  $carte/nom
```

Résultat (Quip seulement ...)

<nom>Bekkers</nom>

**Transforme une
séquence de chaînes
en une chaîne**

XQuery - Y. Bekkers

64

Quantification existentielle

Question fournisseurs fournissant un produit vert

Expression

```
for $p in
  (: question Q2 <fournisseur, produit, couleur> :) ...
where
  some $c in $p/Couleur satisfies $c = 'vert'
return $p
```

Composition

Remarque

La quantification existentielle existe naturellement dans xpath

Where \$p/Couleur = 'vert'

XQuery - Y. Bekkers

65

Fonctions agrégat

avg, count, max, min, sum

XQuery - Y. Bekkers

66

Problème des doubles (Question Q3)

• Question : *fournisseur fournissant des produits verts*

```
for $fournisseur in collection("xml")/fournisseur/Row,
  $maFourniture in collection("xml")/maFourniture/Row,
  $produit in collection("xml")/produit/Row
where
  $maFourniture/F = $fournisseur/F and
  $maFourniture/P = $produit/P and
  $produit/Couleur = 'vert'
return
<prod>{$fournisseur/Nom/text()}</prod>
```

• Réponse :

```
<prod>Barnibus</prod>
<prod>Barnibus</prod>
<prod>Bossuet</prod>
...
```



Génère des doubles

XQuery - Y. Bekkers

67

fonction distinct-values()

• Rôle

- Permet d'éliminer les doubles (*par valeur*) au sein d'une liste d'éléments
- Retourne une liste de `xs:string`

• Question

- *fournisseur fournissant des produits verts*

```
for $v in distinct-values(
  ... tout le texte de la question Q3 de la page précédente ...
) return <prod>{$v}</prod>
```

• Réponse :

```
<prod>Barnibus</prod>
<prod>Bossuet</prod>
<prod>Tanguy</prod>
```

XQuery - Y. Bekkers

68

Résultat de la fonction distinct-values()

- La fonction ne s'applique qu'à des chaînes
 - Le paramètre est «forcé» en une liste de `xs:string` avant évaluation
 - Le résultat est aussi une liste de `xs:string`

XQuery - Y. Bekkers

69

Élimination des doubles par valeur

• Question

```
<result> {
  let $v := (<e>a</e>,<e>a</e>,<e>b</e>)
  for $e in distinct-values($v)
  return <ee>{$e}</ee>
} </result>
```

• Réponse

```
<result>
  <ee>a</ee>
  <ee>b</ee>
</result>
```

Remarquez l'élimination des doubles par valeur

XQuery - Y. Bekkers

70

Tri : clause order by

Question : *couples triés <fournisseur, produits verts fourni>*

```
for $fournisseur in collection("xml")/fournisseur/Row,
  $maFourniture in collection("xml")/maFourniture/Row,
  $produit in collection("xml")/produit/Row
where
  $maFourniture/F = $fournisseur/F and
  $maFourniture/P = $produit/P and
  $produit/Couleur = 'vert'
order by $fournisseur/Nom ascending,
  $produit/Nom_p descending
return
<prod>{$fournisseur/Nom,$produit/Nom_p}</prod>
```

XQuery - Y. Bekkers

71

Tri : order by

Résultat

```
<prod><Nom>Barnibus</Nom><Nom_p>salade</Nom_p></prod>
<prod><Nom>Barnibus</Nom><Nom_p>cornichon</Nom_p></prod>
<prod><Nom>Bossuet</Nom><Nom_p>salade</Nom_p></prod>
<prod><Nom>Bossuet</Nom><Nom_p>huitre</Nom_p></prod>
<prod><Nom>Bossuet</Nom><Nom_p>cornichon</Nom_p></prod>
<prod><Nom>Tanguy</Nom><Nom_p>huitre</Nom_p></prod>
```

XQuery - Y. Bekkers

72

Exemple complet

- Question : *pour chaque producteur fournissant un produit vert donnez la liste de ses produits verts fournis*
- Résultat attendu

```
<r f="Barnibus">
  <p>salade</p>
  <p>cornichon</p>
</r>
<r f="Bossuet">
  <p>huitre</p>
  <p>salade</p>
  <p>cornichon</p>
</r>
<r f="Tanguy">
  <p>huitre</p>
</r>
```

XQuery - Y. Bekkers

73

Réponse

```
let
  $produit := doc("produit.xml")/Import,
  $fournisseur := doc("fournisseur.xml")/Import,
  $maFourniture := doc("maFourniture.xml")/Import
for
  $f in $fournisseur/Row
Return
  <r f="{ $f/Nom/text() }">{
    for
      $mf in $maFourniture/Row[F = $f/F],
      $p in $produit/Row[$mf/P = P and
        Couleur = 'vert']
    return
      <p>{ $p/Nom_p/text() }</p>
  }</r>
```

Problème !
Génère des
éléments vides

XQuery - Y. Bekkers

74

Réponse sans les éléments vides

```
let
  $produit := doc("produit.xml")/Import,
  $fournisseur := doc("fournisseur.xml")/Import,
  $maFourniture := doc("maFourniture.xml")/Import,
for
  $f in $fournisseur/Row
return
  let $r :=
    for
      $mf in $maFourniture/Row[F = $f/F],
      $p in $produit/Row[$mf/P = P and Couleur = 'vert']
    return
      <p>{ $p/Nom_p/text() }</p>
  return
    if ($r) then
      <r f="{ $f/Nom/text() }">{ $r }</r>
    else ()
```

XQuery - Y. Bekkers

75

Comparaison XQuery-xsdt

- Question (jointure): *pour chaque fournisseur donner pour chaque couleur la liste des produits fournis*
 - Requête XQuery optimisée [lien]
 - Programme XSLT [lien]
- La taille des programmes est équivalente
- Les temps d'exécutions sont les mêmes
 - De 1s à 3s pour les trois implémentations
 - XQuery :
 - Galax 0.3.0 (du 20/01/03)
 - Tamino-Quip : semble le plus rapide
 - XSLT : Saxon 6.5 (version 1.1 d'XSLT)
- Une conversion automatique de XQuery vers xsdt est envisageable ...

XQuery - Y. Bekkers

76

Modèles de données

XQuery - Y. Bekkers

77

Syntaxe "modèle de donnée"

Le modèle de donnée

```
element note {
  "enseigne",
  element cles { "XML" },
  "au SEP de Rennes1"
}
```

Est équivalent à

```
<note>
  enseigne
  <cles>XML</cles>
  au SEP de Rennes1
</note>
```

XQuery - Y. Bekkers

78

modèle de donnée (suite)

```
element carteDeViste {
  element prenom { "Yves" },
  element nom { "Bekkers" },
  element organisme { "IFSIC" },
  element adresse {
    element voie {
      attribute tt { "avenue",
        "Général Leclerc"
      },
      element ville {
        attribute codePostal { "35042",
          "Rennes"
        }
      },
      element note { "enseigne", element cles { "XML" },
        "au SEP de Rennes1" }
    }
  }
}
```

XQuery - Y. Bekkers 79

Résultat du modèle de donnée

```
<carteDeViste>
  <prenom>Yves</prenom>
  <nom>Bekkers</nom>
  <organisme>IFSIC</organisme>
  <adresse>
    <voie tt="avenue">Général Leclerc</voie>
    <ville codePostal="35042">Rennes</ville>
  </adresse>
  <note>enseigne<cles>XML</cles>au SEP de
  Rennes1</note>
</carteDeViste>
```

XQuery - Y. Bekkers 80

Intérêt du "modèle de donnée"

Calcul dynamique du nom d'élément ou d'attribut

```
element {name()} {text()}
attribute {name()} {text()}
```

Equivalent en XSLT

```
<xsl:element name="{...}">
<xsl:attribute name="{...}">
```

XQuery - Y. Bekkers

81

Définition de fonctions

XQuery - Y. Bekkers

82

Fonction somme

Obligation d'utiliser un espace de nom

```
(: fonction somme :)
declare namespace local =
  "http://www.irisa.fr/bekkers/test";
declare function local:somme
  ($p1, $p2) {$p1 + $p2};
<result> {
  local:somme(1,4)
}
</result>
```

2 Paramètres Résultat

```
<result>5</result>
```

XQuery - Y. Bekkers

83

Pas de return pour une fonction

```
declare function local:somme($p1, $p2){
  $p1 + $p2
};
```

XQuery - Y. Bekkers

84

Fonction reverse

```
(: fonction reverse :)
declare namespace local =
  "http://www.irisa.fr/bekkers/test";
declare function local:reverse ($items) {
  let $count := count($items)
  for $i in 0 to $count - 1
  return $items[$count - $i]
};
<result> {
  local:reverse(1 to 5)
}
```

Type du Paramètre et du résultat non spécifié
Type par défaut :
une séquence d'items de n'importe type

<result>5 4 3 2 1</result>

XQuery - Y. Bekkers

85

Fonction bonjour : les types

Obligation d'utiliser un espace de nom

```
declare namespace
  local = "http://www.irisa.fr/bekkers";
declare function local:test() as xs:string {
  'bonjour'
};
<result> {
  local:test()
}</result>
```

Une fonction sans paramètre
Type du résultat de la fonction

<result>bonjour</result>

XQuery - Y. Bekkers

86

Fonction "tripler"

```
declare namespace
  local = "http://www.irisa.fr/bekkers";
declare function local:tripler
  ($x as xs:integer) as xs:integer {
  3 * $x
};
<result> {
  local:tripler(5)
}</result>
```

Déclaration de paramètre formel

Résultat d'exécution
<result>15</result>

XQuery - Y. Bekkers

87

Type prédéfinis

| | |
|------------------------------|--|
| element() | N'importe quel élément |
| element(nom) | N'importe quel élément nom |
| attribute() | N'importe quel attribut |
| document-node() | N'importe quel nœud document |
| node() | N'importe quel nœud |
| processing-instruction() | N'importe quel nœud instruction |
| processing-instruction(proc) | N'importe quel nœud instruction dont la cible est proc |
| comment() | N'importe quel commentaire |
| empty() | N'importe quelle séquence vide |
| item() | N'importe quel valeur atomique |

XQuery - Y. Bekkers

88

Type prédéfinis – schema xml

| | |
|-------------|--|
| xs:integer | Entier |
| xs:string | Chaîne |
| xs:boolean | Booléen |
| xs:decimal | Valeur décimale |
| xs:float | Flottant simple |
| xs:double | Flottant double |
| xs:anyType | N'importe type |
| xs:integer? | Séquence de zéro ou un entier |
| element()+ | Séquence de un élément ou plus |
| element()* | Séquence de zéro ou plusieurs éléments |

XQuery - Y. Bekkers

89

Fonction "extraire nom d'élément"

```
declare namespace
  local = "http://www.irisa.fr/bekkers/test";
declare function local:nom
  ($x as element()*) as xs:string {
  name($x[1])
};
<result> {
  local:nom(<p/>)
}</result>
```

Séquence d'éléments quelconques

Résultat d'exécution
<result>p</result>

XQuery - Y. Bekkers

90

Fonction "profondeur maximum"

```
declare namespace
  local = "http://www.irisa.fr/bekkers";
declare function local:prof
  ($x as element()*) as xs:integer
{
  if ($x/*) then
    1 + max(for $e in $x/*
      return local:prof($e))
  else 1
}
Local:prof(doc("produit.xml"))
```

XQuery - Y. Bekkers

91

Comparaison avec XSLT

- Paramètre \$x un ensemble de nœuds

```
<xsl:template name="prof">
  <xsl:param name="x"/>
  <xsl:choose>
    <xsl:when test="$x">
      <!-- cas où $x est non vide -->
      ... (c.f. page suivante)
    </xsl:when>
    <xsl:otherwise>0</xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

XQuery - Y. Bekkers

92

En XSLT - Cas où \$x est non vide

```
<xsl:variable name="prof1">          Profondeur maximum du premier
  <xsl:call-template name="prof">
    <xsl:with-param name="x" select="$x/*[1]"/>
  </xsl:call-template>
</xsl:variable>
<xsl:variable name="prof2">          Profondeur maximum des suivants
  <xsl:call-template name="prof">
    <xsl:with-param name="x" select="$x/*[position() != 1]"/>
  </xsl:call-template>
</xsl:variable>
<xsl:choose>                          Choix entre les deux maximum
  <xsl:when test="$prof1 > $prof2">
    <xsl:value-of select="1 + $prof1"/>
  </xsl:when>
  <xsl:otherwise>
    <xsl:value-of select="1 + $prof2"/>
  </xsl:otherwise>
</xsl:choose>
```

XQuery - Y. Bekkers

93

Convertir des noms d'élément

```
declare function local:convert($x as element()*)
as element()* {
  if (string(name($x[1])) = "nom") then
    <Name>{$x[1]/node()}</Name>
  else if (string(name($x[1])) = "prénom") then
    <first>{$x[1]/node()}</first>
  else $x[1]
}
element carnetDAdresse {
  for $carte in doc("carnetDAdresse.xml")
  /carnetDAdresse/carteDeVisite
  return
    element carte {
      for $elem in $carte/*
      return local:convert($elem)
    }
}
```

Résultat d'exécution
 <nom>nl</nom>
 <prénom>nl</prénom>
 Devient
 <name>nl</name>
 <first>nl</first>

XQuery - Y. Bekkers

94

Fonction les fournisseurs par ville

```
declare namespace
  local = "http://www.irisa.fr/bekkers/test";
declare function local:fournisseurParVille
  ($ville as xs:string) as element()* {
  doc("fournisseur.xml")/
    listeFournisseur/fournisseur[Ville=$ville]
};
<result> {
  local:fournisseurParVille('Paris')/Nom
} </result>
<result>
  <Nom>Barnibus</Nom>
  <Nom>Mercier</Nom>
  <Nom>Dupont</Nom>
</result>
```

XQuery - Y. Bekkers

95

Variables dans le prologue

```
(: fonction somme :)
declare variable $pi := 3.14116;
declare namespace local =
  "http://www.irisa.fr/bekkers/test";
declare function local:somme
  ($p1, $p2) { $p1 + $p2 };
<result> {
  local:somme(1, $pi)
}
</result>
<result>4.14116</result>
```

XQuery - Y. Bekkers

96

Variable typée

```
declare variable $zero as xs:integer := 0;
declare variable $p as element()* := //p;
• Variable locale à un espace de noms
declare namespace loc = "test";
declare variable $loc:p as element()* := //p;
```

XQuery - Y. Bekkers

97

Portées

- Portée d'une variable `declare $i := ...`
 - Tout le programme XQuery
- Portée d'un `let $i := ...`
 - Le return qui suit

XQuery - Y. Bekkers

98

Les Types

XQuery - Y. Bekkers

99

Types dans xquery

- Types prédéfinis
 - Types de XMLSchéma
`xs:string`, `xs:boolean`, `xs:integer`, ...
 - Types XPath
`element()`, `comment()`, `attribut()`, `text()`,
`node()`, `processing-instruction()`, ...
 - Types sequence
`element()*`, `comment()+node()?`
- Types définis par l'utilisateur
 - Norme très complète sur ce sujet
 - Reste à voir ...

XQuery - Y. Bekkers

100

Un système de type très puissant

- Exemple
 - extrait du document *use case* du W3C ...
- ```
declare function local:address-ok
($a as element(*, ipo:Address)) as xs:boolean
{
 typeswitch ($a)
 case $zip as element(*, ipo:USAddress)
 return $zip=zip-ok($zip)
 case $postal as element(*, ipo:UKAddress)
 return $postal=postal-ok($postal)
 default return false()
};
```

XQuery - Y. Bekkers

101

## Import de types

- La norme prévoit l'import
    - d'éléments,
    - d'attributs,
    - de types
- définis dans un XMLSchema :
- Exemple import des types XHTML
- ```
import schema namespace
xhtml="http://www.w3.org/1999/xhtml" at
"http://example.org/xhtml/xhtml.xsd";
```

XQuery - Y. Bekkers

102

Modularité

XQuery - Y. Bekkers

103

Déclaration de module

- Créer un fichier "monModule.xqy" contenant :

```
module namespace <prefixe> = "<uri>";

declare function <prefixe>:<mafonc>(...) {
  ...
};
```

XQuery - Y. Bekkers

104

Utilisation d'un module

```
import module "<uri>" at "monModule.xqy";

declare namespace <prefixe1> = "<uri>";
{
  <prefixe1>:<mafonc> ( ... );
}
```

XQuery - Y. Bekkers

105

Ce qui n'est pas dans la norme

XQuery - Y. Bekkers

106

Entrée et sortie d'un processeur XQuery

- La manière de désigner le contexte de la recherche n'est pas spécifiée, dépend de l'application
 - Exemple : la fonction `collection()` pas connu par Galax
- La forme du résultat n'est pas spécifiée, :
 - Des collection d'arbres xml (Galax)
 - Des fragments d'arbres xml (Kawa)
 - Un arbre unique qui enveloppe les résultats (Quip)

XQuery - Y. Bekkers

107

Conclusion

XQuery - Y. Bekkers

108

XQuery un langage fonctionnel

- Système de type *fort*
 - Lien fort avec XMLSchema
 - *Quid des implémentations à ce sujet ?*

XQuery - Y. Bekkers

109

Un concurrent d'XSLT ?

- Navigation : `$var/xpath`
- Projection : `$var/xpath`
- Sélection : `for ... where ... return ...`
- Jointure : `for $v1 in ..., $v2 in ... where ... return ...`
- Tri : `... sortby ...`
- Construction : `<xml> ... </xml>`
- Variables : `let $v := ...`
- Fonction : `define function nom(type:var, ...) returns ...`
- Agrégat : `avg, count, max, min, sum`

XQuery - Y. Bekkers

110

XSLT versus XQuery

- Semblent offrir des possibilités voisines
- Le plus XQuery
 - Les types
 - Les structures de contrôle
 - La procédure
 - `if`, `switch`, ...
 - Énumération d'entiers consécutifs 1 to 6
- Le plus XSLT
 - Expérience plus grande
 - Robustesse des implémentations

XQuery - Y. Bekkers

111

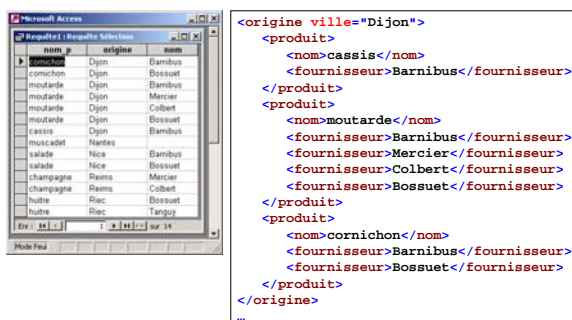
SQL versus XQuery

- XQuery est inspiré de SQL
 - Ce sont deux langages d'interrogation de données.
 - La syntaxe d'XQuery (xpath mis à part) est inspirée de celle de SQL
- Constructions plus souple pour XQuery
 - SQL construit une table à deux dimensions
 - XQuery construit un arbre

XQuery - Y. Bekkers

112

SQL versus XQuery – résultats



| nom_p | origine | nom |
|-----------|---------|----------|
| cornichon | Dijon | Barnibus |
| cornichon | Dijon | Bossuet |
| moutarde | Dijon | Barnibus |
| moutarde | Dijon | Mercier |
| moutarde | Dijon | Colbert |
| moutarde | Dijon | Bossuet |
| moutarde | Dijon | Barnibus |
| muscadet | Nantes | |
| salade | Nice | Barnibus |
| salade | Nice | Bossuet |
| champagne | Reims | Mercier |
| champagne | Reims | Colbert |
| huile | Rue | Bossuet |
| huile | Rue | Tanguy |

```

<origine ville="Dijon">
  <produit>
    <nom>cassis</nom>
    <fournisseur>Barnibus</fournisseur>
  </produit>
  <produit>
    <nom>moutarde</nom>
    <fournisseur>Barnibus</fournisseur>
    <fournisseur>Mercier</fournisseur>
    <fournisseur>Colbert</fournisseur>
    <fournisseur>Bossuet</fournisseur>
  </produit>
  <produit>
    <nom>cornichon</nom>
    <fournisseur>Barnibus</fournisseur>
    <fournisseur>Bossuet</fournisseur>
  </produit>
</origine>
  
```

XQuery - Y. Bekkers

113

Outils

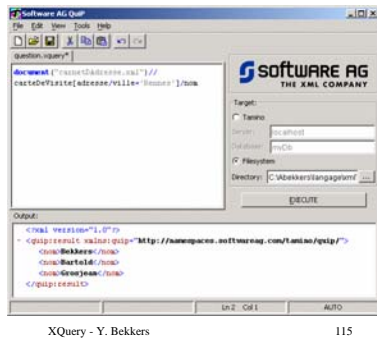
- Quip 2.2.1.1 [[lien](http://www.tamino.com/)]
 - Interface d'accès à Tamino, Base de donnée native XML
 - <http://www.tamino.com/>
- Qexo [[lien](http://www.gnu.org/software/kawa/)] S'installe sous TOMCAT
 - un traducteur de programmes Scheme en Java
 - <http://www.gnu.org/software/kawa/>
- Galax 0.3.0 [[lien](http://db.bell-labs.com/galax)] Par les laboratoires Bell-labs
 - pas de version windows pour l'instant
 - <http://db.bell-labs.com/galax>
- XQEngine [[lien](http://www.fatdog.com/)] écrit en Java
 - <http://www.fatdog.com/>
- Saxon 7.7 [[lien](http://saxon.sourceforge.net/)] écrite en Java
 - <http://saxon.sourceforge.net/>

XQuery - Y. Bekkers

114

Quip

- Une application interactive permettant d'apprendre le langage XQuery en l'utilisant



XQuery - Y. Bekkers

115

Eclipse + saxon7.7

- Un excellent outil de test grâce à une tâche Ant

```
<!-- java net.sf.saxon.Query [opt] query [params] -->
<target name="saxonXquery"
      description="Transformation XQuery">
  <java classname="net.sf.saxon.Query" fork="true">
    <arg line="-o"/>
    <arg path="{prefix}.xml"/>
    <arg line="-s"/>
    <arg path="epicerie.xml"/>
    <arg path="{prefix}.xquery"/>
    <classpath>
      <pathelement path="{saxon}/saxon7.jar"/>
    </classpath>
  </java>
</target>
```

XQuery - Y. Bekkers

116

Niveaux très variés des mises-en-œuvres (30/05/03)

- Problème de jeunesse : XQuery restait jusqu'à maintenant une "belle norme"
 - Il y a un an il n'y avait pas de mise-en oeuvre gratuite sérieuse,
 - cela est en train de changer : trois implémentations gratuites testées à ce jour, mais ...
- Saxon7.7 et plus semble en bonne voie pour être le plus complet

XQuery - Y. Bekkers

117

XQuery a deux syntaxes

- Syntaxe humaine
 - Celle présentée dans ce document
- Syntaxe XML : espace de noms XQueryx
 - <http://www.w3.org/TR/XQueryx>
- Problème
 - Qui prend en compte la syntaxe XML ?
 - A quoi sert-elle ?

XQuery - Y. Bekkers

118

Références

- XML Query Use Cases [\[lien\]](http://www.w3.org/TR/xquery-use-cases/)
 - <http://www.w3.org/TR/xquery-use-cases/>

XQuery - Y. Bekkers

119