UE Ingénierie documentaire
Master 2ème Année

Fabrice Lefèvre

2021

AVIGNON UNIVERSITÉ

XML

Partie 5 - XQuery

Fabrice Lefèvre
fabrice.lefevre@univ-avignon.fr
2021

Présentation

XML Query (*XQuery*) :

- Document XML utilisé comme une base de données
- Permet de lancer des requêtes sur documents XML
- Dérivé de XPath (et un peu concurrent...)
- Chaque requête est une *expression*
- Types d'expressions
  - `expressions de chemin`
  - `constructeurs d'éléments`
  - `expressions conditionnelles`
  - `expressions FLWOR`

Exemple XQuery

```
for $x in doc("books.xml")/bookstore/book
where $x/price>30
order by $x/title
return $x/title
```

Historique

- Langages de requêtes pour données semi-structurées : POQL de l'INRIA (1996), UnQL de Penn. Univ. (1996), Lorel de Stanford Univ. (1997)
- Langages de requêtes pour XML: XOQL (Xyleme), XML-QL, XQL, Lore, ...
- XQuery : W3C Working Draft 02 May 2003, Recommandation November 2017
- Objectif :

XQuery est à XML ce que
SQL est aux Bases de Données

Expressions simples

- Une requête XQuery est une **composition d'expressions**
  - `encore un langage fonctionnel !`
- Chaque expression a une valeur ou retourne une erreur
- Les expressions n'ont pas d'effets de bord
  - `par exemple, pas de mise-à-jour`
- Expressions (requêtes) simples :
  - `Valeurs atomiques : 46, ''Salut''`
  - `Valeurs construites : true(), date(''2002-10-23'')`

Expressions Complexes

- Expressions de chemins (XPath 2.0) :

  $FILM//ACTEUR$

- Expressions de construction d'éléments
- Tests (if-then-return-else-return)
- Expressions FLWOR (for-let-where-order-return)
- Fonctions
  - `racines : input, collection("url"), doc("url")`
  - `fonctions prédéfinies : XQuery 1.0 and XPath 2.0 Functions and Operators`
    - https://www.w3.org/TR/xpath-functions/
  - `fonctions utilisateurs`

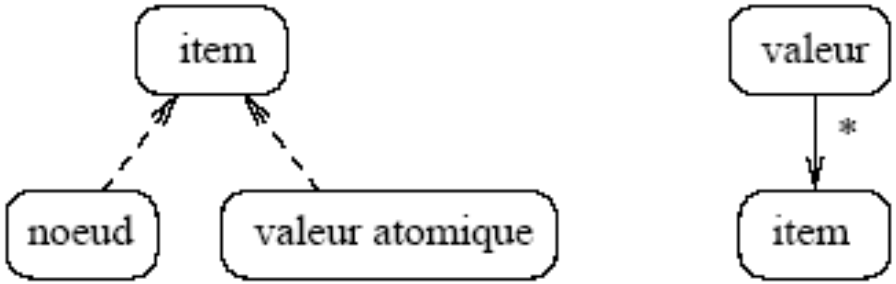
Règles syntaxiques

Quelques règles syntaxiques de base :

- XQuery est sensible a la casse
- les éléments, attributs et variables XQuery doivent avoir des noms XML valides
- les chaînes de caractères XQuery sont délimitées par de simple ou double quotes
- une variable XQuery est un identifiant précèdé de "$", par exemple $ma_variable
- les commentaires XQuery sont délimités par (: et :)

Modèle de données (1)

- Une **valeur** est une *séquence ordonnée d'items*.
- Un **item** est un **noeud** ou une **valeur atomique**.
- Chaque noeud et chaque valeur a un **type**.



Modèle de données (2)

Séquence	**Collection ordonnée d'*items*, éventuellement vide**
Item	**Une valeur atomique ou un noeud**
Valeur atomique	**Instance de type (cf. XML Schéma) : *string*, *integer*, *date*, etc.**
Nœud	**7 types : élément, attribut, texte, document, commentaire, ordre d'exécution, *namespace*** **•Peut avoir des nœuds fils et le tout forme une hiérarchie** **•Certains nœuds, comme les éléments et les attributs, peuvent avoir des noms et/ou des valeurs typées** **•Sont identifiés indépendamment de leur noms et de leur valeurs.**
Valeur typée	**Séquence de valeurs atomiques**

Valeurs et séquences

- Pas de distinction entre un item et une séquence de longueur 1 :

```
47 = (47)
```

- Une séquence peut contenir des valeurs hétérogènes :

```
(1, ''toto'', <toto/>)
```

- Pas de séquences imbriqués :

```
(1, (2, 6), "toto", <toto/>) = (1, 2, 6, ''toto'',
  <toto/>)
```

- Une séquence peut être vide :

```
()
```

- Les séquences sont triées :

```
(1,2)
```

Exemple, DTD

```
<!ELEMENT bib (book*) >
<!ELEMENT book ((author)+, publisher, price) >
<!ATTLIST book year CDATA #IMPLIED title CDATA #REQUIRED>
<!ELEMENT author (la, fi )>
<!ELEMENT la (#PCDATA )>
<!ELEMENT fi (#PCDATA )>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT price (#PCDATA)>
```

Exemple, document XML

```
<bib>
<book title="Comprendre XSLT">
   <author><la>Amann</la><fi>B.</fi></author>
   <author><la>Rigaux</la><fi>P.</fi></author>
   <publisher>O'Reilly</publisher>
   <price>28.95</price>
</book>
<book year="2001" title="Spatial Databases">
   <author><la>Rigaux</la><fi>P.</fi></author>
   <author><la>Scholl</la><fi>M.</fi></author>
   <author><la>Voisard</la><fi>A.</fi></author>
   <publisher>Morgan Kaufmann Publishers</publisher>
   <price>35.00</price>
</book>
<book year="2000" title="Data on the Web">
   <author><la>Abiteboul</la><fi>S.</fi></author>
   <author><la>Buneman</la><fi>P.</fi></author>
   <author><la>Suciu</la><fi>D.</fi></author>
   <publisher>Morgan Kaufmann Publishers</publisher>
   <price>39.95</price>
</book>
</bib>
```

Exemple, requête

- *Requête :*

```
doc("bib.xml")//author
```

- *Résultat :*

<author><la>Amann</la><fi>B.</fi></author>,
<author><la>Rigaux</la><fi>P.</fi></author>,
<author><la>Rigaux</la><fi>P.</fi></author>,
<author><la>Scholl</la><fi>M.</fi></author>,
<author><la>Voisard</la><fi>A.</fi></author>,
<author><la>Abiteboul</la><fi>S.</fi></author>,
<author><la>Buneman</la><fi>P.</fi></author>,
<author><la>Suciu</la><fi>D.</fi></author>

Exemple, requête...

- Chaque étape est une expression XQuery (XPath 3.1) :

```
doc("bib.xml")/bib/book/author
doc("bib.xml")/bib//book[1]/publisher
doc("bib.xml")//book/(author,publisher) :
doc("bib.xml")/(descendant::author,
descendant::publisher)
doc("bib.xml")//book/(@title union publisher)
doc("bib.xml")//book[position() lt last()]
```

Construction de nœuds XML (1)

- Le nom est connu, le contenu est calculé par une expression *expr*
- {} permet de placer une expression dans un constructeur
- Ex. de requête :

```
<auteurs>
{ document("bib.xml")//book[2]/author/la }
</auteurs>
```

- Résultat :

```
<? xml version="1.0" ?>
<auteurs>
<la>Rigaux</la>
<la>Scholl</la>
<la>Voisard</la>
</auteurs>
```

Construction de nœuds XML (2)

- Expression dans {} crée les nouveaux nœuds selon leur type initial (éléments et attributs)

- Ex de requête :

```
<FILM>
  { $b/@HEURE }
  { $b/TITRE }
</FILM>
```

- Résultat :

```
<FILM HEURE="20">
  <TITRE>XQuery</TITRE>
</FILM>
```

Construction de noeuds XML (3)

- Lorsque le nom ET le contenu des nouveaux moeuds sont calculés :
  - `element { expr-nom } { expr-contenu }`
  - `attribute { expr-nom } { expr-contenu }`

- Ex. de requête :

```
element{ document("bib.xml")//book[1]/name(@*[1])} {
attribute{ document("bib.xml")//book[1]/name(*[3])}{
document("bib.xml")//book[1]/*[3]}
}
```

- Résultat :

```
<title publisher="O'Reilly"/>
```

Différences de nœuds

- Ex. de requête :

```
<livre>
Tous les sous-elements sauf les auteurs:
{ document("bib.xml")//book[1]/(* except author) }
</livre>
```

- Résultat :

```
<livre>
Tous les sous-elements sauf les auteurs:
<publisher>O'Reilly</publisher>
<price>28.95</price>
</livre>
```

Concaténation de séquences

- Ex. de requête :

```
<livre>
Le prix suivi des auteurs:
{ document("bib.xml")//book[1]/(price,author) }
</livre>
```

- Résultat :

```
<livre>
Le prix suivi des auteurs:
<price>28.95</price>
<author><la>Amann</la><fi>B.</fi></author>
<author><la>Rigaux</la><fi>P.</fi></author>
</livre>
```

- On a changé l'ordre des nœuds (pas une union)

Transformation nœud en valeur

- Fonctions pour la conversion de type
  - Ex : *data* est une fonction prédéfinie de XQuery qui retourne le contenu d'un élément

- Ex de requête :

```
"Les auteurs du premier livre sont",
document("bib.xml")//book[1]/author/data(la)
```

- Résultat :

```
Les auteurs du premier livre sont, Amann, Rigaux
```

Expressions XQuery

Expression de chemin, basée sur la syntaxe XPath

- Titres des films dans le document films.xml:

```
document("films.xml")//FILM/TITRE
```

- Expressions conditionnelles : IF/THEN/ELSE

```
...
<utilisateur>
{ $u/numero }
{ $u/nom }
{
  IF (empty($b)) THEN
       <status>inactif</status>
  ELSE
       <status>actif</status>
}
</utilisateur>
```

Expression FLWOR

Formes générales :

- FOR $<var> in <forest>, $<var> ... // itération
- LET $<var> := <subtree> // affectation
- WHERE <condition> // filtrage
- ORDER BY // tri
- RETURN <result> // construction

- Similitude avec les requêtes SQL

Opérateurs

On retrouve les même opérateurs que dans les autres langages :

```
<result>
   LET $a := avg(//item/prix)
   FOR $b IN /produit
   WHERE $b/prix > $a
   ORDER BY $b/prix
   RETURN
   <article-cher>
        { $b/item }
        <difference_prix>
               { $b/prix - $a }
        </difference_prix>
   </article-cher>
</result>
```