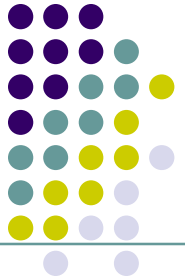
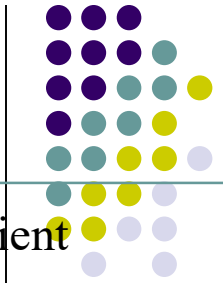
Université Gaston Berger de Saint-Louis
Année académique 2016-2017



XQUERY

Langage d'interrogation de données XML



Références

1. XML Programming Success in a Day: Beginner's Guide to Fast, Easy, and Efficient Learning of XML Programming. Sam Key, 2015.

2. XML Programming: The Ultimate Guide to Fast, Easy, and Efficient Learning of XML Programming. Christopher Right, 2015

3. Beginning XML. Joe Fawcett and Danny Ayers, 2012.

4. XML, Cours et exercices. Modélisation, Schémas et DTD, design patterns, XSLT, DOM, Relax NG, XPath, SOAP, XQuery, XSL-FO, SVG, eXist. *Alexandre Brillant*, Édition : Eyrolles 2eédition, 2010

5. Schémas XML, Jean-Jacques Thomasson, Edition Eyrolles 2002

6. World Wide Web Consortium (W3C) : w3.org 00336241511074

7. Tutoriel XSLT de World Wide Web School : w3schools.com/xsl

8. xsl.developpez.com

9. Youtube.com

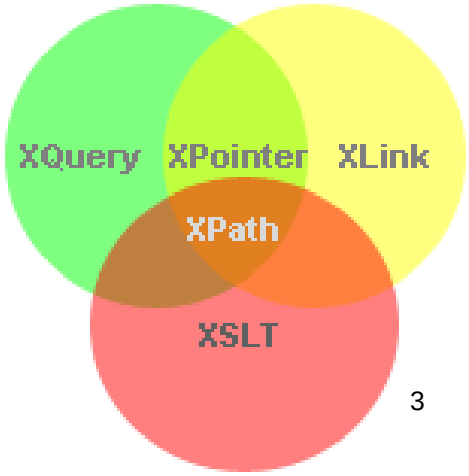
Introduction

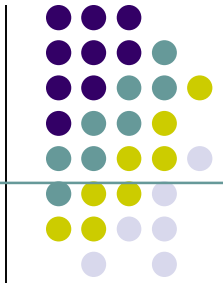
Qu'est-ce que XQuery?

- XQuery est à XML ce que SQL est aux bases de données relationnelles.
- XQuery est conçu pour interroger des données XML - pas seulement des fichiers XML, mais tout ce qui peut apparaître comme XML, y compris les bases de données.
- XQuery est construit sur des expressions XPath
- XQuery est pris en charge par toutes les principales bases de données
- XQuery est une recommandation du W3C depuis le 23 Janvier 2007.

Exemple:

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





Introduction

Qu'est-ce que XQuery?

- XQuery 1.0 et XPath 2.0 partagent le même modèle de données et prennent en charge les mêmes fonctions et les opérateurs. Si vous avez déjà étudié XPath vous aurez aucun problème pour comprendre XQuery.

XQuery peut être utilisé pour:

- Extraire des informations à utiliser dans un service Web
- Générer des rapports de synthèse
- Transformer des données XML en XHTML
- Chercher des documents pertinentes sur le Web

Les nœuds XQuery

Terminologie:

Nœud:

- Dans XQuery, il y a sept types de nœuds: les élément, les attributs, du texte, les espaces de noms, les instructions de traitement, les commentaires et les nœuds du document.
- Les documents XML sont traités comme des arbres de nœuds.
- L'élément le plus haut de l'arbre est appelé l'élément racine.

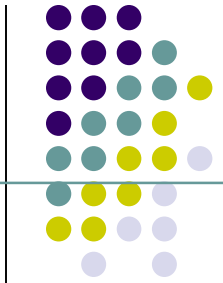
Exemple:

```
<?xml version="1.0" encoding="UTF-8"?>
<librairie>
  <livre categorie="Histoire">
    <titre">Long walk to freedom</titre>
    <auteur>Nelson Mandela</auteur>
    <annee>1995</annee>
  </livre>
</librairie>
```

Nœud élément racine

Nœud attribut

Nœud élément



Les nœuds XQuery

Terminologie:

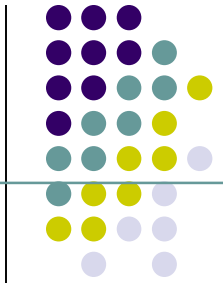
Valeurs atomique:

Une valeur atomique est un nœud sans enfant ou parent.

Exemple : `1955, "Histoire“`

Item:

Un item est une valeur atomique ou un nœud.



Les nœuds XQuery

Relation entre nœuds

Parent: Chaque élément et attribut a un parent.

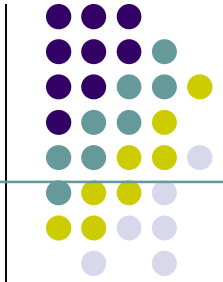
Dans l'exemple suivant livre est le parent de titre, auteur et année.

Enfant :

Un nœud élément peut avoir zéro, un ou plusieurs enfants.

Dans l'exemple suivant; titre, auteur et année sont des enfants de livre:

```
<?xml version="1.0" encoding="UTF-8"?>
<librairie>
  <livre categorie="Histoire">
    <titre">Long walk to freedom</titre>
    <auteur>Nelson Mandela</auteur>
    <annee>1995</annee>
  </livre>
</librairie>
```



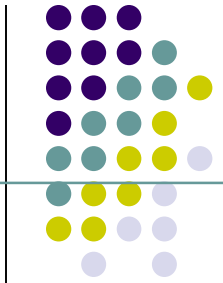
Les nœuds XQuery

Relation entre nœuds

Frères : Ce sont des nœuds ayant le même parent.
Dans l'exemple suivant, titre, auteur et année sont tous frères.

Ancêtres : Le parent d'un nœud, le parent de parent, etc.
Dans l'exemple suivant, les ancêtres de titre sont livre et librairie:

```
<?xml version="1.0" encoding="UTF-8"?>
<librairie>
  <livre categorie="Histoire">
    <titre">Long walk to freedom</titre>
    <auteur>Nelson Mandela</auteur>
    <annee>1995</annee>
  </livre>
</librairie>
```



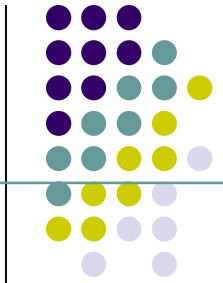
Les nœuds XQuery

Relation entre nœuds

Descendant: C'est l'enfant d'un nœud, l'enfant d'un enfant, etc.

Dans l'exemple suivant, les descendants de librairie sont livre, titre, auteur et année.

```
<?xml version="1.0" encoding="UTF-8"?>
<librairie>
  <livre categorie="Histoire">
    <titre">Long walk to freedom</titre>
    <auteur>Nelson Mandela</auteur>
    <annee>1995</annee>
  </livre>
</librairie>
```

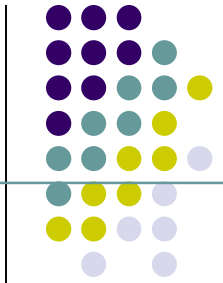


Syntaxe XQuery

Règles de syntaxe de base

- XQuery est sensible à la casse
- Les éléments, attributs et variables doivent être des noms XML valides
- Une valeur de chaîne XQuery peut être entre guillemets simples ou doubles
- Une variable XQuery est définie avec un $ suivi d'un nom, par exemple $librairie
- Les commentaires sont délimités par (: et :), par exemple
  (: Commentaire XQuery :)

Syntaxe XQuery



L'expression conditionnelle "If-Then-Else" :

```
for $x in doc("books.xml")/bookstore/book
return if ($x/@category="CHILDREN")
       then <child>{data($x/title)}</child>
       else <adult>{data($x/title)}</adult>
```

Les opérateurs de comparaison

Dans XQuery il y a deux façons de comparer les valeurs.

1. comparaisons générales:! =, =, <, <=,>,> =
2. Value comparisons: eq, ne, lt, le, gt, ge

`Ex: $bookstore//book/@q > 10`

renvoie true si tous les attributs q ont une valeur supérieure à 10.

Ex: `$bookstore//book/@q gt 10`

retourne vrai si un seul attribut q est renvoyé, et sa valeur est supérieure à 10. Si plus d'un q est renvoyé, une erreur se produit:

Les expressions FLWOR

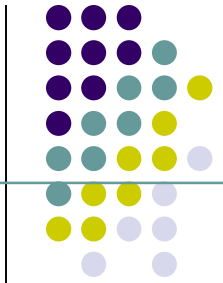
FLWOR est l'acronyme de "For, Let, Where, Order by, Return" où:

- For- sélectionne une séquence de nœuds
- Let- lie une séquence à une variable
- Where- filtre les nœuds
- Order by - trie les nœuds
- Return- contient la valeur à retourner

Exemple: Soit le doc books.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<librairie>
<livre categorie="Histoire">
  <titre langue="en">Long walk to freedom</titre>
  <auteur>Nelson Mandela</auteur>
  <annee>1993</annee>
  <prix>3000</prix>
</livre>
...
</librairie>
```

```
for $x in doc("books.xml")/librairie/livre
where $x/prix>2900
return $x/titre
```



Les expressions FLWOR

Retourner des éléments et attributs

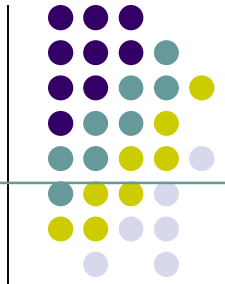
La requête

```
<ul>
{
    for $x in doc("books.xml")/bookstore/book/title
    order by $x
    return <li>{$x}</li>
}
</ul>
```

Donne comme résultat:

```
<ul>
<li><title lang="en">Everyday Italian</title></li>
<li><title lang="en">Harry Potter</title></li>
<li><title lang="en">Learning XML</title></li>
<li><title lang="en">XQuery Kick Start</title></li>
</ul>
```

Les expressions FLWOR



Retourner des éléments avec du texte

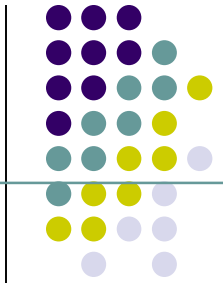
Si on veut éliminer l'élément title et montrer que les données à l'intérieur de l'élément de title, on met:

```
<ul>
{
for $x in doc("books.xml")/bookstore/book/title
order by $x
return <li>{data($x)}</li>
}
</ul>
```

Donne comme résultat:

```
<ul>
<li>Everyday Italian</li>
<li>Harry Potter</li>
<li>Learning XML</li>
<li>XQuery Kick Start</li>
</ul>
```

Les expressions FLWOR



Retourner des attributs d'éléments HTML

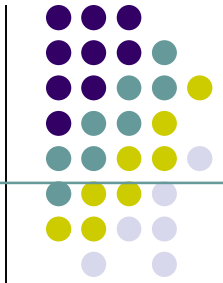
On peut modifier la valeur d'un attribut comme le montre l'exemple suivant:

```
<ul>
{
for $x in doc("books.xml")/bookstore/book
order by $x/title
return <li class="{data($x/@category)}">{data($x/title)}</li>
}
</ul>
```

Donne comme résultat:

```
<ul>
<li class="COOKING">Everyday Italian</li>
<li class="CHILDREN">Harry Potter</li>
<li class="WEB">Learning XML</li>
<li class="WEB">XQuery Kick Start</li>
</ul>
```

Les expressions FLWOR



La clause FOR

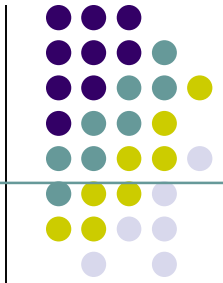
- La clause **for** lie une variable à chaque item retourné par l'expression **in**. La clause for se traduit par itération.
- Il peut y avoir plusieurs clauses **for** dans la même expression FLWOR.
- Pour boucler un certain nombre de fois dans une clause for, vous pouvez utiliser le mot-clé **to**.

Exemple:

```
for $x in (1 to 5)
return <test>{$x}</test>
```

Résultat:

```
<test>1</test>
<test>2</test>
<test>3</test>
<test>4</test>
<test>5</test>
```



Les expressions FLWOR

La clause FOR

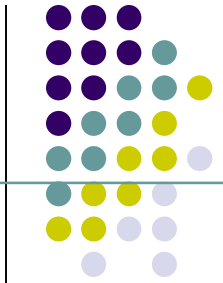
- Le mot-clé **at** peut être utilisé pour compter l'itération:

Exemple:

```
for $x at $i in doc("books.xml")/bookstore/book/title
return <book>{$i}. {data($x)}</book>
```

Résultat:

```
<book>1. Everyday Italian</book>
<book>2. Harry Potter</book>
<book>3. XQuery Kick Start</book>
<book>4. Learning XML</book>
```



Les expressions FLWOR

La clause FOR

- Il est également permis de mettre plus d'une expression dans la clause for. Utilisez des virgules pour séparer chaque expression:

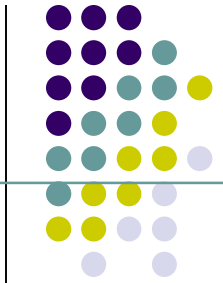
Exemple:

```
for $x in (10,20), $y in (100,200)
return <test>x={$x} and y={$y}</test>
```

Résultat:

```
<test>x=10 and y=100</test>
<test>x=10 and y=200</test>
<test>x=20 and y=100</test>
<test>x=20 and y=200</test>
```

Les expressions FLWOR



La clause LET

- La clause let permet des affectations de variables et il évite de répéter plusieurs fois une même d'expression. La clause let ne se traduit pas par itération.

Exemple:

```
let $x := (1 to 5)
return <test>{$x}</test>
```

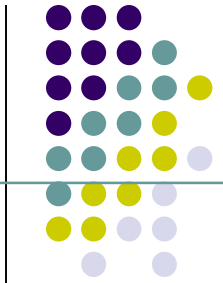
Résultat:

```
<test>1 2 3 4 5</test>
```

La clause where

- La clause where est utilisé pour spécifier un ou plusieurs critères pour le résultat:

Exemple: `where $x/price>30 and $x/price<100`



Les expressions FLWOR

La clause ORDER

- La clause order by est utilisé pour spécifier l'ordre de tri du résultat. Ici, nous voulons trier le résultat par catégorie et titre:

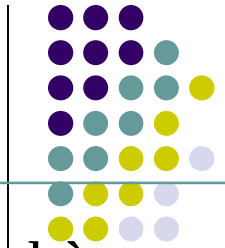
Exemple:

```
for $x in doc("books.xml")/bookstore/book
order by $x/@category, $x/title
return $x/title
```

Résultat:

```
<title lang="en">Harry Potter</title>
<title lang="en">Everyday Italian</title>
<title lang="en">Learning XML</title>
<title lang="en">XQuery Kick Start</title>
```

Les fonctions XQuery



- XQuery 1.0, XPath 2.0 et XSLT 2.0 partagent la même bibliothèque de fonctions.
- Mais on peut également définir ses propres fonctions XQuery.
- XQuery partage les mêmes types de données que XML Schema 1.0 (XSD) (String, Date, Numeric, Misc)

Exemples d'appels de fonction :

Un appel de fonction peut apparaître dans une expression XQuery:

- Dans un élément

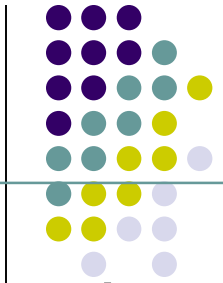
```
<name>{upper-case($booktitle)}</name>
```

- Dans le prédicat d'une expression de chemin

```
doc("books.xml")/bookstore/book[substring(title,1,5)='Harry']
```

- Dans une clause let

```
let $name := (substring($booktitle,1,4))
```



Les fonctions XQuery

Fonctions définies par l'utilisateur

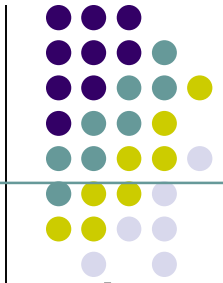
On peut écrire sa propre fonction qui peut être définie dans une requête ou dans une bibliothèque séparée.

Syntaxe

declare function *prefix:function_name* ($*parameter as datatype*)
as *returnDatatype*{
 ... *Mettre le code de la function ici*...
};

Remarques sur les fonctions définies par l'utilisateur:

- Utilisez le mot-clé declare function
- Le nom de la fonction doit être préfixé
- Les type de données des paramètres sont essentiellement les mêmes que celles de XML Schema
- Le corps de la fonction doit être entouré par des accolades



Les fonctions XQuery

Fonctions définies par l'utilisateur

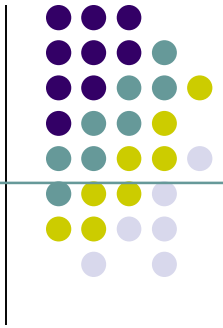
On peut écrire sa propre fonction qui peut être définie dans une requête ou dans une bibliothèque séparée.

Exemple

```
declare function local:minPrice($p as xs:decimal?,$d as
xs:decimal?)
as xs:decimal?
{
  let $disc := ($p * $d) div 100
  return ($p - $disc)
};

Voici un exemple d’appel de la fonction ci-dessus:

<minPrice>{local:minPrice($book/price,$book/discount)}</minPrice
>
```



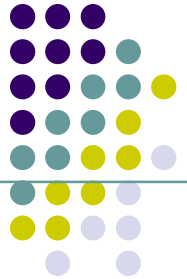
Implémentation de XQuery

Voici une liste de outils implémentant Xquery:

- **BaseX**, open source. Requêtes en direct
- eXist, open source.
- (en) Galax, implémentation en Ocaml
- (en) Qizx/open, interpréteur open source écrit en Java
- (en) **Saxon**, processeur open source XQuery et XSLT 2, réalisé par Michael Kay, concepteur de la norme XSLT 2.0.
- (en) QuiXQuery, réalisé par Innovimax et INRIA écrit en Java.
- Le portail XQuery du W3C liste une cinquantaine d'implémentations de XQuery, open-source ou commerciales, ou de produits basés sur XQuery. https://www.w3.org/XML/Query/#implementations
- Les bases de données XML natives supportant XQuery sont (en 2009) au nombre d'une quinzaine, dont environ le tiers sont open-source.
- **XQuery sous PHP avec Zorba 5**

https://www.ibm.com/developerworks/library/x-zorba/x-zorba-pdf.pdf

FIN



Merci de votre attention au cours de cette première partie.