UNIVERSITE
Abdelhamid Ibn Badis
MOSTAGANEM

Données Semi-Structurées

L3 INFORMATIQUE

Cours : XQuery

N. HOCINE

XML et Base de Données

- On utilise une base de données (BD) XML dans plusieurs cas:
  - Quand les données utilisées sont sous format XML et qu'on souhaite traiter et publier (exp. Open Data)
  - Quand on veut stocker des pages Web (avec les structures associées)
  - Quand l'application utilise une structure de données de type XML.
    - C'est le cas des applications de e-commerce (exp Amazon)
  - ...
- Différents SGBD ont été conçus afin de faciliter le traitement de base de données XML

Comment contenir un fichier XML dans une BD?

- **Solution "simple"** : transformer d'un modèle XML vers un modèle relationnel
  - Très coûteux, pas toujours bijective
- **Solution "naïve"** : garder XML comme attribut de table de type LOB (Large Object)
  - exploitation difficile

- XML comme type de base :
  - **bases de données relationnelles implémentant une extension XML** (e.x. Oracle, SQL Server, etc.)
  - **base de données XML native** (BaseX, eXist, Sedna, etc)

Principe de système de base de données XML native (SGBD-XML)

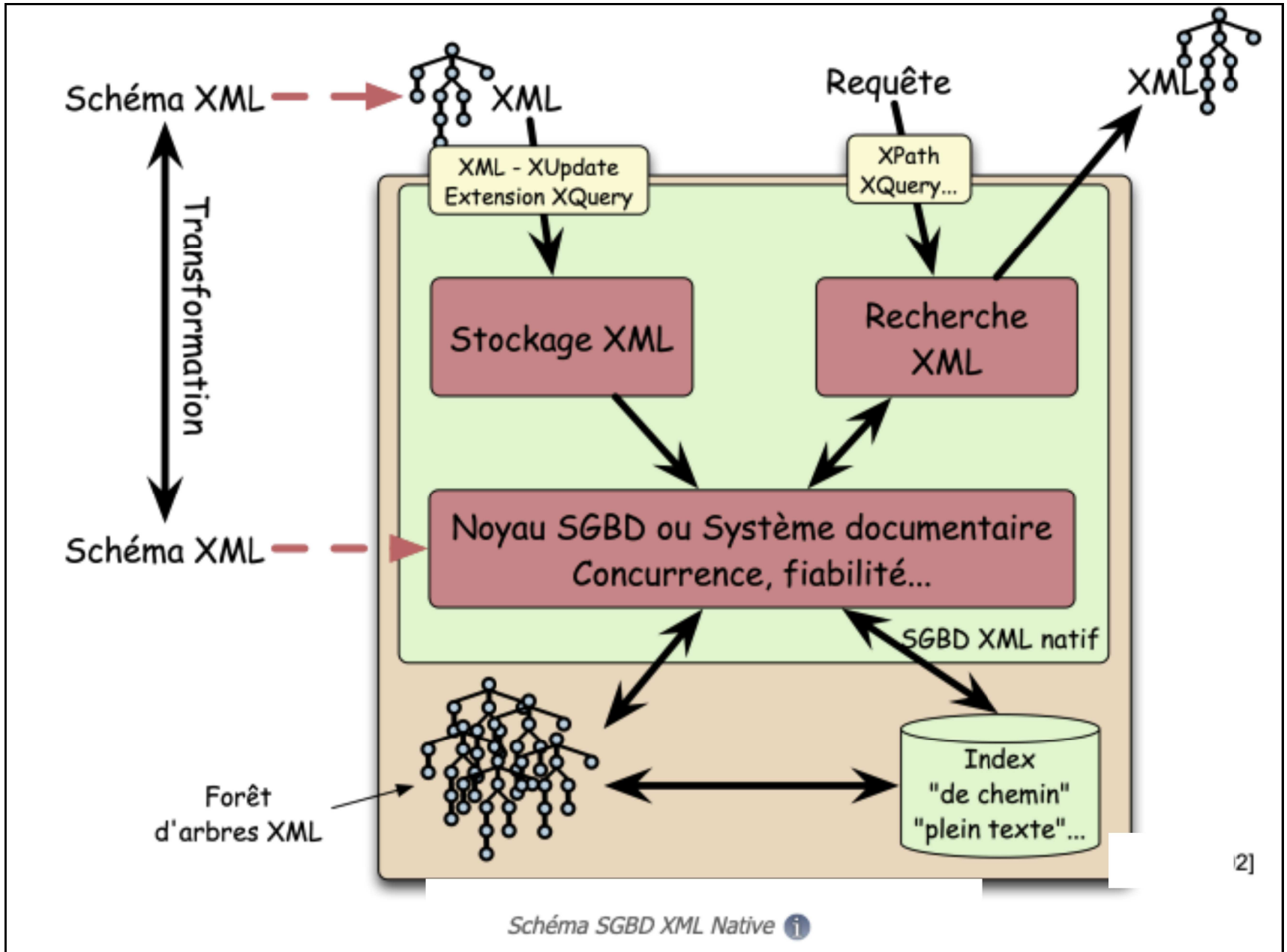

Schéma SGBD XML Native

2]

Caractéristiques d’un SGBD-XML natif

- Langages de requête : XPath/XQuery
- Lisibilité de documents
- Optimisation de l'accès aux données : indexation (chemin, sous-arbres, nœuds, etc.)
- "Forêts" d'arbres XML : La notion de collection (similaire qu’ une table pour une BDD relationnelle)
- …

XQuery

- XQuery est le langage de requêtes pour XML défini et standardisé par le W3C

- XQuery s'impose comme le langage de requêtes:
  - Pour les bases de données XML natives
  - Pour les documents XML textuels (XQuery Text)
  - Pour l'intégration de données (bases de données virtuelles)

XQuery - syntaxe

- Forme de requête élémentaire:
  - **for … let.. where … order by …return**
  - Uniquement **return** qui est obligatoire

FOR $<var> **in** <forest> **[, $**<var> **in** <forest>**]** //itération

LET $<var> **:=** <subtree> // assignation

WHERE <condition> // élagage

RETURN <result> // construction

- Les forêts sont sélectionnées par des expressions Xpath (document ou collection)
- Le résultat est une forêt (un ou plusieurs arbres)

Syntaxe

- XQuery est sensible à la casse
- Les éléments, attributs et variables doivent avoir un nom valide en xml
- string avec ‘ ‘ ou " "
- Une **Variable** est définit avec **$nom_variable**
- **(: Ceci est un commentaire :)**

XML

Exemple «librairie.xml »

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<librairie>
    <livre categorie="SPORT">
        <titre lang="en">Yoga</titre>
        <auteur> Chian J. </auteur>
        <prix>1200.99</prix>
    </livre>

    <livre categorie="ENFANT">
        <titre lang="fr"> Dessins </titre>
        <auteur> Claude M. </auteur>
         <prix>500</prix>
    </livre>
     …
</librairie>
```

XML

Fonctions Xquery

- **doc("librairie.xml") :** ouvrir le fichier librairie.xml
- **doc("librairie.xml")/librairie/livre/titre :** expression de chemin
  - Résultat:

    <titre lang="en"> Yoga </titre>

    <titre lang="fr"> Dessins </titre>

XML

Fonctions Xquery

- **doc("librairie.xml")/librairie/livre[prix>600] :** utilisation de prédicats

- Résultat:

```
<livre categorie="SPORT">
        <titre lang="en">Yoga</titre>
        <auteur> Chian J. </auteur>
        <prix>1200.99</prix>
</livre>
```

Fonctions XQuery <=> Expression FLOWER

doc("librairie.xml")/librairie/livre[prix>600]/titre

- Expression FLWOR équivalente:

```
for $x in doc("librairie.xml")/librairie/livre
where $x/prix>600
return $x/titre
```

- Résultat:

```
<titre lang="en">Yoga</titre>
```

Expression FLOWER : Ordonner le résultat

```
for $x in doc("librairie.xml")/librairie/livre/titre
order by $x
return $x
```

- Résultat:

<titre lang="fr"> Dessins </titre>

<titre lang="en"> Yoga </titre>

Expression FLOWER :

Expressions conditionnelles

```
for $x in doc("librairie.xml")/librairie/livre
return if ($x/@categorie="SPORT")
        then <titre_sport>{data($x/titre)}</titre_sport>
        else <titre>{data($x/titre)}</titre>
```

- Résultat:
  <titre_sport> Yoga </titre_sport>
  <titre> Dessins </titre>

XML

Expression FLOWER :

Opérateurs de comparaison

- Comparaison de valeurs:
  - Général: =, !=, <, <=, >, ≥
  - eq, ne, lt, le, gt, ge

- Exemple:
  - **$librairie//livre/@q> 10** retourne vrai si un attribut q est retourné par l'expression et il est >10.
    - Si plusieurs valeurs sont retournées, cela produit une erreur
    - Solution: **$librairie//livre/@q gt 10**

XML

La clause for

```
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>
```

XML

La clause for

- Le mot clé **at** peut être utilisé pour compter le nombre d'itérations

```
for $x at $i in doc("librairie.xml")/librairie/livre/titre
return <livre>{$i}.{data($x)}</livre>
```

Résultat:

<livre>1. Yoga </livre>

<livre>2. Dessins </livre>

XML

La clause for

- Il est possible de combiner plusieurs expressions séparées par des virgules:

```
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>
```

La clause let

- Let permet d'assigner une valeur à une variable pour éviter la répétition des expressions.

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

- Résultat:

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

XML

Type de données et fonctions

- XQuery utilise les même types de données que XSD (string, date, decimal, ….)
- Il est possible aussi de définir des fonctions

```
declare function prefix:fun_name($par as type) as returnType
{
 …  code ...
};
```

Appel:

<fun_name>{prefix:**fun_name (…, …)**}</fun_name>

XML

Type de données et fonctions

- 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)
};
```

Appel:

```
<minPrice>{local:minPrice($livre/prix,$livre/
promotion)}</minPrice>
```

Conclusion

- XQuery est le langage de requêtes pour XML défini et standardisé par le W3C
- L'avantage de XQuery
  - XQuery permet d'interroger des données de différents types (hiérarchique, tables, arbres, graphes)
  - Il peut être utilisé pour concevoir des pages Web
  - Il peut être utiliser pour transformer les documents XML en XHTML
  - …

Bibliographie

- W3schools
  https://www.w3schools.com/xml/default.asp
- https://www.javatpoint.com/xquery-tutorial
- https://exist-db.org/exist/apps/demo/examples/basic/basics.html

Activité

Exemple «librairie.xml »

```
<librairie>
    <livre categorie="SPORT">
        <titre lang="en">Yoga</titre>
        <auteur> Chian J. </auteur>
        <prix>1200.99</prix>
    </livre>

    <livre categorie="ENFANT">
        <titre lang="fr"> Dessins </titre>
        <auteur> Claude M. </auteur>
         <prix>500</prix>
    </livre>
     …
```

```
…
    <livre categorie="SPORT">
        <titre lang="fr">Golf</titre>
        <auteur> Alain </auteur>
        <prix>200.99</prix>
    </livre>

    <livre categorie="ENFANT">
        <titre lang="en"> Games </titre>
        <auteur> Alain </auteur>
         <prix>9000</prix>
    </livre>
     …
</librairie>
```

XML

- Donnez le résultat des requêtes XQuery suivantes:

```
<bib>
 {
 for $b in doc("librairie.xml")//livre
 where $b/auteur = "Alain" and  $b/@lang = "fr"
 return
    <livre lang="{$b/@lang}">
      {$b/titre}
    </livre>
 }
</bib>
```

Solution :

Sélectionne les livres dont l'auteur est Alain et la langue est fr. Elle retourne pour chaque résultat, un élément livre contenant le titre et un attribut lang.

L'ensemble de ces éléments livre est placé dans un élément bib.

```
<bib>
    <livre lang="fr">
     Golf
    </livre>
    <livre lang="en">
     Games
     </livre>
</bib>
```

```
<results>
  {
  for $b in doc("librairie.xml")//livre,
      $t in $b/titre,
      $a in $b/auteur
  return
     <result>
        {$t}
        {$a}
     </result>
}
</results>
```

Solution:

Construit une liste d’éléments result contenant les associations titre de livre - auteur (jointure). Ces éléments sont placés dans une balise results.

Ps. L'enchaînement de variables avec des virgules dans un for correspond à plusieurs boucles for imbriquées.

```
<results>
    <result>
            <titre lang="en">Yoga</titre>
            <auteur> Chian J. </auteur>
      </result>
    <result>
            <titre lang="fr"> Dessins </titre>
            <auteur> Claude M. </auteur>
    </result>
    <result>
            <titre lang="fr">Golf</titre>
            <auteur> Alain </auteur>
    </result>
    <result>
            <titre lang="en"> Games </titre>
            <auteur> Alain </auteur>
    </result>
 </results>
```