LES BD'S ET LE WEB

Partie2

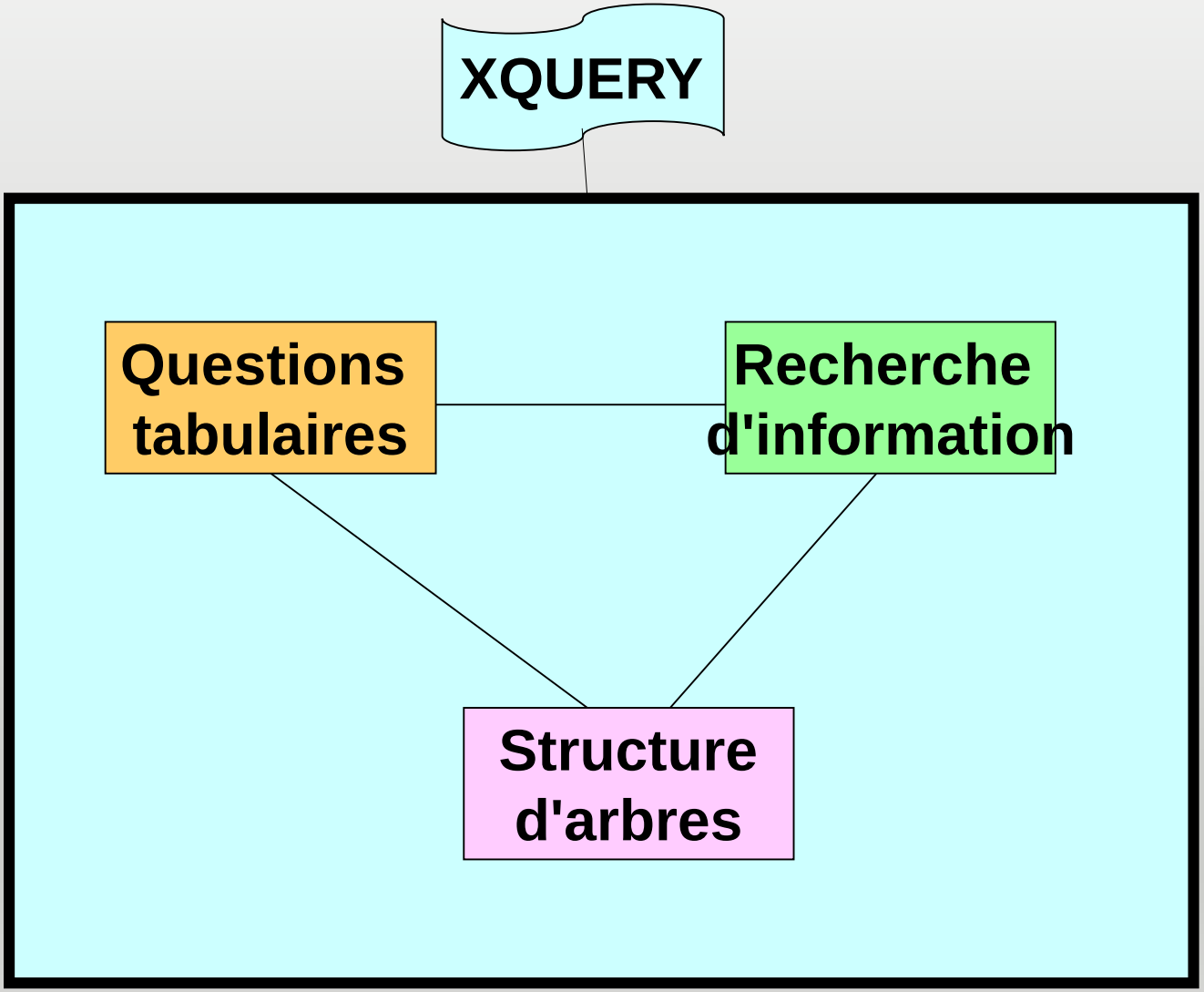
XQUERY

XML Query Language

Exemple de documents

```
<?xml version="1.0" encoding="UTF-8"?>
<Guide Version= "2.0">
 <Restaurant type="francais" categorie="***">
      <Nom>Le Moulin</Nom>
      <Adresse> <Rue>des Vignes</Rue>
           <Ville>Mougins</Ville>
      </Adresse>
      <Menu Nom="Mer" Prix="15">
           <Entrée>Moules</Entrée>
           <Plat>Poisson frais </Plat>
           <Dessert>…</Dessert>
      </Menu>
      <Menu> … </Menu>
      <Manager>Denise Fabre</Manager>
 </Restaurant>
<Restaurant type="francais«   categorie ="**">
      <Nom>La Licorne</Nom>
      <Adresse><Rue>Des Moines</Rue>
           <Ville>Paris</Ville>
      </Adresse>
      <Téléphone>0148253278</Téléphone>
      <Téléphone>0614105407</Téléphone>
      <Fax>0133445500</Fax>
      <Manager>Dupuis</Manager>
      <Comments>Bla Bla</Comments>
 </Restaurant>
 <Hotel type = « Français">
      <Nom>Ibis</Nom>
      <Adresse><Rue>Des Moines</Rue>
           <Ville>Paris</Ville>
      </Adresse>
 </Hotel>
 <Hotel>  </Hotel>
</Guide>
```

La puissance de XQuery



Interrogation des documents XML
Xpath, XQuery

XPath

- XPath est une abréviation pour *XML Path Language* (« langage de chemin XML »).
- utilise une syntaxe non-XML pour cibler différentes parties d'un document XML.
  - Utilise des expressions de chemin pour désigner des nœuds dans l'arbre
  - Les expressions XPath sont utilisées dans d'autres langages: XSLT, XQuery, …

XQuery

- Langage de requêtes pour bases de données XML

Expressions de chemins (XPath)

Selector	Selected nodes
/	Document root
//	Any sub-path
*	Any element
name	Element of tag name
@*	Any attribute
@name	Attribute of name name
text()	Any text node
processing-instruction(‘name’)	Processing instruction of given name
comment()	Any comment node
node()	Any node
id(‘value’)	Element of id value

Exemple 1 : XPath

- Noms de tous les restaurants :
  - *collection(“Restaurants”)/Restaurant/Nom*

Exemple 2 et 3 : XPath

- *Menu de tous les restaurants*
  - *collection(“Restaurants”)//Menu*

- Accès via index à attribut
  - *Donnez le nom des menus du premier restaurant*
  - *collection(“Restaurants”)/Restaurant[1]/Menu/@Nom*

Exemple 4 : Sélection

- Lister le nom des restaurants de Cabourg:
  - *collection(“Restaurants”)/Restaurant [Adresse/Ville= “Cabourg"] /Nom*

<resultat>

FOR $R IN collection("Restaurants")/Restaurant

WHERE $R/Adresse/Ville = “Cabourg”

RETURN {$R/Nom}

</resultat>

- La requête fait partie d’un document XML, le résultat est transformé en fragment XML

Exemple 5 : Jointure

- Lister le nom des Restaurants avec téléphone dans la rue de l'Hôtel Ibis:
  - *FOR $R IN collection("Restaurants")/Restaurant,*
    $H IN collection("Hotels")/Hotel

```
WHERE  $H//Rue = $R//Rue
AND $H//Nom = “Ibis"
RETURN
        <Result>
            {$R/Nom}
            {$R/Téléphone}
        </Result>
```

Exemple 6 : Restructuration d'arbre

- Construire une liste de restaurants par Ville

  - *FOR $c IN distinct(collection(“Restaurants”)/Restaurant//Ville)*

    RETURN

    <Ville>{$c}</Ville>

    <Restaurants>

      FOR $r IN collection(“Restaurants”)/Restaurant

      WHERE $r//Ville = $c

      RETURN {$r}

    <Restaurants>

Exemple 7 : Imbrication

- Adresses des hotels dans des villes ayant des restaurants trois étoiles

  - *FOR $h IN collection(“Hotels”)/Hotel*
    WHERE $h/Adresse/Ville IN
    FOR $r IN collection(“Restaurants”)/Restaurant
    *WHERE $r/@categorie = "***"*
    RETURN {$r/Adresse/Ville/text()}
    RETURN {$h/Adresse}

Exemple 8 : Agrégat simple

- Combien de restaurants y-a-t-il en collection ?

LET $R := collection(“Restaurants”)/Restaurant

RETURN

<NombreRestaurant > count ($R) </NombreRestaurant>

Exemple 9 : Agrégat partitionné

- Lister le nom de chaque restaurant avec le prix moyens des menus proposés
  - *FOR $r IN collection(“Restaurants”)//Restaurant*

    LET $a := avg(collection(“Restaurants”)// [Restaurant = $r]//Menu/@Prix)

    RETURN

    <resultat>

    {$r/Nom}

    <avgPrix>{$a}</avgPrix>

    </resultat>

Exemple 10 : recherche textuelle

- Lister les bons restaurants de Paris

  - *FOR $r IN collection(“Restaurants”)//Restaurant*
    WHERE CONTAINS ($r/Comments, “Bon”)
    OR CONTAINS ($r/Comments, “Excellent”)
    AND $r/Adresse/Ville = “Paris”
    RETURN $r/Nom

Exemple 11 : ordre et désordre

- Lister les bons restaurants de Paris par ordre alphabétique

FOR $r IN unordered(collection(“Restaurants”)//Restaurant)
WHERE $r/Comments CONTAINS (“Excellent”, “Good”)
AND $r/Adresse/Ville = “Paris”
RETURN $r/Nom
SORTBY ($r/Nom DESCENDING)

Exemple 12 :

Construire un document avec en-tête, titre, liste restaurants peu chers, titre, liste restaurants chers

```
<XML_document>
<Very_Expensive_Restaurants>
<Title>List of very expensive
    restaurants</Title>
FOR $r IN
    collection(Restaurants")//Re
    staurant
WHERE EVERY p in
    $r/Menu/@Prix SATISFIES
    (p>1000)
RETURN
{$r}
</Very_Expensive_Restaurants>
<Very_Inexpensive_Restaurants>
<Title>List of very inexpensive
    restaurants</Title>
FOR $r IN
    collection("Restaurants")//Rest
    aurant
WHERE SOME p in $r/Menu/@Prix
    SATISFIES (p<500)
RETURN
{$r}
</Very_Inexpensive_Restaurants>
</XML_document>
```

Mise à jours des documents XML
XUpdate

Comme tout SGBD, les SGBD-XML possèdent
- un langage de recherche (XQuery).
- un langage de modification :
    * L'extension du langage de requête XQuery (comme XQuery Update Facility),
    * un langage spécifique (comme XUpdate).

principe

XUpdate est caractérisé par un nom de domaine : "http://www.xmldb.org/xupdate" et un numéro de version "1.0". Il utilise XPath pour décrire les localisations des modifications. La racine d'un document XUpdate est "**modifications**" (avec le bon nom de domaine et l'attribut "version" à "1.0"). Il contient une suite d'opérations à effectuer sur la base de données (notion implicite de transaction). Les opérations autorisées sont des opérations :

- d'insertion : "insert-before" et "insert-after" ;
- d'ajout : "append" ;
- de mise à jour : "update", "rename" et "remove".
- test If , value-of

-Chaque opération opère sur l'attribut **select**

Exemple

```
<!DOCTYPE xupdate:modifications SYSTEM "xupdate.dtd">
<xupdate:modifications version="1.0" xmlns:xupdate="http://www.xmldb.org/xupdate">

   <!-- (1) - Insertion d'un élément -->
   <xupdate:insert-before select="/annuaire/personne[@id='p18']">
      <xupdate:element name="personne">
         <xupdate:attribute name="id">p17</xupdate:attribute>
         <xupdate:element name="nom">Toto</xupdate:element>
      </xupdate:element>
      <xupdate:comment>Ceci est un commentaire</xupdate:comment>
   </xupdate:insert-before>

   <!-- (2) - Mise à jour du contenu d'un élément -->
   <xupdate:update select="/annuaire/personne[@id='p17']/nom">Bill</xupdate:update>

   <!-- (3) - Suppression d'un élément -->
   <xupdate:remove select="/annuaire/personne[@id='p18']"/>

</xupdate:modifications>
```

Exemple

```
<?xml version="1.0"?>
 <annuaire>
    <personne id="p18"/>
 </annuaire>
```

La première modification permet d'obtenir un nouvel élément :

```
<?xml version="1.0"?>
      <annuaire>
            <personne id="p17">
               <nom>Toto</nom>
            </personne>
       <!-- Ceci est un commentaire -->
           <personne id="p18"/>
     </annuaire>
```

Exemple

Ensuite, la seconde modification opère une petite modification du contenu de la balise "nom" :

```
<?xml version="1.0"?>
 <annuaire>
      <personne id="p17">
            <nom>Bill</nom>
      </personne>
      <!-- Ceci est un commentaire -->
    <personne id="p18"/>
</annuaire>
```

Exemple

la troisième modification supprime le second élément "personne" :

```
<?xml version="1.0"?>
 <annuaire>
    <personne id="p17">
       <nom>Bill</nom>
    </personne>
    <!-- Ceci est un commentaire -->
</annuaire>
```