haute école
neuchâtel berne jura
arc ingénierie
saint-imier le locle delémont

Séance 9

Bases de données XML et XQuery

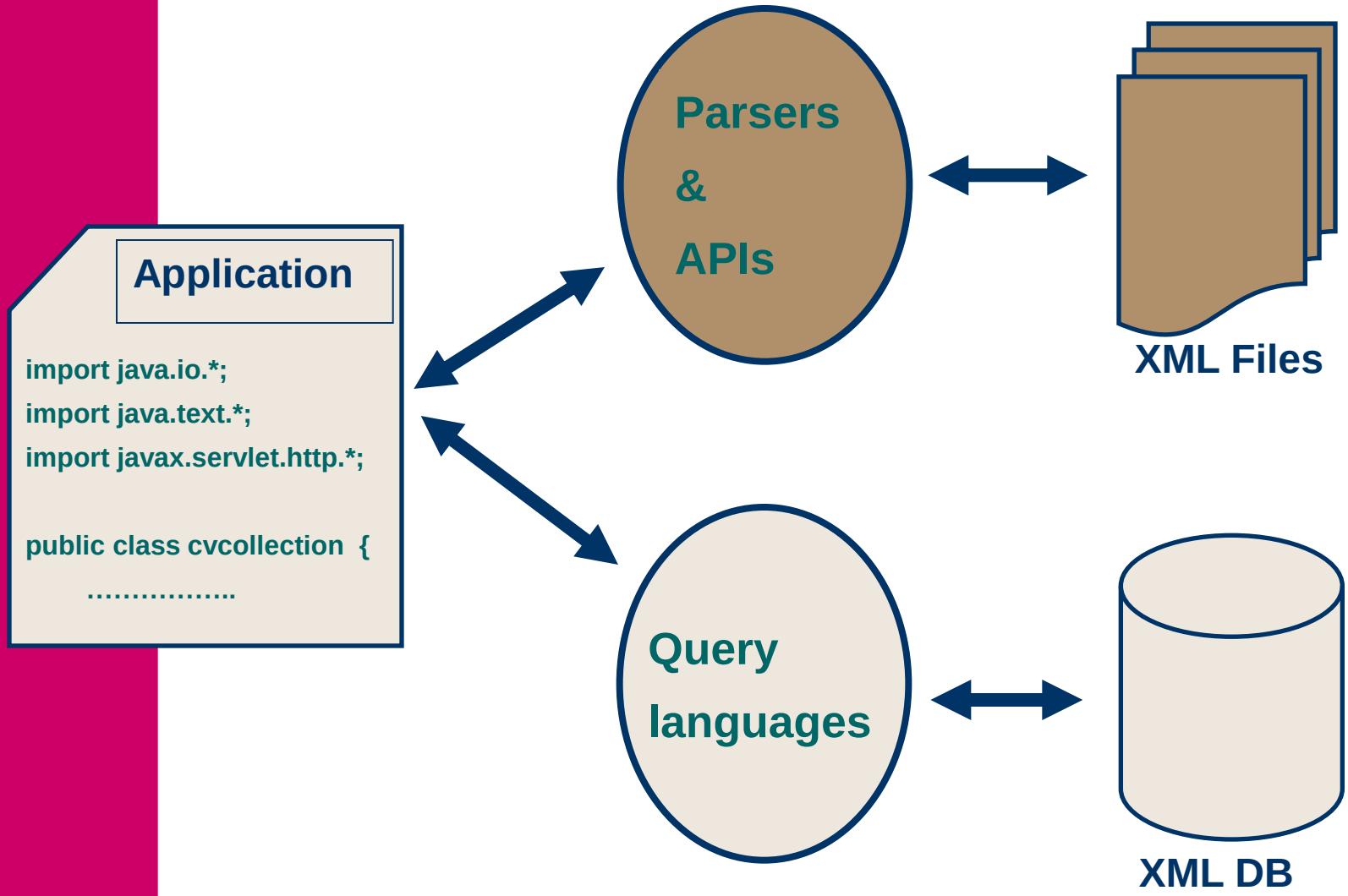
Prof. Yassin Aziz REKIK
Yassin.rekik@he-arc.ch

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale

Références

- **Divers slides de Prof. G. Gardarain**
- **Cours : C. Vanoirbeek (EPFL)**
- **Divers slides de *Yves Bekkers***

Gestion de documents XML orientée BD



XML et bases de données

- **Les fichiers XML : une base de donnée ?**
  - Stockage : fichiers textes
  - Schemas : DTDs, XML schemas, …
  - APIs pour le processing
- **Si l'application nécessite**
  - Stockage efficace, indexation, gestion de la sécurité, gestion des transactions, intégrité, accès multiple, interrogation sur plusieurs documents, triggers, ….

Technologie BD avancée nécessaire

Document XML centré-données

- **Documents XML centrés-données :**
  - Utilisés généralement pour le transport et échange de données
  - Généralement prévus pour une utilisation par machine
- **Caractérisés par :**
  - Structure régulière,
  - Données avec une granularité fine,
  - Absence ou faible nombre d'éléments mixtes,
  - Ordre des données non significatif.
- **Exemples :**
  - Ordre d'achats facture,
  - Plan et horaire de vols,
  - Données scientifiques.

Document XML centré-donnée

```
<SalesOrder SONumber="12345">
  <Customer CustNumber="543">
     <CustName>ABC Industries</CustName>
     <Street>123 Main St.</Street>
     <City>Chicago</City>
     <State>IL</State>
     <PostCode>60609</PostCode>
  </Customer>
  <OrderDate>981215</OrderDate>
  <Item ItemNumber="1">
     <Part PartNumber="123">
        <Description> <p><b>Turkey wrench:</b><br/>
        Stainless steel, one-piece construction, lifetime guarantee.
        </p>
        </Description>
        <Price>9.95</Price>
     </Part>
     <Quantity>10</Quantity>
  </Item>
  <Item ItemNumber="2">
     <Part PartNumber="456">
        <Description> <p><b>Stuffing separator:<b><br />
        Aluminum, one-year guarantee.</p>
        </Description>
        <Price>13.27</Price>
     </Part>
     <Quantity>5</Quantity>
  </Item>
</SalesOrder>
```

Document XML centré-document

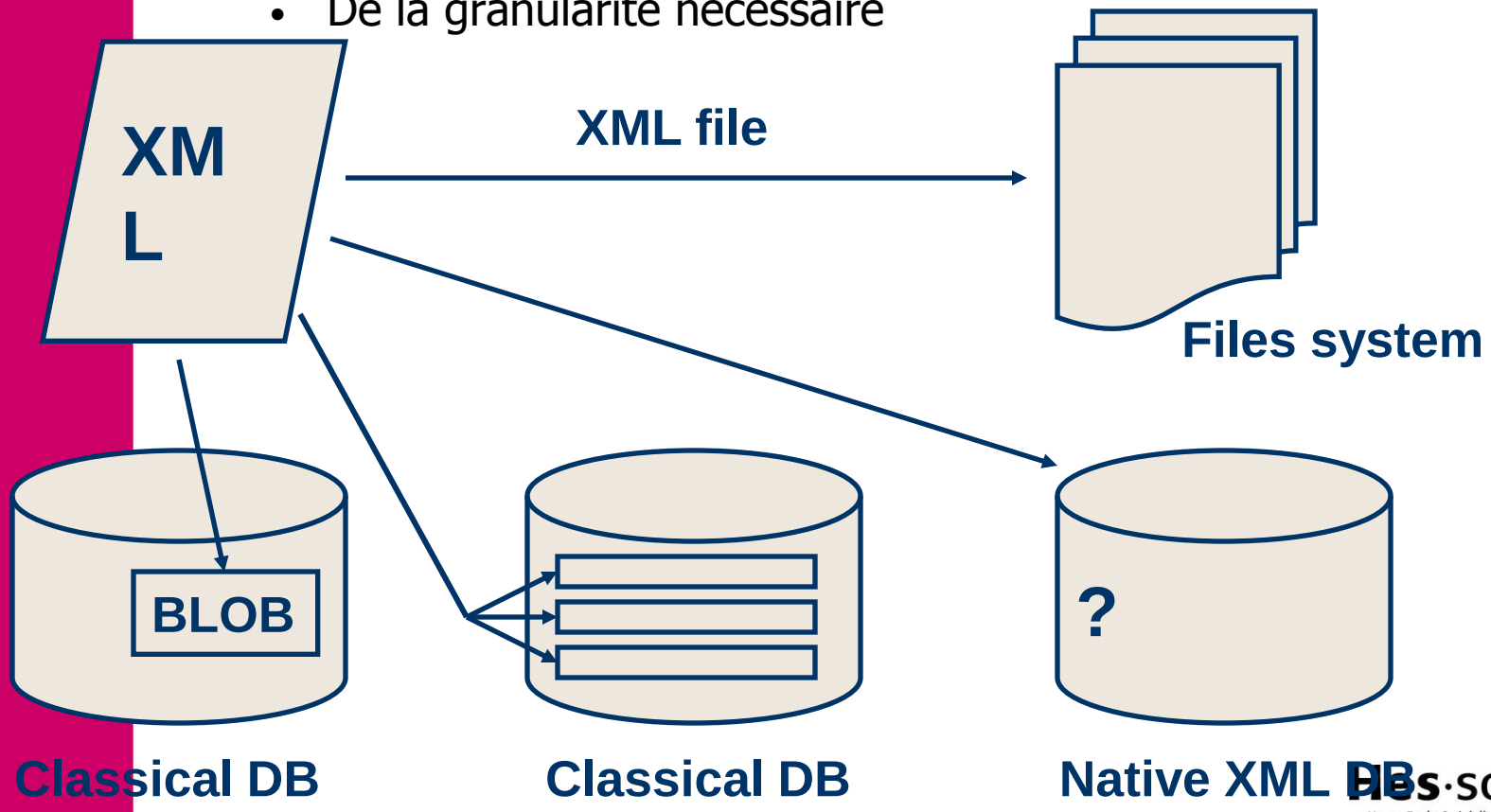
- **Documents XML centré-documents**
  - Orientés vers une utilisation humaine aussi bien en génération qu'en consultation.
- **Caractérisés par :**
  - Structure moins régulière,
  - Granularité plus grande, elle peut même coincider avec les document entier
  - Eléments mixtes nombreux,
  - L'ordre des éléments et des données est significatif.
- **Exemples:**
  - Emails,
  - Messages publicitaires
  - Manuels d'utilisation

Document XML centré-documents

```
<Product>
  <Intro> The <ProductName>Turkey Wrench</ProductName> from
    <Developer>Full Fabrication Labs, Inc.</Developer> is
    <Summary>like a monkey wrench, but not as big.</Summary>
  </Intro>
  <Description>
    <Para>The turkey wrench, which comes in <i>both right- and
    left- handed versions (skyhook optional)</i>, is made of
    the <b>finest stainless steel</b>. The Readi-grip
    rubberized handle quickly adapts to your hands, even in the
    greasiest situations. Adjustment is possible through a
    variety of custom dials.</Para>
    <Para>You can:</Para>
      <List>
          <Item><Link URL="Order.html">Order your own turkey  wrench</Link></Item>
        <Item><Link URL="Wrenches.htm">Read more about       wrenches</Link></Item>
        <Item><Link URL="Catalog.zip">Download the catalog</Link></Item>
      </List>
    <Para>The turkey wrench costs <b>just $19.99</b> and, if you order
    now, comes with a <b>hand-crafted shrimp hammer</b> as a bonus gift.
    </Para>
  </Description>
</Product>
```

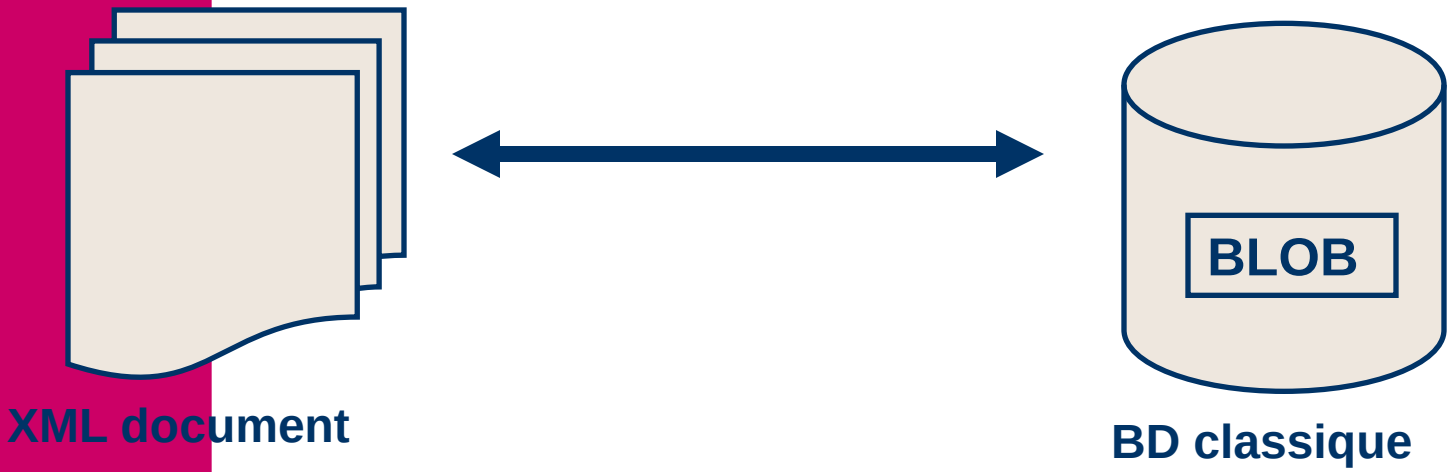
Où stocker les documents XML ?

- **Le choix dépend :**
  - Du type des document : data-centric ou document-centric,
  - De la nature des manipulations à réaliser par la suite,
  - De la granularité nécessaire



XML dans des bases de données classiques

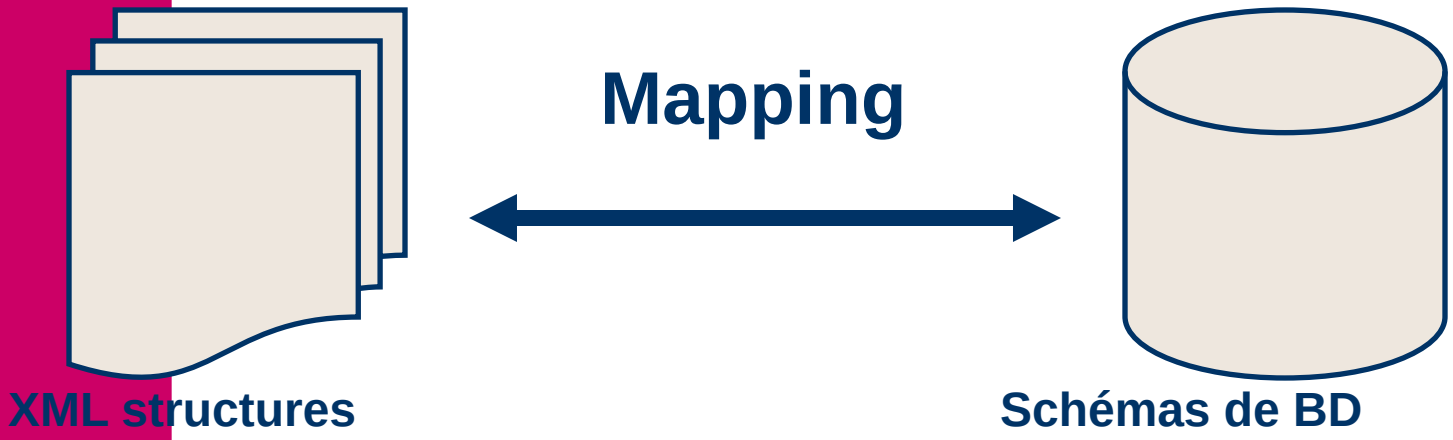
- **Documents centré-documents**



- **La granularité est le document entier**
- **Une application doit recourir au parsing classique par la suite de l'extraction du document**

XML dans des bases de données classiques

- **Documents centré-données**



- **La granularité est généralement l'élément ou l'attribut**
- **L'application peut se baser sur l'interrogation pour manipuler les documents**

Base de données natives XML

- **Terme uniquement « Marketing » proposé par Software AG (Tamino)**

- **Base de donnée native XML :**
  - Doit se baser sur un modèle logique du document,
  - Le stockage et l'interrogation des documents se base sur ce modèle,
  - Le modèle doit au moins considérer les éléments, les attributs et l'ordre des éléments.
  - La mécanique interne de la base de donnée peut être basée sur un modèle relationnel, objet, textuel, etc.

Base de données natives XML

- **Architecture des bases de données native XML**
  - Base de donnée Native XML basé sur un modèle textuel
    - *Les documents XML sont stoqués comme texte.*
    - *Des mécanismes d'optimisation appropriés basés sur des modèles interne divers : relationnel, objet, hiérarchique, etc.*
  - Base de donnée native basé sur un modèle documentaire
    - *Construit un modèle objet du document et le stocke sur la base de ce modèle.*
    - *Comment le modèle est stocké dépend de la base.*

Base de données natives XML

- **Caractéristique d'une BD native XML (1)**
  - Collection de Documents
    - *Définition de collection de document et lancement d'interrogation sur cette collection.*
  - Langages d'interrogation
    - *Tous les bases de données native XML implémentent au moins un langage de requête XML*
    - *XPath, XQL, langages propriétaire, …*
    - *Dans un future proche, l'ensemble des BDs native XML vont supporter le langage Xquery.*
  - Gestion des transactions et du vérouillage
    - *Toutes les BDs native XML supporte la gestion des transactions.*
    - *Généralement le vérouillage se fait sur le document entier.*
    - *Dans le future, le vérouillage sera basé sur les éléments ou les fragments*

Base de données natives XML

- **Caractéristique d'une BD native XML (2)**
  - Application Programming Interfaces : API
    - *Offre généralement des APIs.*
    - *La majorité sont des APIs propriétaires.*
    - *Possibilité d'interrogation et de réponse via HTTP.*
  - Restitution
    - *Restitution des documents stockés sans perte d'information.*
    - *Cette fonction est implémentée généralement au niveau des éléments, attributs, éléments textuels et ordre des éléments.*
  - Intégration de données distants
    - *Quelques bases de données natives XML permettent d'intégrer des données distants d'une autres base.*

BD native XML : Tamino

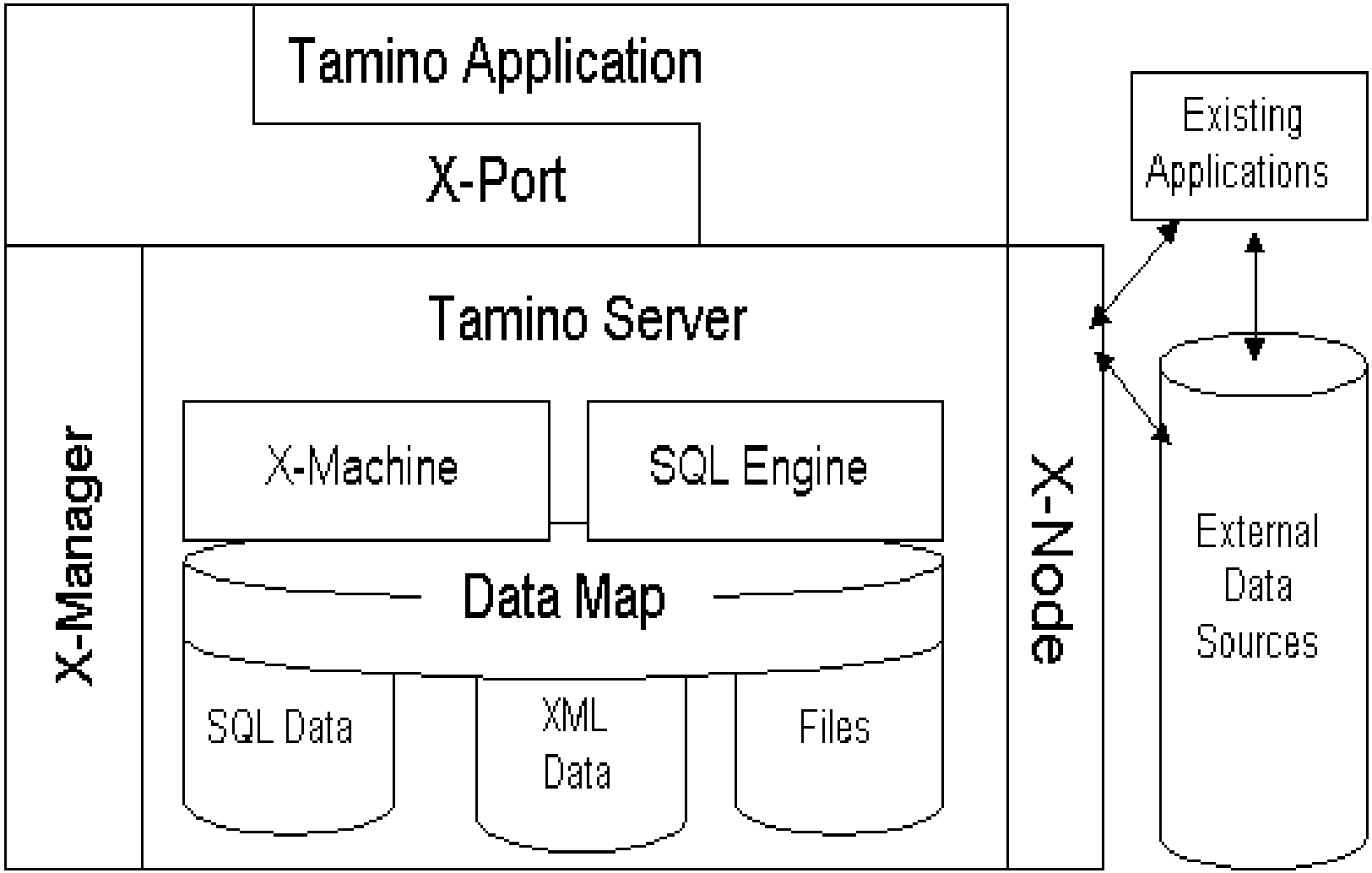
- **Base de donnée XML orienté vers les applications eBusiness**
  - Transaction Architecture for the Management of INternet Objects

- **Tamino est une base de données internet permettant de gérer les documents XML de manière native sans passer par des modèle intermédiaires**

Tamino - Caractéristiques

- **Caractéristiques:**
  - Format de stockage interne spécifique permettant un accès efficaces aux documents
  - Stockage et recherche de données via HTTP et TCP/IP
  - Connexion avec les serveurs Web standards
  - Intégration de données à partir de BD existantes
  - Interface avec des outils XML : Schema, Authoring, etc.
  - Basée sur des standards XML

Tamino - Architecture



Interrogation des documents XML

SGMLQL

StruQL

HyOQL

W3QL

Lorel

Before XML

XMLQL ….. XQL Xpath

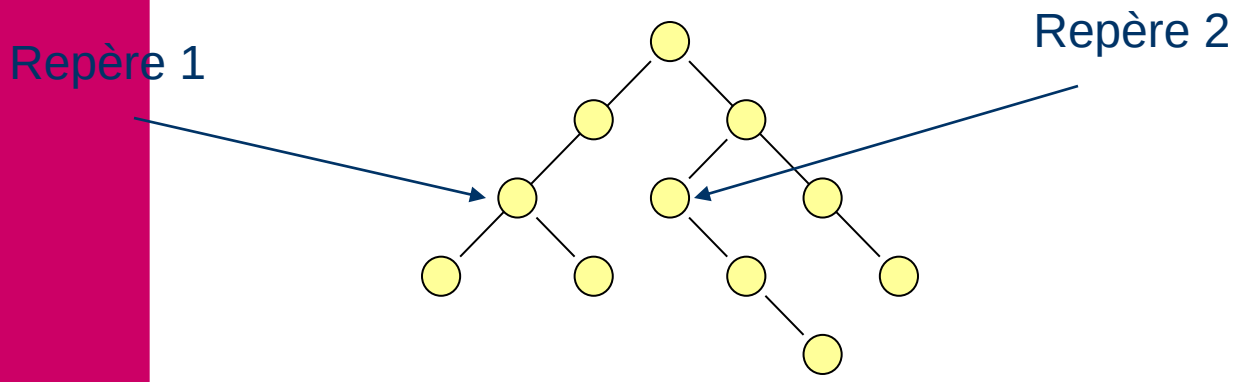
XML

Xquery **W3C standard**

XPath

Problème de repérage

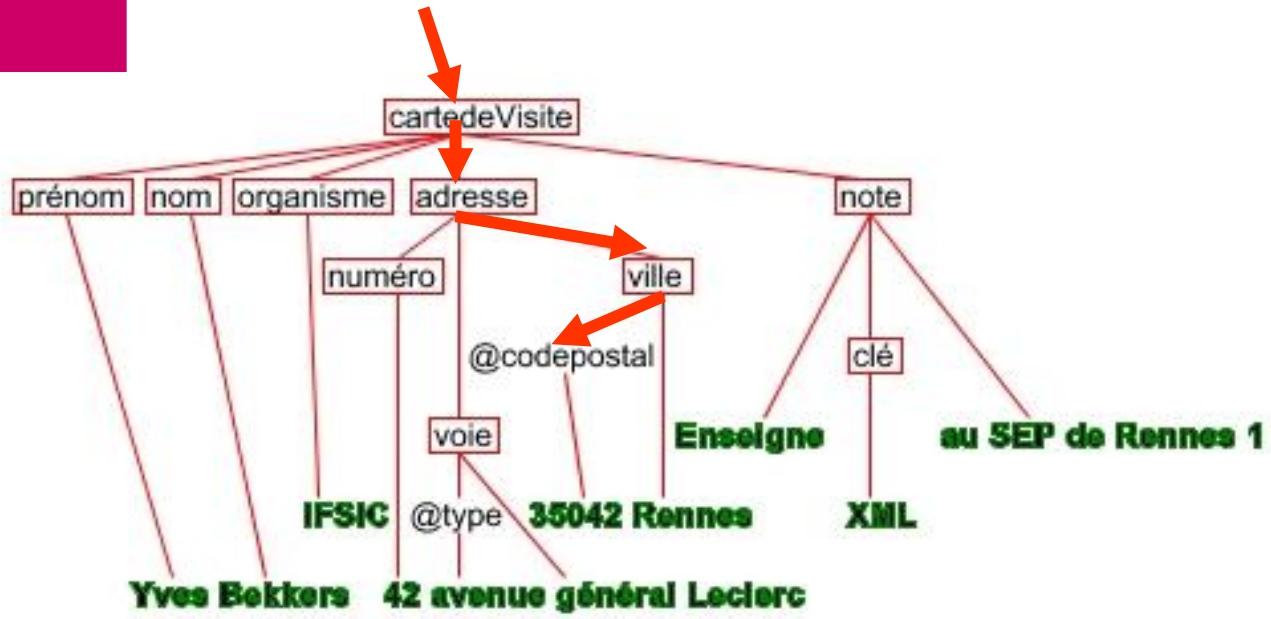
- **Au sein des feuilles de style XSLT**
  - on doit désigner les branches de l'arbre d'entrée (document source)



- Solution : « *expression de chemin* »

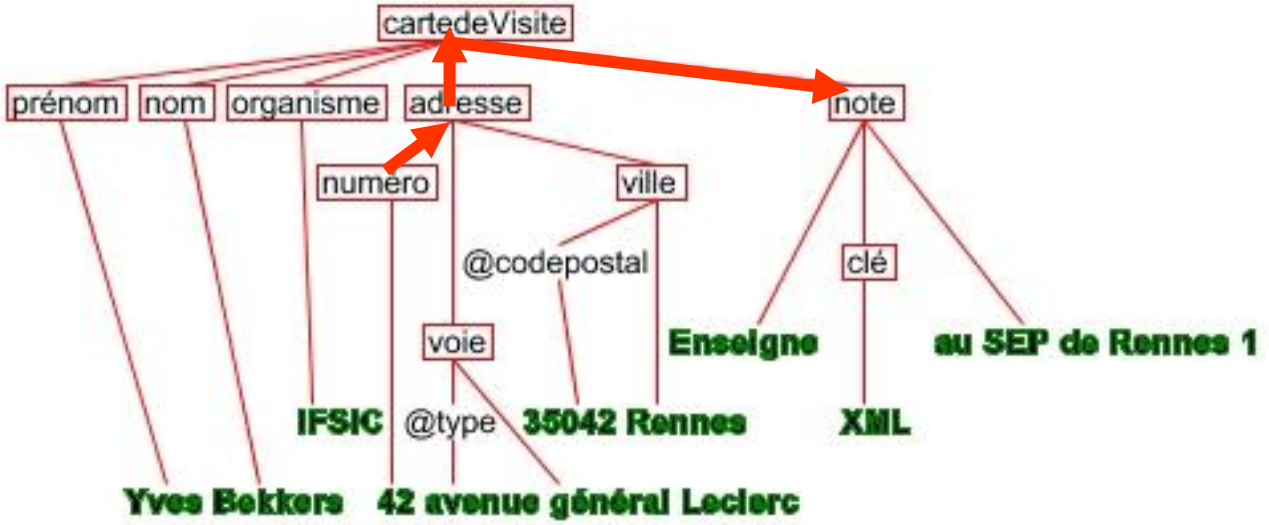
Chemin absolu

- Codepostal de la carte de visite :
- /cartedeVisite/adresse/ville/@codepostal



Chemin relatif

- A partir du numéro, la note :
- ../../note



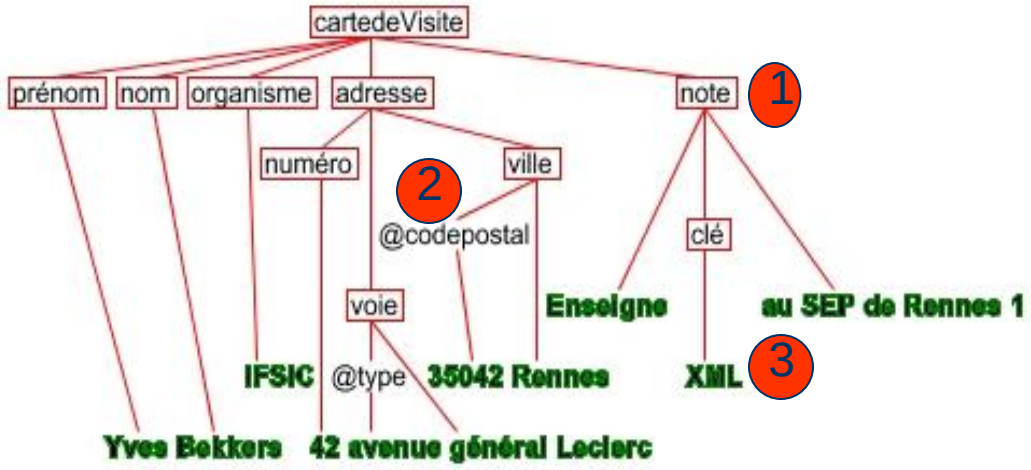
Racine d'un document

- **La racine d'un document est au dessus de l'élément racine du document, elle contient**
  - des commentaires éventuels
  - des instructions de traitements éventuelles
  - un et un seul élément (l'élément racine)

`/`	Racine du document
`/html`	Elément racine du document si c'est html
`/*`	Elément racine du document

Type de nœud d'un arbre XPath

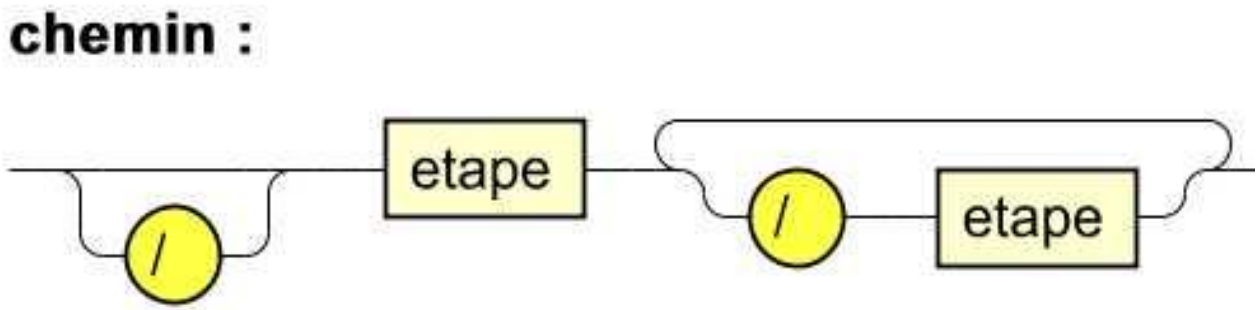
- **1. Nœud « élément » /carteDeVisite/note**
- **2. Pseudo-nœud « attribut »**
  - /carteDeVisite/adresse/ville/attribute::codepostal
  - /carteDeVisite/adresse/ville/@codepostal
- **3. Nœud « texte »**
  - /carteDeVisite/note/clé/text()



Expression de chemin

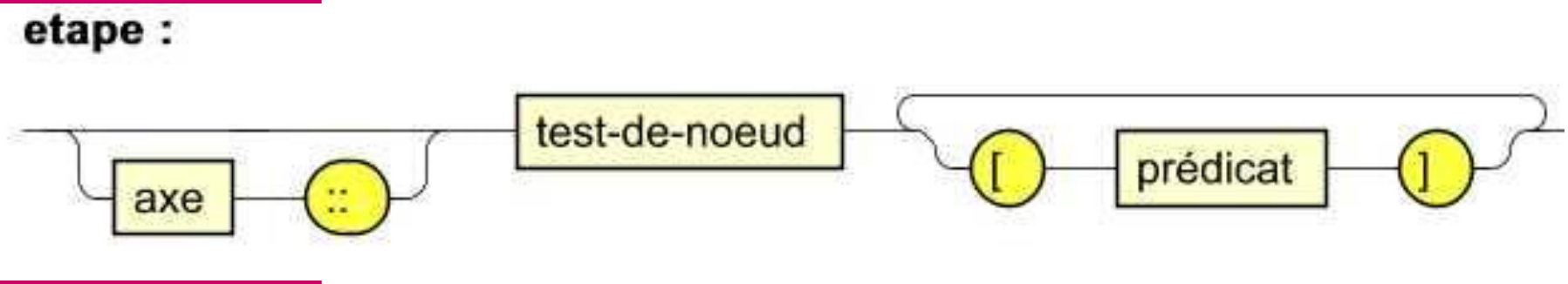
- **séquence d'étapes**

- **Chemin absolu**
  - /étape1/étape2/étape3/...
- **Chemin relatif**
  - étape1/étape2/étape3/...



Étape

- **Une étape est composée de trois parties**
  - axe de déplacement (optionnel)
  - test de nœud (obligatoire)
  - prédicat (optionnel)



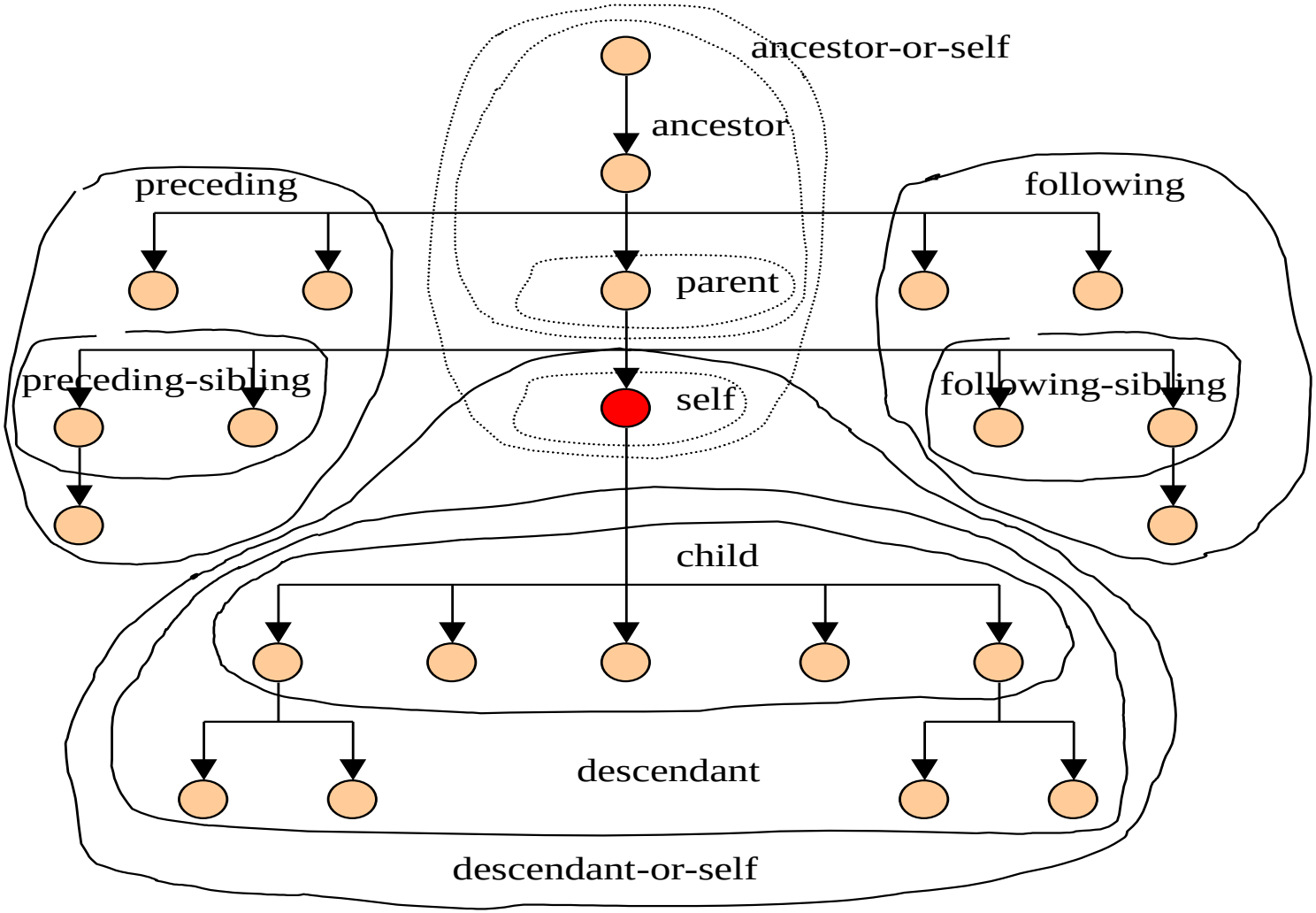
Syntaxe d'une section

- **Exemple**

  - child::text()[position()=1]
  - carteDeVisite[nom='Bekkers']
  - comment()
  - ancestor::nom
  - attribut::codepostal

- **La partie test de nœud est obligatoire**
- **La partie axe se termine par ::**
- **La partie prédicat est délimitée par [...]**

Axes de déplacement



+ attribute

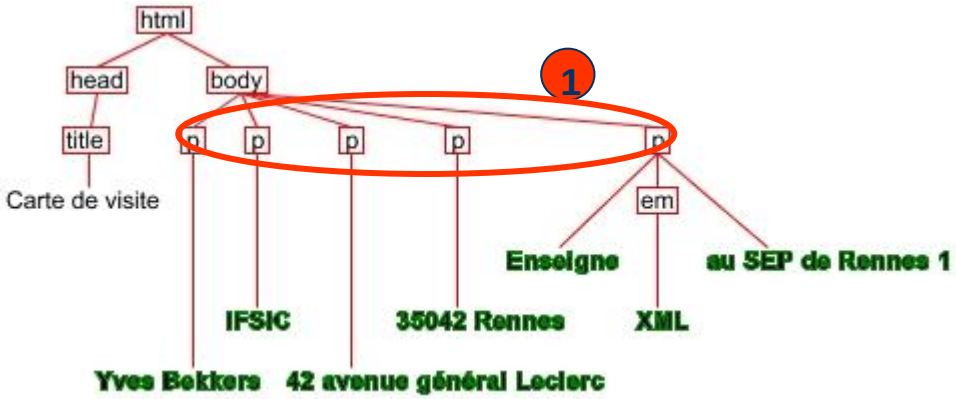
Partie test

- **Un test c'est :**
  - Un nom d'élément, exemple : ville, prénom
  - * : n'importe quel élément
  - text()
  - comment()
  - processing-instruction()
  - processing-instruction(nom)
  - node() identique à l'union de
    **, text(), comment() et de processing-instruction()*

Filtrage à l'aide du prédicat

Une expression Xpath s'évalue en un ensemble de nœuds

1 `/html/body/p`



Filtrage par la position

```
/html/body/p[position()=1]
/html/body/p[position()=last()]
```

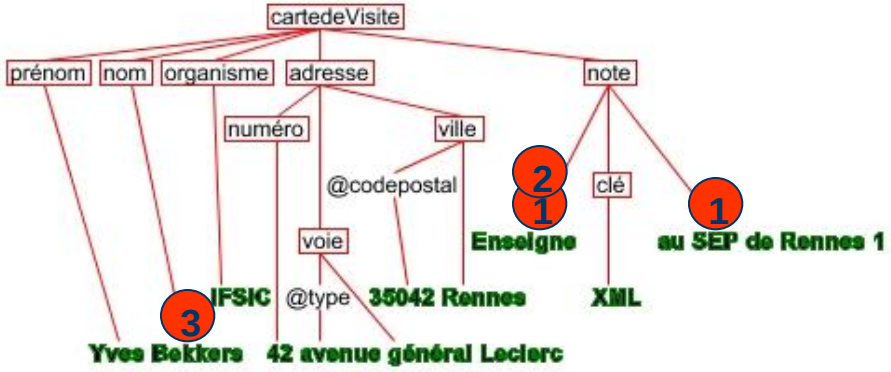
Filtrage par le contenu

```
/html/body/p[em]
```

Quelques raccourci

```
//    /descendant-or-self::node()
@id   attribut::id
.     self::node()
..    parent::node()
note  child::note
*     child::*
```

Quelques exemples



1 • //note/text()

2 • //note/text()[position()=1]

3 • //nom[.='Bekkers']

Quelques exemples

`//figure`	Tous les éléments "figure" du document
`figure`	Tous les éléments "figure" fils du nd courant
`./figure`	Tous les éléments "figure" fils du nd courant
`*/figure`	Tous les éléments "figure" petit fils du nd courant
`adresse/@rue`	Les attributs "rue" des éléments "adresse" fils du nd courant
`adresse/text()`	Tous les textes situés directement sous l'élément "adresse"

Quelques exemples - suite

`*[last()]`	Le dernier fils du nœud courant
`figure[lg]`	Tous les éléments "figure" fils du nd courant, pourvu qu'ils aient un fils "lg"
`ancestor::tst`	L'élément "tst" englobant le plus intérieur
`./@*`	Tous les attributs de l'élément courant
`XXX[@WIDTH and not(@WIDTH="20")]`	Les éléments XXX fils du nd courant pourvu qu'ils aient un attribut "WIDTH" avec une valeur différente de 20

Constantes

- **Chaînes**
  - 'Paris'
  - "That's rubbish"
  - 'He said "Boo"'
- **Valeurs numériques**
  - 12, 3.05, - 5.25

Prédicat - exemples

- [@codepostal='35700']
- [.='Bekkers']
- [nom='Bekkers']
- [position()=last()-1]
- [not(position()=1)]

Opérateurs

- **Expression numériques :**
  - +, -, *, div, mod
  - position(), last(), count(nds),
  - string-length(expr)
- **Expression booléenne :**
  - or, and, not(...), false(), true(),
  - boolean(...),
  - =, !=, <, <=, >= (à écrire < et >)
- **Expression nœud**
  - id(chaîne)

Opérateurs (suite)

- **Expressions chaîne**
  - string(exp)
  - concat(exp1, exp2, …)
  - substring(expr,start),
  - substring(expr,start,length)
  - substring-before(expr,expr)
  - substring-after(expr,expr)

haute école
neuchâtel berne jura
arc ingénierie
saint-imier le locle delémont

XQuery

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale

La base

- **Proposé par IBM , MS, AT&T, Data Direct, …**
- **Langage fonctionnel type CAML**
- **Forme de requête élémentaire**
  - FOR $<var> in <forest> [, $<var> in <forest>]+ //itération
  - LET $<var> := <subtree> // assignation
  - WHERE <condition> // élagage
  - RETURN <result> // construction
- **Les forêts sont sélectionnées par des Xpath (document ou collection)**
- **Le résultat est une forêt (un ou plusieurs arbres)**

Exemple :

```
<Guide Version= "2.0">

 <Restaurant type="francais"
    categorie="***">
    <Nom>Le Moulin</Nom>
    <Adresse> <Rue>des
    Vignes</Rue>
                <Ville>Mougins</Ville>
    </Adresse>
    <Manager>Dupuis</Manager>
 </Restaurant>

 <Restaurant type="francais"
    categorie ="**">
    <Nom>La Licorne</Nom>
    <Adresse><Rue>Des
    Moines</Rue>
                <Ville>Paris</Ville>
    </Adresse>
    <Téléphone>0148253278</Téléph
    one>
    <Manager>Dupuis</Manager>
 </Restaurant>

 <Bar type = "anglais">
    <Nom>Rose and Crown</Nom>
 </Bar>

</Guide>
```

Exemple 1 : XPath

- **(Q1) Noms de tous les restaurants :**
  - collection(“Restaurants”)/Restaurant/Nom/text()
  - collection(“Restaurants”)/Restaurant/Nom

Exemple 2 et 3 : XPath +

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

- **Accès via indice 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>
```

Exemple 5 : Jointure

- **Lister le nom des Restaurants avec téléphone dans la rue de l'Hôtel Lutecia:**

```
for   $R in collection("Restaurants")/Restaurant,
                  $H in collection("Hotels")/Hotel
  where  $H//Rue = $R//Rue
  and $H//Nom = "Le  Lutecia"
  return
                  <Result>
                           {$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 en Where

- **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 := collection("Restaurants")//
                [Restaurant = $r]//Menu/@Prix
   return
   <resultat>
      {$r/Nom}
     <avgPrix>{AVG($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 $r in unordered(collection("Restaurants")//Restaurant)
where (contains($r/Comments, "Excellent")
or contains($r/Comments, "Good"))
and $r/Adresse/Ville = "Paris"
return {$r/Nom}
orderby ($r/Nom descending)
```

Exemple 12 : Multi-requêtes

- **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")//Restaurant
where every $p in $r/Menu/@Prix satisfies ($p>100)
return {$r}}
</Very_Expensive_Restaurants>
<Very_Inexpensive_Restaurants>
<Title>List of very inexpensive restaurants</Title>
{for $r in collection("Restaurants")//Restaurant
where some $p in $r/Menu/@Prix satisfies ($p<10)
return {$r}}
<Date>{date()}</Date>
</Very_Inexpensive_Restaurants>
</XML_document>
```

Exemple 13 : String

- **Trouver les livres dans lequel le nom d'un élément se termine par "or" et le même élément contient la chaîne "Suciu" quelque part. Pour chaque tel livre, retourner le titre et l'élément qualifiant.**

```
for $b in document("document")//book
let   $e := $b/*[contains(string(.), "Suciu")
                    and ends-with(local-name(.), "or")]
where exists($e)
return <book> { $b/title } { $e } </book>
```

Fonctionnalités XQuery Text

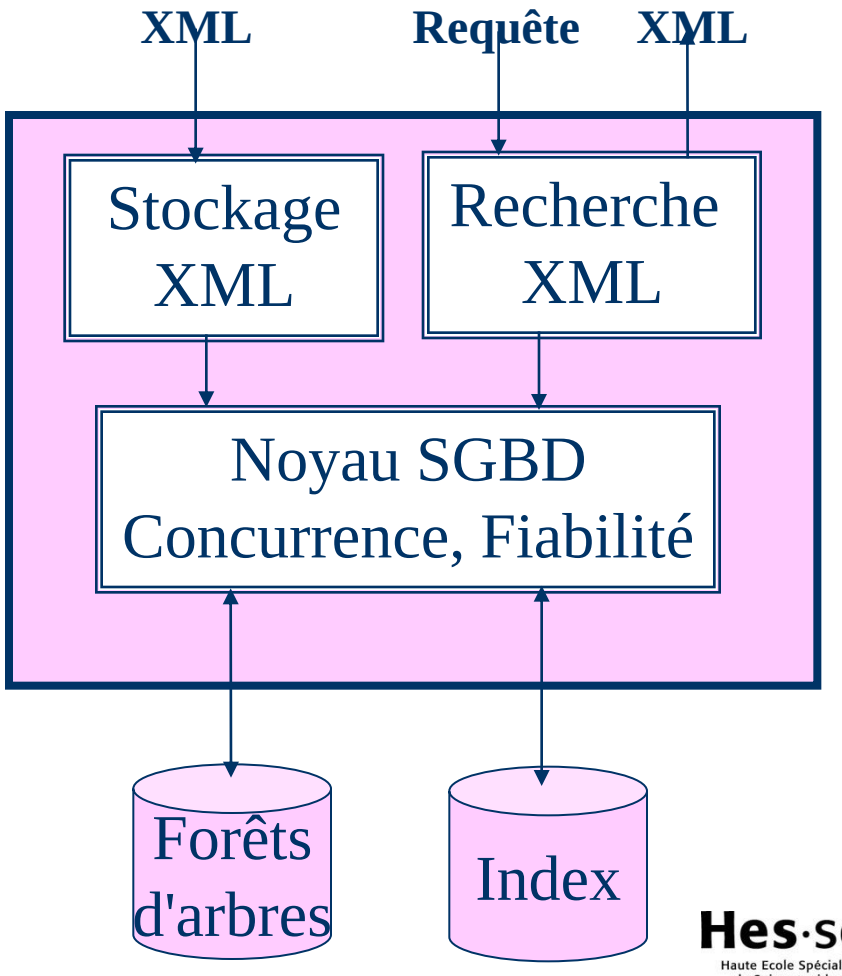
- Recherche sur mot-clés
- Recherche de phrase
- Support des mots de laiaison
- Recherche sur préfix, suffix, infix
- Normalisation des mots, accents, capitales, …
- Recherche par proximité (unité = mots)
- Spécification de l'ordre des mots
- Combinaison logic avec AND, OR , NOT
- Recherche par similarité
- Tri des résultats par pertinence

4. Aperçu des produits

- **Systèmes natifs**
  - Technique spécialisée de stockage et recherche
  - Extension des techniques documentaires à l'élément
- **SGBD relationnels étendus**
  - Séparation des éléments et du graphe
  - Mapping en tables
- **SGBD objet adapté**
  - Utilisation d'une structuration objet (DOM)
  - Un produit : Excelon (Object Store)
    - *Racheter par Progress Software*

4.1 SGBD Natif XML

- **SGBD**
  - conçu pour XML,
  - stockant les documents en entiers sans les décomposer en éléments,
  - utilisant de techniques d'indexation d'arbres spécifiques.



Indexation Plein Texte

- **Utilisation d'un thésaurus au chargement**
  - ensemble de termes reliés
  - liste des mots importants
  - synonymes et préférés
  - spécialisations, traductions
  - Standards ISO 2788 et ANSI Z39.19
- **Stemisation (racine) ou lemmisation (préféré)**
- **Listes inverses**
  - fichiers de mots significatifs
  - pour chaque mot, adresse document (élément+offset)

Principaux produits

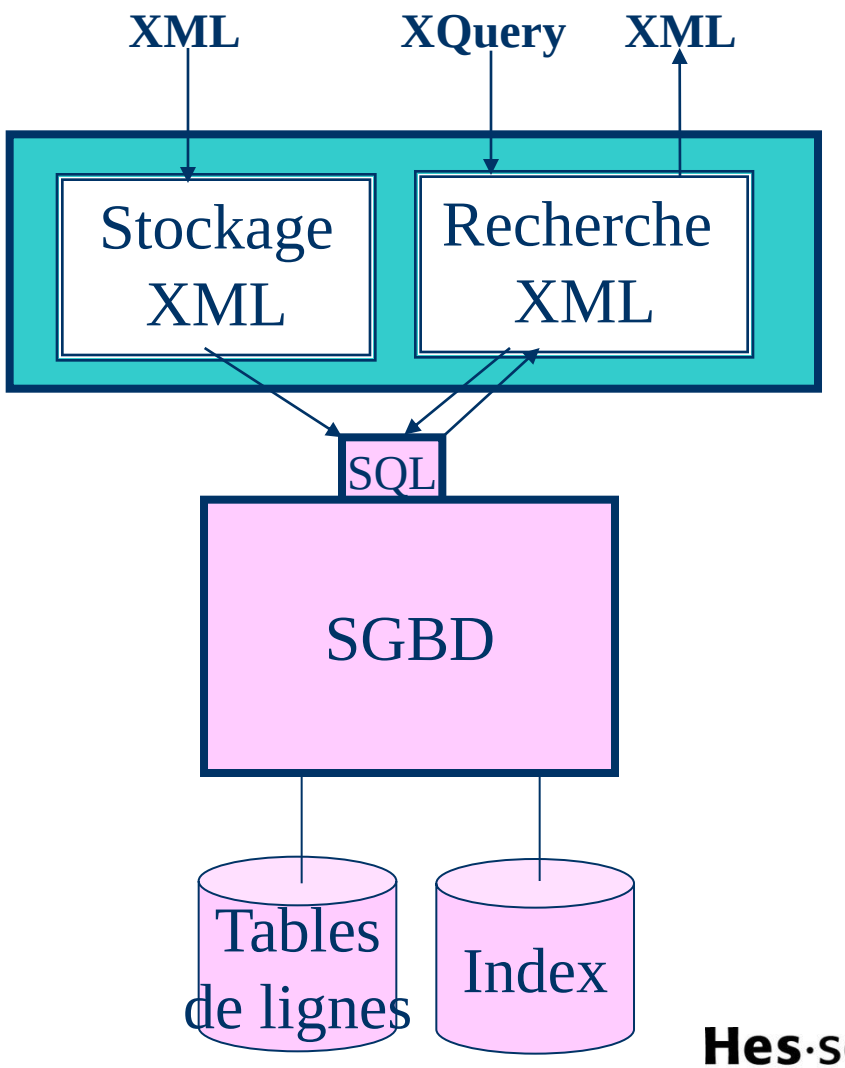
- **De multiples start-up**
  - Software A.G. Tamino
    http://www.softwareag.com/
  - X-Hive/Db http://www.x-hive.com/
  - Coherity
    http://www.coherity.com/
  - IXIA soft http://www.ixiasoft.com/
  - XML Global http://www.xmlglobal.com/
  - NeoCore http://www.neocore.com/
  - Xyleme http://www.xyleme.com/
- **Intégration comme type spécialisé à SGBD OR**
  - DB2 XML Extender
    - *Stockage en BLOB, Fonctins d'accès Xpath intégrées à SQL/XML*
  - Oracle 9.i XML DB
    - *Support XMLType, Interrogation via SQL/XML*

Xyleme

- **Efficient XML warehouse**
- **Distributed architecture**
  - Cluster of PCs
  - Communicating with Corba
- **Developed with Linux and C++**
- **Currently support XyQL**
  - Extended OQL with path expressions
  - Efficient full text search in elements

4.2 Mapping SGBDR

- **Composant logiciel au-dessus d'un SGBDR assurant:**
  - le stockage et l'interrogation de documents XML
  - en transformant le XML en tables
  - et les tables en XML



Oracle : XSU

- **Stockage et publication**
  - Mapping de XML plat sur une table
  - Mapping de XML imbriqué en tables imbriquées
  - Balises spéciales <rowset> et <row>
  - Commandes PutXml et GetXml
  - Passage par iFS et XSL possible

- **Interrogation**
  - Servlet XSQL
    - *document XML avec requêtes SQL*
    - *transformation naïve du résultat des requêtes*

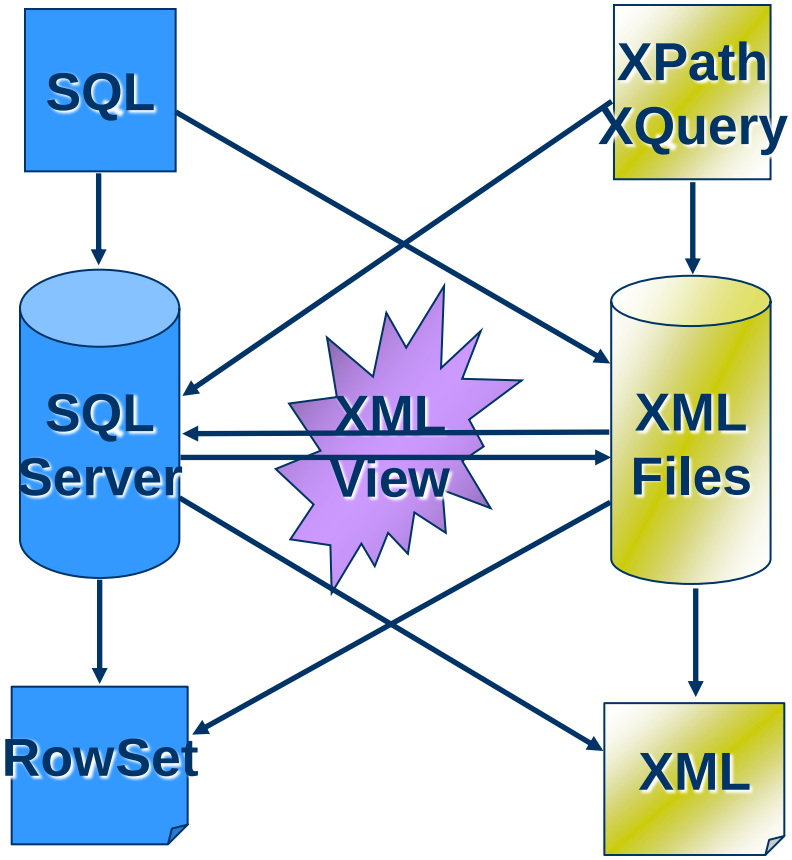
Oracle : XML DB

- **Intègre XSU (mapping) et type natif XMLType**
- **Interrogation via SQL étendu (SQL/XML) avec des fonctions**

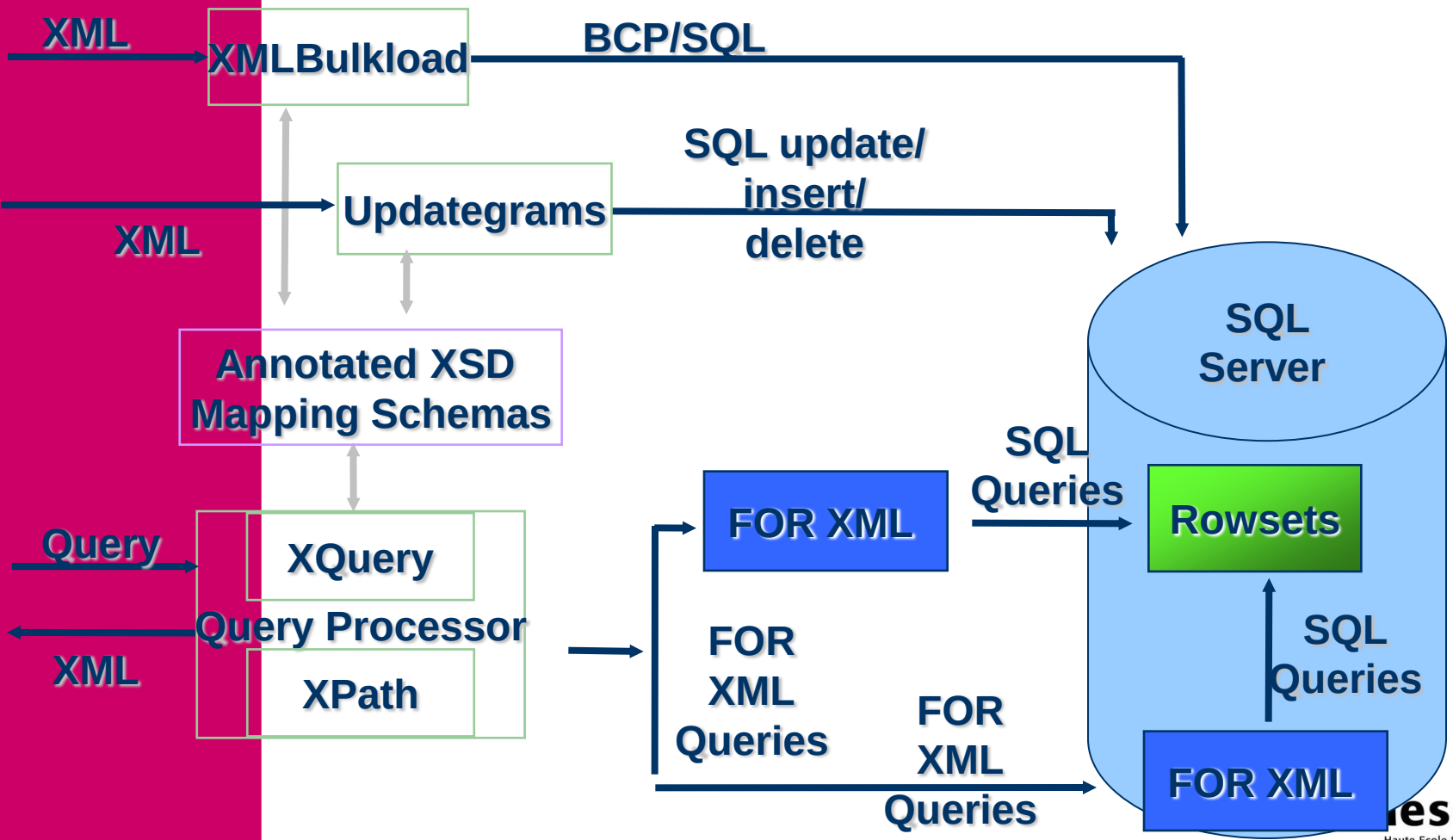
Fonction	Rôle
XMLAgg	prend en argument une collection de fragments et retourne un document XML agrégé ;
XMLConcat	reçoit en argument une série d'instances XMLType correspondant aux valeurs d'une colonne pour les lignes d'une table et retourne les instances concaténées ;
XMLElement	prend en argument un nom d'élément, une collection d'attributs optionnels, un contenu d'élément et retourne une instance XMLType ;
XMLForest	convertit la suite de ses argument en XML et retourne un fragment XML concaténation des arguments convertis ;
XMLColAttVal	converti une valeur de colonne en XML ;
XMLSequence	transforme une suite de lignes référencées par un curseur en séquence XML ;
XMLTransform	applique une feuille de style XSL à une instance XMLType et retourne une instance XMLType ;
ExtractValue.	reçoit en argument une instance XMLType et une expression XPath et retourne la valeur scalaire des nœuds sélectionnés
ExtractXML	reçoit en argument une instance XMLType et une expression XPath et retourne une instance XML représentant les nœuds sélectionnés.

Microsoft

- **SQL Server 2000**
  - Mapping de XML sur tables
    - *défini par assistants*
    - *exécuté par procédures stockées*
  - Génération de résultats en XML
    - *par SELECT … FOR XML*
- **Projet Yukon 2003**
  - SGBD natif ?

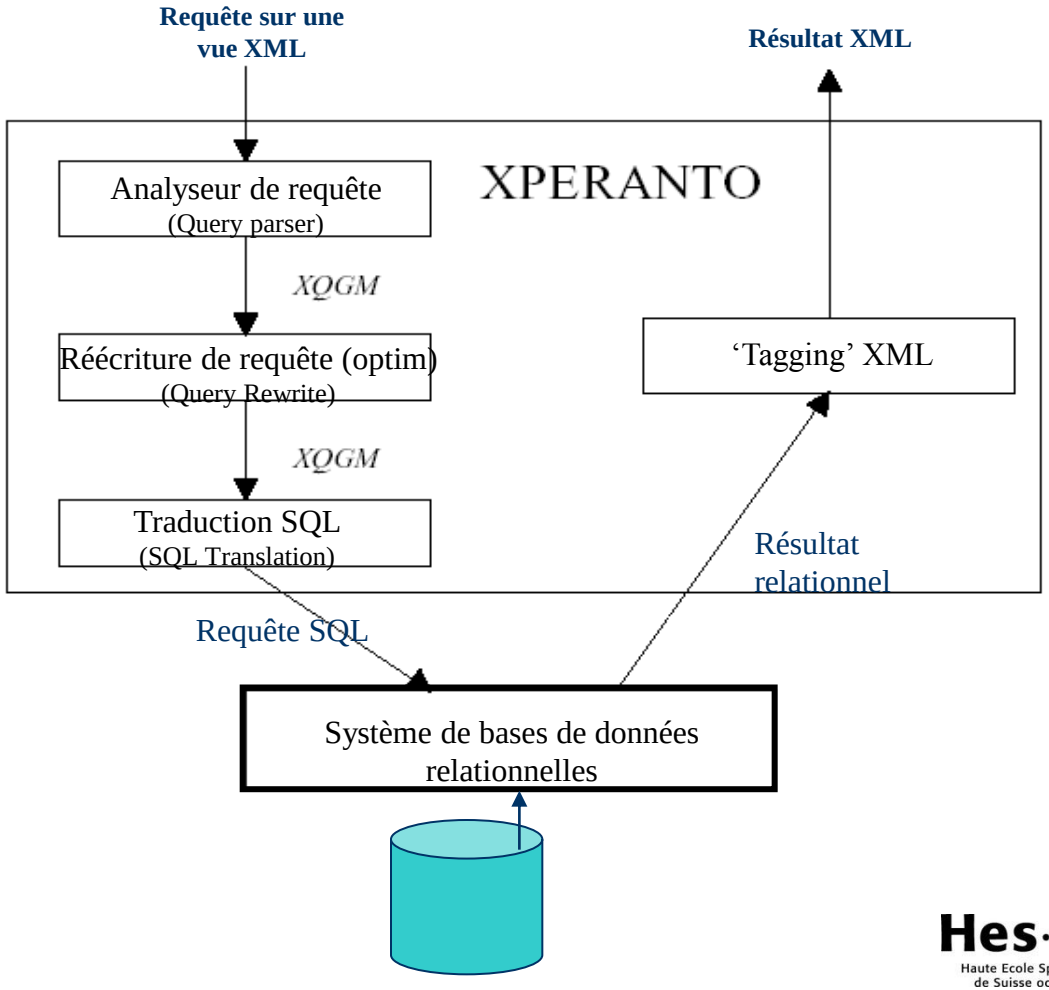


Microsoft : SQL Server XML



IBM XPeranto : Architecture

- **Vues XML au-dessus de SGBDR (DB2)**
- **Traducteur et optimiseur de XQuery en SQL**
- **Intégré à DB2 dans une future version**



XQuark : Open Source XMLizer

- **Extraction XML**
  - via XQuery traduit en SQL
- **Stockage XML en base**
  - Mapping via schema
- **Indépendant du SGBD**
  - MySQL, Oracle,
  - SQLServer, Sybase, …