XQuery
XML Query Language

Sébastien Laborie
Sebastien.Laborie@iutbayonne.univ-pau.fr

Christian Sallaberry
Christian.Sallaberry@univ-pau.fr

Bibliographie

- http://www.w3.org/XML/
- http://www.w3.org/TR/xquery/
- http://www.w3.org/TR/xquery-semantics/
- http://www.w3.org/TR/xpath-full-text-10-use-cases/
- http://www.gnu.org/software/qexo/XQuery-Intro.html
- http://www.w3schools.com/xquery/
- http://bibliotheques.univ-pau.fr/live/livres-electroniques/edition-ENI
- http://www.gchagnon.fr/cours/xml/index.html
- http://www.datypic.com/IntroductionToXQuery.pdf
- http://www.datypic.com/books/xquery/

Xquery : présentation

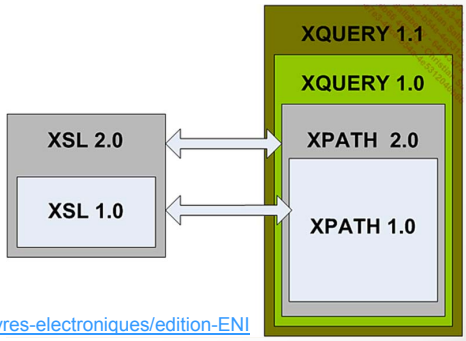
What is XQuery?

- XQuery is ***the*** language for querying XML data
- XQuery for XML is like SQL for databases
- XQuery is built on XPath expressions
- XQuery is supported by all major databases
- XQuery is a W3C Recommendation

http://www.w3schools.com/xquery/

XQuery : présentation

- **XQuery** est un langage de requêtes pour les documents et bases de documents XML
- **XQuery** est souvent comparé :
  - au langage **SQL** des bases de données relationnelles
  - au langage **XSLT** gérant des structures hiérarchisées XML



http://bibliotheques.univ-pau.fr/live/livres-electroniques/edition-ENI

XQuery : présentation

- Langages pour interroger des arbres :
  - **XQuery**, plus pour interroger que pour transformer
  - **XSLT**, plus pour transformer
  - **XQuery** et XSLT utilisent des expressions XPath
- Langages pour interroger des relations :
  - **SQL** , pour interroger deux niveaux : relations & attributs

XQuery : présentation

- Analogie XML et BD

Vue XML	Vue BD
XML	Données
XSLT	HTML & ASP,JSP
XPath	Référence à un champ
XQuery	SQL

Semi-structuré	Structuré
Echange & partage	Stockage
Auto-descriptif	Normes et contraintes SGBDR

XQuery : présentation

- Les données relationnelles sont denses :
  - le schéma existe et est stocké séparément
  - les attributs de chaque instance ont une valeur
  - problème des valeurs nulles
- Ce n'est pas le cas de XML pour qui :
  - le schéma existe ou pas (s'il existe, il est stocké dans le document ou bien séparément)
  - il peut y avoir des éléments vides
  - il peut y avoir des éléments absents

Degré de liberté supérieur pour les documents XML dits semi-structurés

XQuery : présentation

- Les requêtes SQL retournent des relations,
  - ensembles résultats homogènes
- Les requêtes XML retournent des arbres,
  - de type différent
  - de structure complexe
    - //*[couleur="rouge"] *peut retourner une cerise, une voiture, …*

On trouve côte à côte des éléments et des valeurs atomiques ; des transformations structurelles sont supportées

XQuery : présentation

- Les BD XML - relationnel ➲ XML :
  - Traduction en attributs

Vins	nv	cru	mill	degré
	100	Jurançon	1999	12
	200	Madiran	1996	12

```
<Vins>
<Tuple nv='100' cru='Jurançon' mill='1999' degré='12'/>
<Tuple nv='200' cru='Madiran' mill='1996' degré='12'/>
</Vins>
```


XQuery : présentation

- Les BD XML - relationnel ➲ XML :
  - Traduction en éléments

Vins	nv	cru	mill	degré
	100	Jurançon	1999	12
	200	Madiran	1996	12

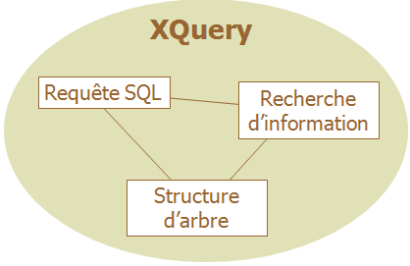
```
<Vins>
<Tuple> <nv>100</nv> <cru>Jurançon</cru>
<mill>1999</mill> <degré>12</degré> </Tuple>
<Tuple> <nv>200</nv> <cru>Madiran</cru>
<mill>1996</mill> <degré>12</degré> </Tuple>
</Vins>
```

XQuery : présentation

- Les BD XML – modèle :
  - BD relationnelle ⇔ Collection de relations
  - Relation ⇔ Collection tuples
  - BD XML ⇔ Collection de forêts
  - Forêt XML ⇔ Collection d'arbres
  - Arbre XML ⇔ Document XML

XQuery : présentation

- Les extensions par rapport à SQL sont :
  - extraction de sous-arbres : XPath
  - sélection de sous-arbres (avec prédicats) : XPath
  - variables d'itération sur des collections d'arbres : XPath
  - jointures pour combinaisons d'arbres : XQuery
  - réordonnancement d'arbres : XQuery
  - définition et intégration de fonctions dans des requêtes : XQuery
  - transformation de document (arbres) : XSLT



XQuery : expression

- Syntaxe :
  - **let** : permet l'affectation de valeurs à une variable
  - **for** : itération sur une liste de partie de document XML
  - **return** : forme de l'expression à retourner
  - **where** : clause de restriction de la requête Xquery
  - **order by** : tri des résultats
  - If then else : expression conditionnelle

XQuery : expression

- Création d'éléments :
  - Création de l'élément XML <valeur> contenant la chaîne "ok "

    ```
    <valeur>ok</valeur>
    ```
  - Création de l'élément XML <valeur> qui a pour contenu la valeur de la variable i

    ```
    let $i := 1 return <valeur>{$i}</valeur>
    ```

    Un constructeur d'élément XML peut contenir tout type d'expression XQuery. L'expression doit alors être placée entre accolades { }
  - Création de l'élément <resultat> contenant l'ensemble des éléments <titre> de l'ensemble des recettes du document

    ```
    let $i := /collection/recette
    return <resultat>{$i/titre}</resultat>
    ```

    Lorsque la variable est une séquence, il y a autant d'éléments générés que de valeurs dans la séquence d'origine

    ```
    <resultat>
      <titre>Poivrée de steak d'autruche sur purée de céleri</titre>
      <titre>Salade de chèvres chauds</titre>
    </resultat>
    ```

XQuery : expression

- Création d'éléments :
  - Création, pour chacune des recettes du document, d'un élément <resultat> contenant l'élément <titre>

```
for $i in /collection/recette
return <resultat>{$i/titre}</resultat>
```

La différence avec l'exemple précédent est la structure retournée. La fermeture de la balise <resultat> après chaque balise <titre> ou une seule fois en fin de document ne donne pas la même signification au document XML

```
<resultat>
  <titre>Poivrée de steak d'autruche sur purée de céleri</titre>
</resultat>
<resultat>
  <titre>Salade de chèvres chauds</titre>
</resultat>
```


XQuery : expression

- Création d'éléments :
  - Intégration d'une séquence dans le constructeur d'élément

```
<recette>
{for $i in /collection/recette
  return <resultat>{$i/titre, $i/commentaire}</resultat>}
</recette>
```

```
<recette>
    <resultat>
      <titre>Poivrée de steak d'autruche sur purée de céleri</titre>
      <commentaire auteur="Laborie">Très bon !</commentaire>
    </resultat>
    <resultat>
      <titre>Salade de chèvres chauds</titre>
      <commentaire auteur="Laborie">Un classique !</commentaire>
    </resultat>
</recette>
```

sous eXist :

```
<recette>
{
for $i in collection("Recettes")/collection/recette
return <resultat>{$i/titre, $i/commentaire}</resultat>
}
</recette>
```

XQuery : expression

- Création d'éléments :
  - Calcul de la valeur d'un attribut par un constructeur d'élément

```
for $i in /collection/recette
return <recette image="{$i/image/@src}">{$i/titre}</recette>
```

```
<recette image="http://www.mesrecettes.com/imgRecette1.jpg">
  <titre>Poivrée de steak d'autruche sur purée de céleri</titre>
</recette>
<recette image="http://www.mesrecettes.com/imgRecette2.jpg">
  <titre>Salade de chèvres chauds</titre>
</recette>
```


XQuery : expression

- Création d'éléments :
  - Utilisation du mot-clé "element" pour évaluer le nom de l'élément créé ("attribut" pour évaluer le nom de l'attribut)

```
let $i := "message", $j := "ok"
return element {$i}{$j}
```

```
<message>ok</message>
```

```
let $i := "val", $j := "ok"
return element message {attribute {$i} {$j}, "bon"}
```

```
<message val="ok">bon</message>
```

Ces exemples montrent comment générer à partir de valeurs de variables des séquences ou des attributs XML

XQuery : expression

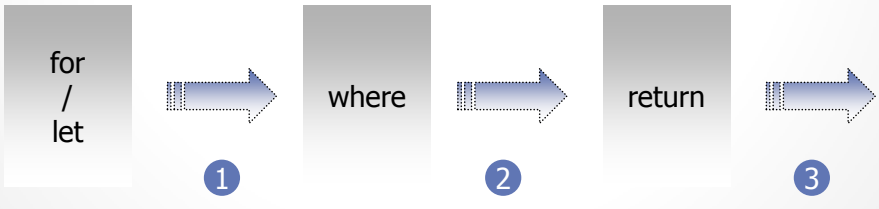
- Expression FLWOR :

for $<var_f1> in <forest_1> [, $<var_f2> in <forest_2>] ... //itération
let $<var_s1> := <subtree_1> [, $<var_s2> := <subtree_2 >] ... //assignation
where <condition> //élagage
order by <criterion> //trie
return <result> //construction

XQuery : expression

- Expression FLWOR :

for $<var_f1> in <forest_1> [, $<var_f2> in <forest_2>] ... //itération
let $<var_s1> := <subtree_1> [, $<var_s2> := <subtree_2 >] ... //assignation
where <condition> //élagage
order by <criterion> //trie
return <result> //construction



XQuery : expression

- Expression FLWOR : différence entre F et L
  - for $i in (2,3)
    return <res> {2 * $i} </res>
    <res>4</res>
    <res>6</res>
  - for $i in (2,3)
    return <res> {2 * 1} </res>
    <res>2</res>
    <res>2</res>
  - let $i := (2,3)
    return <res> {2 * 1} </res>
    <res>2</res>
  - let $i := (2,3,4)
    return <res> {count($i)} </res> *(≠ for $i in (2,3,4) return <res> {count($i)} </res>)*
    <res>3</res>

XQuery : expression

- Expression FLWOR : sélection d'éléments
  - for $i in /collection/recette
    return $i/titre
    <titre>Poivrée de steak d'autruche sur purée de céleri</titre>
    <titre>Salade de chèvres chauds</titre>

XQuery : expression

- Expression FLWOR : restriction
  - for $i in /collection/recette
    where $i/@categorie="entree"
    return $i/titre

    <titre>Salade de chèvres chauds</titre>

  - for $i in /collection/recette
    where $i/@prix<15
    return $i/titre

    <titre>Salade de chèvres chauds</titre>

XQuery : expression

- Expression FLWOR : tri
  - for $i in /collection/recette
    order by $i/titre ascending
    return $i/titre

    <titre>Foie frais aux pommes</titre>
    <titre>Poivrée de steak d'autruche sur purée de céleri</titre>
    <titre>Salade de chèvres chauds</titre>
  - for $i in /collection/recette
    where $i/ingredient/@nom = "salade"
    order by $i/@prix ascending
    return $i/titre

    <titre>Foie frais aux pommes</titre>
    <titre>Salade de chèvres chauds</titre>

XQuery : expression

- Expression FLWOR : fonction SI
  - for $i in /collection/recette
    return <recette> {$i/titre,
    element
    {if ($i/ingredient/@nom = "armagnac") then "avec_alcool" else "sans_alcool"}
    {$i/ingredient}} </recette>

```
<recette>
  <titre>Poivrée de steak d'autruche sur purée de céleri</titre>
  <sans_alcool>
    <ingredient nom="céleri" qte="500" unite="g"/>
    <ingredient nom="pommes de terre" qte="1" unite="kg"/>
    <ingredient nom="sel fin"/>
  </sans_alcool>
</recette>
...
<recette>
  <titre>Foie frais aux pommes</titre>
  <avec_alcool>
    <ingredient nom="foie" qte="300" unite="g"/>
    <ingredient nom="armagnac" qte="5" unite="cl"/>
    <ingredient nom="pommes" qte="3"/>
    <ingredient nom="salade" qte="1"/>
  </avec_alcool>
</recette>
```


XQuery : expression

- Expression FLWOR : fonctions prédéfinies (valides XPath)
  - let $R := doc("recettes.xml")//recette
    return
    <NombreRecettes>
    {count($R)}
    </NombreRecettes>

```
<NombreRecettes>3</NombreRecettes>
```

XQuery : expression

- Expression FLWOR : fonctions prédéfinies
  - let $a := //commentaire
    for $d in distinct-values($a/@auteur)
    return <auteur> {$d} </auteur>

    <auteur>Laborie</auteur>
    <auteur>Sallaberry</auteur>

XQuery : expression

- Expression FLWOR : fonctions prédéfinies
  - for $d in doc("recettes.xml")//recette
    return <res> {$d/titre}{data($d/preparation)} </res>

    <res>
     <titre>Poivrée de steak d'autruche sur purée de céleri</titre>
       Peler le céleri et les pommes de terre.
       Faire cuire dans l'eau bouillante.
       Arroser du jus de viande
    </res>
    <res> …

XQuery : expression

- Expression FLWOR : fonctions personnalisées
  - declare function local:quantiteCommentaires($l as node()) as xs:integer
    {count($l/commentaire)};
    for $i in //recette
    return <res> {$i/titre} <com>{local:quantiteCommentaires ($i)} </com> </res>

```
<res>
        <titre>Poivrée de steak d'autruche sur purée de céleri</titre>
        <com>2</com>
</res>
<res>
        <titre>Salade de chèvres chauds</titre>
        <com>1</com>
</res>
<res>
        <titre>Foie frais aux pommes</titre>
        <com>1</com>
</res>
```


XQuery : expression

- Expression FLWOR : quantificateur universel
  - let $seuil := 40
    return
    <message>
    {if (**every** $i in //recette **satisfies** $i/@prix > $seuil)
    then "Trop cher" else "Pas cher"}
    </message>

```
<message>Pas cher</message>
```

XQuery : expression

- Expression FLWOR : quantificateur universel
  - for $i in /collection/recette
    where **every** $c in $i/commentaire **satisfies** contains($c, "bon")
    return $i/titre

    <titre>Poivrée de steak d'autruche sur purée de céleri</titre>

XQuery : expression

- Expression FLWOR : quantificateur existentiel
  - let $seuil := 15
    return
     <message>
      {if (**some** $i in //recette **satisfies** $i/@prix > $seuil)
       then "Des réponses" else "Pas de réponse"}
     </message>

    <message>Des réponses</message>

XQuery : expression

- Expression FLWOR : quantificateur existentiel
  - for $i in /collection/recette
    where **some** $c in $i/commentaire **satisfies**
    $c/@auteur="Laborie"
    return $i/titre

    <titre>Poivrée de steak d'autruche sur purée de céleri</titre>
    <titre>Salade de chèvres chauds</titre>

XQuery : expression

- Cas particulier de la jointure : exemple gestion commerciale

```
for $p in doc("gestionCommerciale-v2.xml")//produit,
   $cp in doc("gestionCommerciale-v2.xml")//commande/detail/produit
where $p/designation = "Bâton de colle" and $cp/@ref = $p/@id
return <res> {$p/designation}
   <com>
   <numCom>{$cp/../../@id}</numCom>
   </com>
   </res>

for $p in doc("gestionCommerciale-v2.xml")//produit
where $p/designation = "Bâton de colle"
return <res> {$p/designation}
   <com>
   {for $cp in doc("gestionCommerciale-v2.xml")//commande/detail/produit
   where $cp/@ref = $p/@id
   return <numCom> {data($cp/../../@id)} </numCom> }
   </com>
   </res>
```

XQuery : expression

- Expression FLWOR : rappel

Nom	Notation	Rôle
Assignation et construction	let $var := expr	Construction d'un élément XML et assignation à une variable $var
Projection	$var / xpath	Extraction de séquences d'éléments ou d'attributs d'un arbre XML à l'aide d'une expression de chemin XPath
Accès données	$var / xpath / data()	Extraction de séquences de données d'un arbre XML à l'aide d'une expression XPath ou de la fonction data()

XQuery : expression

- Expression FLWOR : rappel

Nom	Notation	Rôle
Itération	for $var IN seq return expr	Itération sur les éléments d'une séquence et construction d'une séquence résultat dérivée
Sélection (restriction)	for $var IN seq where qual return cons	Itération avec sélection des arbres satisfaisant la qualification qual : expression de la forme expr1 Θ expr2
Quantification existencielle	some $var in seq satisfies qual	Expression de qualification vraie si un des éléments de la séquence satisfait la qualification qual

XQuery : expression

- Expression FLWOR : rappel

Nom	Notation	Rôle
Quantification universelle	every $var in seq satisfies qual	Expression de qualification vraie si tous les éléments de la séquence satisfont la qualification qual
Jointure	for $var1 in seq1 $var2 in seq2 where qual return expr	Expressions d'itération imbriquées permettant de joindre seq1 et seq2 selon la qualification qual
Tri	expr1 sortby expr2	Tri de la séquence Expr1 selon les données extraites par Expr2

XQuery : expression

- Expression FLWOR : rappel

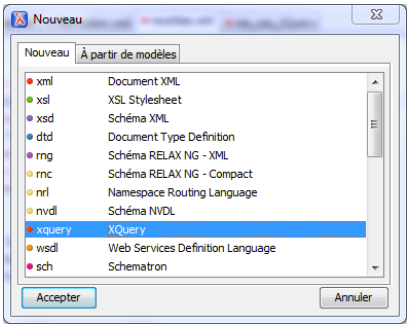
Nom	Notation	Rôle
Fonctions intégrées	distinct-value … avg, count, …	Élimination des doubles … Fonctions d'agrégats
Définition de fonction	define function nom ([type:var]*) returns collection	Définition d'une fonction paramétrée retournant une collection d'arbres XML

Exercices !

Recherche d'information avec XQuery

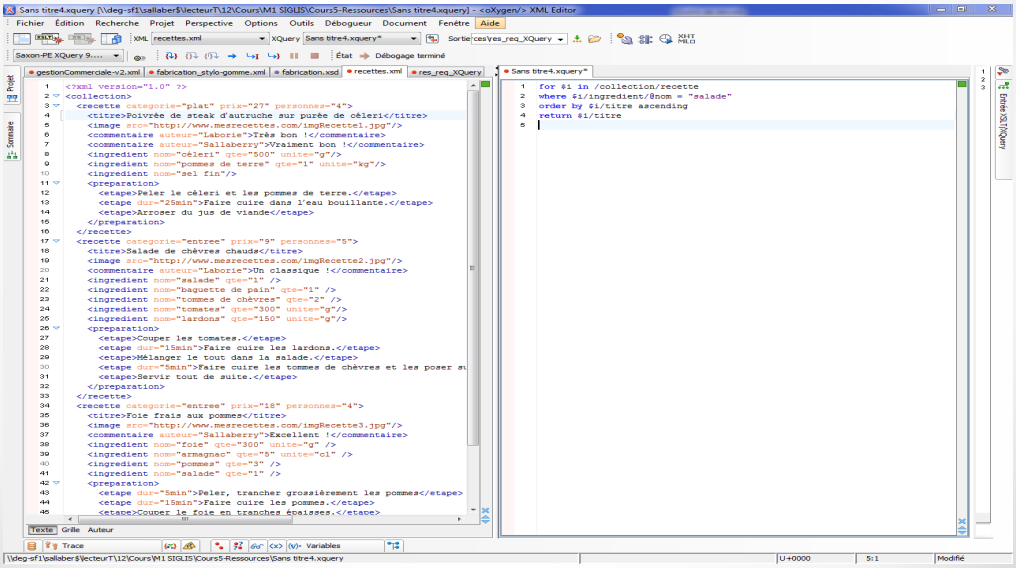
Exercice 1 : éditeur oXygen

- Ouvrir un nouveau fichier



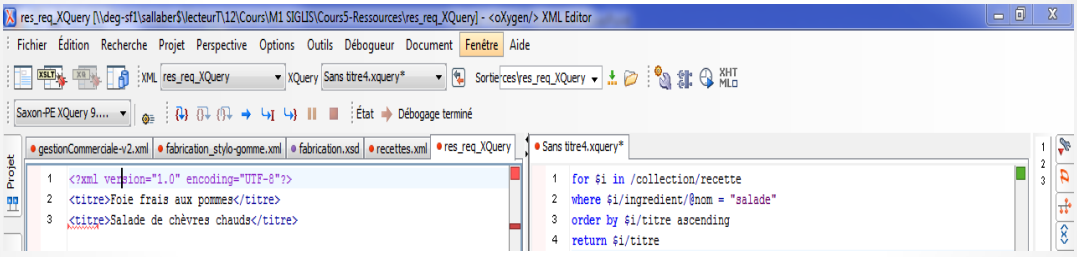
Exercice 1 : éditeur oXygen

- Ecrire et exécuter une requête XQuery



Exercice 1 : éditeur oXygen

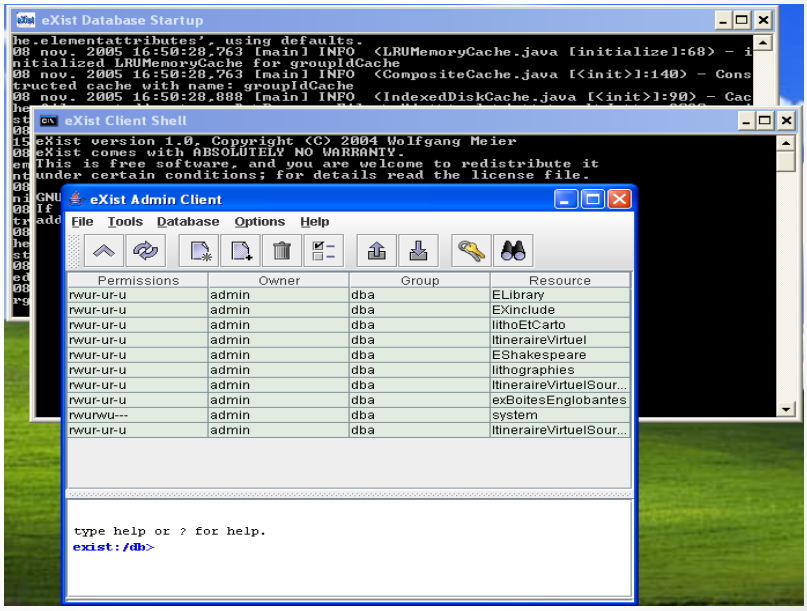
- Ecrire et exécuter une requête XQuery



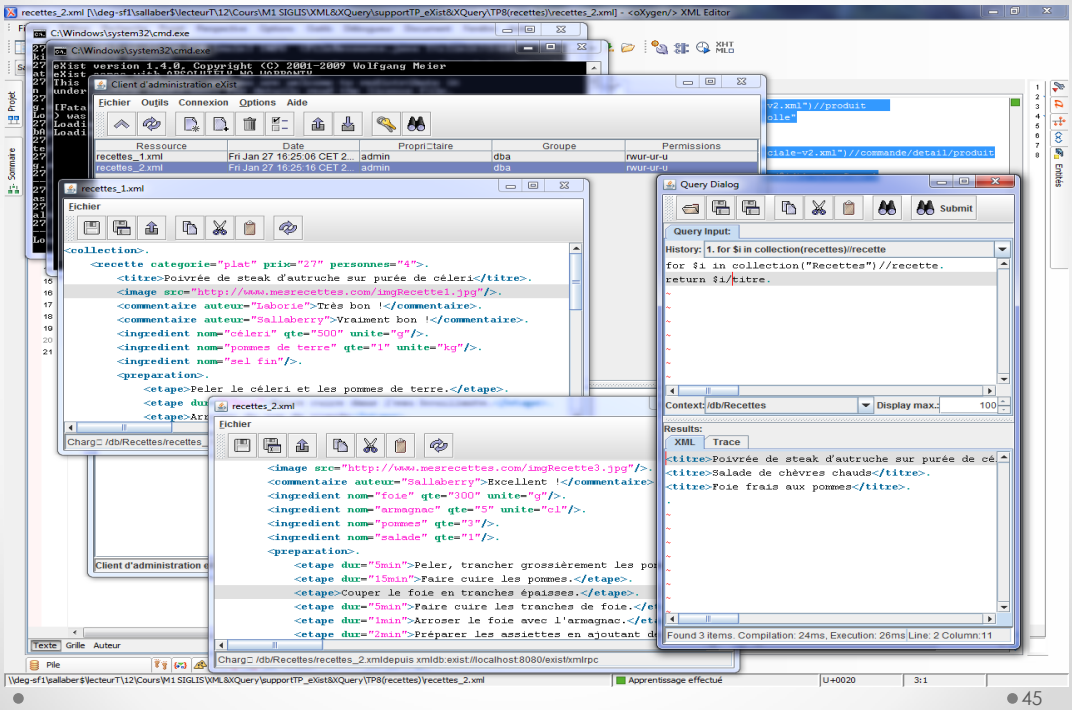
Exercice 2 : Oxygen & XQuery

- Interroger le fichier gestion_commerciale.xml :
  - Noms et prénoms des clients
  - Clients qui ont des commandes
  - Clients et nombre de commandes
  - Clients qui ont plus de 2 commandes
  - Clients qui n'ont pas de commande
  - Désignation et prix des produits à plus de 10 euros
  - Produits en rupture de stock (quantité en stock < 2)
  - Produits commandés
  - Code des commandes comprenant plus de deux lignes de commande
  - Nombre total de bâtons de colle commandés
  - Nombre de bâtons de colle en stock
  - Nombre de commandes

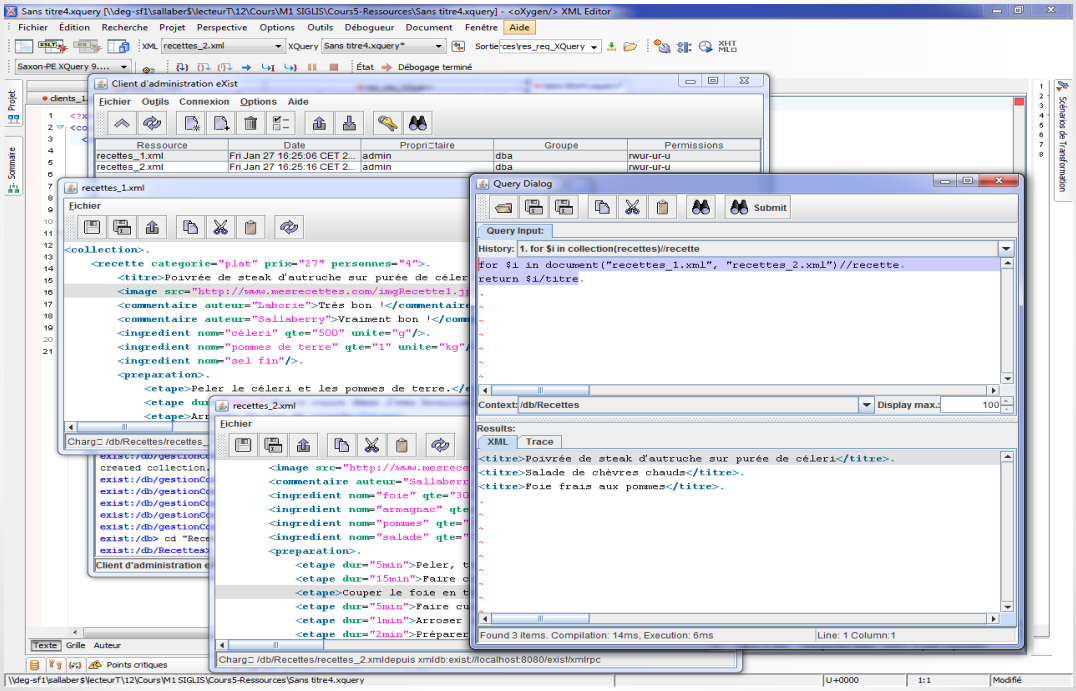
Exercice 3 : éditeur eXist



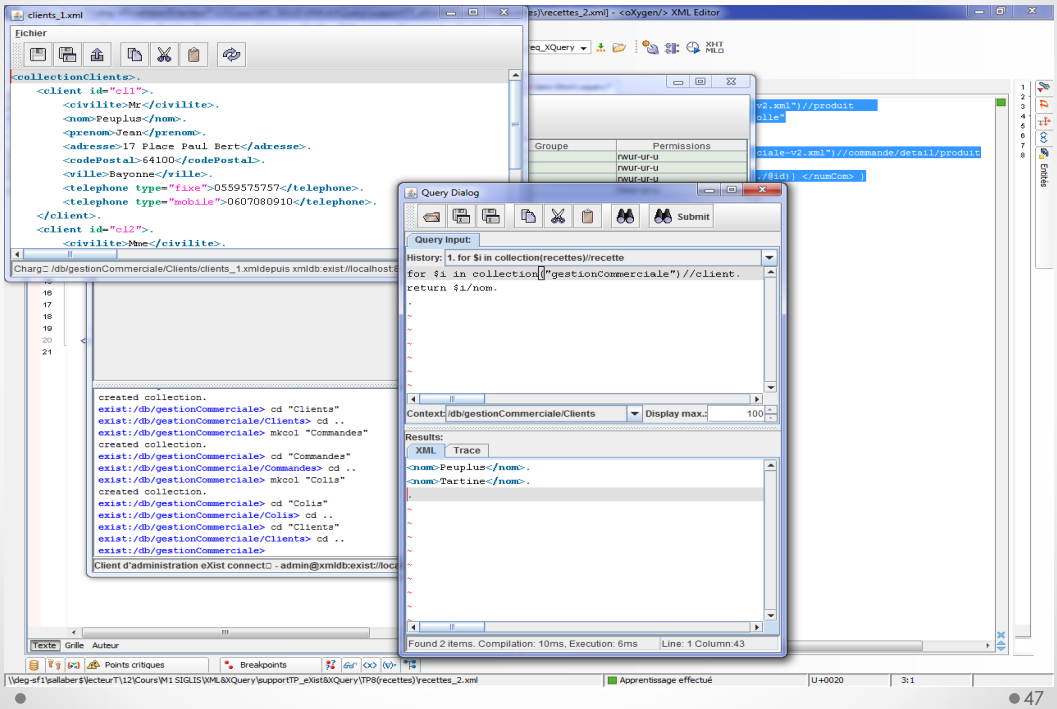
Exercice 3 : éditeur eXist



Exercice 3 : éditeur eXist



Exercice 3 : éditeur eXist



Exercice 4 : eXist & XQuery

- **Interroger le fichier gestion_commerciale.xml :**
  1. Noms et prénoms des clients
  2. Clients qui ont des commandes
  3. Clients et nombre de commandes
  4. Clients qui ont plus de 2 commandes
  5. Clients qui n'ont pas de commande
  6. Désignation et prix des produits à plus de 10 euros
  7. Produits en rupture de stock (quantité en stock < 2)
  8. Produits commandés
  9. Code des commandes comprenant plus de deux lignes de commande
  10. Nombre total de bâtons de colle commandés
  11. Nombre de bâtons de colle en stock
  12. Nombre de commandes