LES BD'S ET LE WEB

Partie1

Plan

- Les BD’s et le Web
  - *Les serveurs Web dynamiques*
  - *L’accès aux BD depuis Java*
  - *XML et les BD semi structurées*

BDDW

- Données
  - *Comment gérer (stocker/interroger) des données ?*
- Web
  - *Comment développer des pages web, en particulier avec des données qui évoluent/sont mises à jour ?*
- et encore un peu de Réseau
  - *Comment se gère la communication via un réseau ?*

Serveurs web dynamiques

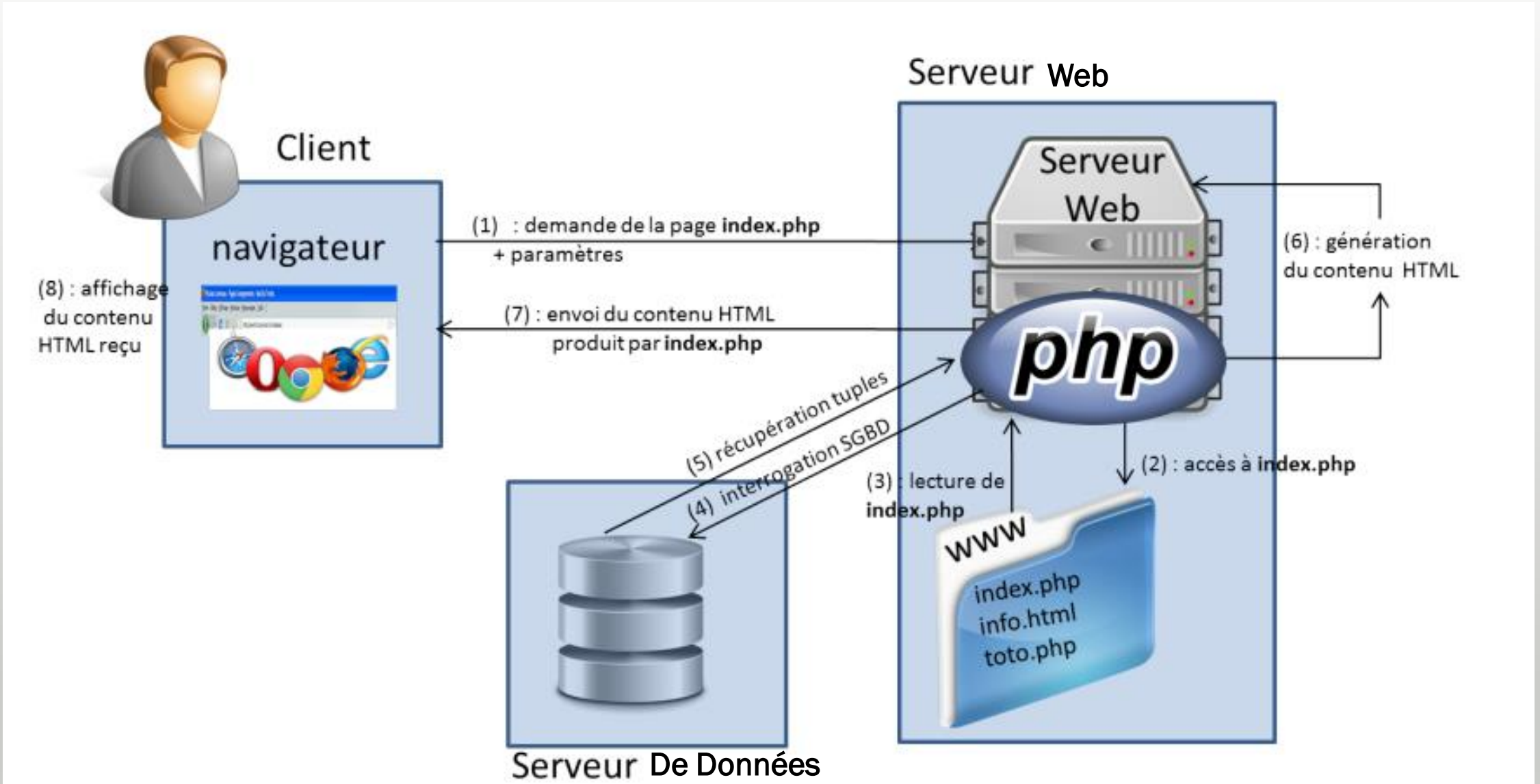
- Initialement, le Web était statique, ce qui veut dire qu'on ne pouvait que charger des pages HTML (ou des images, vidéos) existantes et stockées sur les disques des serveurs Web.
  - *Le web n'était alors qu'un simple graphe de documents, un document pouvant contenir un ensemble de liens (HTML) vers d'autres documents.*
- Le besoin s'est très vite fait sentir d'être en mesure d'exécuter des programmes dans les serveurs Web, ces programmes générant des pages Web.
  - *On dit alors qu'on a des* ***serveurs Web dynamiques*** *: dont les pages sont générées dynamiquement (par des programmes).*

Serveurs web dynamiques

- L'autre besoin qui s'est fait sentir très vite et de passer des paramètres à ces programmes, ces paramètres étant saisis et transmis par l'utilisateur.
- Exemple d'une application Web de réservation de billet d'avion.
  - *L'utilisateur doit pouvoir saisir des paramètres (ville de départ, d'arrivée, horaires) et le serveur web doit pouvoir générer la page Web montrant les vols disponibles.*

Serveurs web dynamiques

- Pour répondre à ces 2 besoins, on a ajouté :
  - *des formulaires dans HTML.*
    - Ces formulaires permettent une interaction avec l'usager (qui peut saisir du texte ou choisir des options avec des boutons) et ces données sont envoyées comme paramètres dans la requête HTTP.
  - *des scripts CGI (Common Gateway Interface).*
    - Ce sont des programmes installés dans le serveur Web (comme des documents, mais exécutables).
    - Lorsque le serveur reçoit une requête adressant un tel programme, au lieu de retourner le document (le programme), le programme est exécuté et c'est le résultat de l'exécution du programme qui est retourné en réponse au client (navigateur)


Client
navigateur
(8) : affichage du contenu HTML reçu
(1) : demande de la page index.php + paramètres
(7) : envoi du contenu HTML produit par index.php
Serveur Web
Serveur Web
php
(6) : génération du contenu HTML
(2) : accès à index.php
(3) : lecture de index.php
(5) récupération tuples
(4) interrogation SGBD
WWW
index.php
info.html
toto.php
Serveur De Données

Accès aux BDD depuis java(JDBC)

JDBC

- JDBC: Java database connectivity
  - *Cette API a était développée par SUN pour permettre à des applications Java d'accéder à des bases de données relationnelles quelconques.*
- Les étapes principales
  - *Se connecter à une base de données*
  - *Envoyer une requête SQL*
  - *Manipuler le résultat*
- JDBC: un driver (pilot) fournissant des outils pour ces fonctions

installation

- Installer un driver JDBC
  - *Aller sur https://dev.mysql.com/downloads/file/?id=480292*
  - *Télécharger et Décompresser l'archive .zip*
- Intégrer le driver dans votre projet
  - *Faire un clic droit sur le nom du projet et aller dans New > Folder*
  - *Renommer le répertoire « lib » puis valider*
  - *Copier le .jar de l'archive décompressée dans « lib »*
- Ajouter JDBC au path du projet
  - *Faire clic droit sur .jar qu'on a place dans « lib »*
  - *Aller dans Build Path et choisir Add to Build Path*

JDBC

- Remarque
  - *Tous les imports de ce chapitre sont de java.sql.*;*
- Chargement du driver

```
try {
    Class.forName("com.mysql.jdbc.Driver");
}
catch (ClassNotFoundException e) {
    System.out.println(e.getMessage());
 }
```

Se connecter a la base de données

- Il faut spécifier l'URL (de la forme ´ jdbc:mysql://hote:port/nombd)
  - *hote : le nom de l'hote sur lequel le serveur MySQL est installé (dans notre cas localhost ou 127.0.0.1)*
  - *port : le port TCP/IP utilise par défaut par MySQL est 3306*
  - *nombd : le nom de la base de données MySQL*
- Il faut aussi le nom d'utilisateur et son mot de passe (qui permettent de se connecter a la base de données MySQL)

JDBC

Connexion a la base

```
String url = "jdbc:mysql://localhost:3306/jdbc";
String user = "root";
String password = "";
Connection connexion = null;
try {
    connexion = DriverManager.getConnection(url, user, password);
} catch (SQLException e) {
    e.printStackTrace();
}
finally {
    if (connexion != null)
        try {
            connexion.close();
        } catch (SQLException ignore) {
            ignore.printStackTrace();
        }
}
```

JDBC

Préparation et exécution de la requête

```
// création de la requête (statement)
Statement statement = connexion.createStatement();

// Préparation de la requête
String request = "SELECT * FROM Personne;";

// Exécution de la requête
ResultSet result = statement.executeQuery(request);
```

On utilise

- executeQuery() pour les requêtes de lecture : ˆ select.
- executeUpdate() pour les requêtes d'écriture : ´ insert, update, delete et create.

JDBC

- Récupération des données

```
while (result.next()) {
    // on indique chaque fois le nom de la colonne
       et le type
    int num = result.getInt("num");
    String nom = result.getString("nom");
    String prenom = result.getString("prenom");
    System.out.println(num + " " + nom + " " +
       prenom);
}
```

- Pour faire une insertion

```
Statement statement = connexion.createStatement();
String request = "INSERT INTO Personne (nom,prenom)
   VALUES ('Wick','John');";
int nbr = statement.executeUpdate(request);
if (nbr ! = 0)
    System.out.println("insertion réussie");
```

La méthode executeUpdate() retourne
• 0 en cas d'echec de la requête d'insertion, et
• 1 en cas de succès le nombre de lignes respectivement mises a jour ou supprimées

Manipuler le résultat

- On peut identifier chaque colonne de la base de donnée
  - *Par son index*
  - *Par son nom*

```
String strQuery = "SELECT * FROM T_Users;";
ResultSet rsUsers = stUsers.executeQuery(strQuery);
while(rsUsers.next()) {
    System.out.print("Id[" + rsUsers.getInt(1) + "]"
        + rsUsers.getString(2)
        + "[" + rsUsers.getString("Password") + "] "
        + rsUsers.getInt("ConnectionNumber") ); }
rsUsers.close();
```

Récapitulation

- Les éléments pour faire fonctionner un programme
  - ***Matériels:*** *mémoire, CPU, bus, registres, disque, périphériques, …*
  - ***Système d'exploitation:*** *gérer les matériels, outils de base pour télécommunication, gestion et manipulation de fichiers, …*
  - ***Applications:*** *base de données, programme d'utilisateur*
  - ***Communication*** *entre des applications et entre des ordinateurs*
    - Protocole
    - Connexion et transfert

XML et les BD semi structurées

Introduction

XML devient le standard d'échange des systèmes d'information ouvert au dessus du protocole http. Il devient donc nécessaire de :

- stocker les document xml reçus dans une base de donnée
- publier les données des bases directement sur le web

Depuis 1998, l'idée de bases de données XML a émergé, deux approches se sont opposées

- l'approche Middleware
- l'approche native

Rappel sur XML

- Un document XML est un document créé en utilisant les facilités XML

```
<?xml version="1.0" encoding="utf-8"?>
    <voeux genre="simple">
        Bonne fête.
    </voeux>
```

- <? ...?> déclaration de document XML.
- Le reste est formé d'une suite d'**éléments** XML.
- Un élément est composé de **balise de début**, **données** (*character data*) , **balise de fin**.
- Le mot voeux est utilisé pour parler de balise "voeux" (<voeux> </voeux>)
- Pour un élément on peut définir des **attributs** *nom*="*valeur*", à l'intérieur de la balise de début.
- L'attribut genre="simple", sert ainsi à compléter l'information portés par un élément de document XML.
- Un document XML est destiné à être compris par un humain aussi bien qu'une machine (facile à traiter par une application).
- Deux instances d'un même élément peuvent être de tailles différentes.

Document Bien Formé et Document Valide

- La spécification XML définit deux niveaux de conformité pour un document XML. **Document bien formé** (*well-formed*) et **document valide** (*valid*)
- Un document *X* est dit bien formé vis à vis de XML ssi:
  - il satisfait la grammaire et les règles définies par la spécification XML
  - chaque document *Y* référencée directement ou indirectement par *X* est bien formé à son tour
- Les règles sont de type: un seul élément racine, si balise ouvrante *alors* balise fermante de même nom, imbrication propre, etc…
- Un document *X* est dit valide ssi:
  - il est bien formé
  - est conforme à une *DTD* donnée.

«XML, est-il une base de données? »(1)

- XML et les technologies qui lui sont associées est une sorte de SGBD
  - *Le stockage (les documents XML),*
  - *Des Schémas (DTDs, XML Schemas, RELAX NG, etc.),*
  - *Des langages de requêtes (XQuery, XPath, XQL, etc.),*
  - *Des interfaces de programmation (SAX, DOM, JDOM)*
- Mais:
  - *Un stockage efficace,*
  - *Les index,*
  - *La sécurité,*
  - *La gestion des transactions,*
  - *L'accès multi-utilisateurs,*
  - *Les déclencheurs (triggers),*
  - *Les requêtes sur plusieurs documents, etc.*

«XML, est-il une base de données? »(2)

- **En résumé**

  bien qu'il soit possible d'utiliser un ou plusieurs documents XML comme une base de données dans les environnements qui présentent une faible quantité de données, qui comportent peu d'utilisateurs, et qui se satisfont de performances modestes, cela échouera dans la plupart des environnements de production qui disposent de nombreux utilisateurs, requièrent une **stricte** intégrité des données et ont besoin de bonnes performances.

«Comment choisir sa base de données? »(1)

Quand vous commencez à penser à XML et aux bases de données vous devez d'abord répondre à certaines questions:

- Avez-vous des données déjà existantes que vous souhaitez publier ?

- Recherchez-vous un endroit où stocker vos pages Web ?

- La base est-elle exploitée par une application de e-commerce dans laquelle XML est utilisé comme vecteur de données ?

Les réponses à ces questions influenceront grandement votre choix de base de données et de logiciel intermédiaire [*middleware*] (s'il y a lieu) ainsi que la manière d'utiliser cette base.

«Comment choisir sa base de données? »(2)

Application e-commerce (XML format de transport)

- Données à structure régulière
- Données utilisées par des applications non XML

→ Base de données relationnelle (ou objet) et des techniques de sérialisation

Site Web (documents orientés texte)

- Données à structure moins régulière
- L'organisation des données est importante (ordre, entités,etc.)
- Recherche sur le contenu, structure

→ Base de données native XML

Données ou Documents ?

Document XML centré-données (*data-centric)*

- conçus pour les stocker et traiter à partir d'une base de donnée.
- stockage relationnel avec middleware de transformation en XML est bien adapté avec ce type de contenu
- le document XML n'est pas conservé (reconstituable) et les données sont extraites des fichiers XML puis stockées dans une BD classique (relationnelle)

Document XML centré-données (data-centric)

- **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-données

Exemple: l'ordre de ventes suivant est *orienté données* :

```
<OrdreDeVentes NumeroOrdreDeVentes="12345">
  <Client NumeroClient="543">
    <NomClient>ABC Industries</NomClient>
    <Rue>123 Main St.</Rue>
    <Ville>Chicago</Ville>
    <Etat>IL</Etat>
    <CodePostal>60609</CodePostal>
  </Client>
  <DateOrdre>981215</DateOrdre>
  <Item NumeroItem="1">
    <Lot NumeroLot="123">
      <Description>
        <p><b>Turkey wrench:</b><br />
        Stainless steel, one-piece construction,
        lifetime guarantee.</p>
      </Description>
      <Prix>9.95</Prix>
    </Lot>
    <Quantite>10</Quantite>
  </Item>
</OrdreDeVentes>
```

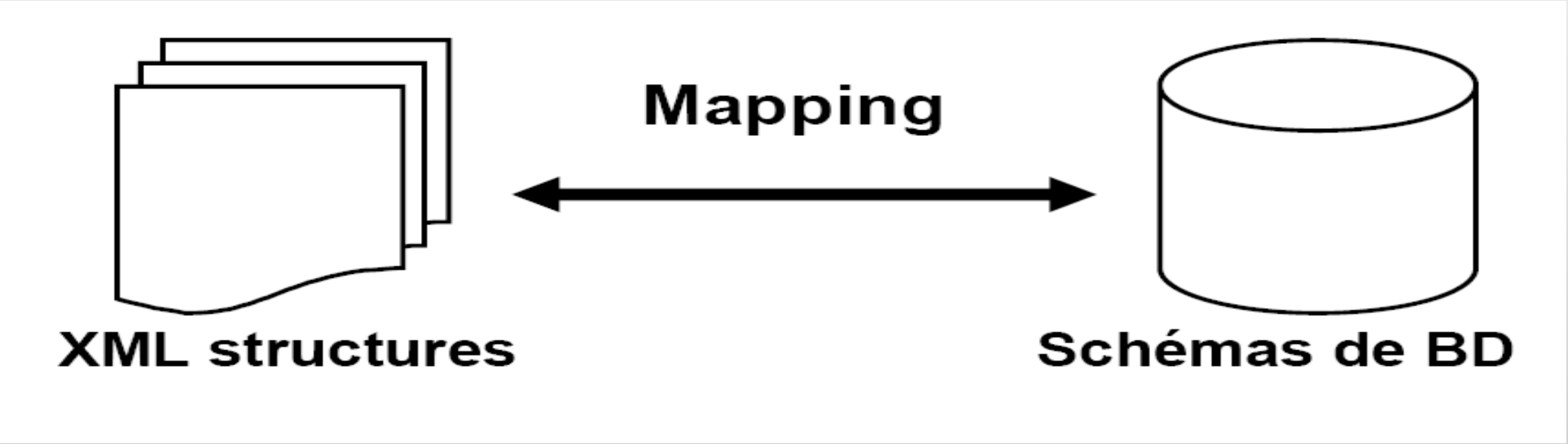
Document XML centré-documents (*document-centric*)

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

```
<Produit>
  <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>
  <Liste>
  <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>
  </Liste>
  <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>
</Produit>
```

XML dans des bases de données classiques

•Documents centré-données



• La granularité est généralement l'élément ou l'attribut

La correspondance entre schémas de documents et schémas de base de données (1)

1-La correspondance basé sur des tables

- Utilisée le plus souvent par les middleware
- Les documents XML sont modélisés comme une ou plusieurs tables
- Les XML documents suivent un format prédéfini

```
<database>
    <table>
        <row>
            <column1> …
             /column1>
        </row>
      ……..
    </table>
</database>
```

La correspondance entre schémas de documents et schémas de base de données(2)

2-La correspondance basée sur un modèle Objet-Relationnel

– Utilisée par la plupart des Bases de données « XML-Enabled»(compatibles XML)

- Les données du document sont modélisées comme un arbre d'objets :
  - les types d'éléments possédant des attributs, les contenus d'éléments ainsi que les contenus mixtes sont généralement modélisés comme des classes.
  - Les types d'éléments contenant seulement des PCDATA, les attributs et les PCDATA elles même sont modélisés comme des propriétés scalaires.
  - Le modèle est alors mis en correspondance avec la base relationnelle tel que les classes correspondent à des tables et les propriétés scalaires à des colonnes.

Les langages de requêtes(1)

- Plusieurs produits transfèrent les données directement selon le modèle sur lequel ils sont construits.
- Étant donné que la structure du document XML est souvent différente de la structure de la base, ces produits incluent ou sont souvent utilisés avec XSLT.

→

- Ceci pour transformer des documents selon la structure commandée par le modèle avant de transférer les données à la base (et vice-versa)
- traitement XSLT peut être coûteux
- **Solution** : l'implémentation de langages de requête renvoyant du XML

Les langages de requêtes(2)

1 Les langages de requête basés sur des modèles

- Dans ces langages, il n'existe pas de correspondance prédéfinie entre le document et la base
- Les ordres SELECT sont englobés dans un modèle et les résultats sont traités par le logiciel de transfert de données

le modèle suivant utilise les éléments <**SelectStmt**> pour inclure les ordres SELECT et les valeurs **$column-name** pour déterminer où les résultats doivent être placés :

Les langages de requêtes(3)

```
<?xml version="1.0"?>
  <InformationsVol>
    <Introduction>The following flights have available seats:</Introduction>
    <SelectStmt>SELECT Compagnie, NumeroVol, Depart, Arrivee FROM
Vols</SelectStmt>
      <Vol>
        <Compagnie>$Compagnie</Compagnie>
        <NumeroVol>$NumeroVol</NumeroVol>
        <Depart>$Depart</Depart>
        <Arrivee>$Arrivee</Arrivee>
      </Vol>
    <Conclusion>We hope one of these meets your needs</Conclusion>
  </InformationsVol>
```

Les langages de requêtes(4)

Résultat

```
<?xml version="1.0"?>
  <InformationsVol>
    <Introduction>The following flights have available seats:</Introduction>
    <Vols>
      <Vol>
        <Compagnie>ACME</Compagnie>
        <NumeroVol>123</NumeroVol>
        <Depart>Dec 12, 1998 13:43</Depart>
        <Arrivee>Dec 13, 1998 01:21</Arrivee>
      </Vol>
      ...
    </Vols>
    <Conclusion>We hope one of these meets your needs.</Conclusion>
  </InformationsVol>
```

Les langages de requêtes(5)

2 - Les langages de requête basés sur SQL

- **extensions XML au langage SQL** : SQL/XML introduit des types de données XML et ajoute un certain nombre de fonctions à SQL de telle façon qu'il soit possible de construire des éléments et des attributs XML à partir de données relationnelles.
- Utilisable uniquement sur des bdd relationnelles

```
SELECT
        XMLELEMENT(NAME "Order",
                XMLATTRIBUTES(Orders.SONumber AS SONumber),
                XMLELEMENT(NAME "Date"),
                XMLELEMENT(NAME "Customer")) AS xmldocument
  FROM Orders
```

Les langages de requêtes(6)

3- Les langages de requête XML

Xquery, XQL

Oracle et XML(1)

XML DB est le nom de la technologie d'Oracle qui permet de gérer du contenu XML en base (stockage, mises à jour et extractions).

Les deux principales caractéristiques de XML DB sont d'une part l'interopérabilité entre SQL et XML (documents et grammaires), et, d'autre part la gestion de ressources XML dans un contexte multiutilisateurs avec *XML Repository*

Oracle et XML(2)

- *Le type de données XMLType*

Le type de données XMLType fournit de nombreuses fonctionnalités, la plupart relatives à XML (validation de schéma XML et transformation XSL), d'autres qui concernent SQL :

· Définition d'une colonne d'une table (jouant le rôle d'un type de données) ;

· Définition d'une table (jouant le rôle d'un type d'objet) ;

· Déclaration de variables PL/SQL ;

· Appel de procédures PL/SQL cataloguées.

```
CREATE TABLE tab OF XMLTYPE
```

```
INSERT INTO tab VALUES (XMLType('
<element>
...
</element>
'))
```

```
SELECT t.OBJECT_VALUE.EXTRACT('XPath').GETSTRINGVAL() FROM tab t
```

BDD XML Native(1)

- Les *bases de données XML natives* sont des bases conçues spécialement pour stocker des documents XML.
- Comme toutes les autres bases, elles possèdent des fonctionnalités telles que les transactions, la sécurité, les accès multi-utilisateurs, un ensemble d'APIs, des langages de requête
- Les bases de données XML natives sont plus franchement utiles pour le stockage des contenus *orientés document*
- préservent des choses telles que l'ordre interne du document, les instructions de traitement, les commentaires, les sections CDATA, l'utilisation des entités, etc.
- supportent les langages de requête XML qui permettent de poser des questions telles que : « Donnez-moi tous les documents dans lesquels le troisième paragraphe après le début d'une section contient un mot en caractères gras. » De telles recherches sont manifestement difficiles à formuler dans un langage tel que SQL.

BDD XML Native(2)

- Une base de données XML native définit un modèle (logique) de document XML, stocke et retrouve les documents en fonction de ce modèle. Le modèle doit au minimum inclure les éléments, les attributs, les PCDATA et l'ordre interne du document.

- Le document XML est l'unité fondamentale du stockage (logique) dans une base de données XML native, tout comme une ligne d'une table constitue l'unité fondamentale du stockage (logique) dans une base relationnelle.

- base de données XML native ne repose pas sur un modèle physique particulier pour le stockage. Elle peut par exemple être bâtie aussi bien sur une base relationnelle, hiérarchique, orientée-objet, ou bien utiliser des techniques de stockage propriétaires comme des fichiers indexés ou compressés.

Les caractéristiques des bases de données XML natives(1)

1 Les collections de documents

Ce concept joue un rôle similaire à la table dans une base de données relationnelle ou au répertoire dans un système de fichiers.

il est possible d'imbriquer les collections

2 Les langages de requête

Xpath,XQL,XQuery

3 Mises à jour et effacements

DOM

4 Transactions, verrouillages et accès concurrentiels

- Pratiquement toutes les bases XML natives supportent les transactions

- Le verrouillage est réalisé le plus souvent au niveau du document dans son intégralité plutôt qu'au niveau des fragments du document

Les caractéristiques des bases de données XML natives(2)

5 Les APIs [Application Programming Interfaces]

Presque toutes les bases XML natives proposent des APIs

avec des méthodes permettant la connexion à la base, l'exploration des métadonnées, l'exécution des requêtes et la recherche des résultats.

6 L'aller-retour des documents [Round-Tripping]

Une caractéristique importante des bases XML natives est qu'elles permettent l'aller-retour des documents [*Round-Tripping*]. Cela signifie que l'on peut stocker un document XML dans une base XML native et obtenir à nouveau le "même" document.

7 Les index

Presque toutes les bases XML natives supportent l'indexation des valeurs des éléments et des attributs.

Exemple de BDD native

- eXist : Base de données native XML Open Source entièrement écrite en Java
  - http://exist.sourceforge.net/
  - *Bien qu'il possède quelques caractéristiques*
    - DTD, XML Schemas
    - des langages de requêtes -XQuery, XPath,
    - des interfaces de programmation : SAX, DOM, JDOM
  - *Il ne propose pas :*
    - un stockage efficace,
    - les index,
    - la sécurité,
    - les transactions et l'intégrité des données,
    - l'accès multi-utilisateur