

# XML et données semi-structurées

Serge Abiteboul

[Serge.Abiteboul@inria.fr](mailto:Serge.Abiteboul@inria.fr)

Data on the Web: Abiteboul, Buneman, Suciu, Morgan  
Kaufmann

---

# Plan

- Contexte et cahier des charges
- A la découverte de XML
- Héritage et principes fondamentaux
- Exemples d'applications XML
- Avantages de XML
- Zoom sur le langage
- Typage rapidement
- Quelques gadgets importants



# Contexte et cahier des charges

---

# La révolution du Web...

- Depuis dix ans, Internet révolutionne l'informatique « grand public »
- HTML est le langage du Web...
  - Même si on trouve aussi du .doc, .ps, .pdf... et des images (jpg, gif), du son, de la vidéo...
- Des milliards de pages existent actuellement
  - public/privé, statique/dynamique, visible/caché
- Support naturel pour l'information distribuée
  - À destination d'êtres humains et
  - de plus en plus, pour des applications



---

# Distribution de données sur le Web: les applications arrivent...

- Exemples:
  - B2C (commerce électronique)
  - B2B (achats groupés)
  - Bibliothèques et fonds documentaires en ligne
  - G2C (impôts)...
- HTML n'est pas adapté pour ces applications
  - Ces applications (programmes) ont besoin de typage pour représenter la structure des données



---

# Constat1

- Pour écrire des programmes, il faut du typage
  - On ne peut pas se contenter d'obtenir des ensembles de pages HTML comme avec les moteurs de recherche du Web
- Les modèles documentaires à la HTML ne proposent pas assez de structure
- Quel modèle et quelles structures?
  - Un typage à la « bases de données »?



---

# Particularités – différences avec SGBD

1. On ne connaît pas bien la structure
  - Dans un SGBD, on connaît la structure des tables et la sémantique des colonnes; pas sur le Web
2. La structure est irrégulière
  - Données manquantes, ou en plus (annotations)
  - Variations de type: dollars, euros
  - Standards différents sur les adresses
3. La structure peut être implicite
  - Il faut « parser » pour la découvrir
4. La structure peut n'être que partielle
  - Une partie des données est sans structure: plein texte, images, son
5. Le typage peut n'être qu'indicatif
  - On tolère des données non strictement conforme



---

# Particularités – différences avec SGBD)

6. Le schéma est souvent a posteriori
  - On a des données et on en déduit un type
7. Le schéma est souvent gros et complexe
8. Le schéma est parfois ignoré par les requêtes
  - Requête par mots clé et broutage
9. Le type d'une information peut dépendre du contexte
  - An objet MyBib est d'abord une chaîne de caractères, puis un fichier BibTex après analyse/classification, il peut obtenir un « propriétaire » une date\_création; peut devenir ensuite un ensemble de Références après analyse syntaxique
10. Le schéma évolue très rapidement
  - Souvent une raison de ne pas choisir un SGBD



---

# Constat2

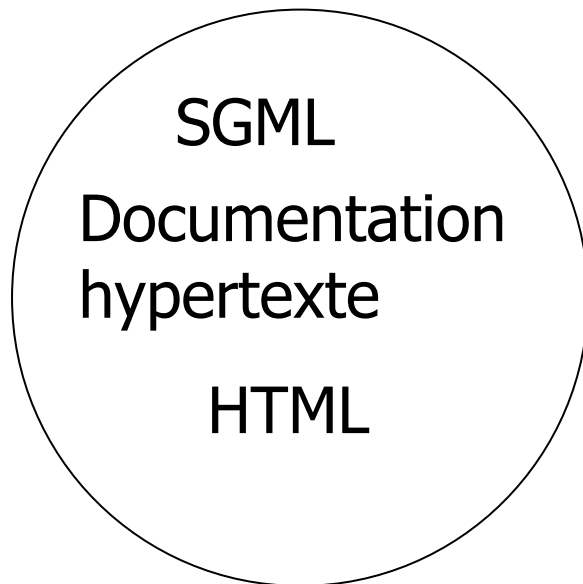
- Les modèles bases de données classiques proposent une structure trop rigide
- Ni modèle documentaire, ni modèle base de données, on va prendre un modèle de données semi-structuré qui marrie les deux



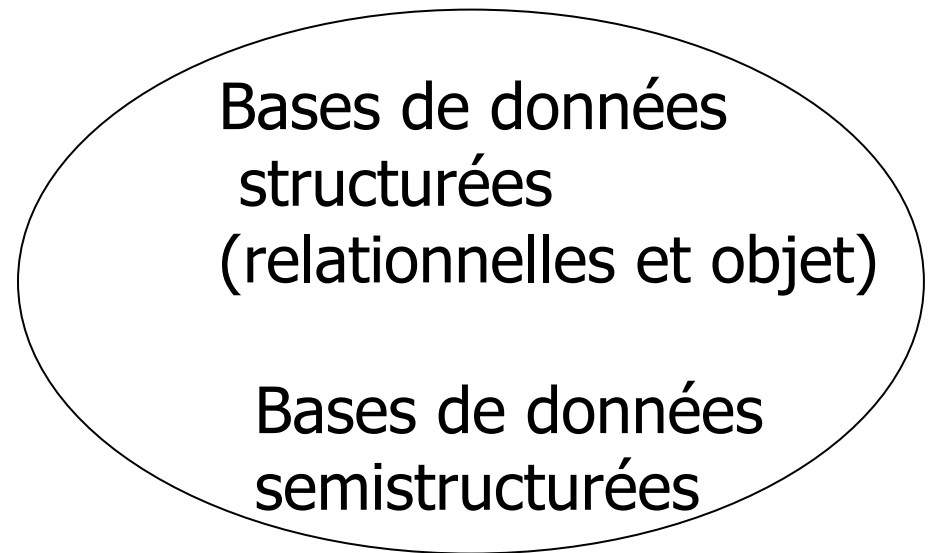
---

# Deux mondes se rejoignent dans les modèles semi-structurés

## Gestion de documents



## Gestion de données



XML



---

# Données semi-structurées = XML

- Cible: la gestion du champ le plus large possible de contenus
- SGBD
- Données textuelles
- Données documentaires
- Data eXchange Formats
  - ASN1, BibTex, EDI...
  
- Architecture
  - Client/serveur
  - médiateurs or entrepôts



---

# Accès à de telles données

Requêtes – aspects principaux

- Bases de données style SQL/OQL
- Navigation (style broutage Web)
- Mots clés (style moteur recherche Web) et pattern matching (style grep)
  
- Requêtes sans connaître la structure des données – interroger données et structure simultanément
- Requêtes tenant compte du temps, requêtes sur des versions, archives, requêtes sur les changement
- Appui linguistique: synonymes, correction de fautes d'orthographe



# A la découverte de XML

---

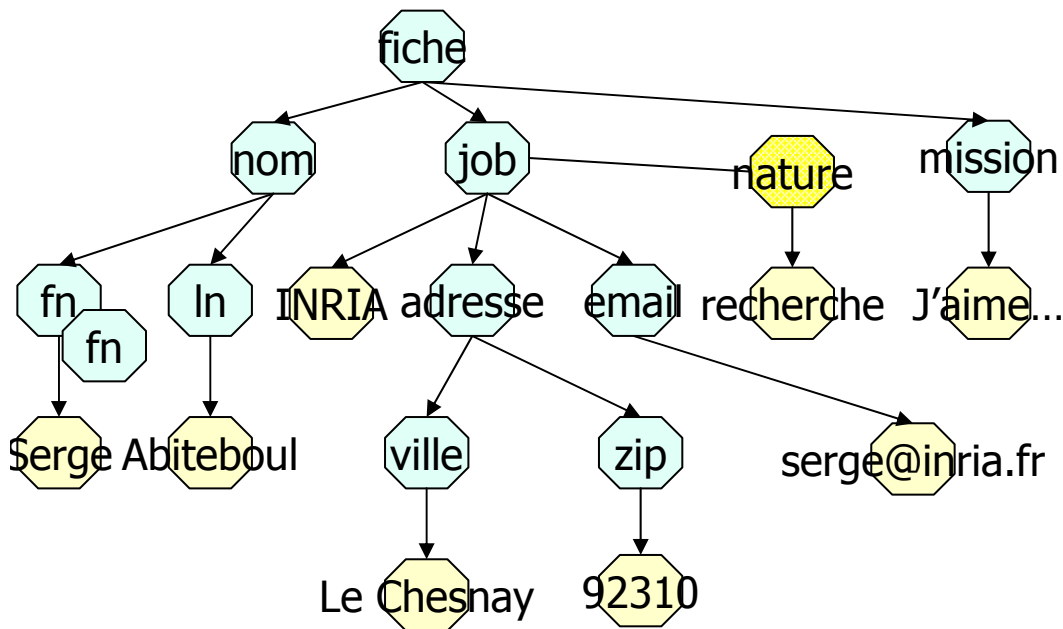
# XML – eXtensible Markup Language




- Un format universel pour les documents et données (semi-) structurés sur le Web
- Une version simplifiée de SGML (ISO 8879)
- L'espéranto du Web qui va remplacer HTML
- Beaucoup de bruit pour rien... du marketing
- Une famille de standards en train de naître
  - XLink, XPointer, XSchema, DOM, SAX, Xpath, XSL, XQuery, SOAP, WSDL, ...
- Un modèle de données fondé sur des arbres et un langage de représentation basé sur le balisage (« bannières »)



# Exemple

*La structure est dans les bannières*  
*La sémantique aussi* <a>...</a>



Éléments  fn  
Attributs  nature  
Données  92310

```
<fiche>
  <nom>
    <fn>Serge</fn>
    <ln>Abiteboul</ln>
  </nom>
  <job nature="recherche">
    INRIA
    <adresse>
      <ville>Le Chesnay</ville>
      <zip>92310</zip>
    </adresse>
    <email>serge@inria.fr</email>
  </job>
  <mission>J'aime enseigner ☺ ☹
</mission>
</fiche>
```

Syntaxe



---

# Remarques

- XML fournit une *syntaxe*, pas de *sémantique* « a priori »
- Les balises n'ont pas de présentation ou de signification définie par le langage mais elles peuvent bien sûr avoir un sens pour les applications
  - `<nom>jean</nom>`
  - `<matière>jean</matière>`
- XML ne définit que la structure et le contenu d'un document, pas son comportement ni son traitement



---

# Remarques

- Développement et promotion par W3C
  - Industriels: tous les poids lourds, notamment Oracle, IBM, Compaq, Xerox, Microsoft, etc..
  - Laboratoires de recherche: MIT (représentant les US), INRIA (Europe), université Keio au Japon (Asie)
- Sur Internet: publication et échange d'information
- Simplicité
  - Production
  - Lecture, analyse syntaxique, compréhension
- Les mêmes données (avec différentes feuilles de style) disponibles pour de nombreux supports (pc, pda...) et de nombreuses applications



---

# XML est un univers impitoyable

## Les standards naissent et meurent

- XML: données
- [DTD], Xschema: typage
  - Les documents doivent être bien formés mais le typage (« valide ») n'est pas obligatoire
- XSL/T, [Xquery]: transformation et requêtes
- XPATH: chemins
- XLink: liens entre documents
- DOM: Application programming interface
- SOAP: distributed computing
- [WebDAV]: distributed authoring and versioning
- Plus des tas d'autres et de logiciels



# Héritage et principes fondamentaux

---

# XML : « successeur » de HTML

- HTML Hypertext Markup Language.
  - Un ensemble prédéfini et limité de balises surtout de présentation, défini par une norme (HTML 2.0, 3.2, 4.0).
- Sémantiques des balises :
  - `h1,...,h6, title, address, ...` donnent des indications structurelles
  - `center,hr,b,i,big,small,...` ne servent qu'à décrire une mise en page
- Tim Berners-Lee (le créateur de HTML) a lui-même prêché pour un successeur. Pourquoi?



---

Des méta données  
+  
Surtout de la  
présentation

## Exemple de html



---

# Problèmes liés à HTML

- L'affichage d'un document est fortement dépendant de l'interprétation qu'en fait le navigateur
- Il est nécessaire de disposer de plusieurs versions du document en fonction du média de rendu
- L'indexation de documents ne peut se faire que sur la partie textuelle
- ***Document et pas donnée***



---

# SGML et le balisage structurel

- Il fallait passer d'un de balisage de *présentation* à un ***BALISAGE STRUCTUREL***
- XML comme SGML dont il est un descendant utilisent un balisage structurel
- SGML : Standardized Generalized Markup Language
  - Très utilisé en documentation technique
  - Airbus: la doc doit être précise et non ambiguë
  - SGML + vocabulaire contrôlé: ontologie
- Ce sont des *métalangages* de *description* et *d'échange de documents structurés*
  - Métalangage: possibilité de définir des « dialectes » dans des domaines particuliers



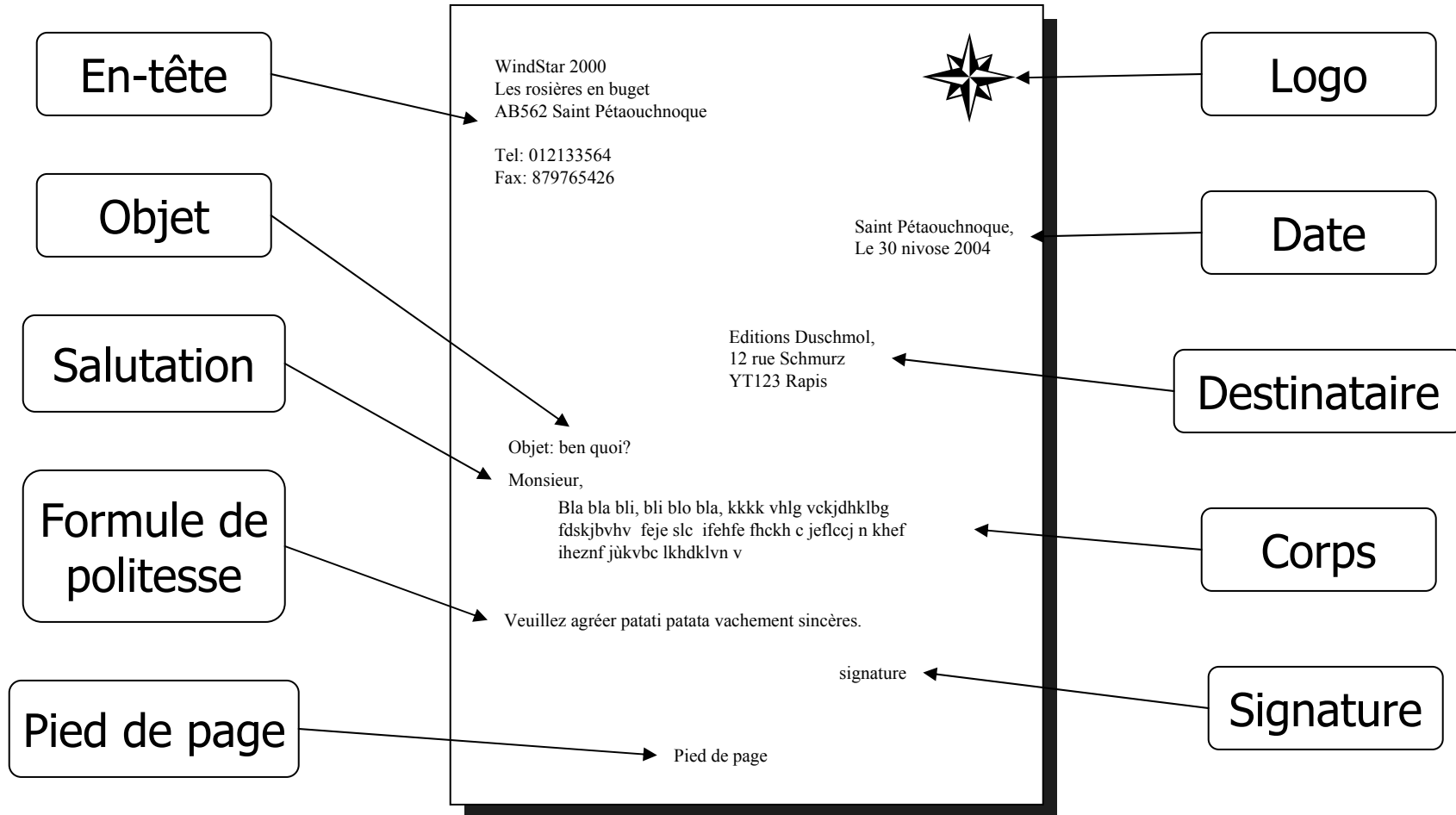
---

# XML contre SGML

- SGML norme ISO 8879:1986
- Très utilisé dans l'industrie pour de grandes documentations techniques.
- Trop complexe pour une utilisation « grand public » ou dans des domaines moins exigeants sur la précision
- SGML: trop de trucs compliqués et inutiles
- XML utilise 10% de SGML pour représenter efficacement la plupart des besoins des applications



# Exemple de document



# Représentation XML

## **<lettre>**

```
<entete>
  <logo loc="logo-graph.vml"/>
  <adresse>
    &abrev-adresse;
  </adresse>
</entete>
<destinataire>
  <nom> Mr Schnock </nom>
  <adresse>
    <rue>
      rue des églantiers
    </rue>
    <ville>
      Saint Glin
    </ville>
  </adresse>
</destinataire>
<objet> bla bla </objet>
...
```

...

```
<date>
  30 Nivose 2004
</date>

<salutation>
  Monsieur,
</salutation>

<corps>
  <para>
    Ici le premier paragraphe
  </para>
  <para>
    et là le deuxième
  </para>
</corps>

</lettre>
```

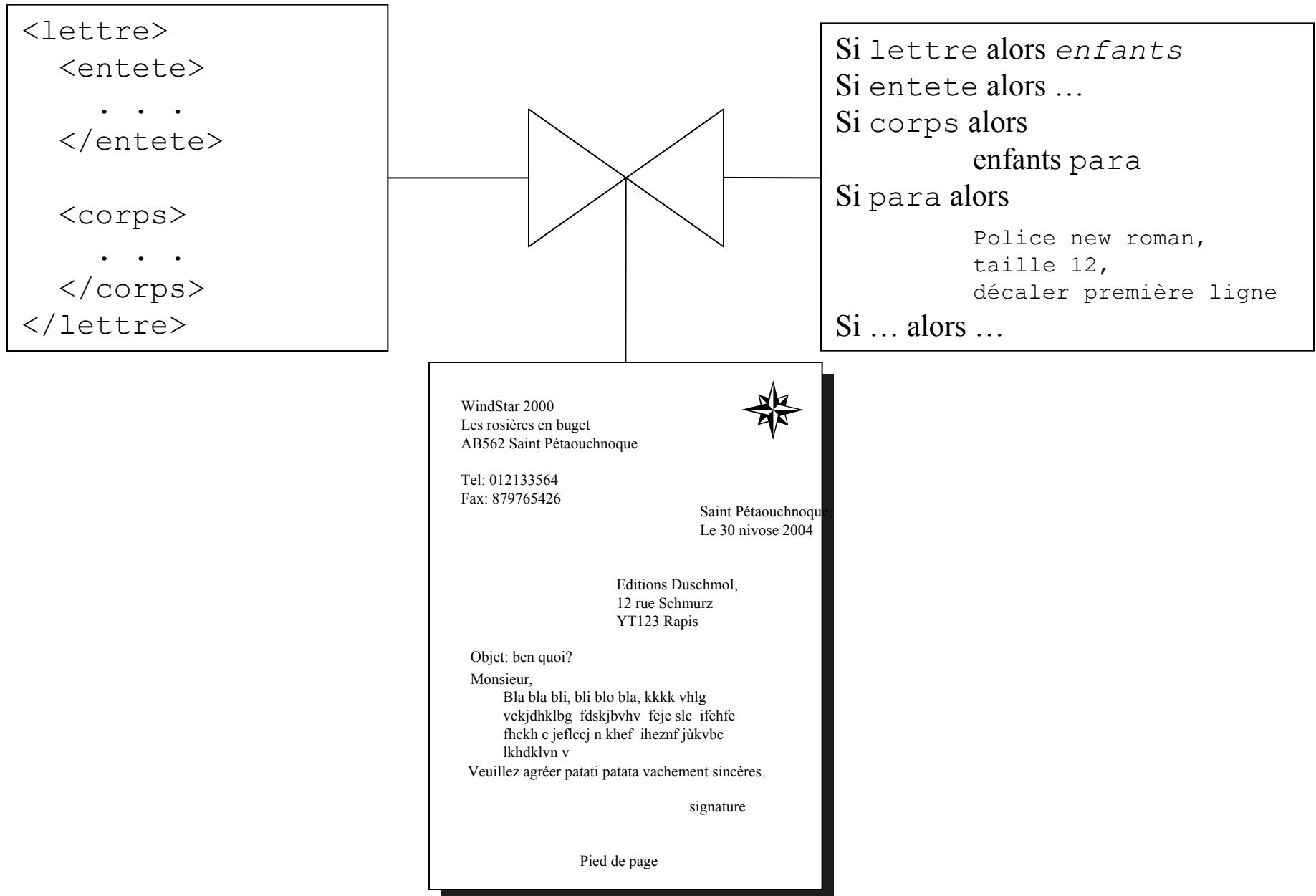
---

# Points importants

- La représentation de cette lettre en XML ne comporte aucune indication sur sa mise en page
  - Les aspects graphiques ou typographiques sont absentes du source XML
  - Ces aspects seront définis par l'intermédiaire d'une *feuille de style*
- Une feuille de style est un *ensemble de règles* pour spécifier la *réalisation concrète* d'un document sur un *média* particulier



# Principe de fonctionnement des feuilles de style



# Exemples d'applications

---

# Principe général

- Pour une application particulière
- On se définit une syntaxe: un dialecte XML
- On définit la sémantique de ce dialecte
  
- Pour me demander un rendez-vous, il faut m'envoyer le document xml-rdv du type suivant

```
<rdv><d><n>$x</n><p>$y</p></d>  
<h232>$z</h232><p>$p</p></rdv>
```

Où \$x est votre nom, \$y votre prénom, \$z la date et l'heure du rdv au format ISO... et \$p optionnel, un lieu de rdv.



---

# Exemples d'applications XML

- XHTML
- MathML
- SVG
- XSL
- SOAP
- WSDL
- XML Schema



---

# XHTML = HTML avec un syntaxe XML

- Reformulation de HTML en tant qu'application XML
  - En gros: on ferme ce qu'on a ouvert...
- Intérêt
  - Syntaxe plus rigoureuse
  - Importation de fragments de documents d'autres domaines nominaux
  - Possibilité d'utiliser les applications XML standard



---

# MathML : les maths en XML

- Permettre l'échange et le traitement d'expressions mathématiques sur le Web
- Insertion aisée d'expressions mathématiques dans des documents HTML ou XML
- Communication d'expressions entre applications au plan *sémantique*



---

# SVG : le graphique 2D en XML

- Langage de description de graphiques 2D
- Graphiques vectoriels
- Interactifs et dynamiques
  - Animations déclaratives
  - Programmation ECMAScript
- Recommandation du 04/09/2001



---

# SOAP : calcul distribué en se passant du XML

- Simple Object Access Protocol
- SOAP 1.1 : soumission au W3C du 08/05/2000
- Protocole d'échange de données entre applications distantes
- Adapté pour être utilisé au-dessus du protocole HTTP (méthode POST)
- Structure d'un message SOAP
  - Enveloppe **Envelope**; Entête **Header**;  
Corps **Body**



# Exemple

```
<env:Envelope
xmlns:env='http://schemas.xmlsoap.org/soap/envelope/'
env:encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'>
  <env:Header>
    <t:Transaction
xmlns:t='http://stock.org/registry/updPriceList/'>
      env:mustUnderstand='1'> 5
    </t:Transaction>
  </env:Header>
  <env:Body>
    <m:GetLastTradePrice
xmlns:'http://stock.org/registry/updPriceList'>
      <symbol>DEF</symbol>
      <company>DEF Corp</company>
    </m:GetLastTradePrice>
  </env:Body>
</env:envelop>
```



# Avantages de XML

---

# Echange et partage d'information

- En XML, une communauté d'auteurs invente librement les balises qui lui paraissent utiles pour représenter les informations qu'ils comptent échanger ou partager
- Exemple: diverses façons de représenter une date
  - `<date> 5 Janvier 2000 </date>`
  - `<date>  
    <a>2000</a><m>01</m><j>05</j>  
</date>`
  - `<date format='ISO-8601'> 2000-01-05 </date>`
- Exemples
  - Commandes en commerce électronique ou toute transaction
  - Publication et recherche d'information dans un domaine industriel comme l'industrie des biotechnologies



---

# Interopérabilité des outils de traitement

- Existence d'outils pour données XML
  - Parseurs, éditeurs, browser...
- Conséquences:
  - Un serveur de documents XML est susceptible de répondre à l'ensemble des besoins d'une organisation.
  - Un seul éditeur permet de traiter l'ensemble des données d'une organisation.



---

# Modularité et réutilisation

- Chaque utilisateur est libre de définir ses propres structures de document
- Il peut aussi se conformer à des structures types, appelées DTD
- Chaque communauté peut ainsi proposer des structures normalisées
- La conformité à une DTD permet l'automatisation des traitements et assure une possibilité de contrôle de validité



---

## Accès à des sources d'information hétérogènes

- L'interrogation et l'échange de données entre systèmes d'information hétérogènes est souvent complexe
- XML contribue à résoudre ce problème
  - format d'échange normalisé indépendant de toute plateforme
- L'indexation et l'interrogation de grosses bases documentaires
  - informations structurelles en plus d'informations textuelles.



# Zoom sur le langage

---

# Exemples de documents XML

```
<d/>
```

```
<document> </document>
```

```
<document> Bonjour! </document>
```

```
<document>  
  <salutation> Bonjour! </salutation>  
</document>
```

```
<?xml version="1.0" standalone="yes" ?>  
<document>  
  <salutation> Bonjour! </salutation>  
</document>
```



---

# XML 1.0

## Structure d'un élément

- Un élément est de la forme:

`<nom attr='valeur'> contenu </nom>`

- `<nom>` est la *balise d'ouverture*
- `</nom>` est la *balise de fermeture*
- [ éléments vides, indifféremment `<nom> </nom>` ou `</nom>` ]
- `contenu` est le contenu d'un élément ☺
  - composé d'une liste (peut-être vide) de texte, d'autres éléments, d'instructions de traitement et de commentaires
- `attr='valeur'` représente un *ensemble* éventuellement vide d'*attributs*, c'est à dire de paires (nom,valeur). Un élément ne peut posséder qu'un seul attribut de nom donné



---

# Exemples d'éléments

- `<a></a>`                      `</a>`
- `<a>Bonjour comment va?</a>`
- `<a><b>...</b><b>...</b><a>...</a></a>`
- `<a><b>...</b>Bonjour<b>...</b>Salut</a>`
  
- Contenu d'un élément = Forêt d'éléments ou de texte
- Texte UNICODE: peut représenter n'importe quel alphabet (russe, hébreu, arabe, japonais, chinois...)



# XML 1.0

## Contrainte sur les noms (détail)

- Un nom d'élément ou d'attribut est une suite non vide de caractères pris parmi
  - les *caractères alphanumériques*; le tiret-souligné (*underscore*); le signe *moins*; le *point*; le caractère *deux-points* (:) sens particulier
- qui doit satisfaire les contraintes suivantes
  - le premier caractère doit être alphabétique ou un tiret-souligné
  - les trois premiers caractères ne doivent pas former une chaîne dont la représentation en lettres minuscules est "xml".

Exemples de noms d'éléments	
corrects	incorrects
<code>_toto</code>	<code>1998-catalogue</code>
<code>Nom_société</code>	<code>XmlSpécification</code>
<code>xsl:rule</code>	<code>nom société</code>
<code>X.11</code>	



---

# XML 1.0

## Syntaxe des attributs

- Un attribut est une paire `nom='valeur'` qui permet de caractériser un élément. Un élément peut avoir plusieurs attributs. Dans ce cas, les paires `nom='valeur'` seront séparées par un espace.
  - `<rapport langue='fr' dern-modif='08/07/99'>`
  - `<annuaire generator='SQL2XML V2.0'  
update='07.08.99'>`
- La *valeur* d'un attribut est une chaîne encadrée par des guillemets (") ou des apostrophes simples ('). Une valeur d'attribut ne doit pas contenir les caractères ^, % et &.
- Un élément a un ensemble d'attributs (ordre n'a pas de sémantique pour les attributs)



---

# XML 1.0

## Document *bien formé*

- Un document XML doit représenter un *arbre d'éléments*
  - Il existe dans un document un et un seul élément père qui contient tous les autres. C'est la *racine* du document.
  - Un élément distinct de la racine est totalement inclus dans son père
    - `<p> <b> bla bla </p> bla </b>` NON!
- On dit qu'un document XML doit être bien formé



# Typage de XML

---

# Structure d'un document

- Un document XML se compose
  - d'un *prologue*, éventuellement vide

```
<?xml version="1.0" standalone="yes" ?>
```

- d'un *arbre d'éléments*

```
<document>  
  <salutation> Bonjour! </salutation>  
</document>
```

- de *commentaires* et *d'instructions de traitement*, facultatifs



---

# Le prologue contient

- Une *déclaration XML*, facultative

```
<?xml version="1.0" encoding="ISO-8859-1"
standalone="yes">
```

- indique au processeur qui va traiter le document:
  - la version du langage XML utilisée
  - le codage de caractères utilisé
  - l'existence de déclarations extérieures au document

- Une *déclaration de type de document*,  
facultative

```
<!DOCTYPE exemple SYSTEM "exemple.dtd" [
déclarations ]>
```

- indique la structure particulière à laquelle doit se conformer un document



---

# Document bien formé sans typage

- Exemple

```
<?xml version="1.0"  
  standalone="yes" ?>  
<document>  
  <salutation>  
    Bonjour!  
  </salutation>  
</document>
```



---

# Documents *valides*

- Un document est dit *valide* si:
  - son prologue contient une déclaration de type de document
  - son arbre d'éléments respecte la structure définie par la déclaration de type

```
<?xml version="1.0" encoding="ISO-8859-1"
  standalone="yes" ?>
<!DOCTYPE document [
  <!ELEMENT document (salutation)>
  <!ELEMENT salutation (#PCDATA)>
]>
<document>
  <salutation> Bonjour! </salutation>
</document>
```



---

# Exemple de DTD simple

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
<!DOCTYPE document SYSTEM "accueil.dtd">
<document>
  <salutation> Bonjour! </salutation>
</document>
```

```
<!-- fichier accueil.dtd. Exemple de DTD simple -->
<!-- Auteur:      -->
<!-- Date:       -->

<!-- la déclaration XML n'est pas obligatoire dans une DTD -->
<!-- permet de s'assurer que les documents qui la référence -->
<!-- utilisent la même version de XML -->
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>

<!-- Définition de l'élément racine -->
<!ELEMENT document (salutation)>

<!-- Un élément salutation ne contient que du texte -->
<!ELEMENT salutation (#PCDATA)>
```



---

# Une autre DTD un peu plus compliquée

```
<!ELEMENT note (to, from, heading, body)>  
<!ELEMENT to (#PCDATA)>  
<!ELEMENT from (#PCDATA)>  
<!ELEMENT heading (#PCDATA)>  
<!ELEMENT body (#PCDATA)>
```

Pour chaque élément, on définit une expression régulière qui définit le langage des sous éléments



---

# La même chose en XML schema – c'est du XML

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="note">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="to" type="xs:string"
          minOccurs='1' maxOccurs='1'/>
        <xs:element name="from" type="xs:string"/>
        <xs:element name="heading" type="xs:string"/>
        <xs:element name="body" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```



# Quelques gadgets importants autour de XML

---

# DOM

- Document Object Model – DOM
  - modèle de document
  - interface de programmation indépendante du langage et des plates-formes
- Permet d'accéder
  - la structure des documents (HTML et XML)
  - le contenu des documents
  - Exemple: Premier-Enfant, Premier-Enfant(Section) Suivant, Parent, Attribut(ID)...
- API définie pour trois langages
  - OMG IDL, Java, ECMAScript



---

# XPath

- Langage *d'expressions de chemin*
  - adresser des parties de documents XML
- Une expression élémentaire XPath contient
  - un axe qui spécifie la relation structurale : fils, descendants, ancêtres, frères, attributs,...
  - un test qui spécifie le type de noeud
  - des prédicats pour raffiner la sélection
- Brique de base d'autres applications XML
  - Xlink, XSLT, Xquery, ...



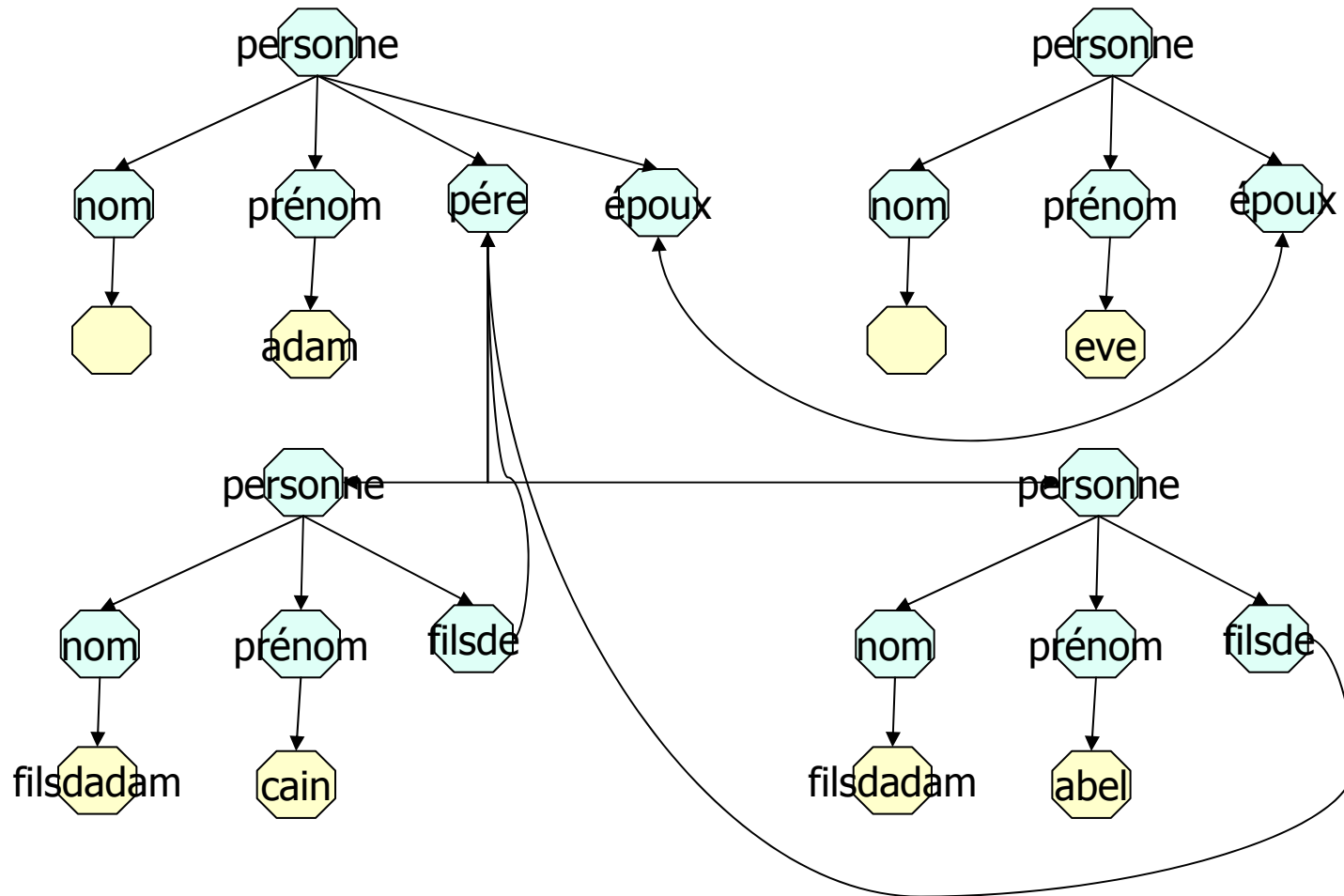
---

# XLink

- XML Linking Language
  - Hypertexte avancé
  - permet d'insérer dans les documents XML de quoi décrire les liens entre ressources Web
- Liens simples unidirectionnels à la HTML
- Liens hors document
- Liens multidirectionnels
- Comportement des liens
- XLink utilise XPath pour adresser l'intérieur des structures XML



Si on ajoute des références: cela devient un graphe



---

# Transformation de documents

- Le Web présente de multiples cas de transformation
  - Formatage
  - Réutilisation de document
  - Adaptation aux outils de présentation
  - Conversion de format, par exemple XML vers HTML
- Trois approches
  - Des programmes utilisant le DOM
  - Les feuilles de transformation XSLT
  - Le langage de requêtes XQuery



---

# XSLT

- Langage de transformation
  - « le Perl de XML »
- Une feuille de transformation XSLT contient un ensemble de règles pattern/template
- Pattern
  - Contexte structurel dans l'arbre source
  - Expression XPath
- Template
  - Un fragment du résultat à produire pour le pattern correspondant
- Principe
  - Quand le pattern est reconnu dans le document source, le template correspondant est engendré dans le document résultat



---

# XQuery

- « le SQL de XML »
- Pas complètement fixé
- Les requêtes XQuery
  - Peuvent sélectionner des documents entiers ou des sous-arbres qui répondent à la requête
  - Peuvent construire des documents nouveaux fondés sur ce qui est sélectionné



---

# Exemple de XQuery

- FLW prononcer "flower"
  - Dans le style du SFW de SQL

- Exemple

```
FOR $p IN document("bib.xml")//publisher
LET $b := document("bib.xml")//book[publisher = $p]
WHERE count($b) > 100
RETURN $p
```

- \$p : parcourt la séquence des éléments publisher
- \$b : contient la séquence des book associés à \$p
- WHERE filtre la liste des tuples (\$p,\$b)
- RETURN construit pour chaque tuple le résultat



---

# XML et traitement de données

- Compilateurs Xquery non persistents sur le marché
- SGBD relationnelles adaptées pour XML
  - Oracle a maintenant un XML-Box, IBM...
- SGBD objets refaits à la mode XML
  - Excellon vient de Object Design
- SGBD « natif » XML
  - Tamino de Software AG (recyclage?)
- Repository XML natif
  - Xyleme – on verra la prochaine fois



Merci