Bases de Données XML Natives

Préparé par
M.G. BELKASMI

Modèles

- Modèle relationnel
  - Modèle simple, puissant avec des fondements théoriques bien connus, stockage efficace, langage de requêtes, cohérence, …
- Modèle des Fichiers
  - Information pas ou peu structurée, pas de schéma des données, pas de langage de requêtes, pas d'optimisation, de cohérence, Beaucoup de souplesse
- Modèles de données semi-structurés
  - Intermédiaire entre modèle relationnel et modèle de fichier, présence d'un schéma mais souple, possibilité de langages de requêtes

XML et BD relationnelles

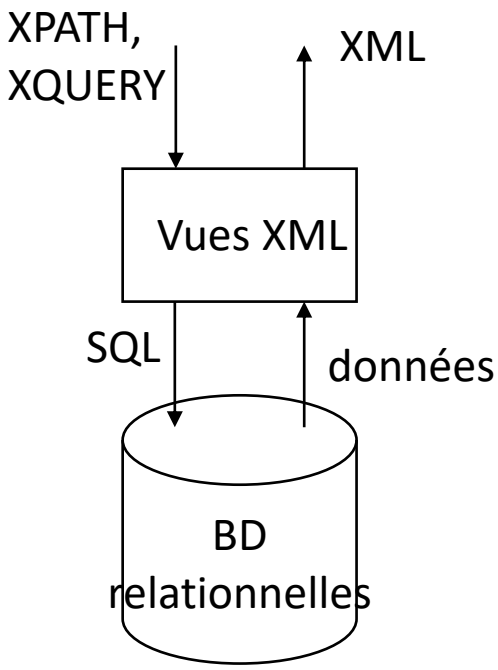
SQL	XSLT, XPATH, XQUERY
Modèle relationnel	Modèle XPATH
JDBC/ODBC/SQL-CLI	DOM et SAX API
Bases de données relationnelles	Document XML

BD XML

- BD XML = forêt de documents XML
- Schéma de BD =
  - Schéma XML si existant
  - Guide de données :
    - Schéma faible généré à partir d'un ensemble de documents par union des arbres de structure décrivant tous les cheminements possibles dans la collection et par typage des données en texte
  - Schéma plus flexible que schéma relationnel

Publication XML de données relationnelles

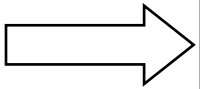
- Exporter une relation dans un format canonique (simple)
  - Intéressant pour publication de données sur le web ou intégration de données
- Exporter une base de données dans un format prédéfini (complexe)
  - Ressemble à du stockage XML dans un SGBDR
  - Spécification du mapping :Besoin d'un langage, pas de standard (un par vendeur)
  - Schéma annoté



Format canonique

HOTELS

Nom	Catégorie	Adresse	prix
Napoléon	3	Paris	600
Gare	2	Evry	300



```
<row>
    <nom>Napoleon</nom>
    <categorie>3</categorie>
    <adresse>Paris</adresse>
    <prix>600</prix>
</row>
<row>
    <nom>Gare</nom>
    <categorie>2</categorie>
    <adresse>Evry</adresse>
    <prix>300</prix>
</row>
```

Schéma annoté

- Spécification du schéma de sortie souhaité annoté par la provenance des données

```
<xs:element name="Acteur">
        <xs:complexType>
                <xs:sequence>
                        <xs:element name="nom" type="xs:string"/>
                        <xs:element name="prenom" type="xs:string"/>
        </x
</xs:element>
<xs:element n
        <xs
</xs:element>
```

```
<xs:element name="Acteur"  sql:relation="Acteur">
   <xs:complexType><xs:sequence>
      <xs:element name="nom" type="xs:string"  sql:field="nom"/>
      <xs:element name="prenom" type="xs:string" sql:field="prenom"/>
      <xs:element ref="Film" sql:relationship="ActDist" sql:relationship="DistFilm"/>
   </xs:sequence></xs:complexType>
</xs:element>
<xs:element name="Film"  sql:relation="Film" sql:field="titre">
   <xs:attribute name="annee" type="xs:dateTime"  sql:field="annee"/>
</xs:element>
```

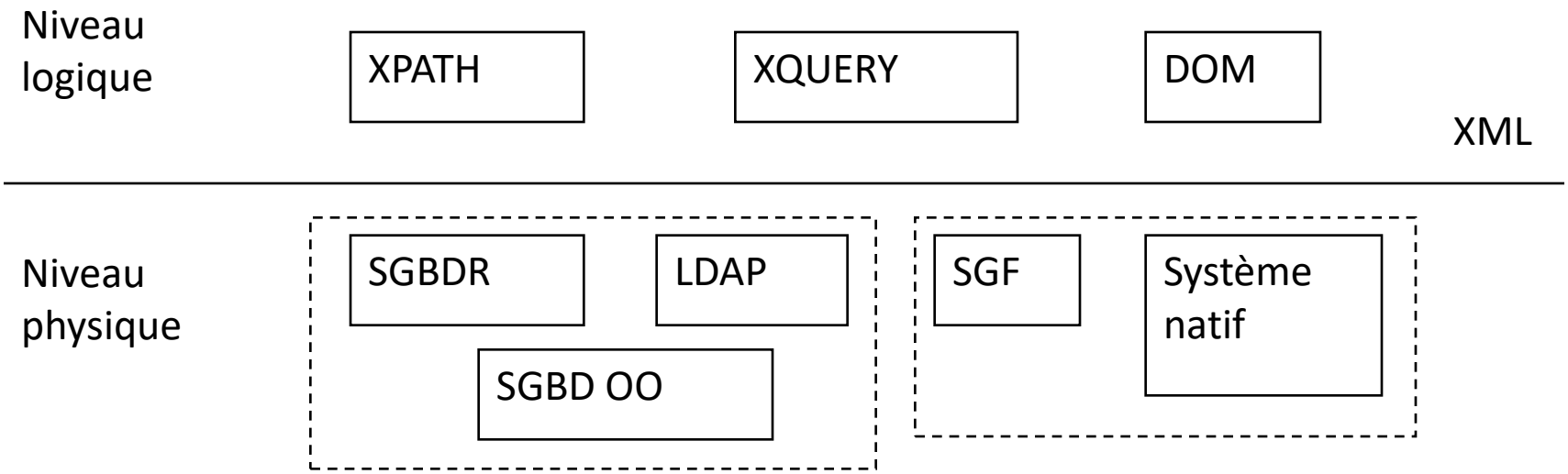
Schéma annoté

```
<xs:annotation>
<xs:appinfo>
<sql:relationship name=ActDist
        parent=Acteur parent-key=idact
        child=Distribution child-key=idact
</sql:relationship>
</xs:appinfo>
</xs:annotation>
```

```
<xs:annotation>
<xs:appinfo>
<sql:relationship name=DistFilm
        parent=Distribution parent-key=idfilm
        child=Film child-key=idfilm
</sql:relationship>
</xs:appinfo>
</xs:annotation>
```

Stockage de données XML

Système XML


Niveau logique
XPATH
XQUERY
DOM
XML
Niveau physique
SGBDR
LDAP
SGBD OO
SGF
Système natif

Eléments de choix

- Données :
  - Plat vs structuré; petit ou volumineux; avec ou sans schéma, ...
- Requêtes :
  - Avec ou sans maj; accès full-text; navigationnel; relationnel (jointures, ...)
- Besoins applicatifs
  - Transactions, contrôle de concurrence, réplication, ...

Principes de stockage XML

- Plat :
  - Un document XML est stocké dans un BLOB
  - Simple, mais pas de requêtes possibles et maj difficiles
- Natif
  - Définir un nouveau SGBD adapté au stockage de documents XML
  - Redéfinir toutes les fonctions classiques d'un SGBD (transactions, concurrence, …)
- Par conversion ou « mapping »
  - Utiliser un SGBD existant (souvent relationnel) pour stocker des documents XML
  - Nécessite certaines extensions (index spécifiques par exemple)

Systèmes natifs

- Besoins
  - Représentation concise des documents
  - Support efficace des API XML
  - Possibilité de maj données et structure
- Correspondance entre structure d'arbre et des pages physiques
  - De nombreuses possibilités

Indexation

- Un placement physique ne peut être optimal pour toutes les requêtes possibles
- Besoin de plusieurs index
- XML a besoin d'index
  - Sur les valeurs
  - Sur la structure (navigation)
  - Full-text (mots-clés)

Systèmes à base de mapping

- Stockage : convertir modèle de données XML vers graphe, relations, objets
- Chargement de données : convertir données XML vers arcs, tuples, objets
- Réécriture de requêtes : transformer requêtes XML vers requêtes cibles
- Transformation du résultat : système cible vers XML

Stockage de données XML vers relationnel

- Défini par l'utilisateur
- Générique (fixe)
- Dirigé par les données
- Dirigé par le schéma
- Basé sur un modèle de coût (adaptation aux besoins réels de l'application)

Mapping défini par l'utilisateur

- Supporté par la majorité des SGBDR commerciaux
- Utilisateur spécifie comment transformer éléments en relations
- Flexible mais
  - Nécessite de connaitre XML et BD Relationnelles
  - Beaucoup de solutions possibles (laquelle choisir ?)
  - Maj des données implique maj du mapping

En Résumé

XML <> BD SQL

- Conversion nécessaire
  - Impact sur les performances
  - Traduction vers de multiples tables.
  - Risque de perte d'information
    - Document 1 = > BD
    - BD => Document 2
    - Risque que Document 1 <> Document 2

En Résumé

XML <> BD SQL

- L'ordre des nœuds XML a un sens
- Agrégation de données provenant de sources diverses
  - Structure irrégulière , Schéma non défini au préalable => difficulté de traduction vers bases SQL
- Dans le Mapping XML <-> Bd SQL, une mise à jour de nœud peut impacter plusieurs enregistrements dans plusieurs tables différentes

En Résumé

BD XML Natives

- Performances de XML grâce à l'indexation
- Gestion de la sécurité pour l'accès au données
- Gestion des transactions pour les mises à jour
- Automatisation des traitements
- Accès aux données par des requêtes XQuery
- Gestion des backups.

Offre NXD

- BaseX. Open Source, XQuery Update et Full Text
- DB-XML (Sleepycat).
- eXist
- IXIASOFT TEXTML Server, XPath, Full Text
- MarkLogic MarkLogic, XQuery et Full Text
- Qizx
- xDB EMC², XQuery Update et Full Text
- Xindice (retiré par Apache)
- Oracle Berkeley DB XML. Open Source basé sur Oracle Berkeley DB, parser, XQuery Update et Full Text

eXist

- *eXist* est une base de données *Open Source* native *XML*, entièrement écrite en *Java*.
- *eXist* s'intégre très facilement dans une application *Java*
- *eXist* peut également être utilisée dans de nombreux autres langages grâce à des API
- *SOAP* est également supporté dans le mode servlet.

eXist

eXist utilise plusieurs méthodes d'indexation :

- un regroupement en collections de documents (similaire à un système de fichiers), une série de *DOM* persistants,
- une indexation structurelle (basée sur les relations type parent/enfant…) pour les nœuds d'élément et d'attribut
- une indexation textuelle pour les nœuds texte et les valeurs d'attribut.
- Les mises à jour (contrôlées par un mécanisme d'accès multi-utilisateur) peuvent être effectives au niveau du document ou d'un nœud spécifique.

eXist

- *eXist* permet de formuler des requêtes via *XPath*/*XQuery*
- *eXist* fournit des extensions à *XPath*, comme des fonctions adaptées à la recherche textuelle et aux concepts plus orientés bases de données.
- Des modules *XQuery* complémentaires peuvent être écrits soit directement en *XQuery* soit en *Java*.
- Les technologies *XML XInclude*, *XPointer*, *XUpdate* sont également partiellement supportées.

eXist

- Site du projet

http://exist.sourceforge.net/

(http://exist-db.org/exist/apps/homepage/index.html)