

Intégration

Données semi-structurées

Modèle de données semi-structurées

- **Objectif** : améliorer l'intégration du texte sous des formats divers (HTML, SGML, Latex, ...), du son, du graphique, des images, de la vidéo, etc.
- Utiliser les modèles de données des SGBD pose des problèmes
 - les données peuvent ne pas être conformes au schéma (==> valeurs nulles, difficultés de traitement et ambiguïtés)
 - l'évolution fréquente de la structure de données conduit à des évolutions de schéma pas toujours maîtrisées.
 - les données du Web sont indexées par des moteurs de recherche dont les services sont limités, l'interrogation souvent imprécise.

➡ Nécessité d'un modèle général et souple, «sans schéma», avec un langage de requêtes associé : modèles semi-structurés

Caractéristiques

- Ces données sont « autodescriptives », cad. contiennent leur propre type, et peuvent être interprétées indépendamment de toute autre information.
- Il n'y a plus de séparation entre données et types (en général, on définit un type puis on y attache des valeurs, ou des données).

- Exemples :

Guide de restaurants : chaque restaurateur décrit son restaurant à son idée. On a généralement des informations sur la carte, les prix, l'adresse, etc. Quelques régularités, beaucoup d'irrégularités (pas d'uniformité).

Intégration de données hétérogènes : structures et conventions différentes, informations absentes, etc.

Génome, bibliothèques de programmes, commerce électronique, etc...

Caractéristiques

- Structure irrégulière :
nécessité de modéliser et d'interroger ces structures
- Structure implicite :
ex: un document électronique a un texte et une grammaire. On peut parser pour détecter des informations et des relations entre ces informations. Mais cela nécessite des outils spéciaux.
- Structure partielle :
il manque des informations, certaines informations sont stockées hors de la base, et ne sont pas structurées.
- Structure indicative (et non pas contrainte, comme dans les BD) :
pas de typage, pas de schéma (trop contraignants)

Caractéristiques

- Un schéma a posteriori, et non a priori :
en semi-structuré, le schéma est souvent postérieur aux données (c'est l'inverse en BD). Mais parfois, on peut suggérer ou guider (ex: pages personnelles).
- Un schéma très vaste (on ne sait pas tout),
=> il faut pouvoir l'interroger
- Un schéma ignoré (on parcourt tout à la recherche d'une chaîne)
=> pas de SQL, il faut trouver d'autres langages
- Un schéma qui évolue rapidement
=> à prendre en compte dans le langage de requêtes
- La distinction entre schéma et données est peu claire.
- Le typage est flou.

Caractéristiques

- Un schéma a posteriori, et non a priori :
en semi-structuré, le schéma est souvent postérieur aux données (c'est l'inverse en BD). Mais parfois, on peut suggérer ou guider (ex: pages personnelles).
- Un schéma très vaste (on ne sait pas tout),
=> il faut pouvoir l'interroger
- Un schéma ignoré (on parcourt tout à la recherche d'une chaîne)
=> pas de SQL, il faut trouver d'autres langages
- Un schéma qui évolue rapidement
=> à prendre en compte dans le langage de requêtes
- La distinction entre schéma et données est peu claire.
- Le typage est flou.

Modèles

Les modèles semi-structurés utilisent des graphes annotés pour représenter les données.

Les différents modèles diffèrent par

- l'endroit où sont situées les annotations (arêtes et/ou nœuds)

- l'existence ou non d'un ordre sur les fils d'un nœud

- la façon de représenter le partage d'information

Ex : OEM : annotations sur arcs et feuilles, pas d'ordre

XML : annotations sur les nœuds et feuilles, existence d'un ordre

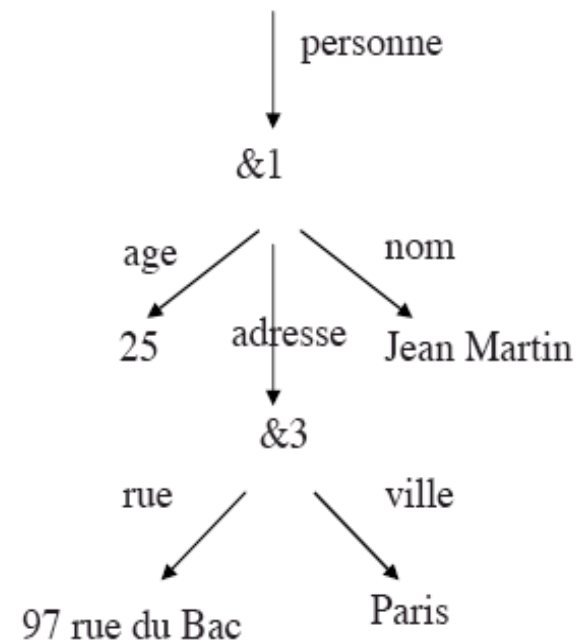
UnQL : annotations sur les arcs, pas d'ordre

Modèles

OEM Object Exchange Model

```
<personne id=&1  
  nom= " Jean Martin "  
  age=25  
  adresse=&3  
>
```

```
<adresse id=&3  
  rue= " 97 rue du Bac "  
  ville= " Paris "  
>
```



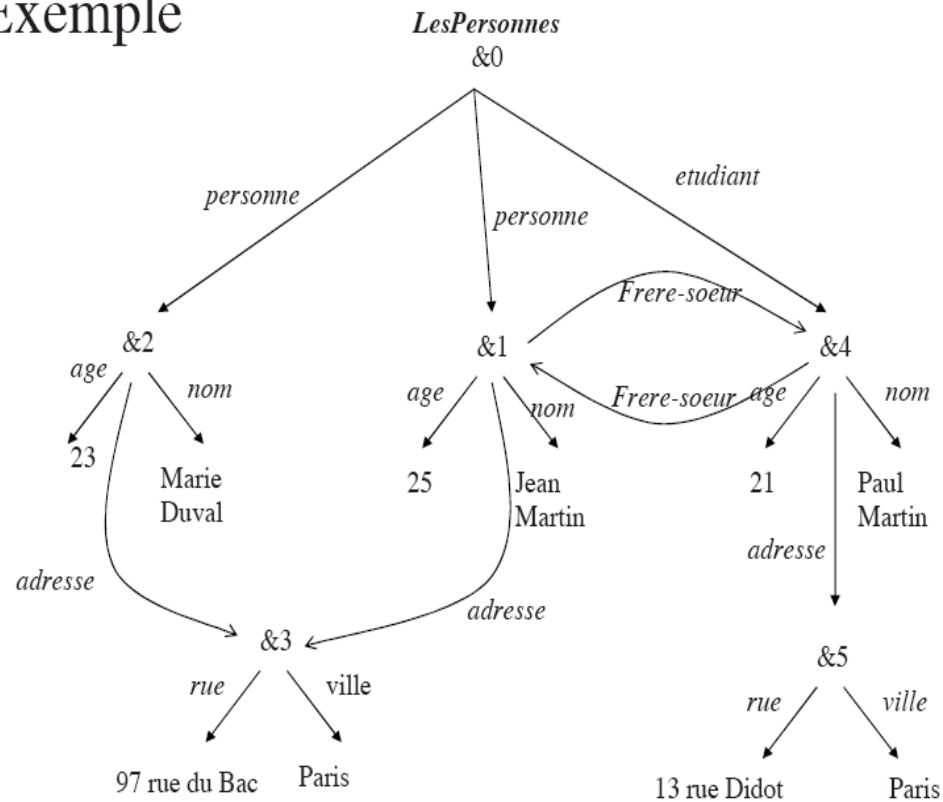
Modèles

OEM Object Exchange Model - Interrogation

- Les langages classiques ne sont plus appropriés:
 - la structure exacte des données n'est pas connue
 - la structure est irrégulière
 - des concepts similaires sont représentés différemment
 - ensembles hétérogènes
- Rester proche du style SQL/OQL
 - ⇒ l'interrogation est navigationnelle, (parcours de graphe) et s'appuie sur des expressions de chemins.

Modèles

Exemple



```

<LesPersonnes id=&0
  <Personne id=&1
    nom="Jean Martin"
    age="25"
    frere-soeur=&4
    adresse=&3 />
  <Personne id=&2
    nom="Marie Duval"
    age="23"
    adresse=&3 />
  <Etudiant id=&4
    nom="Paul Martin"
    age="21"
    frere-soeur=&1
    adresse=&5 />
  <Adresse id=&3
    rue="97 rue du Bac"
    ville="Paris" />
  <Adresse id=&5
    rue="13 rue Didot"
    ville="Paris" />
/>

```

Modèles

OEM Object Exchange Model - Interrogation

Expression de chemin (suite d'étiquettes) :

Une expression de chemin est une séquence $Z.l_1.l_2\dots.l_n$, où les l_i sont des étiquettes et Z est un nom d'objet (ou une variable dénotant un objet).

Exemple : LesPersonnes.nom.adresse

Chemin de données :

Un chemin de données est une séquence $O_0, l_1, O_1, \dots, l_n, O_n$, où les O_i sont des objets, et pour chaque i , il existe un arc étiqueté l_i entre O_{i-1} et O_i .

Partant d'un objet $Z=O_0$, il peut exister plusieurs chemins de données correspondant à l'expression $Z.l_1.l_2\dots.l_n$.

Modèles

OEM Object Exchange Model - Interrogation Langage LOREL

- Syntaxe et sémantique proche de SQL
SELECT <liste d'attributs>
FROM <liste de variables de nœuds>
WHERE <predicat>
- Clause FROM : produit cartésien de nœuds
- Clause WHERE : élimine certains nœuds
- Clause SELECT : projection

Modèles

OEM Object Exchange Model - Interrogation Langage LOREL

- *Quel est l'âge de Jean Martin ?*

```
SELECT personne.age FROM Lespersonnes  
WHERE personne.nom = "Jean Martin "
```

Ou

```
SELECT p.age FROM Lespersonnes.personne p  
WHERE p.nom = "Jean Martin "
```

Les variables de noeuds s'utilisent comme les variables de relations.

- *Noms des personnes habitant Paris ou ayant 25 ans ?*

```
SELECT P.nom FROM Lespersonnes.personne P  
WHERE P.#.ville = "Paris" OR P.age = "25"
```

Prendre en compte les valeurs nulles, les chemins manquants.

Modèles

XML

Format/langage standard pour les données semi-structurées du Web

- Objectif : un formalisme pour la description et l'échange de données sur le Web, version simplifiée de SGML (DTD facultative)
- Consortium W3C (Oracle, IBM, MS, MIT, INRIA....) 1996
- Début en 1996, version 1.0 en 1998, nombreux développements depuis.
- Successeur de HTML, héritier de SGML

Principes de XML :

balisage structurel (SGML)

balisage défini par les auteurs : souplesse

séparer la *structure logique* des données de leur *présentation*

feuille de style (XSL) : ensemble de règles pour la réalisation sur un médium particulier

Modèles

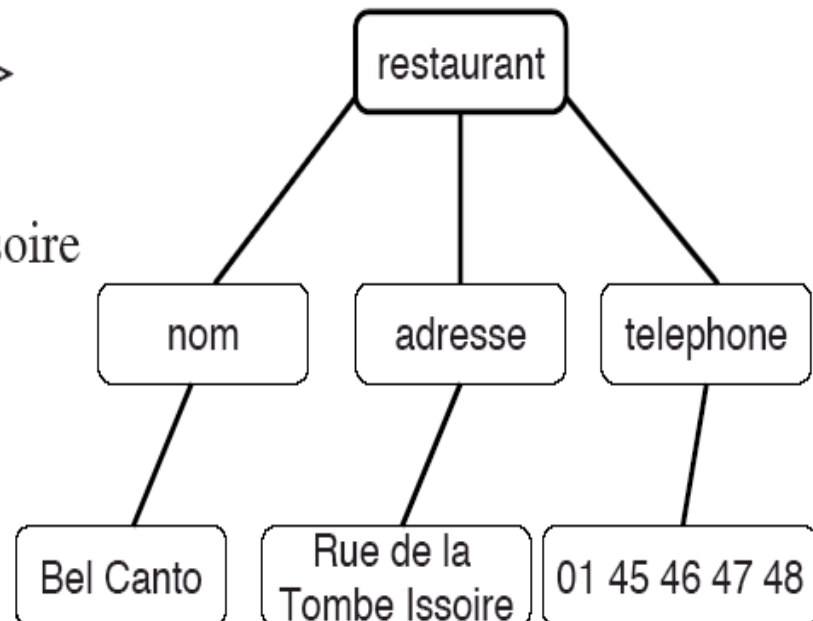
XML

- Un document XML peut se représenter sous deux formes:
 - La forme sérialisée est la forme courante (contenu marqué par des balises). Elle est utilisée pour
 - Stocker un document dans un fichier
 - Echanger des documents
 - La forme arborescente met en évidence la structure du document (facilite la conception des traitements).
 - Elle permet de spécifier des manipulations de données XML
 - Elle peut être utilisée par certaines applications qui gèrent les documents en mémoire (éditeurs XML).

Modèles

XML

```
<restaurant>  
  <nom>Bel Canto </nom>  
  <adresse>  
    Rue de la Tombe-Issoire  
  </adresse>  
  <telephone>  
    01 45 46 47 48  
  </telephone>  
</restaurant>
```



Modèles

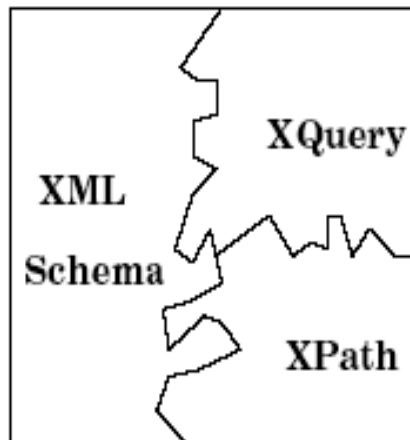
XML Interrogation Xquery

- Spécification du W3C (version 1.0, oct.2004)
 - inspiré de SQL
 - satisfait les contraintes émises (requêtes essentielles, Use Cases)
 - Construit au-dessus de XPath
- Une requête en XQuery est une expression qui
 - lit un ensemble de documents XML (ou des fragments)
 - renvoie une séquence de fragments XML bien formés
- Use Case « XMP » : Experiences and Exemplars
- Use Case « TREE » : requêtes préservant la hiérarchie
- Use Case « SEQ » : requêtes basée sur des séquences
- Use Case « R » : accès à des données relationnelles
- Use Case « SGML » : standard generalized markup language
- Use Case « TEXT » : recherche full-text
- Use Case « NS » : requêtes utilisant des espaces nominaux (namespaces)
- Use Case « PARTS » : recursive parts explosion
- Use Case « REF » : requêtes utilisant des références
- Use Case « FNPARM » : requêtes utilisant des fonctions et paramètres

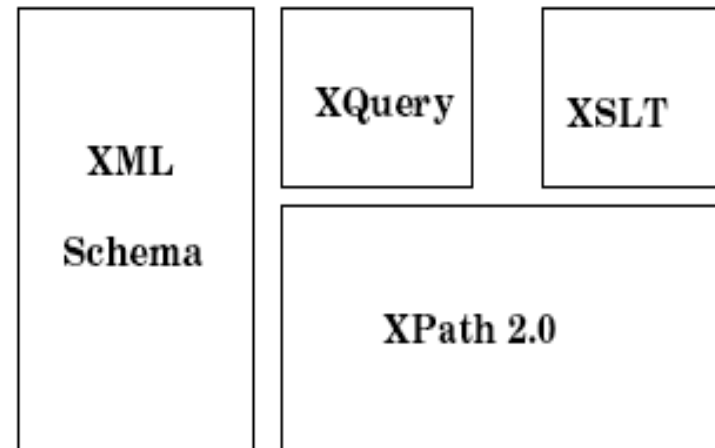
Modèles

XML Schema

Modèles et Langages:



Groupes de travail:



Stockages

Fichiers 'plats' :

- Petits documents
- Langage de requêtes : `grep`, index plein texte
- Avantages : temps de chargement/reconstruction

SGBD (objet-) relationnelle étendu avec des outils pour le traitement de documents XML :

- Définition d'un schéma relationnel pour stocker des documents XML
- Nouveau type d'attributs XML
- Interrogation avec SQL

Avantages :

- On peut traiter en même temps des données XML et des tables classiques
- Passage doux du Relationnel vers XML

Stockages

Mapping XML Objet-relationnel

Mapping via deux tables :

- une table pour stocker l'ordre, les balises et les relations parent/enfant
- une table pour les valeurs

Systemes :

- Oracle XML
- IBM DB2 XML Extender
- Microsoft OpenXML
- Excellon

Caractéristiques :

- Importation générique/guidé par le schéma
- Langages: SQL + TAD pour XML

Stockages

Bases de données XML natives (NXD)

Bases de données spécifiquement conçues pour XML

- Modèle conçu pour le stockage et l'accès à des arbres ordonnés.
- Le document XML est l'entité centrale de la base (comme une relation dans une BD relationnelle)

Avantages :

- Chargement efficace de gros documents
- Mises-à-jour efficaces
- Tamino de Software AG
- Xyleme/Natix de l'Université de Mannheim
- XIndex de Apache
- X-Hive DB
- IXIA Soft TextML