Intégration de données et XML

Groupe de recherche Bases de Données

DEA SIR

Dan VODISLAV

CNAM Paris

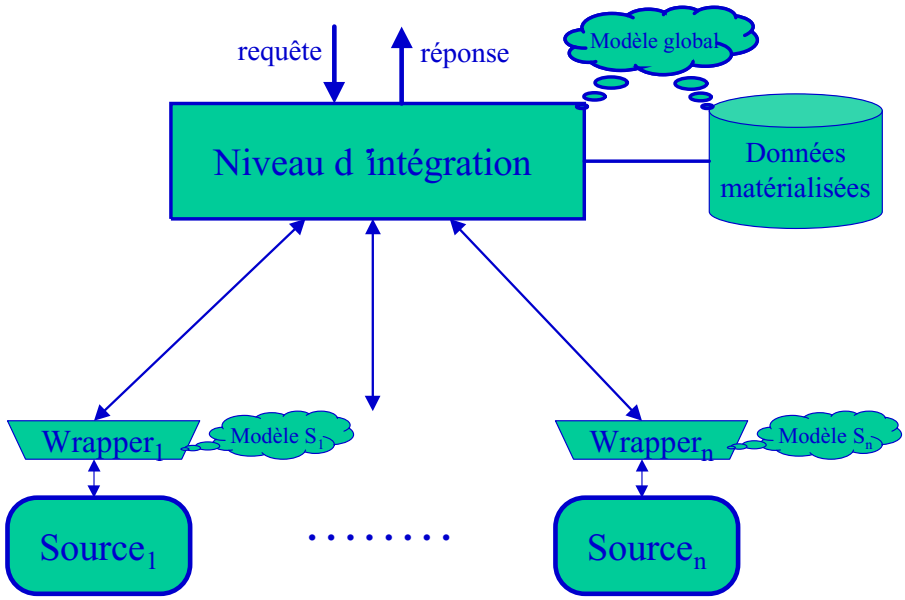
Plan

- Principes d 'intégration de données
- Classification et exemples
- Les mappings
- Intégration basée sur XML
  - XML et relations
  - Intégration de sources XML
    - STYX
    - Xyleme

1. Principes

- Intégration de données
  - nouvelles applications autour du Web : commerce électronique, portails, communautés Web, etc.
  - accès à des sources d 'informations très hétérogènes
    - bases de données
    - documents : HTML, XML, mail
    - données multimédia : vidéo, son, images
- Objectif
  - offrir un accès unique, homogène, à des sources hétérogènes
- Autonomie des sources
  - les sources préexistent et ont une vie indépendante les unes des autres
  - autonomie par rapport au système d 'intégration

Architecture générale



Composants

- Sources
  - bases de données, fichiers, formulaires web
- Wrappers (traducteurs)
  - interfaces entre les sources et le niveau d 'intégration
  - le modèle des sources : description des sources *pour l 'intégration*
- Niveau d 'intégration
  - vue uniforme, modèle global
  - matérialisation possible de données intégrées

Particularités

- Modèle de données d 'intégration
  - schéma global + description des schémas des sources
  - *mapping*: modèle global ↔ modèles sources
  - hétérogénéité des sources : format, vocabulaire, structure
  - redondances et contradictions entre sources
- Distribution des données
  - passage à l 'échelle
  - problèmes de performances
- Transformation des données
  - *wrappers* : les données des sources transformées vers le modèle global

Particularités (suite)

- Interrogation
  - traduction de requêtes : modèle global → modèle source
    - tâche complexe, dépendant du langage de description des sources
  - la traduction doit être
    - *saine* : elle donne des réponses correctes pour la requête initiale
    - *complète* : toutes les réponses de chaque source sont extraites
    - *efficace* : seules les sources pertinentes pour la requête sont choisies
  - optimisation
    - difficile d 'avoir des statistiques sur les sources pour l 'optimisation
    - les sources peuvent avoir des possibilités de traitement très différentes
    - les temps de transfert sont difficiles à estimer

2. Classification

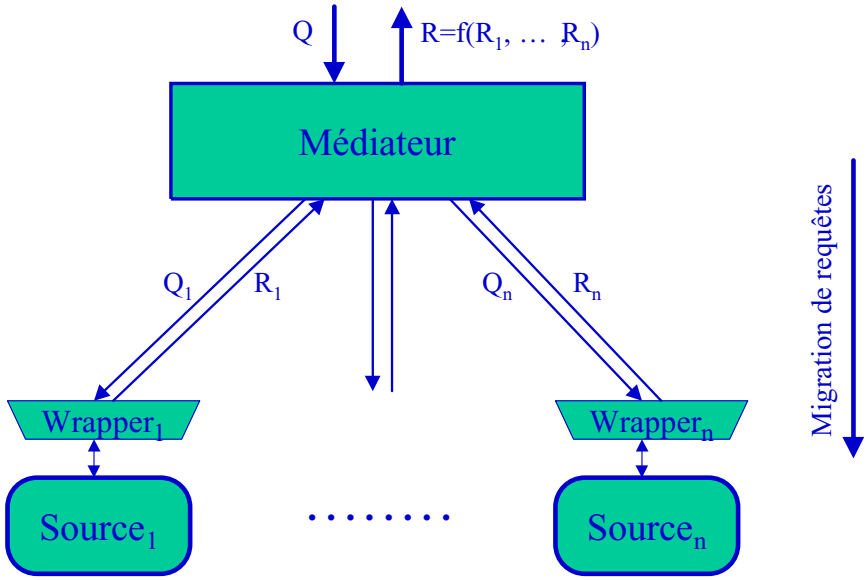
- Selon le degré de matérialisation
  - entrepôt
  - médiateur
- Selon la relation entre le modèle global et celui des sources
  - « local-as-view »
  - « global-as-view »
- Autres critères
  - le modèle global : relationnel, XML, ontologie, logique
  - le modèle de description des sources
  - le type de mapping
  - la taille : quelques sources, échelle du Web
  - la complétude, les possibilités de relaxation, ...

Médiateur et entrepôt de données

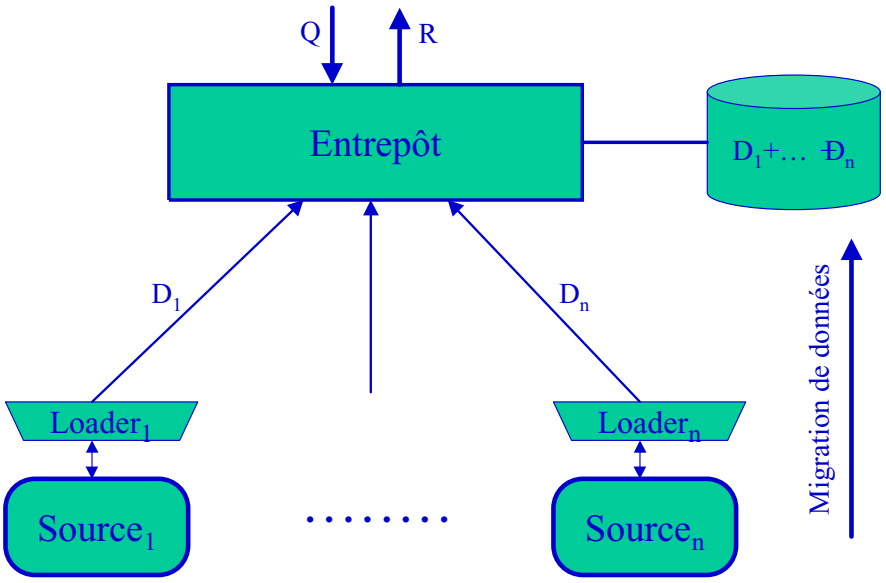
- Médiateur : accès direct aux sources
  - approche « paresseuse », pas de matérialisation
  - migration de requêtes vers les sources
  - *avantages* : cohérence, données réelles
  - *inconvénients* : performances, traduction de requêtes, capacités sources

- Entrepôt de données : accès efficace à une copie des données
  - matérialisation des sources au niveau du modèle global
  - migration de données vers l 'entrepôt
  - *avantages* : performances, personnalisation des données, versions
  - *inconvénients* : mise-à-jour, cohérence, volume de données

Architecture de médiation



Architecture d 'entrepôt



« Global-as-View »

- Le modèle global = vue sur les sources
  - élément global = f(éléments des sources)
    M = V(S_1, … S_n)
- Avantages
  - approche naturelle
  - la traduction de requêtes se fait facilement
    - « expansion » de la requête dans la vue
- Inconvénients
  - nouvelle source → modification du modèle global
    - il faut considérer l 'interaction de la nouvelle source avec les autres

« Local-as-View »

- Les sources = vues matérialisées du modèle global
  - une source décrit les données du modèle global qu 'elle peut fournir
  - élément source = f(éléments modèle global)

    $S_i \subseteq V_i (M)$
- Avantages
  - les sources sont décrites indépendamment les unes des autres
    - très simple de rajouter une nouvelle source
- Inconvénients
  - traduction de requêtes plus complexe

Exemple « Global-as-View »

- TSIMMIS (Stanford)
  - modèle semi-structuré OEM
    - objet : <id, nom, type, valeur>
  - langage de spécification de médiateur MSL
    - règles : PM :- P_1, … P_k, avec PM, Pi « patterns »
- Exemple
  - Sources : informations sur les personnes d 'une université
    - **Inf** :BDR avec des employés et des étudiants du département Informatique

      Employé(Nom, Prénom, Titre, Supérieur)

      Étudiant(Nom, Prénom, Année)
    - **Ann** : Annuaire pour l 'université (nom, département, catégorie, e-mail, …)
  - Médiateur : les personnes du département Informatique
    - nom, catégorie, titre, supérieur, e-mail, année, ...

TSIMMIS (suite)

Wrapper **Inf**

```
<&e1, employé, set, {&n1, &p1, &t1, &s1}>
   <&n1, nom, string, 'Dupont '>
   <&p1, prénom, string, 'Michel '>
   <&t1, titre, string, 'professeur '>
   <&s1, supérieur, string, 'Jean Martin '>
<&s2, étudiant, set, {&n2, &p2, &a2}>
   <&n2, nom, string, 'Hugo '>
   <&p2, prénom, string, 'Victor '>
   <&a2, année, int, 2>
...
```

Wrapper **Ann**

```
<&p1, personne, set, {&i1, &d1, &c1, &e1}>
   <&i1, nom, string, 'Michel Dupont '>
   <&d1, dept, string, 'Informatique '>
   <&c1, categ, string, 'employé '>
   <&e1, email, string, ' md@univ.fr '>
<&p2, personne, set, {&i2, &d2, &c2, &y2}>
   <&i2, nom, string, 'Zoé  Durand '>
   <&d2, dept, string, 'Informatique '>
   <&c2, categ, string, 'étudiant '>
   <&y2, année, int, 3>
...
```

Médiateur

```
<&pi1, pers_inf, set, {&nm1, &cm1, &tm1, &sm1, &em1}>
   <&nm1, nom, string, 'Michel Dupont '>
   <&cm1, catégorie, string, 'employé '>
   <&tm1, titre, string, 'professeur '>
   <&sm1, supérieur, string, 'Jean Martin '>
   <&em1, email, string, ' md@univ.fr '>
...
```

Spécification MSL du médiateur

```
<pers_inf {<nom N> <catégorie C> Reste1 Reste2}> :-
   <personne {<nom N> <dept 'Informatique '>
                <categ C> | Reste1}>@Ann   AND
   decomp(N, NF, P)   AND
   <C {<nom NF> <prénom P> | Reste2}>@Inf
```


TSIMMIS (suite)

- Exemple de requête
  - trouver toutes les informations sur Michel Dupont

    <pers_inf {<nom 'Michel Dupont '>}>@ Med

  - substitution des éléments de la requête dans la définition du médiateur

```
<pers_inf {<nom 'Michel Dupont '> <catégorie C> Reste1 Reste2}>   :-
   <personne {<nom 'Michel Dupont '> < dept 'Informatique '> < categ C> | Reste1}>@Ann
   AND decomp('Michel Dupont ', NF, P)
   AND <C {<nom NF> <prénom P> | Reste2}>@Inf
```

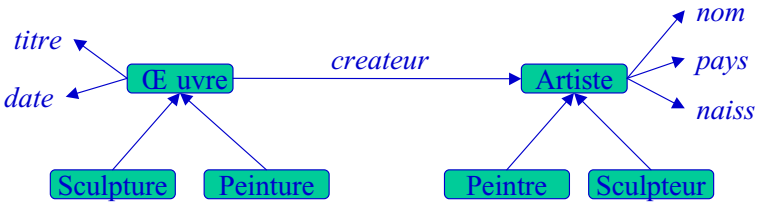
  - chaque source répondra à la sous-requête qui la concerne

Exemple « Local-as-View »

- Information Manifold (AT&T)
  - modèle global : mélange de modèle objet et relationnel
    - classes → objets → attributs → valeurs
    - relations: ensembles de nuplets contenant des objets ou des valeurs
      - chaque classe C est une relation C(o) contenant les objets de la classe
      - chaque attribut A est une relation A(o, v) avec les valeurs d 'attribut des objets
  - sources
    - vision « accès à une BD à travers des formulaires »
    - contenu = vue sur le modèle global (requête conjonctive + inégalités)
    - capacités = décrivent comment la source peut être exploitée
      - paramètres d 'entrée (dans les requêtes)
      - paramètres de sortie (dans les résultats)
      - paramètres qui acceptent une sélection de type <paramètre><inégalité><valeur>

Information Manifold (suite)

- Exemple de modèle global



- Exemple de description de source
  - Nom de peintres nés après 1880 et les titres/dates de leurs peintures

  Contenu : S(o, p) ⊆ Peintre(p), Peinture(o), createur(o, p), naiss(p) ≥1880

  Capacités : **in**(S) = {titre(o), nom(p)}, min = 1

  out(S) = {titre(o), date(o), createur(o), nom(p)}

  sel(S) = {date(o)}

3. Les mappings

- Mapping
  - correspondance entre le schéma global et les schémas des sources
  - utilisé pour la traduction des requêtes et la structuration des résultats
- Diversité
  - les schémas : relationnel, XML, orienté-objet, entité-association
  - les mappings : paires d 'éléments, fonctions, règles, degré de similarité
- Objectifs
  - calcul automatique des mappings (autant que possible)
    - validation par l 'utilisateur
  - création d 'outils aussi génériques que possibles

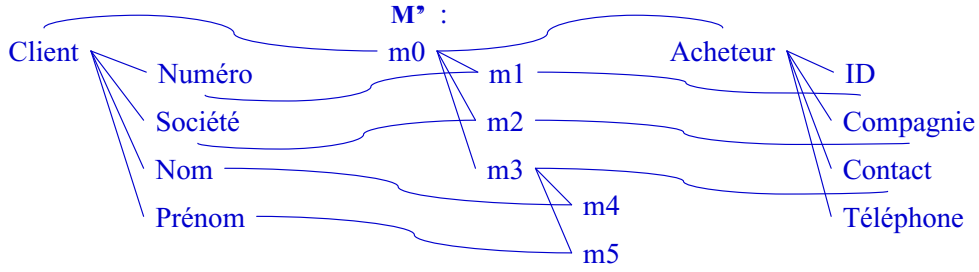
Exemple de mapping

S_1 : Client — Numéro
— Société
— Nom
— Prénom

S_2 : Acheteur — ID
— Compagnie
— Contact
— Téléphone

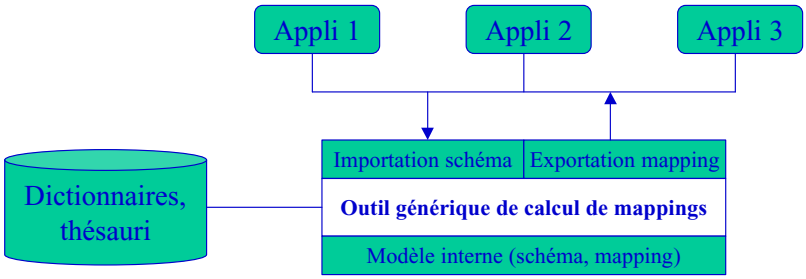
M : Client → Acheteur
Client.Numéro → Acheteur.ID
Client.Société → Acheteur.Compagnie
Client.Nom → Acheteur.Contact
Client.Prénom → Acheteur.Contact

M' : Client ↔ Acheteur
Client.Numéro → Acheteur.ID
Client.Société → Acheteur.Compagnie
conc(Client.Nom, Client.Prénom) →
Acheteur.Contact



Outils génériques de calcul de mappings

- Approche
  - modèle interne générique de schéma et de mappings
  - *schéma* : éléments + structure (composition, héritage, association, lien)
  - *mappings* : éléments S_1 + éléments S_2 + expression
  - traduction : modèle externe → modèle interne (schéma)
    modèle externe ← modèle interne (mapping)
- Architecture



Techniques de calcul de mappings

- Utilisation du schéma uniquement
  - *granularité* : mapping d ĕléments ou de structures d ĕléments
  - *cardinalité:* mappings individuels - 1:1, 1:n, n:1, n:m
  - *linguistique* : similarité des noms/descriptions des éléments
    - égalité : stricte, forme canonique, synonymes, hypernymes
    - sous-chaînes communes, distance d ĕdition, similarité phonétique
    - similarité indiquée par l ŭtilisateur
  - *contraintes* : type, domaine, multiplicité, valeur clé, cardinalité relations
  - *réutilisation* : mappings déjà calculés pour des structures qui apparaissent souvent
- Utilisation des instances
  - extraction de caractéristiques du schéma absentes dans sa description
  - calcul de mappings d ĭnstances → généralisés au schéma

Techniques (suite)

- Combinaison de plusieurs critères
  - *hybride* : algorithme qui combine plusieurs critères
  - *composition* : combinaison flexible de plusieurs algorithmes
    - poids ajustables
    - ordre d 'exécution flexible, un algorithme utilise les résultats d 'un autre
- Intervention de l 'utilisateur
  - décision sur les mappings candidats
  - nouveaux mappings loupés par le système
  - paramétrage fin de la composition d 'algorithmes

4. Intégration basée sur XML

- Les avantages de XML
  - *sources* : format naturel pour les documents
  - *wrappers* : format d 'échange idéal, auto-descriptible, structuré, flexible
  - *médiateur* : format général, indépendant du domaine d 'application
  - *intégration* : un seul format peut décrire des données, des structures, des vocabulaires très hétérogènes
- Les principaux types de systèmes
  - sources non-XML, médiateur XML
  - sources XML, médiateur XML ou autre

XML et relations

- Pourquoi mélanger XML et relations?
  - XML est très bien adapté à l 'échange et à l 'intégration
  - … mais les sources de données restent essentiellement relationnelles
- Un problème essentiel : données relationnelles → documents XML
  - publication de documents XML
  - interrogation
- Opérations à réaliser pour la transformation
  - *extraction* : extraction de nuplets de la source relationnelle
  - *structuration* : création des structures imbriquées
  - *balisage* : libellés (« tags ») associés aux structures
- Question : quelle est la stratégie la plus efficace?
  - qui (source/wrapper), quand et comment réalise ces opérations?

Exemple de transformation

Schéma relationnel

Client(Id, Nom)
Compte(Id, ClientId, No)
Commande(Id, ClientId, CptId, Date)
Produit(Id, CmdId, Desc)

Document XML

```
<client id="c1">
  <nom> Jean Dupont </nom>
  <comptes>
    <compte id="cpt1"> B2345 </compte>
    <compte id="cpt2"> A9513 </compte>
  </comptes>
  <commandes>
    <cmd id="cmd1" compte="cpt1">
      <date> 25/02/2003 </date>
      <produits>
        <prod id="p1"> Vin </prod>
        <prod id="p2"> Fromage </prod>
      </produits>
    </cmd>
    …
  </commandes>
</client>
```

Requête SQL étendu

```
select CLIENT(cl.Id, cl.Nom,
              (select XmlAgg(COMPTE(cpt.Id, cpt.No))
               from Compte cpt
               where cpt.ClientId = cl.Id),
               (select XmlAgg(COMMANDE(cmd.Id, cmd.CptId, cmd.Date,
                                       (select XmlAgg(PRODUIT(p.Id, p.Desc))
                                        from Produit p
                                        where p.CmdId = cmd.Id)))
                from Commande cmd
                where cmd.ClientId = cl.Id))
from Client cl;
```

Constructeur XML

```
Define XML Constructor
 CLIENT(clId: integer, clNom: varchar(20), cptListe: xml, cmdListe: xml)
 AS{
  <client id=$clId>
    <nom> $clNom </nom>
    <comptes> $cptListe </comptes>
    <commandes> $cmdListe </commandes>
  </client>
}
```

Commentaires sur l 'exemple

- Langage d 'extraction : SQL étendu
  - constructeur de fragments XML
  - agrégation de fragments XML : ***XmlAgg***
  - respect de l 'ordre sur les collections (pour ***XmlAgg***)
- Les opérations de transformation
  - *structuration* : requêtes imbriquées
  - *balisage* : constructeurs XML
- Choix d 'implémentation possibles
  - tout dans le moteur relationnel ou juste une partie
  - le rapport extraction des nuplets ↔ structuration/balisage

Intégration de sources XML

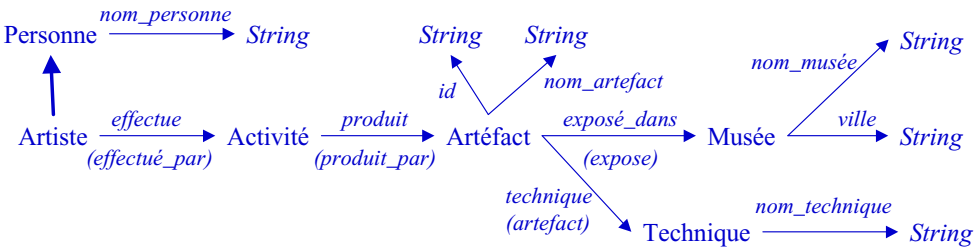
- Deux systèmes: STYX et Xyleme
  - approches caractéristiques pour petite et grande échelle
  - STYX : petite échelle, communautés Web
  - Xyleme : très grande échelle, tout le Web
- Brève comparaison
  - STYX : modèle de médiation riche
    - ontologie, mappings de chemins utilisant XPath
    - complétude, jointure par clé
    - problèmes : gestion manuelle, performances
  - Xyleme : nombre de sources très important
    - performances, passage à l 'échelle
    - calcul semi-automatique des mappings
    - problèmes : modèle plus simple, pouvoir d 'expression, complétude

STYX

- Médiateur pour communautés Web
  - CNAM : B. Amann, I. Fundulaki, M. Scholl + C. Beeri (Tel Aviv)
- Caractéristiques
  - données dans un domaine d 'application bien déterminé
  - sources XML décrites par *DTDs*
  - modèle global décrit par *une ontologie*
  - mappings (chemin ontologie - chemin DTD XPath)
  - approche « local-as-view »
  - pas de matérialisation au niveau du médiateur
  - notion de *clé* pour les jointures entre sources
  - requêtes de type arbre dans l 'ontologie traduites en requêtes de type arbre dans les sources XML

STYX : le modèle du médiateur

- Ontologie ≈ schéma entité - association
  - concepts : relation d héritage (ISA)
  - rôles : associations binaires entre concepts
    - réversibles : pour tout rôle, il existe un rôle inverse
  - attributs : associés aux concepts
    - type atomique



STYX : les sources

- Ressources XML
  - identifiées par une URL
  - décrites par une DTD
- Exemple : Œ uvres de peintres *http://www.peintres.com*

```
<!ELEMENT Collection (Peintre*)>
<!ELEMENT Peintre (Peinture+, Sculpture+)>
   <!ATTLIST Peintre Nom CDATA #REQUIRED>
<!ELEMENT Peinture (Technique?, Musee)>
   <!ATTLIST Peinture Titre CDATA #REQUIRED>
<!ELEMENT Sculpture (Technique?, Musee)>
   <!ATTLIST Sculpture Titre CDATA #REQUIRED>
<!ELEMENT Technique #PCDATA>
<!ELEMENT Musee #PCDATA>
```


STYX : les mappings

- Publication d 'une source dans le médiateur
  - spécification de l 'ensemble de mappings de la source

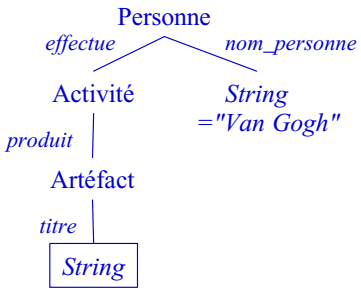
A_1 : **http://www.peintres.com/Collection/Peintre** as u_1 → Personne
A_2 : u_1/**@Nom** as u_2 → nom_personne
A_3 : u_1/**Peinture** as u_3 → effectue.produit
A_4 : u_1/**Sculpture** as u_3 → effectue.produit
A_5 : u_3/**Musee** as u_4 → exposé_dans.nom_musée

- Caractéristiques
  - chemins XPath dans la source (langage puissant)
  - utilisation de variables liées à des instances (fragments) XML
    - factorisation de mappings (m+n mappings au lie de m*n)
    - un même fragment de la source peut jouer des rôles différents

STYX : les requêtes

- Requêtes de type arbre dans l 'ontologie
  - langage de type OQL
- Exemple
  - trouver les titres des œuvres de Van Gogh

select x_3
from Personne x_1, x_1.nom_personne x_2,
x_1.effectue.produit.titre x_3
where x_2 = "Van Gogh"



  - remarque : des requêtes équivalentes sont possibles (*ex*. avec Artéfact comme racine et utilisant les rôles inverses)

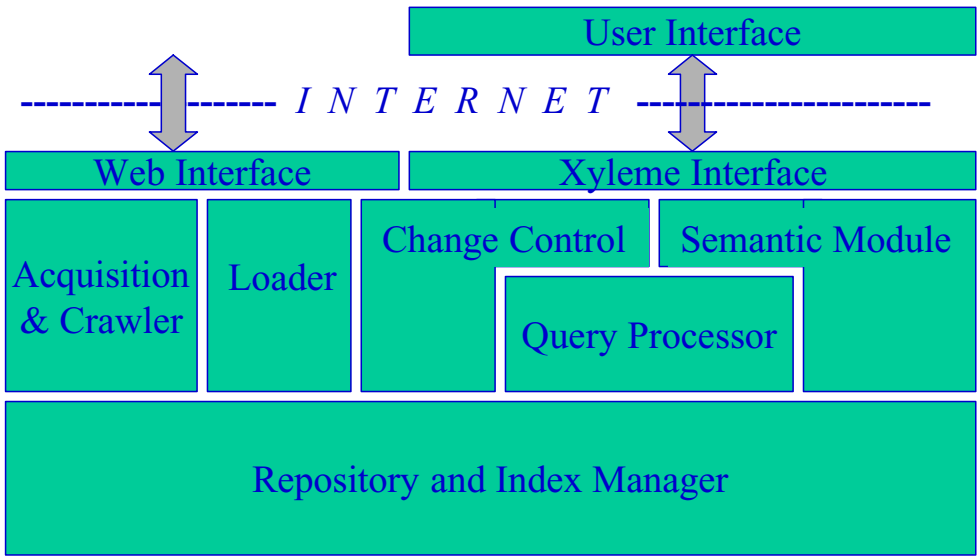
STYX : les clés

- Clés pour les concepts de l 'ontologie
  - clés sémantique, qui « identifient » un concept
    - intuitivement : un attribut du concept (*ex*. nom musée, identifiant artéfact)
  - plus généralement : ensemble de *chemins d àttributs* partant du concept
- Jointure
  - une instance de concept peut être éclatée entre plusieurs sources
  - une requête est traduite vers toutes les sources
    - différentes sources peuvent répondre avec des fragments d 'une instance
  - fusion des fragments concernant une même instance de concept
    - jointure basée sur l 'égalité des clés du concept

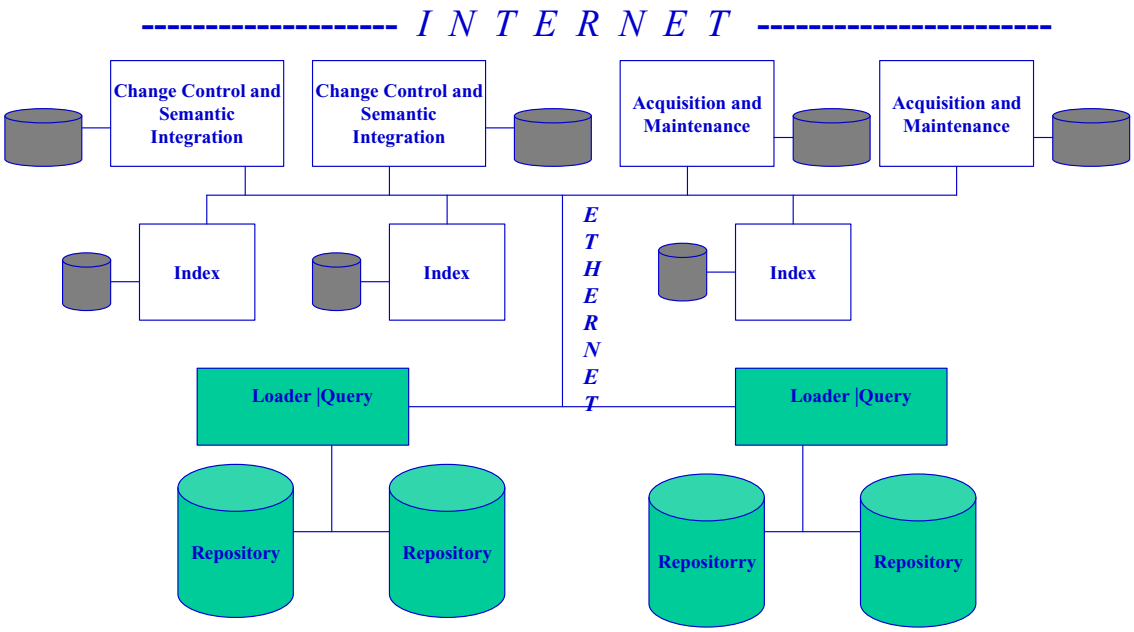
Xyleme

- Entrepôt de données XML à l'échelle du Web
  - projet INRIA (Verso) + LRI, CNAM, Univ. Mannheim
    - commercialisé par Xyleme SA
  - problèmes : passage à l'échelle, hétérogénéité, changements
  - architecture distribuée
- Principaux services
  - acquisition de documents à partir du Web ou en local
  - interrogation basée sur la structure
  - notification de changements, gestion de versions
  - intégration sémantique à travers des vues

Architecture fonctionnelle



Architecture physique distribuée



Xyleme : intégration sémantique

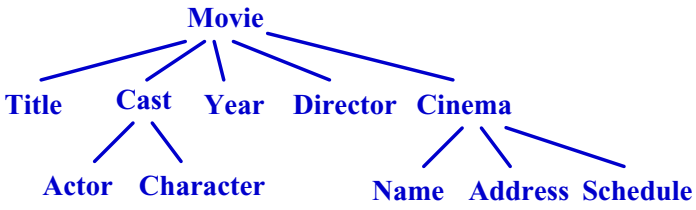
- Organisation de données guidée par la sémantique
  - classification suivant une hiérarchie prédéfinie de *domaines*
  - domaine = cluster de documents XML
- Intégration de données de type entrepôt
  - source = domaine (cluster)
  - modèle global = vue sur les domaines
    - plusieurs vues possibles en même temps dans l 'entrepôt
  - schéma global (de vue) = DTD « abstraite »
  - domaine « abstrait » de la vue = union domaines « concrets » des sources
- Approche mixte
  - *entrepôt / médiateur* : les instances du schéma global non matérialisées
  - *global-as-view / local-as-view* : mappings symétriques

Définition de vues

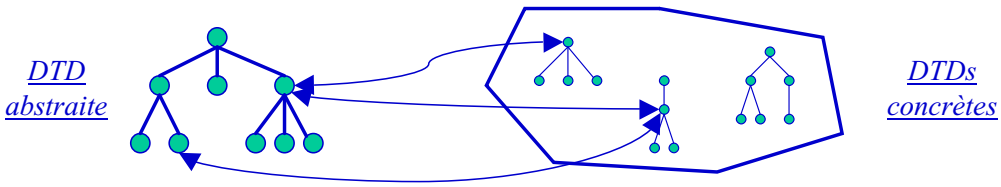
- Domaine
  - ensemble de clusters contenant des documents XML
  - DTDs « concrètes » sous forme d 'arbre
- Schéma
  - DTD abstraite = arbre de concepts
- Définition
  - ensemble de mappings (chemin abstrait - chemin concret)

Exemple de vue

- Vue sur le cinéma
  - domaine **Cinema** : clusters {Films, Acteurs, Spectacles}
  - schéma:



  - définition: mappings

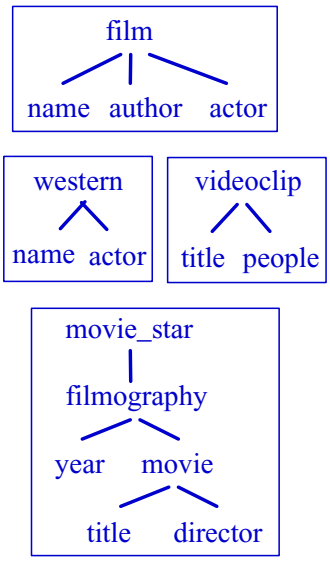


DTD abstraite

- Rôle
  - interrogation homogène de documents hétérogènes en termes de vocabulaire et de structure
  - schéma de structuration unique des résultats
  - interface d 'interrogation pour l 'utilisateur
- Structure: arbre de concepts
  - pas de sémantique pour les liens, multiplicité *
  - le chemin dans l'arbre définit le contexte
    - exemple: **Movie/Cinema/Name**

Exemples de mappings

DTDs concrètes



Quelques mappings

Movie ↔ film
↔ western
↔ videoclip
↔ movie_star/filmography/movie

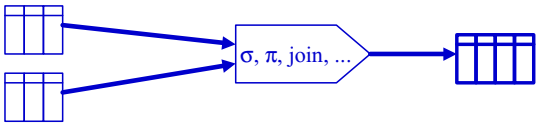
Movie/Title ↔ film/name
↔ western/name
↔ videoclip/title
↔ movie_star/filmography/movie/title

Movie/Director ↔ film/author
↔ videoclip/people
↔ movie_star/filmography/movie/director

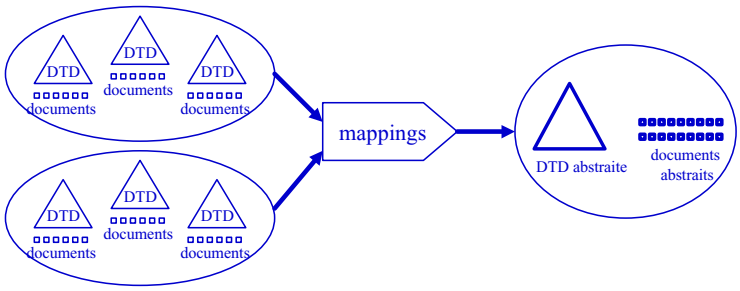
Movie/Cast/Actor ↔ film/actor
↔ western/actor
↔ videoclip/people
↔ movie_star

Comparaison avec le relationnel

Relationnel



Xyleme



Domaine ***Définition*** ***Schéma + extension***

Comparaison (suite)

- Relations ↔ Clusters
  - relations homogènes ↔ clusters très hétérogènes
  - la définition ne peut pas se limiter à une simple requête
    - plusieurs requêtes par DTD concrète !
    - taille de la définition de vue beaucoup plus importante
- Manuel ↔ Automatique
  - taille et hétérogénéité des données → la vue ne peut pas être créée manuellement par l 'utilisateur
  - outils de génération (semi-)automatique des mappings

Mappings

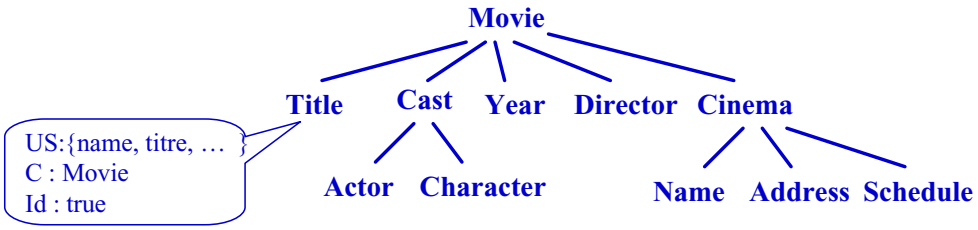
- Mapping = correspondance abstrait - concret
  - correspondance de termes
    - taille réduite et facile à générer
    - requêtes: mauvaise précision et mauvaises performances
  - correspondance d 'arbres (DTDs)
    - le plus proche de la définition classique des vues
    - requêtes: précision parfaite et très bonnes performances
    - stockage coûteux et difficile (impossible) à générer
  - correspondance de chemins
    - le « bon compromis », bonnes propriétés pour tous les critères
    - précision de génération automatique → précision des requêtes

Génération automatique de mappings

- Absolument nécessaire
  - gestion manuelle possible pour la DTD abstraite, mais impossible pour les mappings
  - nb. de mappings proportionnel au nb. de DTDs concrètes
- Principes de génération automatique
  - similarités entre mots
    - lexicale: racine commune, mots composés, abréviations, etc.
    - sémantique: synonymie, généralisation, etc.
  - contexte d 'interprétation : le chemin

Annotation de la DTD abstraite

- Chaque concept (nœud) de la DTD abstraite
  - unité de sens : l'ensemble des mots "similaires"
    - chaque mot: décomposé, forme racine ("stemming"), info langue
  - contexte : nœud ancêtre important pour l'interprétation
    - transitivité
  - identifiant : booléen pour marquer les nœuds identifiants



Phases

- Mise en correspondance sémantique
  - concept abstrait ↔ mot concret, à l'aide de l'US
  - factorisation des mots qui apparaissent souvent
- Mise en correspondance contextuelle
  - un mapping de mots → ***n*** mappings de chemins
  - le contexte permet d'éliminer des mappings incorrects
- Validation
  - aide à la découverte de mappings incorrects ou manquants

Techniques contextuelles et de validation

- Contexte
  - si contexte(A)=A', A $\leftrightarrow$ C valide seulement s 'il existe un mapping de A' vers un préfixe de C
- Validation
  - découverte de concepts ou de zones concrètes peu mappées
  - indication mappings douteux
- Remarque
  - il vaut mieux ne pas perdre de mappings que d'en avoir trop
  - la traduction n 'accepte que des combinaisons *valides* de mappings

Bibliographie

- B. Amann, *Intégration de Données*, Cours DEA SIR 2001
- (*TSIMMIS*) Y. Papakonstantinou, H. Garcia-Molina, J. Ullman, *Medmaker: a mediation system based on declarative specifications*, ICDE 1996
- (*Information Manifold*) A. Levy, A. Rajaraman, J. Ordille, *Querying heterogeneous information sources using source description*, VLDB 1996
- (*Mappings*) E. Rahm, P. Bernstein, *A survey of approaches to automatic schema matching*, VLDB journal, 2001
- (*XML et relations*) J. Shanmugasundaram et al, *Efficiently publishing relational data as XML documents*, VLDB 2000
- (*STYX*) I. Fundulaki, *Intégration et interrogation de ressources XML pour communautés Web*, thèse CNAM Paris, 2002
- (*Xyleme*) S. Cluet, P. Veltri, D. Vodislav, *Views in a large scale XML repository*, VLDB 2001