Intégration de Bases de données hétérogènes par une modélisation conceptuelle XML

Myriam Lamolle, Amar Zerdazi

Intégration de Bases de données hétérogènes par une modélisation conceptuelle XML

Myriam LAMOLLE, Amar ZERDAZI

Laboratoire de Recherche en Informatique (LINC)
IUT de Montreuil – Université Paris 8
140, rue de la Nouvelle France
93100 – Montreuil
{m.lamolle, a.zerdazi}@iut.univ-paris8.fr

Abstract. Le développement exponentiel d'échanges d'informations par le web a mis à jour les difficultés pour retrouver l'information pertinente désirée par un utilisateur final. En effet, les informations sont représentées et stockées dans une multitude de sources de données et cela de façon très hétérogène. Notre travail se situe dans le cadre de l'intégration de données hétérogènes avec la définition d'un modèle de médiation structurelle et sémantique, pour l'extraction et la modélisation des connaissances à partir de données sources, appelé hyperschema XML. Cet article présente plus précisément la phase d'extraction des schémas, en définissant les règles de base permettant d'obtenir une structure logique XML. Cette structure logique sert de fondement pour la phase d'intégration proprement dite.

1 Introduction

Le développement exponentiel d'échanges d'informations par le Web a mis à jour les difficultés pour retrouver l'information pertinente désirée par un utilisateur final. En effet, les informations sont représentées et stockées dans une multitude de sources de données et cela de façon très hétérogène. Les premières approches d'intégration de ces sources de données pour les faire coopérer ont été réalisées dans le cadre de système de bases de données (BDs) relationnelles, objets/relationnelles ou objets au travers de la mise en place d'une fédération de bases de données [11], [22].

Cependant, dans le contexte du Web, les sources de données ont pour but principal de fournir des capacités d'interrogation et non d'exportation des données. Ce contexte nous oblige à repenser la façon dont nous devons intégrer ces différentes sources d'informations pour connaître au plus tôt la sémantique des données mémorisées. Connaître cette sémantique, c'est pouvoir « filtrer » les sources qui sont les plus adéquates pour répondre à une demande d'information.

Comment peut-on intégrer cette sémantique ? Comment peut-on en déduire les concepts présents partiellement ou totalement dans plusieurs sources de données ?

Peut-on faciliter cette intégration lors de la phase d'extraction des concepts ? Après un bref rappel sur les techniques d'intégration, nous proposons une solution d'enrichissement sémantique[1] des concepts lors de la phase d'extraction [24] facilitant la mise en correspondance des schémas à intégrer [23].

Nous prenons en compte le fait qu'intégrer les données ne consiste pas seulement à homogénéiser les données en utilisant un format commun XML mais consiste aussi à exprimer les relations entre les données des sources intégrées [4]. Pour ceci faire, lors de la phase d'extraction de la sémantique d'un schéma de base de données, nous enrichissons la représentation des concepts, des relations entre concepts et des propriétés. Cette représentation utilise le formalisme *XML Schema* [33] qui est plus adapté que les modèles traditionnels les plus courant basés sur des DTD. L'apport du langage XSD permet d'affiner d'une part la sémantique interne d'un concept (typage, contrainte d'unicité, etc.) et d'autre part la sémantique inter-concepts (dépendance, association, agrégation, etc.).

La présentation de notre travail est organisée de la manière suivante. Après avoir présenté un état de l'art sur les travaux déjà réalisés (section 2), nous présentons dans la section 3, l'architecture générale de notre système et l'étape de pré-intégration qui consiste en l'extraction des schémas des sources à intégrer. Nous détaillons les règles d'extraction mises en place pour homogénéiser la représentation des schémas des sources de données ainsi que l'enrichissement sémantique des schémas XML par l'adjonction de métaconnaissances. La section 4 aborde brièvement la phase de mises en correspondance entre les schémas XML étendus où nous définissons les opérateurs de base utilisés lors des appels aux primitives de transformation. Nous conclurons sur le travail déjà réalisé et les différents points que nous devons développer dans le futur.

2 Etat de l'art

L'intégration de données et du *schema matching* ont été étudiés depuis longtemps par la communauté base de données [24]. Nous ne présenterons pas ici de manière exhaustive tous les systèmes existants mais ceux qui nous ont paru intéressant pour les problématiques qu'ils soulèvent et/ou pour les solutions envisagées.

Intégration de schémas

Dans le contexte du Web, il existe deux principales approches d'intégration à savoir l'approche matérialisée (par entrepôt) [31], [29] et l'approche virtuelle (par médiateur) [30]. Dans le cadre de l'approche virtuelle, citons *TSIMMIS* [10] qui utilise un modèle global sous forme d'objets complexes et un langage de règles *(MSL, Mediator Specification Language)* pour l'interrogation. Un des objectifs de *TSIMMIS* est d'intégrer des sources qui sont très hétérogènes, qui peuvent être peu structurées et qui

[1] Processus d'adjonction de métaconnaissances ou méta-informations, généralement réalisé et validé par un expert humain.

sont susceptibles d'évoluer rapidement. Il existe d'autres approches telles que *AGORA* [17], *GARLIC* [12], *YAT* [28], etc.
Dans le cadre de l'approche matérialisée, citons *Xylème* [8] qui est un système d'entrepôt dynamique dont l'objectif est de stocker et d'intégrer semi-automatiquement toutes les ressources XML du Web afin de fournir à l'utilisateur final un accès uniforme et transparent à la localisation et l'hétérogénéité de données. La nouveauté de ce projet se situe dans la volonté de créer un système traitant le stockage efficace de données (via des arbres), l'évaluation de requêtes, la maintenance de données et surtout l'intégration sémantique de données. Il existe d'autres systèmes tels que *e-XMLMédia* [11], *Castor* [6], etc.

Schema matching
De nombreux travaux proposent des méthodes pour automatiser la tâche de *matching* tels que [21], [26], [25], [9], [29]. [27] présente une taxonomie de travaux effectués dans cette voie. Nous pouvons les classer en deux grandes catégories : les systèmes qui travaillent à partir d'une connaissance *a priori* du schéma de données (schéma de base de données, DTD, schéma XML, etc.) alors que la seconde regroupe des systèmes dont les approches utilisent le contenu, c'est-à-dire les données elles mêmes (ou instances). Par exemple, le système *SemInt* [15] se base sur la similarité entre noms d'éléments. Il repose sur une architecture neuronale afin de déterminer les éléments à mettre en correspondance. Le modèle *Similarity Flooding* [20] repose sur une analyse de la structure des données. Cette approche implémente un algorithme de matching *(appelé SF)* basé sur des calculs de points fixes pour déterminer les correspondances sémantiques. [13] constitue un exemple de travaux de la deuxième catégorie. D'autres travaux, tels que *Cupid* [16] qui préfèrent une voie de recherche hybride des deux précédentes, utilisent un matching manuel.

Nous proposons dans cet article une implémentation de l'intégration de schémas selon la deuxième approche (approche virtuellement intégrée) en créant un modèle de médiation au dessus des schémas sources des bases de données à fédérer.

3 Extraction des Schémas

L'objectif de nos travaux est de fournir un modèle d'intégration, en même temps souple et puissant, capable de fédérer les différentes sources de données hétérogènes représentant des dimensions structurelle et sémantique des schémas sources. Pour cela, la définition de notre modèle doit comporter un nombre minimal de type d'*éléments* à manipuler mais suffisant pour traduire les schémas. Notre hypothèse s'appuie sur un modèle semi-structurée (XML) [32] qui par sa souplesse permet une intégration aisée de modèles variés [2]. En effet, sa structure arborescente permet de représenter de façon générique diverses sources de données (structurées : SGBD, semi-structurées : OEM, non-structurés : Web) [1]. Comme nous pouvons le constater sur la figure 1, notre approche est composée de deux parties distinctes à savoir le module d'extraction des schémas sources et le module d'intégration permettant de

construire l'hyperschéma. Au final, l'hyperschéma XML est constitué de l'ensemble des schémas XSL [34] stipulant les règles de transformation entre les éléments des schémas XSD [14].

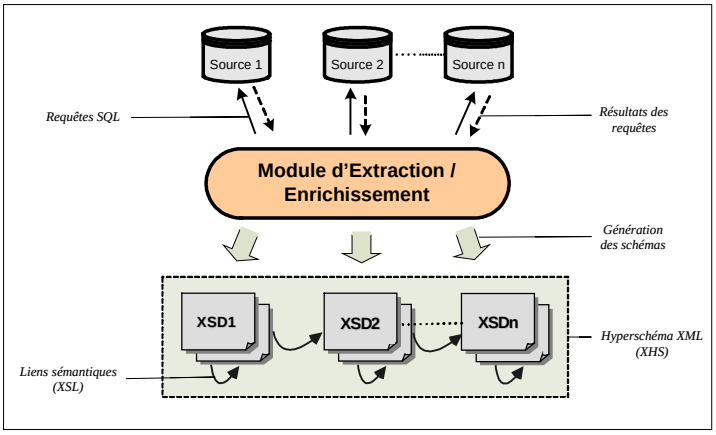


Fig. 1. Architecture d'intégration via XML.

Dans la figure 1, la phase d'extraction homogénéise la représentation des différents schémas des BDs à intégrer en l'exprimant sous la forme d'une définition de schéma XML (i.e XSD*i*) [33]. Une fois cette transformation faite [18], chaque concept (dans les fichiers XSD) est représenté selon deux dimensions à savoir une dimension structurelle et une dimension sémantique. Dans cette phase, la partie sémantique des schémas XML extraits est affinée par l'adjonction de métaconnaissances soient déduites du catalogue des données, soient précisées par les experts de la BD. Ces deux dimensions sont prises en compte dans le calcul du degré de similitude de concepts (la détection des conflits sémantiques et logiques dans les schémas XML lors de la phase de *matching*) qui nous permet de mettre en place les opérateurs de transformation (appelés liens sémantiques sur la figure 1) [7]. Il est à noter que la recherche des « *liens sémantiques entre schémas XSD* » ne rentre pas dans le cadre du présent article.

Le schéma XML issu de cette phase est composé de trois types d'élément à savoir :

- *Concept :* est l'élément le plus important dans le schéma XSD. Il peut engendrer des propriétés et/ou d'autres concepts imbriqués, et porter des relations avec d'autres concepts.

- *Relation :* correspond aux associations structurelles et/ou sémantiques existant entre deux ou plusieurs concepts. Chaque relation possède éventuellement un sens de "lecture" (cardinalités [0,1] ou [1,1]) et un rôle.

- *Propriété :* est l'élément le plus "basique" dans le schéma XSD. Pour chaque propriété, nous définissons son nom, son type et également ses occurrences dans le concept ou la relation, ses contraintes (unicité, renseigné, constante, etc.).

Au final, l'hyperschéma XML *(noté XHS)* est constitué de l'ensemble des schémas des BDs sous forme XSD *étendu* et de l'ensemble des opérateurs de transformation d'un schéma à un autre sous forme XSL [34]. L'extraction des différents schémas suit des règles précises pour les dimensions statique et dynamique d'un concept. Nous allons présenter maintenant plus en détail ces différentes règles.

3.1 Règles d'extraction

Les règles d'extraction de base sont issues de la modélisation des BDs relationnelles. Ces règles sont étendues au cas des BDOOs pour traiter des relations de généralisation/spécialisation, d'agrégation, de composition et de dépendance.

3.1.1 Extraction de la partie statique

La dimension statique d'un concept est constituée de son nom et des *propriétés* qui le caractérisent. Par exemple, un concept prend la forme d'une table et de ses champs dans le cas d'une BD relationnelle, d'une classe et de ses attributs dans le cas d'une BD objet, d'une balise et de ses attributs et/ou de balises de niveau immédiatement inférieur dans le cas d'une BD XML. De plus, il se peut qu'un concept contienne lui-même d'autres concepts (BDOR, BDOO, BDXML).

Règle 1. Toute table, toute classe, toute balise directement au-dessous de la balise racine de la BD considérée représente un concept ; elle est traduite par un type complexe *(complexType)* XSD.

Exemple 1. Soit le schéma partiel d'une base de données *agenda* dont le modèle relationnel est :

Personne [**idPers, nom**, adresse, profession, **#dept**]
Département [**idDep**, nom]

où idPes et idDep sont les clés primaires et #dept est une clé étrangère dont l'ensemble de ses valeurs appartient à l'ensemble des valeurs de idDep.

Alors la table relationnelle *personne* est transformée en :

```
<xsd:complexType name="personne"
type="personneConcept"…/>
```

Dans la suite des règles, nous notons les arguments pour les fonctions d'extraction comme suit :

En entrées: *bd* est un schéma de BD, *tb* une table, *att* un attribut, *op* une opération.

En sorties: *xsd* est un schéma XML, *c* un concept, *p* une propriété, *r* une relation.

Règle 2. Tout attribut *(att)* d'une table ne portant pas de contrainte d'intégrité fonctionnelles *(CIF)*, autrement dit la propriété *(p)* appartenant au concept *(c)*, est représenté par un élément *(element)* du *complexType* XSD vu dans la règle 1. Les occurrences possibles pour cet élément sont 0 ou 1.

$\forall$ tb$\in$bd $\forall$ att$\in$tb $\wedge$ use(att, CIF)=false |
extractAtt(bd, tb, att) = element ⟨name, type, [minOccurs, maxOccurs]⟩

Exemple 2.
extractAtt(agenda, personne, profession) = element ⟨profession, "xsd:string"⟩

```
<xsd:element name="profession" type="xsd:string"…/>
```

3.1.2 Extraction de la partie dynamique

La dimension dynamique d'un concept est constituée des contraintes d'intégrité et des opérations. L'aspect dynamique est constitué des clés primaires et étrangères, des triggers et des procédures cataloguées.

3.1.2.1 Cas des contraintes d'intégrité fonctionnelle

Règle 3.a. Tout attribut identifiant unique d'une table de la BD considérée est traduit par un attribut *(Attribute)* XSD du *complexType* construit d'après la règle 1 dont l'usage est obligatoire. Cet attribut joue le rôle d'une *propriété clé* dans le concept.

$\forall$ tb$\in$bd $\forall$ att$\in$tb $\wedge$ use(att, CIF) |
extractKeyPart1(bd, tb, att) = attribute ⟨name, type⟩

Exemple 3.a.
extractKeyPart1(agenda, personne, idPers) = attribute ⟨idPers, "xsd:positiveInteger"⟩

```
<xsd:attribute name="idPers"
type="xsd:positiveInteger"…/>
```

Règle 3.b. Tout attribut identifiant la BD considérée, intègre le type complexe représentant la BD elle-même par l'élément XSD *key*.

$\forall$ tb$\in$bd $\forall$ att$\in$tb $\wedge$ use(att, CIF) |
extractKeyPart2(bd, tb, att) = key ⟨name, selector ⟨xpath⟩, field ⟨xpath⟩ ⟩

Exemple 3.b.
extractKeyPart2(agenda, departement, idDep) =
key ⟨IDdepartement, selector ⟨.//departement⟩, field ⟨./@idDep⟩ ⟩

```
<xsd:key name="IDdepartement">
    <xsd:selector xpath=".//departement"/>
    <xsd:field xpath="./@idDep"/>
</xsd:key>
```

Règle 4.a. Tout lien d'association (agrégation, composition) intégré directement dans un concept est traduit par la contrainte XSD *keyref*.

∀ tb∈ bd ∀ att∈ tb ∧ use(att, CIF) |
extractRelation(bd, tb, att) = keyref ⟨name, selector ⟨xpath⟩, field ⟨xpath⟩ ⟩

Exemple 4.
extractRelation(agenda, departement, idDep) =
keyref ⟨IDREFdepartement, selector ⟨.//personne⟩, field ⟨./@dept⟩ ⟩

```
<xsd:keyref name="IDREFdepartement"
    refer="IDdepartement">
   <xsd:selector xpath=".//personne"/>
    <xsd:field xpath="./@dept"/>
</xsd:keyref>
```

Règle 4.b. Dans le cas où la clé primaire de la table est constituée d'attributs étant des clés étrangères alors il y a création d'un concept spécifiant les liens vers les autres tables concernées.

3.1.2.2 Cas des opérations
Le cas général des opérations concerne les procédures cataloguées des BDR et des méthodes de classes des BDs objets.

Règle 5. Toute procédure cataloguée ou toute méthode de la BD considérée est traduite par un type élément ***(element)*** XSD qui sera référencé par le concept « propriétaire ».

∀ bd ∀ tb∈ bd ∧ use(op, tb) |
extractOp(bd, tb, op) = element ⟨name, type, sequence ⟨in, out, action⟩ ⟩

Remarque. Nous opérons de la même façon pour traiter le cas des triggers (nous appliquons la fonction d'extraction *extractTrig(bd, tb, trig)*).

3.2 Adjonction de métaconnaissances

Les règles d'extraction présentées précédemment uniformisent la représentation des concepts essentiellement sur leur dimension statique. La dimension sémantique des concepts est enrichie par l'apport de métaconnaissances [19] lors d'une étude plus approfondie de l'état du concept (structure incomplète, incertaine, valeurs optionnelles, etc.), du niveau d'encapsulation du concept qui précise sa visibilité et son type d'association avec d'autres concepts (dépendance, utilisation, agrégation/composition, etc.). L'objectif de cet enrichissement est de représenter des concepts selon des niveaux de granularité différents. Ces métaconnaissances sont utilisées lors de la phase d'intégration proprement dite pour obtenir des mises en correspondance plus précises. Pour cela, nous rajoutons des facettes, autrement dit des attributs (nom/valeur) à notre schéma XSD au même niveau que la définition des concepts.

3.2.1 Complétude d'un concept

Dans le cas des BDs de type XML native, la structure des données est irrégulière, contrairement aux BDs traditionnelles. En d'autres terme, le concept change de forme et/ou de contenu d'un document XML à un autre. Pour pallier cette différence de structuration de concepts, nous introduisons une facette de type logique dans le *complexType* représentant le concept, qui porte le nom « complete ».

Exemple 5.

```
<xsd:complexType name="personneConcept"
complete="false"…>
```

Nous remarquons que le concept *personne* dans l'exemple ci-dessus est incomplet du fait qu'il pourra changer de structure (forme, hiérarchie des propriétés, etc.) d'un schéma XSD à un autre.

3.2.2 Visibilité d'un concept

Afin de garder la structure hiérarchique, notamment dans le cas des BDs XML (comme un document XML peut posséder une structure arborescente parfois très profonde), nous devons mémoriser le niveau d'apparition du concept. En effet, ceci traduit la visibilité du concept par rapport aux autres, c'est-à-dire un concept-clé [3] *(plus pertinent)* ou non pour le domaine représenté. C'est par la facette « level » de type entier dans la définition du concept que ce niveau d'encapsulation est sauvegardé.

Exemple 6.

```
<xsd:complexType name="personneConcept" level="1"…>
```

Le concept *personne* est de niveau 1 dans le schéma XSD, ce qui explique son importance dans son schéma source.

3.2.3 Contraintes sur les propriétés

De même, les propriétés des concepts (représentés pas *<xsd :attribute>* dans le fichier XSD), traduisent les clés primaires dont les contraintes de valeurs sont *not null* et *unique*. Ces contraintes sont présentes dans l'hyperschéma XML par l'utilisation de la facette « use » et « unique » qui prennent respectivement les valeurs *required* et *true*.

Exemple 7.

```
<xsd:attribute name="idPers" type="xsd:positiveInteger"
use="required" unique="true"/>
```

Dans ce cas, la présence de la propriété clé *idPers* appartenant au concept *personne* est obligatoire du fait qu'elle représente la clé primaire dans la table *personne* (BDR de l'exemple 1).

4 Opérations de transformation

L'hétérogénéité dans l'ensemble des schémas XML de l'hyperschéma est de nature logique ou sémantique [5]. Pour lever une partie des ambiguïtés, nous utilisons les métaconnaissances dans les algorithmes des opérations de transformation. Les opérateurs mis à disposition concernent les concepts et leurs propriétés ainsi que les relations entre concepts. Ces opérateurs exploités par les règles de passage XSL entre schémas XML obtenu lors de la phase d'extraction sont :

- **Addition *(add)*** : permet d'ajouter un élément (concept, propriété d'un concept ou relation existante entre concepts) au schéma source.

- **Suppression *(delete)*** : permet de supprimer un ou plusieurs éléments lors du processus de mises en correspondance *(matching)* entre le schéma source et cible *(XSD_i ® XSD_j)*. La suppression est l'opérateur inverse de l'ajout.

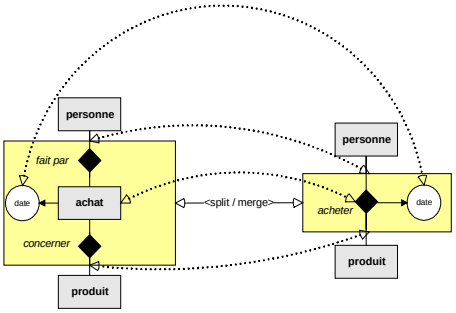
- **Fusion *(merge)*** : permet de fusionner deux éléments distincts en un élément atomique. Généralement, nous appliquons la fusion afin de regrouper des propriétés appartenant au schéma source en un concept dans le schéma cible.

- **Eclatement *(split)*** : permet la décomposition d'un élément du schéma source en un ensemble d'éléments équivalents dans le schéma cible. *Split* est l'opérateur inverse de *merge*.

§ **Renommage *(rename)*** : permet de renommer des éléments lors du *mapping*. Nous appliquons cet opérateur de base sur les éléments de même type, qui sont structurellement équivalents mais syntaxiquement différents *(conflit de nom).*

§ **Identité *(identity)*** : L'identité est appliquée dans le cas où nous voulons mettre en correspondance deux éléments qui sont identiques *(même nom, même structure, même granularité, etc.).*

Exemple 8.



2 relations & 1 concept « 1 relation
(a) mergeá{achat & fait par & concerner}® acheter ; XSD$_1$®XSD$_2$ñ
(b) splitáacheter® {achat & fait par & concerner} ; XSD$_2$®XSD$_1$ñ

Fig. 2. Exemple d'application des opérateurs des primitives de transformations

Le concept ***achat*** et les relations ***fait par*** et ***concerner*** sont fusionnés en une relation ***(acheter)*** dans le XSD$_2$ (voir cas ***(a)*** de la figure 2). Nous remarquons que la propriété ***date*** portée sur le concept ***achat*** (dans le schéma source) est identique à la propriété ***date*** dans le schéma cible qui est attachée à la relation ***acheter***.

Lors de la phase de ***mapping***, le module d'intégration organise l'ordre d'application de ces différents opérateurs dans les fichiers XSL.

5 Conclusion et perspectives

L'intégration de sources de données hétérogènes dans le contexte du Web est un problème d'actualité. L'apport du langage XML comme standard d'échange et de représentation de ces données est indéniable et facilite la mise en place des différentes approches proposées. En effet, XML permet de représenter des données indépendamment de leurs présentations et de leurs stockages. Aussi, tous les systèmes d'intégration actuels par médiateur utilisent XML comme modèle pivot. Cependant,

XML ne permet pas d'éviter les problèmes de conflits sémantiques et logiques. Notre travail de recherche consiste donc à proposer une formalisation de l'intégration pour diminuer ces types de conflits. La phase de ***pré-intégration*** s'appuie sur la puissance de représentation XML.

Nous cherchons à l'heure actuelle à affiner la gestion des règles d'intégration et de résolution de conflits pour avoir des correspondances entre concepts plus précises. La phase de fusion détecte les conflits sémantiques et logiques possibles par l'application de degré de similitude entre concepts selon les deux dimensions structurelle et sémantique. Pour cela, nous nous appuyons sur la théorie des graphes et sur la théorie des catégories. Nous tentons actuellement d'affiner la mesure du degré de similitude afin d'améliorer l'ordre d'applications des opérateurs qui génèrent des liens structurels et sémantiques entre sources de données. L'automatisation partielle des opérateurs devrait s'appuyer sur l'application des techniques d'acquisition de connaissances.

Références

1. Abiteboul, S., Buneman, P., Suciu, D. Data on the Web: From Relations to Semistructured Data and XML. Morgan Kaufmann, (1999)
2. Bergamaschi, S., Castano, S., Vinci M.: Semantic integration of semistructured and structured data sources. SIGMOD Record, March (1999) 54-59
3. Bird, L., Goodchild, A., Halpin, T.: Object Role Modelling and XML-Schema. Conceptual Modeling-ER 2000, 19th International Conference on Conceptual Modeling, Salt Lake City, Utah, USA, October 9-12, Proceedings, , (2000) 309-322
4. Boucelma, O., Lacroix Z.: InterMed–interface de médiation pour les systèmes d'information. ISI-NIS, Vol 6, n°3, (2001) 33-60
5. Boukottaya, A., Vanoirbeek, C., Paganelli, F., Abou-Khaled, O.: Automating XML documents transformations: a conceptual modelling based approach. Proceedings of the first Asian-Pacific conference on Conceptual modelling , ACM Volume 31, Dunedin, New Zealand (2004) 81-90
6. Castor: Web site of Castor Project, http://castor.exolab.org/
7. Cousin, R.: Apport de la théorie des catégories à la représentation des connaissances. Thèse de doctorat, Université Pierre et Marie Curie, Paris VI, (1988)
8. Delobel, C., Reynaud, C., Rousset, M.C., Sirot, J.P., Vodislav, D.: Semantic integration in Xyleme:A uniform tree-based approach. Data and Knowledge Engineering 44, (2003) 267-298
9. Doan, A., Domingos, P., Havely A.: Learning to match the schema of date sources: A multistrategy approach, Machine Learning, (2003) 279-301
10. Garcia-Molina, H., Papakonstantinou, Y., Quass, D., Rajamaran A., Sagiv Y., Ullman, J., Vassalos, V., Windom, J.: The TSIMMIS Approach to mediation : Data Models and Languages. Journal of Intelligent Information Systems, (1997)
11. Gardarin, G., Sha, F., Ngoc, T.: XML-based Conponents for Federating Multiple Heterogeneous Data Sources. Proceedings of ER'99, 18th International Conference on Conceptual Modeling, (1999) 506-519
12. Haas, L.M., Kossmann, D., Wimmers, E.L., Yang, J.: Optimizing Queries Across Diverse DataSources. In 23th international Conference on VLDB, Athens, Greece, (1997) 276-285

13. Kurgan, L., Swiercz, W., Cicos, K.J.: Semantic Mapping of XML Tags Using Inductive Machine Learning, Proceedings of the 2002 International Conference on Machine Learning and Applications (ICMLA'02), Las Vegas, NV (2002) 99-109
14. Lamolle, M., Mellouli, N. : Intégration de bases de données hétérogènes via XML. EGC'2003, Atelier fouille de données, Lyon (2003)
15. Li, W.S., Clifton, C.: Semint: a tool for identifying attributes correspondences in heterogeneous databases using neural networks, Data Knowledge Engineering, (2000)
16. Madhavan, J., Bernstein, P., Rahm, E.: Generic Schema Matching with Cupid. The VLDB Journal, (2001) 49-58
17. Manolescu, I., Florescu, D., Kossmann, D., Olteanu, D., Xhumari, F.: Agora: Living with XML and Relational. Proc. of the Int'l Conf. on Very Large Databases (VLDB), Cairo, (2000)
18. McBrien, P., Poulavassilis, A.: Schema Evolution in Heterogeneous Database Architectures, A Schema Transformation Approach. In Proc. CAiSE'02, Toronto, Ontario, Canada (2002)
19. Melnik, S., Rahm, E., Bernstein, P.A.: Developing metadata-intensive applications with Rondo. Journal of web semantics, (2003)
20. Melnik, S., Molina-Garcia, H., Rahm, E.: Similarity Flooding: A Versatile Graph Matching Algorithm, Proceedings of International Conference on Data Engineering, (2002)
21. Milo, T., Zohar, S. Using Schema Matching to Simplify Heterogeneous Data translation. Proceedings of International Conference on VLDB, (1998)
22. Millan, T., Mulatéro, F., Lamolle, M.: Design, Share and Re-use of Data and Applications into a Federate Database System. Onzièmes Journées Internationales le Génie Logiciel et ses Applications, Paris, France (1998)
23. Papotti, P., Torlone R.: An Approach to Heterogeneous Data Translation based on XML Conversion. Journal of Web Engineering, (2003)
24. Parent, C., Spaccapietra, S.: Intégration de bases de données: Panorama des problèmes et des approches. Ingénierie des systèmes d'information Vol.4, N°3. Lavoisier (1996)
25. Pottinger, R.A., Bernstein, P.A. Merging models bases on given correspondences. Proceedings of International Conference on VLDB,(2003)
26. Rahm, E., Do, H.D., Maßmann S., Matching large XML schemas, ACM SIGMOD Record, v.33 n.4, December (2004)
27. Rahm, E., Bernstein, P. A survey of approaches to automatic schema matching, VLDB journal, (2001) 334-350
28. Siméon, J.: Data Integration with XML: A Solution for Modern Web Applications. Lecture at Temple University, March (2000)
29. Tranier, J., Baraër, R., Bellahsène, Z., Teisseire, M. Where's Charlie: Family-Based Heuristics for Peer-to-Peer Schema Integration. IDEAS, (2004) 227-235
30. Vargas-Solar G., Doucet, A. : Médiation de données: solutions et problèmes ouverts, Actes des deuxièmes assises nationales du GdRI3, (2002)
31. Widom, J.: Research Problems in Data Warehousing. In Proceedings of the 1995 International Conference on Information and Knowledge Management (CIKM), Baltimore, Maryland, (1995)
32. XML 1.0 (Second Edition), 6 October 2000. W3C Consortium Recommendation, Available http://www.w3.org/TR/REC-xml-20001006.
33. XML Schema, W3C Recommendation: XML Schema Primer, W3 Consortium, Available at http://www.w3.org/TR /xmlschema-0, (2001)
34. XSLT 1.0. W3C Recommendation : XSL Transformations XSLT Version 1.0, Available at http://www.w3.org/TR/xslt, (2002)