Intégration de sources hétérogènes par matching semi-automatique de schémas XML étendus

Amar Zerdazi, Myriam Lamolle

Intégration de sources hétérogènes par matching semi-automatique de schémas XML étendus

Amar ZERDAZI, Myriam LAMOLLE

LINC – Université Paris VIII
IUT de Montreuil
140, rue de la Nouvelle France
93100 – Montreuil, France
{a.zerdazi, m.lamolle}@iut.univ-paris8.fr

RÉSUMÉ. *Lors de l'intégration de sources de données hétérogènes, une des tâches les plus importantes est la transformation (semi-)automatique de ces sources par une phase de recherche de correspondances appelée schema matching. Le but de cet article est de proposer une solution lors de la phase de matching entre des schémas XML étendus appelés EXS (Enhanced XML Schemas). L'ensemble des schémas EXS, dans notre approche, est considéré comme une fédération de schémas XML issus de différents schémas de sources hétérogènes (relationnel, objet, XML, etc.) et enrichis par des métaconnaissances sémantiques. Nous présentons ici les problèmes majeurs liés à cette tâche cruciale, notamment en ce qui concerne la sémantique des schémas. Nous proposons pour cela un algorithme basé sur l'identification sémantique des entités équivalentes afin de produire une série de mappings.*

ABSTRACT. *At the time of the integration of heterogeneous data sources, one of the most important tasks is the semiautomatic transformation of these sources by schema matching. The goal of this paper is to propose a solution at the time of matching phase between Enhanced XML Schemas, called EXS. In our approach, the set of the EXS schemas, are considered like a federation of XML schema descended of different heterogeneous sources schemas (relational, object, XML, etc.) and enriched by the semantic metaknowledge. We present here the major problems bound to this crucial task, notably with regard to the semantic of schemas. So, we propose an algorithm based on the semantic identification of the equivalent entities in order to produce a set of mappings.*

MOTS-CLÉS: *appariement de schémas, hétérogénéité, intégration sémantique, schéma XML.*

KEYWORDS: *heterogeneity, schema matching, semantic integration, XML schema.*

Catégorie: Jeune Chercheur

1. Introduction

Le *schema matching* est le processus de manipulation de schémas afin de détecter des correspondances entre deux schémas hétérogènes et de déterminer un ensemble de transformations appelées *mappings.* Le schema matching apparaît comme une phase essentielle dans la majorité des applications qui manipulent diverses données avec diverses structures dans divers modèles (tels que schéma relationnel, schéma orienté-objet, DTD XML, etc.). Ces derniers sont représentés par des schémas hétérogènes. Le schema matching permet donc de manipuler ces schémas hétérogènes (Bernstein *et al.,* 2000), (Andritsos *et al.,* 20002). Actuellement, cette tâche est généralement réalisée manuellement, ce qui explique son coût élevé et son manque de fiabilité. En conséquence, beaucoup de travaux de recherche ont essayés d'automatiser ou semi-automatiser ce processus pour des raisons multiples. Selon (Drew *et al.,* 1993), les éléments des schémas sont appariés à partir de leur sémantique. Cette dernière peut être implicite et cachée dans les sources de données ; aussi, le processus de matching tient compte de la structure et des instances des schémas. Les schémas des différentes sources et applications sont de nature hétérogènes, bien que la plupart des données qui les décrivent sont sémantiquement équivalentes (Abiteboul *et al.,* 1997). Dans ce cas, la structure et la syntaxe employées peuvent être, elles aussi, différentes. Pour la résolution des conflits sémantiques et structurels, la tâche de matching relie souvent des éléments par rapport à leurs noms, leurs types, leurs contraintes, leurs valeurs, etc. Cependant, de telles informations sont souvent incertaines ou incomplètes. Enfin, la tâche de matching ne peut être complètement automatique, du fait qu'elle requiert une intervention humaine ayant une expérience dans le domaine de données (Doan *et al.,* 2001).

Nous présentons le reste de l'article de la manière suivante. Dans la section 2, nous parlons des principaux prototypes de schema matching concernant XML. La section 3 illustre notre approche avec la définition du schéma EXS et son graphe EXS qui sont nécessaires pour son exécution. La section 4 est consacrée entièrement à notre processus de matching. Pour cela, nous définissons pour chaque étape son algorithme approprié. Enfin, nous concluons et présentons les évolutions à réaliser en section 5.

2. Etat de l'art

Le schema matching n'est pas un problème récent pour la communauté des bases de données. (Castano *et al.,* 1999) ont développés le système ARTEMIS supportant l'intégration de schémas hétérogènes avec la mesure de similarité entre les noms d'éléments, le type de données et la structure. Actuellement, plusieurs travaux s'intéressent à cette problématique afin de résoudre le problème de schema matching (Rahm *et al.,* 2001). Avec l'avènement du langage XML (W3C-XML, 1998), plusieurs approches sur le matching se sont tournées vers ce nouveau format pensant

tirer partie de sa structure hiérarchique et de sa représentation universelle. Dans ce qui suit, nous représentons quelques exemples récents de schema matching qui emploient XML dans leurs algorithmes.

L'idée du matching *(syntaxique)* générique a été implémentée pour la première fois dans le système Cupid (Madhavan *et al.,* 2001) qui est une approche hybride combinant à la fois des techniques linguistiques et structurelles. Il se base sur la comparaison de schémas sans recourir aux instances. COMA (Do *et al.,* 2003) est aussi un outil générique de schema matching implémenté avec des composants plus récents dont le but est d'optimiser l'algorithme de matching. La principale innovation de ce système par rapport à Cupid, réside dans la flexibilité de son architecture.

Aujourd'hui, il existe peu d'approches implémentées et qui exploitent les techniques de matching sémantique. Par exemple les schémas de COMA sont encodés avec DAG *(Directed Acyclic Graphs)* où les éléments du graphe sont représentés par leurs chemins et analysés par des techniques de comparaison textuelles. La même idée est exploitée par (Melnik *et al.,* 2002) dans SF *(Similarity Flooding)*. SF utilise un algorithme de matching hybride basé sur l'idée de propagation de similarité. L'inconvénient de SF est qu'il ignore tous les types de contraintes, ce qui diminue sans doute sa performance. Les contraintes telles que les types de données, les contraintes d'intégrités sont utilisées en post-traitement afin de *filtrer* manuellement les paires de mappings avec l'aide d'experts.

CTXMatch (Serafini *et al.,* 2003) utilise trois niveaux de connaissances sémantiques (sémantique lexicale, sémantique du domaine et la sémantique structurelle) et retourne cinq types de relations sémantiques (moins général, plus général, équivalent, compatible, incompatible) entre les paires de concepts. Le problème de CTXMatch est qu'il est applicable uniquement aux *concepts hiérarchiques.*

3. Notre approche

La majorité des travaux de recherche sur le schema matching reposent sur l'hypothèse que l'utilisateur humain ou l'expert du domaine est capable de décider si le résultat de matching *(mapping)* est correct ou non. Cette décision ou suggestion implique la connaissance des schémas a priori. En effet, chaque utilisateur doit avoir une représentation claire sur l'ensemble des concepts et sur la sémantique des relations entre ces derniers (Boukottaya *et al.,* 2004). Les schémas sont considérés comme formels, car leur syntaxe est basée sur une représentation concrète de l'ensemble des concepts et de leurs relations (par exemple, schéma relationnel, schéma XML, etc.).

3.1. *Schéma EXS (Enhanced XML Schema)*

L'objectif de notre approche est de fournir un modèle d'intégration souple, capable de fédérer différentes sources de données hétérogènes en prenant en compte les dimensions structurelle et sémantique des schémas sources. Pour cela, la définition de notre modèle doit comporter un nombre minimal d'entités à manipuler mais suffisant pour traduire les schémas (Lamolle *et al.,* 2003). Dans (Lamolle *et al.,* 2005), nous avons définis des règles d'extraction qui homogénéisent la représentation des différents schémas sources à intégrer en l'exprimant sous forme de schémas XML *(i.e. XSD)* (W3C-XSD, 2001). Cette étape est appelée ***modélisation structurelle***. Une fois cette transformation faite, la partie sémantique des XSDi créés est affinée par l'adjonction de métaconnaissances *(modélisation sémantique)* (Zerdazi *et al.,* 2005), soient déduites (par exemple du catalogue des données dans le cas de bases de données relationnelles), soient précisées par l'expert du domaine.

Nous décrivons ici brièvement les deux étapes primordiales pour la définition des schémas XML étendus *(EXS : Enhanced XML Schemas)*. Dans (Zerdazi, 2005), nous détaillons amplement ces deux phases de modélisation des schémas EXS.

3.1.1. *Modélisation structurelle*

Le schéma XML étendu *(noté EXS)* tel qu'il est modélisé lors de la première étape (modélisation structurelle) est composé de trois types d'entité à savoir :

Concept : un concept dans le schéma EXS est équivalent à la notion d'entité dans le modèle Entité-Relation, de classe dans le modèle objet, et d'élément dans les documents XML. Il peut engendrer des propriétés et/ou d'autres concepts imbriqués, et porte des relations avec d'autres concepts.

Relation : une relation dans le schéma EXS correspond aux associations structurelles et/ou sémantiques existantes entre deux ou plusieurs concepts. Cette relation possède un degré et une catégorie prédéfinie et d'autres métaconnaissances.

Propriété : une propriété dans le schéma EXS est équivalente à la notion d'attribut dans le modèle relationnel, objet, ou semi-structuré. Selon le schéma d'origine, les propriétés peuvent apparaître dans les concepts et/ou les relations.

3.1.2. *Modélisation sémantique*

La phase d'enrichissement sémantique suit l'étape de modélisation structurelle des schémas EXS. Cette dimension sémantique des entités (concepts, relations, propriétés) est enrichie par l'apport de métaconnaissances lors d'une étude plus approfondie de l'état des entités (structure, complétude, niveau d'encapsulation, type d'association, contraintes sur les propriétés, etc.). Ces métaconnaissances sont utilisées lors de la phase d'intégration afin d'obtenir des mises en correspondance plus précises. L'enrichissement sémantique consiste à étendre la structure des entités (concepts, relations, propriétés) grâce à des facettes.

Complétude d'un concept : Cette métaconnaissance nous renseigne sur l'état du concept, c'est-à-dire s'il est représenté avec la même structure dans la totalité des schémas XML.

Niveau d'un concept : Ceci traduit la visibilité d'un concept par rapport aux autres. Le niveau d'encapsulation est sauvegardé par l'attribut «level» dans la définition du concept. Via cette métaconnaissance, nous distinguons les concepts les plus pertinents *(level =1 ou 2)* dans la hiérarchie des concepts *(level >2).*

Catégorie des relations : Nous avons introduit une métaconnaissances «category» qui déterminera les différentes catégories possible dans le schéma EXS *(dépendance, généralisation/spécialisation, Association).*

Orientation et rôle des relations : Pour l'orientation, nous employons deux métaconnaissances «source» et «target» de type logique *(true/false)* qui expriment le(s) concept(s) source et le(s) concept(s) cible. Le rôle est défini par la métaconnaissance «role» qui permet de distinguer les relations explicitées sémantiquement de celles qui ne sont que des associations.

Contraintes sur les propriétés : Les propriétés clé des concepts doivent être présentes obligatoirement dans les schémas EXS. En particulier, ces attributs doivent être déclarés avec la restriction de spécification «use» qui prendra la valeur *required.*

Contraintes de cardinalités : Nous employons les facettes *minOccurs* et *maxOccurs* des XML-Schemas afin de préciser les cardinalités des propriétés et des concepts dans le schéma EXS.

3.2. *Graphe EXS*

La majorité des approches abordant le problème de schema matching tiennent compte de deux situations distinctes :

i. Représentation des données ou du modèle conceptuel sous forme de graphe.

ii. Exécution du processus de matching sur le graphe résultant.

Nous représentons le schéma EXS par un graphe possédant un ensemble de nœuds, d'arcs et de contraintes. Les nœuds sont connectés entre eux via des arcs. Des contraintes peuvent apparaître sur les nœuds et sur les arcs du graphe EXS.

Définition 1 (graphe EXS)

Le graphe EXS **G** est représenté par le triplet $\langle N_G, A_G, C_G \rangle$ où :

$\mathbf{N_G}$: est l'ensemble des nœuds du graphe G,
$\mathbf{A_G}$: est l'ensemble des arcs existants entre les nœuds du graphe G,
$\mathbf{C_G}$: est l'ensemble des contraintes sur les nœuds et les arcs du graphe G.

3.2.1. *Nœuds du graphe EXS*

Nous distinguons deux types de nœuds dans le graphe EXS. Les nœuds atomiques et les nœuds complexes qui représentent respectivement les propriétés et les concepts dans notre schéma EXS. Les nœuds complexes possèdent un contenu complexe qui peut être lui-même constitué de nœuds atomiques ou d'autre nœuds complexes liés par des arcs. Les nœuds atomiques possèdent un contenu simple avec un domaine de définition tels qu'une chaîne de caractères, entier, date, etc. Dans la figure 1, les nœuds *projet*, *membre* et *publication* sont des nœuds complexes. Tandis que *libelle*, *nom*, *titre*, etc. sont des nœuds atomiques.

Définition 2 (nœud)

Un nœud $n \in N_G$ est représenté par le triplet $\langle n_{nom}, n_{type}, n_{contenu} \rangle$
$\mathbf{n_{nom}}$: est le nom du nœud,
$\mathbf{n_{type}}$: est le type du nœud (atomique ou complexe),
$\mathbf{n_{contenu}}$: est le contenu du nœud.

Où N_A et N_C sont respectivement les nœuds atomiques et les nœuds complexes.
Avec $N_A \cap N_C = \Phi$ et $N_A \cup N_C = N_G$.

3.2.2. *Arcs du graphe EXS*

Chaque arc dans le graphe EXS lie deux nœuds ou plus, en capturant l'aspect structurel des relations existantes dans le schéma EXS. Nous distinguons donc trois types d'arc : dépendance, généralisation/spécialisation et association (qui représentent les types des relations existantes dans les schémas EXS).

Définition 3 (arc)

Un arc $a \in A_G$ est représenté par le triplet $\langle a_{type}, a_{source}, a_{cible} \rangle$
$\mathbf{a_{type}}$: est le type de la relation entre les nœuds complexes,
$\mathbf{a_{source}}$: le nœud source $\in N_G$ est lié à a_{source},
$\mathbf{a_{cible}}$: le nœud cible $\in N_G$ est lié à a_{cible}.

Où A_d, A_g et A_a sont respectivement les arcs de *dépendance*, *généralisation/spécialisation* et *association.*
Avec $A_d \cap A_g \cap A_a = \Phi$ et $A_d \cup A_g \cup A_a = A_G$.

3.2.3. *Contraintes dans le graphe EXS*

Pour la définition des contraintes dans le graphe EXS, nous utilisons certaines contraintes et/ou métaconnaissances employées dans le schéma EXS tels que l'unicité, le domaine de définition, les catégories des relations, etc.

Définition 4 (contrainte)

Une contrainte $c \in C_G$ est représentée par le triplet $\langle c_{nom}, c_{type}, c_{valeur} \rangle$
$\mathbf{c_{nom}}$: est le nom de la contrainte (unicité, cardinalité, etc.),
$\mathbf{c_{type}}$: est la catégorie de la contrainte (contrainte sur les nœuds ou sur les arcs),

c_{valeur} : est la valeur de la contrainte.

Définition 5 (chemin)

Le chemin entre les nœuds $n_1$et n_k est représenté par la séquence $ch=(n_1, n_2,\ldots n_k)$, où $\{n_1, n_2,\ldots n_k\} \hat{I} N_G$. De plus, entre tout nœud complexe n_i et n_{i+1} $(1 £ i £ k-1)$, $\$ a_i \acute{a} 1, n_i, n_{i+1} \tilde{n} \hat{I} A_G$. Supposons que n_1 et n_k sont respectivement les nœuds source et cible d'un chemin. Ce dernier peut passer par i nœuds $(1 £ i £ k)$ intermédiaires.

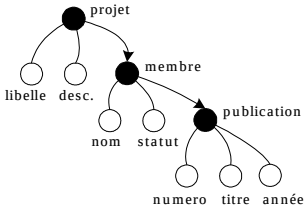


Figure 1. *Exemple d'un graphe EXS*

4. Notre processus de matching

Les méthodes de matching existantes dans le cadre d'une modélisation XML ne combinent pas à l'heure actuelle les dimensions syntaxique, structurelle et sémantique des sources à intégrer pour optimiser le processus de matching.

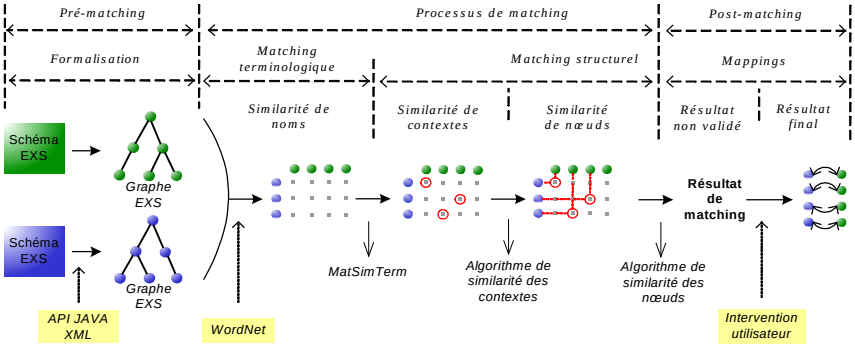


Figure 2. *Processus de matching*

Pour prendre en compte ces trois dimensions, nous procédons en trois étapes à savoir une étape de pré-matching, une de matching et enfin une de post-matching comme le montre la figure 2. Chacune de ces étapes donne lieu à la définition d'algorithmes spécifiques selon des critères précis. Nous détaillons plus précisément les algorithmes de la phase de matching proprement dite.

4.1. *Pré-matching*

Afin de résoudre le problème de matching, notamment pour sa semi-automatisation, nous devons identifier dans un premier temps les informations d'entrée *(input informations)* qui sont nécessaires pour la réalisation de cette tâche. Généralement les informations d'entrée pour le matching incluent tout type de connaissances concernant les schémas de données.

Informations sur le schéma : nous nous référons ici à la structure du schéma concernant les concepts, leurs propriétés, leurs organisations dans le schéma, les différentes relations entre les concepts, les contraintes exprimées, etc. (Lamolle et *al.*, 2005). Dans le cadre du *XML schema matching* et particulièrement dans son matching structurel, il existe peu d'algorithmes qui exploitent les caractéristiques avancées des XML-Schemas. Notre proposition des schémas EXS tient compte de ces caractéristiques importantes dans le processus de matching.

Informations fournies par l'utilisateur : nous rappelons qu'il est impossible d'automatiser d'une façon complète la tâche de matching. L'intervention humaine dans ce cas est absolument requise. Dans la plupart des algorithmes de matching, l'utilisateur ou l'expert est sollicité durant la phase de pré-matching afin de valider ce premier traitement. Le but de notre de travail est de produire un schéma résultat qui représente les sources de données en entrée. Pour cela, nous supposons qu'il soit compréhensible par son utilisateur, et que ce dernier soit capable à son tour de juger le résultat de ce pré-matching (s'il est correct ou non, avec ou sans perte d'informations). Dans ce cas, l'utilisateur peut rejeter ou modifier le résultat.

REMARQUE. — Les instances de données sont aussi considérées comme des informations d'entrée pour le processus de matching. Généralement, seuls les systèmes basés sur le matching par apprentissage *(Learner-based matching)*, comme LSD (Doan *et al.*, 2002) par exemple, tiennent compte de cette information en analysant le contenu des données. Cependant, dans le contexte de notre approche, nous ne tenons pas compte de ce type d'information car elle est basée sur la structure des données à traiter et non aux données elles-mêmes. Nous argumentons notre choix, du fait que le contenu des données ne garantit pas toujours une représentation complète et disponible.

4.2. *Post-matching*

Les systèmes de schema matching fournissent souvent des mappings avec un coefficient, généralement appelé coefficient de similarité compris entre 0 *(forte dissimilarité)* et 1 *(forte similarité)*, afin d'indiquer la plausibilité des correspondances. Notre objectif n'est pas seulement l'implémentation d'un algorithme de matching pour les schémas XML mais aussi l'utilisation de son résultat afin d'automatiser les transformations entre ces schémas et de produire des scripts XSLT stipulant les règles de transformation. Dans ce dernier cas, nous ne réexécutons pas le processus de matching, mais réalisons juste une simple mise à jour entre les éléments modifiés ou rajoutés dans les schémas source et cible. Comme

pour le pré-matching, l'utilisateur intervient lors de cette phase afin de valider et de filtrer ces résultats.

4.3. *Matching*

4.3.1. *Matching terminologique*

Le matching terminologique représente la première étape de notre processus de matching. Son but est de mesurer la similarité entre les différentes entités des schémas EXS (concepts, relations et propriétés), basés sur leurs noms *(labels).* Nous proposons donc une méthode qui prend en entrée les labels des entités à comparer à partir des schémas source et cible et qui produit en sortie un *coefficient de similarité* compris entre [0,1]. Pour l'ensemble des noms des entités, nous dressons une matrice contenant l'ensemble des mesures de similarité terminologique. Nous utilisons WordNet (Miller, 1995) afin de déterminer les liens sémantiques entre les mots. Chaque mot possède plusieurs sens représentés par un ensemble de synonymes *(synset)* et une définition. L'avantage de WordNet (Evens, 1998) est sa structuration par des relations lexicales entre *synset* telles que synonymie, antonymie, hyponymie et de relations sémantiques telles que méronymie et morphologie.

Algorithme 1 – Matching Terminologique (M.T)

1. ***Entrée :*** G_S, G_T *// graphe source et cible*
2. ***Sortie :*** MatSimTerm *// matrice de similarité terminologique*
3. ***Début***
4. $N_{GS} \leftarrow$ nœud(G_S)
5. $N_{GT} \leftarrow$ nœud(G_T)
6. Pour chaque $n_{nomS} \in N_{GS}$ faire
7. Pour chaque $n_{nomT} \in N_{GT}$ faire
8. TSim(n_{nomS}, n_{nomT}) $\leftarrow$ compute_TSim (n_{nomS}, n_{nomT})
9. MatSimTerm $\leftarrow$ MatSimTerm $\cup$ TSim(n_{nomS}, n_{nomT})
10. return MatSimTerm
11. ***Fin.***

Figure 3. *Algorithme de matching terminologique*

L'algorithme de matching terminologique présenté dans la figure 3 prend en entrée les noeuds sources et les nœuds cibles, respectivement des deux graphes source *(G_S)* et cible *(G_T)*, et retourne la matrice de similarité terminologique *(MatSimTerm)* qui stocke les coefficients de similarité entre les nœuds des graphes.

4.3.2. *Matching structurel*

Les techniques de matching décrites dans la section 4.1 comparent seulement les nœuds entre le schéma source et le schéma cible. Ce dernier peut produire des matchings incorrects provoquant des incertitudes et des erreurs dans les représentations des entités candidates. Le matching structurel a pour but de corriger ces anomalies, en se basant sur leurs **contextes structurels**. Soient les deux graphes EXS de la figure 3 à partir desquels nous appliquons le matching terminologique sur les nœuds *laboratoire/adresse* (du graphe source G_{source}) et *auteur/adresse* (du graphe cible G_{cible}). Le matching terminologique définit une forte similarité entre ces deux nœuds, où le premier nœud concerne l'adresse du département, tandis que le second est l'adresse de l'auteur. Si nous appliquons maintenant le matching structurel (étape suivante dans le processus), en comparant leurs contextes, nous pouvons facilement déduire que les deux nœuds *adresse* ne sont pas équivalents et ils sont rejetés dans le matching structurel. Par contre, le nœud *laboratoire/adresse* (G_{source}) peut être apparié vers le nœud *laboratoire/localisation* (G_{cible}). Et la relation source entre *laboratoire/adresse* peut être appariée aussi vers la relation cible *laboratoire/localisation.* Dans notre approche, le matching structurel se base sur la notion de *contexte du nœud*.

4.3.2.1. *Similarité des contextes*

Définition du contexte d'un nœud

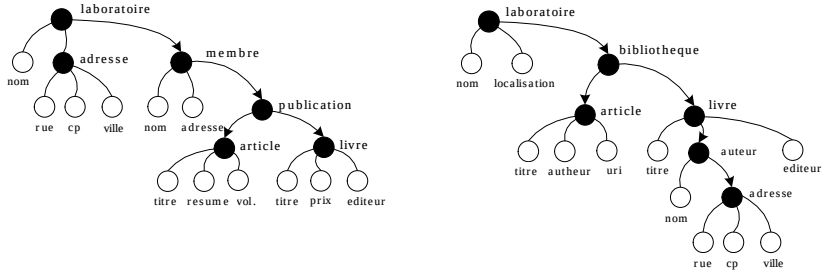
Le rôle du matching structurel est de comparer les contextes structurels de chaque nœud appartenant aux graphes EXS. En conséquence, nous définissons la notion de contexte structurel. Pour cela, nous distinguons trois classes de contexte qui dépendent de la position des nœuds dans le graphe EXS.

Contexte-racine : Le contexte-racine (ou *root*) du nœud n_i est défini par le chemin racine possédant le nœud n_i comme nœud cible et le nœud racine comme nœud source dans le graphe. Par exemple, le contexte-racine du nœud *publication* dans la figure 4(a) est défini par le chemin *laboratoire/membre/publication* qui décrit les publications d'un membre dans son laboratoire. Dans le cas où le contexte-racine du nœud racine *(root node)* est vide, nous affectons la valeur *null.*

Contexte-intermédiaire : Le contexte-intermédiaire du nœud n_i engendre les nœuds directement imbriqués *(complexes et atomiques)* dans ce dernier. Il reflète la structure de ses descendants dans le graphe EXS. Par exemple, le contexte-intermédiaire du nœud *publication* dans la figure 4(a) est défini par *(article, livre)*. Le contexte intermédiaire des nœuds atomiques est assigné à *null.*

Contexte-feuille : Les terminaux ou feuilles dans l'arbre XML représentent les données atomiques décrites par leur schéma. Le contexte-feuille du nœud n_i inclut les feuilles du sous-arbre lié au nœud n_i. Par exemple, le contexte-feuille du nœud *publication* dans le graphe EXS de la figure 4(a) est défini par l'ensemble *(titre,*

résumé, vol., titre, prix, éditeur). Notons que le contexte-feuille des nœuds atomiques, par défaut, est *null*.



(a) Graphe EXS source *(Gsource)* (b) Graphe EXS cible *(Gcible)*

Figure 4. *Graphes EXS (source et cible)*

Enfin, le contexte du nœud est défini par l'union de ces trois types de contextes. Deux nœuds sont considérés structurellement similaires s'ils possèdent des contextes similaires. La notion de ***similarité de contexte*** est utilisée dans Cupid et SF ; néanmoins, aucun des deux n'abordent les trois classes de contexte. Pour le calcul de la ***similarité structurelle*** entre deux nœuds, nous mesurons respectivement la similarité de leurs contexte *racine*, *intermédiaire* et *feuille*.

Similarité du contexte-racine

La similarité du contexte-racine, notée *ctx_rootSim* calcule la similarité entre deux nœuds, basée sur leurs contextes-racines. Tant que les deux nœuds n_1 et n_2 respectivement *source* et *cible* sont décrits par leur chemin racine (chemin de la racine vers le noeud), la mesure de *ctx_rootSim* est équivalente à la mesure des deux chemins *chSim*[1] entre n_1 et n_2, pondérée par leur similarité terminologique.

$$\mathbf{ctx\_rootSim\ (n_1, n_2)\ \neg\ chSim\ ((racine_1, n_1), (racine_2, n_2))\ \acute{}\ TSim(n_1, n_2)}$$

Similarité du contexte-intermédiaire

La similarité du contexte-intermédiaire *(ctx_IntSim)* est obtenue par la comparaison des nœuds immédiatement au-dessous de la racine (qui représente les concepts pertinents dans le schéma EXS). Supposons le nœud n_1 possédant l'ensemble $(n_{11},\ldots n_{1m})$ et le nœud n_2 avec $(n_{21},\ldots n_{2k})$. Afin de déterminer la similarité

[1] Pour la mesure de *chSim* entre n_1 et n_2, nous utilisons l'algorithme de la programmation dynamique classique afin de calculer *Longest Common Subsequence (LCS)* entre les chemins racines $ch_1(racine_1, n_1)$ et $ch_2(racine_2, n_2)$.

entre ces deux nœuds, nous calculons la similarité terminologique *(TSim)* entre chaque paire de nœuds fils dans les deux ensembles. Ensuite, nous sélectionnons les paires de matching avec le degré de similarité le plus élevé. Enfin, nous choisissons la moyenne de ces similarités importantes. L'algorithme 2 de la figure 5 illustre le calcul de cette similarité.

Algorithme 2 – Similarité du Contexte-Intermédiaire (S.C.I)

1. ***Entrée :*** n_1, n_2, MatSimTerm *// possédant respectivement n, m nœuds fils*
2. ***Sortie :*** ctx_IntSim
3. ***Début***
4. Sim_Inter $\leftarrow \max_{i\in[1,n],\ j\in[1,m]} \{(n_{1i}, n_{2j}, TSim) \in MatSimTerm\}$
5. paire_Sim $\leftarrow (n_{1i}, n_{2j}, Sim\_Inter)$
6. $ctx\_IntSim \leftarrow \dfrac{\sum_{(n1i,\ n2j,\ Sim\_Inter)\in paire\_Sim} (Sim\_Inter)}{\max(n, m)}$
7. return ctx_IntSim
8. ***Fin.***

Figure 5. *Algorithme de similarité du contexte-intermédiaire*

Similarité du contexte-feuille

La similarité du contexte-feuille *(ctx_FeSim)* calcule la similarité entre deux nœuds n_1 et n_2 en comparant respectivement l'ensemble des nœuds feuilles de n_1 *(n_1{feuilles})* avec l'ensemble des nœuds feuilles n_2 *(n_2{feuilles})*. Pour la mesure de similarité entre deux nœuds feuilles n_{f1} $\hat{I}$ n_1{feuilles} et n_{f2} $\hat{I}$ n_2{feuilles}, nous comparons les contextes dans lesquels ses derniers apparaissent. Si le nœud feuille n_f $\hat{I}$ n_i{feuilles}, alors son contexte est déterminé par le chemin $ch(n_i,\ldots n_f)$. La similarité de contexte-feuille est donc obtenue par la comparaison de leurs chemins. Le calcul de similarité *ctx_FeSim* entre deux nœuds atomiques est obtenu par l'algorithme 3 de la figure 6.

Algorithme 3 – Similarité du Contexte-Feuille (S.C.F)

1. ***Entrée :*** n_1, n_2 *// possédant respectivement n, m nœuds feuilles*
2. ***Sortie :*** ctx_FeSim
3. ***Début***
4. Pour chaque $n_{fi} \in n_1${feuilles} faire
5. Pour chaque $n_{fj} \in n_2${feuilles} faire
6. Sim_feuilles $(n_{fi}, n_{fj}) \leftarrow chSim((n_1, n_{fi}), (n_2, n_{fj}) \times TSim(n_{fi}, n_{fj})$
7. temp_Sim $\leftarrow \max_{i\in[1,n],\ j\in[1,m]} (n_{fi}, n_{fj}, sim\_feuilles\}$
8. paire_Sim $\leftarrow (n_{fk}, n_{fh}, temp\_Sim)$

9. $ctx\_FeSim \leftarrow \frac{\sum_{(nfi, nfj, Sim\_feuilles) \in paire\_Sim} (Sim\_feuilles)}{\max(n, m)}$

10. return ctx_FeSim

11. ***Fin.***

Figure 6. *Algorithme de similarité du contexte-feuille*

4.3.2.2. *Similarité des nœuds*

Nous proposons dans cette section la mesure de similarité entre deux nœuds appartenant respectivement aux graphes source et cible, avec la combinaison de toutes les mesures et méthodes présentées précédemment. L'algorithme 4 de la figure 7 illustre le calcul de cette similarité. Nous distinguons trois cas possible :

Cas 1 : Les deux nœuds source et cible sont atomiques (N_A) ; c'est-à-dire leurs contextes intermédiaires et feuilles sont ***null*** (voir section 4.2.1). Dans ce cas, la similarité entre deux nœuds atomiques est définie par la similarité de leurs contextes racines, pondérée par leur similarité terminologique (lignes 5, 6 et 7). Par exemple, la similarité entre les nœuds atomiques *nom* (figure 3(a)) et *nom* (figure 4(b)) est égale à *ch((laboratoire/nom), (laboratoire/nom)) ´ TSim(nom, nom) = 1*.

Cas 2 : Un des deux nœuds à comparer est atomique. Supposons les deux nœuds n_1, n_2 avec $n_1 \hat{I} N_A$ (nœud atomique) et $n_2 \hat{I} N_C$ (nœud complexe). Les contextes intermédiaire et feuille du n_1 sont toujours à *null* (voir définitions ci-dessus). La similarité entre n_1, n_2 est obtenue par le calcul de leur contexte-racine (ligne 9) ; puis, le contenu de n_1 est défini par son nœud ***(notion de nœud atomique).*** Par contre, le contenu du nœud complexe n_2 est défini par son contexte-feuille. Nous proposons donc de calculer la moyenne de la similarité terminologique entre n_1 et les nœuds appartenant au contexte-feuille de n_2 (lignes 10, 11 et 12). La similarité entre n_1 et n_2 est obtenue finalement par la forte similarité de leur contexte racine et feuille (ligne 13).

Cas 3 : les deux nœuds source et cible sont complexes (N_C). Leur similarité est obtenue par la somme des similarités des contextes racine, intermédiaire et feuille (lignes 16 jusqu'à 21). Enfin, nous mettons à jour la matrice de similarité terminologique ***(MatSimTerm)*** par le résultat de la similarité obtenue (lignes 26).

Algorithme 4 – Similarité des Nœuds (S.N)

1. ***Entrée :*** n_1, n_2

2. ***Sortie :*** $sim(n_1, n_2)$

3. ***Début***

4. **Cas1. (n_1, n_2) Î N_A**

5. $ctx\_rootSim(n_1, n_2) \leftarrow chSim((racine_1, n_1), (racine_2, n_2)) \times TSim(n_1, n_2)$

6. $sim(n_1, n_2) \leftarrow ctx\_rootSim(n_1, n_2)$
7. return $sim(n_1, n_2)$
8. **Cas2. $n_1 \hat{I} N_A$ et $n_2 \hat{I} N_C$**
9. $ctx\_rootSim(n_1, n_2) \leftarrow chSim((racine_1, n_1), (racine_2, n_2)) \times TSim(n_1, n_2)$
10. Pour chaque $n_{fj} \in n_2\{feuilles\}$ faire
11. Calculer $TSim(n_{fj}, n_1)$
12. $ctx\_FeSim(n_1, n_2) \leftarrow \frac{\sum_{(nfj \in n2\{feuilles\}} TSim(n_{fj}, n_1)}{n_2\{feuilles\}}$
13. $sim(n_1, n_2) \leftarrow ctx\_rootSim(n_1, n_2) + ctx\_FeSim(n_1, n_2)$
14. return $sim(n_1, n_2)$
15. **Cas3. $(n_1, n_2) \hat{I} N_C$**
16. $ctx\_FeSim(n_1, n_2) \leftarrow SCF(n_1, n_2)$ *// Etape1 : Calcul de ctx_FeSim*
17. $ctx\_IntSim(n_1, n_2) \leftarrow SCI(n_1, n_2)$ *// Etape2 : Calcul de ctx_IntSim*
18. *// Etape3 : Calcul de ctx_IntSim*
19. $ctx\_rootSim(n_1, n_2) \leftarrow chSim((racine_1, n_1), (racine_2, n_2)) \times TSim(n_1, n_2)$
20. *// Etape4 : Calcul de la similarité entre n_1 et n_2*
21. $sim(n_1, n_2) \leftarrow ctx\_FeSim(n_1, n_2) + ctx\_IntSim(n_1, n_2) + ctx\_rootSim(n_1, n_2)$
22. *// Etape5 : Génération de la similarité*
23. Pour chaque $n_{nomsS} \in N_{GS}$ faire
24. Pour chaque $n_{nomT} \in N_{GT}$ faire
25. Calculer $sim(n_{nomS}, n_{nomT})$
26. $MatSimTerm \leftarrow MatSimTerm \cup (n_{nomS}, n_{nomT}, sim(n_{nomS}, n_{nomT}))$
27. return MatSimTerm
28. ***Fin.***

Figure 7. *Algorithme de similarité des nœuds*

Ces calculs de similarité (présentés dans la section 4) ont pour but d'optimiser les transformations semi-automatiques entre les schémas EXS, validées ensuite par l'utilisateur. L'étape suivante sera l'utilisation de ces résultats pour générer des scripts XSL entre les schémas source et cible en appliquant un ensemble d'opérateurs de transformations. Ces scripts seront utilisés lors du mapping proprement dit.

5. Conclusion

Le schema matching représente l'étape critique dans le processus de transformation de documents en général et dans l'intégration de schémas en particulier. Dans cet article, nous nous somme intéressés à cette tâche dans le cadre des schémas XML (XML schema matching). Nous avons commencé par une analyse des problèmes liés au matching, ce qui nous a permis de proposer une nouvelle approche qui tient compte de l'hétérogénéité des schémas sources d'un point de vue syntaxique, structurel et sémantique. Pour la réalisation du matching, nous avons

exploité les différentes caractéristiques et types de données employés dans les schémas EXS par une formalisation d'un graphe EXS représentant les différentes entités et contraintes. Ce graphe est utilisé lors du processus de matching. Dans la première tâche du processus, nous proposons une mesure de similarité terminologique entre les noms des nœuds, en se basant sur WordNet afin déterminer les relations sémantiques entre les mots. La seconde tâche est la mesure de similarité structurelle basée sur la similarité des contextes, la similarité des nœuds et la similarité des types (à partir de la hiérarchie des types prédéfinis). L'idée proposée ici est de définir pour chaque nœud du graphe EXS, un contexte basé sur les chemins reliant ces derniers. Nous combinons par la suite l'ensemble de ces mesures de similarités afin d'identifier des équivalences sémantiques entre les nœuds des graphes (source et cible). Une fois cette tâche réalisée, l'intervention de l'utilisateur est requise afin de valider ou de rejeter les résultats obtenus. A partir du résultat validé, nous appliquons une série de transformations, définies par des scripts XSLT.

Nous envisageons maintenant d'améliorer le processus de matching, en optimisant le processus de semi-automatisation pour la recherche de correspondances sémantiques par une prise en compte plus complète des métaconnaissances sémantiques dans l'implémentation de l'algorithme de matching. Ceci facilitera la génération d'opérateur basés sur des primitives de transformations entre les entités des schémas (source et cible). Puis, nous appliquerons ces opérateurs de transformation réalisant la tâche de *mapping* proprement dite. Le second axe à aborder concerne l'efficacité et le temps d'interaction des utilisateurs afin de produire des mappings finaux corrects. L'apport de l'intégration des métaconnaissances dans les phases de matching/mapping permet sans doute de minimiser le coût et le temps du matching tout en préservant son efficacité.

6. Références

Abiteboul S., Cluet S., Milo T., « Correspondance and Translation for heterogeneous data », *In Proceeding of The international Conference on Database Theory (ICDT)*, 1997, p. 351-363.

Andritsos P., Fagin R., Fuxman A., Haas L.M., Hernandez, M.A., Ho C.T., Kementsietsdis R., Miller R.J., Naumann F., Popa L., Velegrakis Y., Vilarem C., Yan L.L., « Schema Management », *IEEE Data Eng. Bull*, vol. 25, n° 3, 2002, p. 32-38.

Bernstein P., Halevy A., Pottinger R., « A vision for management of complex models», *ACM SIGMOD Record*, vol.29, n° 4, 2000, p. 55-63.

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*, vol. 31, 2004, p. 81-90.

Castano S., De Antonellis V., « A schema analysis and Reconciliation Tool Environment For Heterogeneous Databases », *In Proceedings of IDEAS'99 International Database Engineering and Applications Symposium*, 1999.

Do H.H., Rahm E., « COMA – A System for Flexible Combination of Schema Matching Approach », *VLDB Journal*, 2002, p. 610-621.

Doan A., Madhavan J., Domingos P., Halevey A., « Reconciling schemas of disparate data sources: A machine Learning Approach », *In Proceedings ACM SIGMOD conference*, 2001, p. 509-520.

Doan A., Madhavan J., Domingos P., Halevy A., « Learning to map between ontologies on the semantic web », *In Proceedings of the Eleventh International World Wide Web Conference, (WWW2002)*, 2002.

Drew P., King R., McLeod D., Rusinkiewicz M., Silberschatz A., « Report of the Workshop on Semantic Heterogeneity and Interoperation in Multidatabase Systems », *SIGMOD record*, vol. 22, n°3, 1993, p. 47-56.

Evens M., Relational Models of the lexion , Cambridge University Press, 1998.

Lamolle M., Mellouli N., « Intégration de bases de données hétérogènes via XML », *EGC'2003, Atelier fouille de données*, 2003.

Lamolle M., Zerdazi A., « Intégration de Bases de données hétérogènes par une modélisation conceptuelle XML », *Colloque sur l'Optimisation et les systèmes d'Information (COSI'05), 2005*, p. 216-227.

Madhavan J., Bernstein P., Rahm E., « Generic schema matching with cupid », *In Proceeding of International Conference on Very Large Databases (VLDB)*, 2001.

Melnik S., Garcia-Molina H., Rahm E., « Similarity Flooding : A versatile Graph Matching and its Application to Schema Matching », *In Proceedings of the 18th International Conference on Data Engineering*, 2002.

Miller A.G., « WordNet: A lexical Database for English », *ACM*, vol. 38, 1995, p. 39-41.

Rahm E., Bernstein P., « A survery of approaches to automatic schema matching », *In VLDB Journal*, 2001, p. 334-350.

Serafini L., Bouquet P., Magnini B., Zanobini S., « An Algorithm for Matching Contextualized Schemas via SAT », *In Proc. of CONTEXT 03*, June 2003.

W3C-XML: Extensible Markup Language (XML) 1.0, *W3C Recommendation, 1998*, Available at http://www.w3.org/TR/REC-XML

W3C-XSD: XML Schema Part 2: Datatypes, *W3C Recommendation, 2001*, Available at http://www.w3.org/TR/2001/REC-xmlschema-2-20010502.

Zerdazi A., « Modélisation des schémas XML par adjonction de métaconnaissances sémantiques », *ASTI'05*, 2005.

Zerdazi A., Lamolle M., « Hyperschema XML: un modèle d'intégration par enrichissement sémantique de schémas XML », *MajecSTIC'05*, 2005, p.143-150.