HyperSchéma XML: Un modèle d'intégration sémantique par enrichissement de schémas XML

Amar Zerdazi, Myriam Lamolle

HyperSchema XML : Un modèle d'intégration par enrichissement sémantique de schémas XML

Amar ZERDAZI, Myriam LAMOLLE
Laboratoire d'Informatique et de Communication (LINC)
IUT de Montreuil – Université Paris 8
140, rue de la Nouvelle France
93100 – Montreuil
{a.zerdazi, m.lamolle}@iut.univ-paris8.fr

Résumé : Récemment, l'apparition de nouvelles techniques de représentation des données sous forme semi-structurées dédiées au Web repose le problème de l'interopérabilité et donc de l'intégration de différentes sources de données. Notamment, le schema matching est l'un des problèmes majeurs rencontrés lors du processus d'intégration. Il consiste à identifier les correspondances entre les schémas de sources de données à intégrer (relationnel, XML, etc.). Aujourd'hui, cette tâche est généralement réalisée manuellement ou semi-automatiquemment ce qui explique son coût élevé et son manque de fiabilité. Nous nous positionnons ici dans le cas de la médiation de données et nous cherchons à semi-automatiser cette tâche complexe. Pour cela, nous exploitons les schémas XML extraits lors d'une phase de pré-intégration depuis ces sources hétérogènes. Ces schémas XML sont une représentation logique enrichie par des métaconnaissances sémantiques utilisées lors de la phase de matching. Nous améliorons l'intégration des différents schémas XML étendus (appelé EXS) afin de définir des règles de correspondance détectant les différences structurelles et sémantiques. A l'issue de cette étape, nous employons un ensemble d'opérateurs de transformation d'un *schéma EXS* à un autre, mémorisés par un script XSL.
Mots Clés : Bases de données hétérogènes, Correspondance sémantique, Hyperschema XML, Intégration, Schema matching.

1 INTRODUCTION

Récemment, l'apparition de nouvelles techniques de représentation de données sous forme semi-structurées dédiées au Web telles que XML, RDF, OWL, repose le problème de l'interopérabilité et donc de l'intégration de différentes sources de données. Notamment, l'appariement de schéma (schema matching) est l'un des problèmes majeurs rencontrés lors du processus d'intégration soit de données (par exemple, la médiation de données, les entrepôts de données, etc.), soit applications (par exemple, l'e-commerce, le Web sémantique, etc.). Plus récemment, plusieurs travaux se sont intéressés à cette problématique dans le contexte des données semi-structurées et plus particulièrement des données XML [Milo, 1998], [Berlin, 2002], [Melnik, 2002], [Tranier, 2004], [Madhavan, 2003], [McBrien, 2001]. Le schema matching – action d'associer un élément d'une structure de données à un élément sémantiquement équivalent d'une autre structure – a pour but de faire émerger les similitudes et les dissemblances des sources à intégrer pour définir les associations entre éléments de chacune des sources. Malheureusement, la diversité de représentation structurelle ou sémantique des données complique l'automatisation du processus d'intégration. Une démarche applicable pour l'intégration est de procéder en deux tâches principales [Hull, 1995]. La première concerne la traduction de données qui résout le problème de l'hétérogénéité physique/logique des sources et la deuxième tâche, l'intégration sémantique qui résout l'hétérogénéité sémantique. Dans cet article, nous nous intéressons à la deuxième tâche et plus précisément à la recherche des correspondances entre les différents types d'éléments définissant les structures logiques XML (arbre) enrichies sémantiquement par adjonction de métaconnaissances. Lors d'une phase de pré-intégration, les correspondances entre les éléments de différentes structures doivent être soigneusement spécifiées par un expert humain [Boukottaya, 2004]. Par la suite nous appliquons le processus de matching sur des schémas XML étendus. L'avantage de ces schémas est l'uniformité de représentation XML et la manipulation d'un nombre minimal d'entités à savoir des concepts, des relations et des propriétés les caractérisant.

Cet article est organisé comme suit. Un état de l'art recensant les systèmes d'intégration liés au schéma matching est présenté dans la section 2. Nous décrivons dans la section 3 notre modèle d'intégration basé sur l'hyperschéma XML ; nous détaillons par la suite la phase d'enrichissement sémantique affinée

par l'adjonction de métaconnaissances. La section 4 aborde la phase de mise en correspondance entre les schémas XML étendus où nous présentons les différents types de conflits. Dans la section 5, nous définissons les opérateurs de base utilisés lors des appels aux primitives de transformations.

2 ETAT DE L'ART

L'automatisation de l'intégration de données et du schema matching a été étudiée depuis longtemps par la communauté base de données. De nombreux travaux dans ce contexte proposent des méthodes pour automatiser la tâche de matching tels que [Madhavan, 2001], [Doan, 2003].

Une taxonomie de travaux effectués dans cette voie est présentée par [Rahm, 2001]. 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.), autrement dit par approche structurelle.

Par exemple, le système ***SemInt*** [Li, 2000] se base sur la similarité entre noms d'éléments. Il repose sur une architecture neuronale afin de déterminer les éléments à mettre en correspondance. La comparaison/classification est automatique, et c'est le système qui affecte à tout élément une distance entre les seuils de chaque classe.

Le modèle ***Similarity Flooding*** [Melnik, 2002] applique 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. L'idée principale est de représenter cette structure de données par des graphes orientés afin de simplifier la recherche, via les nœuds de ces graphes.

Le système ***LSD (Learning Source Descriptions)*** [Doan, 2003] utilise des techniques d'apprentissage afin d'obtenir un ensemble de règles d'association entre les éléments du schéma source et les éléments du schéma cible.

Dans le cadre de la deuxième catégorie, ces systèmes utilisent le contenu, c'est-à-dire des approches basées sur les données elles-mêmes (ou instances).

C'est le cas de ***Xylème*** [Delobel, 2003] 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 [Reynaud, 2001].

Le modèle ***AGORA*** [Manolescu, 2000] qui s'appuie sur un médiateur ***(leSelect)*** fondé sur la définition de ***mapping*** permet la traduction de requêtes ***(en Quilt vers SQL)***.

Il existe d'autres travaux, tel que Cupid [Madhavan, 2003] qui préfère une voie de recherche hybride des deux précédentes. Il a pour ambition d'être assez générique à travers les modèles et les sources de données (base de données relationnelles, documents XML).

e-XMLMédia [EXML, 2004] permet aussi de construire un système d'intégration hybride. Ce système utilise XML comme modèle commun et trouve son origine dans le projet Miro-web [Fankhauser, 1998]. Il est constitué de trois modules à savoir ***e-XML Mediator*** comme outil de requêtes, ***e-XMLLizer*** qui sert de wrapper, ***e-XML Repositery*** pour stocker et interroger les documents XML dans une BD relationnelle. La structure des documents n'est pas toujours connue mais si c'est le cas, ***e-XML Repositery*** peut générer un schéma relationnel spécifique.

Cette liste n'est pas exhaustive mais recense les approches qui nous ont paru intéressantes dans le contexte de nos propres travaux.

Chaque système présenté ci-dessus, propose ses propres critères d'évaluation. [Do, 2002] donne un aperçu des différentes méthodes d'évaluation utilisées. Cependant ces méthodes ne sont efficaces que si les noms des éléments à comparer ont une très forte similarité. Aussi le problème de la recherche de correspondance sémantique reste entier.

Dans le cadre de notre approche, nous nous situons dans la première classe des systèmes d'intégration en manipulant des schémas XML.

3 INTEGRATION VIA L'HYPERSCHÉMA XML (XHS)

L'objectif de notre approche 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 d'entités à manipuler mais suffisant pour traduire les schémas [Lamolle, 2003]. Notre hypothèse s'appuie sur un modèle semi-structuré (XML) [W3C, 2000] qui par sa souplesse permet une intégration aisée de modèles variés. En effet, sa structure arborescente permet de représenter de façon

générique diverses sources de données (structurées : SGBD, semi-structurées : OEM, non-structurés : Web) [Bergamaschi, 1999]. Comme nous pouvons le constater sur la figure 1, notre approche est composée de deux parties distinctes à savoir le

le module d'extraction de schémas source [Lamolle, 2005] et le module d'intégration permettant de construire l'hyperschéma. L'hyperschéma XML constitue l'ensemble des schémas XML étendus *(EXSi)* et l'ensemble des schémas XSL [W3C, 2002] stipulant les règles de transformation entre les entités des EXS (voir figure ci-dessous).

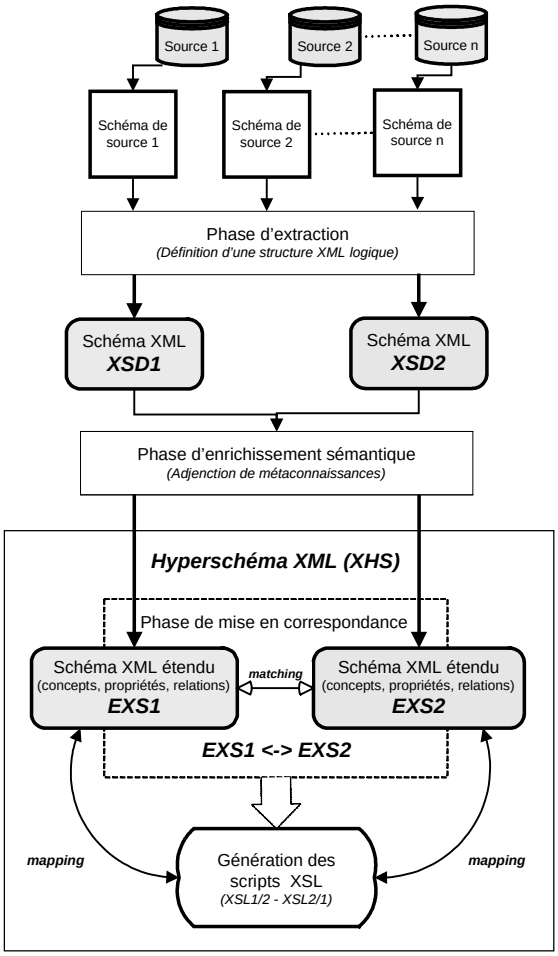


Figure 1 : Architecture d'intégration basée sur l'hyperschéma XML

Dans la figure 1, la phase d'extraction [Lamolle, 2005] homogénéise la représentation des différents schémas sources à intégrer en l'exprimant sous forme d'une définition de schémas XML (i.e XSDi) [W3C, 2001]. Une fois cette transformation faite, la partie sémantique des schémas XML extraits est affinée par l'adjonction de métaconnaissances soit déduites du catalogue des données, soit précisées par l'expert du domaine (phase d'enrichissement sémantique de la figure 1). A l'issue de ces deux phases (extraction et adjonction de métaconnaissances), le schéma XML étendu est composé de trois types d'entité à savoir :

- *Concept (c):* est l'entité la plus importante dans le EXS. Elle peut engendrer des propriétés et/ou d'autres concepts imbriqués, et porte des relations.

- *Relation (r):* correspond aux associations structurelles et/ou sémantiques existantes entre deux ou plusieurs concepts.

- *Propriété (p)*: est l'entité la plus basique dans le schéma EXS. Elles peuvent être incluses dans des concepts ou relations.

Nous définisons donc le schéma XML étendu (EXS) par le triplet ⟨concepts, relations, propriétés⟩ où chacune de ces entités est définie comme suit :

c_i est le concept nommé *C* de type «CConcept» et qui possède le niveau *i* dans l'arborescence. Il possède également d'autres propriétés et métaconnaisances que nous détaillerons plus loin dans cet article.

$r(c_i,\ldots c_j)$ est la relation existante entre deux ou plusieurs concepts $c_i,\ldots c_j$ nommée «c_i..c_jRelationShip» ; cette dernière possède une catégorie prédéfinie, et inclut les concepts en question.

p_k^i est la $k^{éme}$ propriété nommée *p* du concept c_i ; elle possède un domaine de définition et des contraintes (valeur par défaut, obligatoire, etc.).

L'exemple de la figure 3, utilise les notations précédentes pour chacune de ces entités.

3.1 Adjonction de métaconnaissances aux schémas XML

La phase d'enrichissement sémantique (adjonction de métaconnaissances) suit l'étape d'extraction (traduction des schémas des sources hétérogènes dans une structure logique définie par des schémas XSD) *(voir figure 1)*. La dimension sémantique des entités des schémas XML (concepts, relations, propriétés) est enrichie alors 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.) que nous définissons ci-après. Ces métaconnaissances sont utilisées lors de la phase d'intégration afin d'obtenir des mises en correspondance plus précises. L'enrichissement sémantique, dans ce cas, consiste à étendre la structure des schémas XML.

3.1.1 Complétude d'un concept

Dans le cas des sources de données semi-structurées, la structure des données est irrégulière, contrairement

aux BDs traditionnelles. En d'autres termes, 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 un attribut de type logique dans la définition du concept, appelé « complete ». Cet attribut spécifie si le concept est complet ou non ; autrement dit, s'il est représenté avec toutes ses propriétés et relations dans l'ensemble des sources de données.

Exemple.

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

Le concept *personne* dans l'exemple ci-dessus est incomplet du fait qu'il pourra changer de structure (forme, hiérarchie, propriétés, etc.). Par exemple, si nous disposons d'un ensemble de documents XML, instances d'un même domaine, sans DTD ou XML-Schema les validant, le concept *personne* peut avoir une structure interne différente d'un document à l'autre. Dans ce cas, nous le signalons comme concept incomplet. Par contre, si le concept *personne* est issu d'une table relationnelle, nous pouvons le qualifier par *complete="true"*, du fait que son schéma est connu dés l'origine.

3.1.2 Niveau d'encapsulation d'un concept

Afin de garder la structure d'arbre, notamment dans le cas des BDs XML (comme un document XML peut posséder une structure arborescente parfois très profonde), nous mémorisons le niveau d'apparition du concept. En effet, ceci traduit la visibilité du concept par rapport aux autres. Le niveau d'encapsulation est sauvegardé par l'attribut « level » *(de type entier positif)* dans la définition du concept.

Exemple.

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

Le concept *personne* est de niveau 1 dans le schéma EXS, ce qui explique son importance (concept-clé) dans les schémas sources [Bird, 2000].

3.1.3 Catégorie des relations

Il existe plusieurs catégories de relations dans un schéma XML étendu. A l'heure actuelle, nous traitons les catégories suivantes :

Dépendance : nous entendons par dépendance la relation d'utilisation d'un concept par un autre. Autrement dit, la durée de vie du concept englobé dépend de la durée de vie du concept englobant. Dans le schéma EXS, il est également possible d'avoir des dépendances entre plusieurs concepts.

Généralisation/spécialisation : est une relation de classification entre un concept plus général et des concepts plus spécialisés ou dérivés de celui-ci.

Association : le schéma EXS utilise des relations de type association afin de définir un lien entre plusieurs concepts de même niveau (conceptuellement équivalents).

Nous représentons la catégorie d'une relation par un attribut au même niveau que la définition de cette dernière. Pour l'instant cet attribut ne peut prendre comme valeur que les noms des catégories présentées ci-dessus.

Exemple.

```
<xsd:complexType  name="personneadresseRelationShip"
                  category="dependance">
```

3.1.4 Orientation et rôle des relations

Chaque relation possède éventuellement une orientation et un rôle. Dans le schéma EXS, nous les représentons par des attributs de la manière suivante : Pour l'orientation, nous employons deux attributs « source » et « target » de type logique *(true/false)* qui exprime le(s) concept(s) source et le(s) concept(s) destination. Le rôle est défini par l'attribut « role » qui permet de distinguer les relations explicitées sémantiquement de celles qui ne sont que des associations.

Exemple.

```
<xsd:complexType name="personneadresseRelationShip"
                 category="dependance"
                 role="posseder">
   <xsd:sequence>
      <xsd:element name="personne"
                   type="personneConcept"
                    source="true"
                   target="false"  …/>
      <xsd:element name="adresse"
                   type="adresseConcept"
                   source="false"
                   target="true"  …/ >
      …
   </xsd:sequence>
</xsd:complexType>
```

Dans le cas où il n'y a pas de concepts ayant un rôle privilégié (cardinalités 0 à plusieurs ou 1 à plusieurs), les attributs « source » et « target » sont positionnés à *false*.

3.1.5 Contraintes sur les propriétés

De même, les propriétés des concepts qui portent des contraintes d'intégrité sont représentées par des attributs. Ces contraintes 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*, signifiant que la propriété (la

contrainte) doit toujours être présente. Pour les autres propriétés (qui ne portent pas de contraintes) la restriction peut prendre la valeur *optional* (ou *fixed*), selon la cardinalité minimale associée (respectivement si c'est une constante).

Exemple.

```
<xsd:attribute name="id" type="xsd:string"
                    use="required"/>
```

Dans ce cas, la présence de la propriété *id* appartenant au concept *personne* est obligatoire du fait qu'elle représente un attribut dont la valeur est toujours renseignée dans le schéma source.

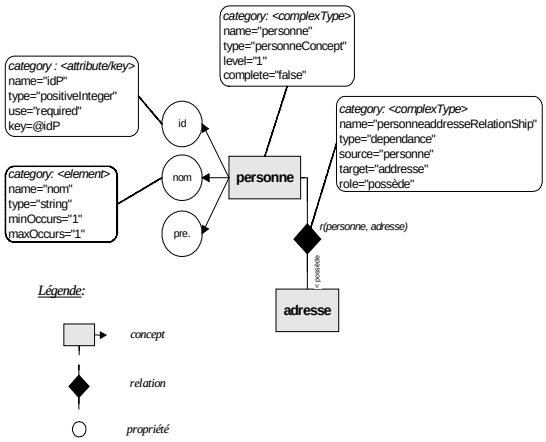


Figure 2. Représentation du concept *personne*

Par exemple, la représentation graphique du concept *personne* avec la relation *r(personne, adresse)* de la figure 2 est traduite après adjonction des métaconnaisances par :

```
<!--définition du concept personne -->
<xsd:complexType name="personneConcept"
                    level="1"  complete="false">
  <xsd:sequence>
     <xsd:element name="name" type="xsd:string"
                  minOccurs="1" maxOccurs="1"/>
     <xsd:element name="profession" type=" xsd:string"
                  minOccurs="1" maxOccurs="1"/>
     <xsd:element name="address"  type="addressConcept"/>
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:positiveInteger"
                 use="required"/>
</xsd:complexType>

<!--définition du concept adresse -->
<xsd:complexType name="adresseConcept"
                    level="2" complete="true">
…
</xsd:complexType>

<!--définition de la relation r(personne, adresse) -->
<xsd:complexType name="personneadresseRelationShip"
                    type="dependance">
 …
    <xsd:element name="personne" type="personneConcept"
                 source="true" target="false" level="1"/>
 …
```

Figure 3. EXS représentant le concept *personne*

4 CONFLITS ENTRE SCHÉMAS XML ETENDUS

La recherche de correspondance entre les différents schémas EXS étendus prend en compte deux aspects à savoir l'aspect sémantique et l'aspect logique. Les comparaisons/correspondances sont établies entre deux entités appartenant chacune à un schéma EXS *(source et cible).* Nous essayons de déterminer les conflits de correspondance *(conflits d'intégration)* entre ces derniers. Après la recherche et la détermination de correspondance, nous appliquons par la suite l'ensemble des opérateurs de base réalisant les primitives de transformation. [Pitoura, 1995], présentent les différents types de conflits qui peuvent survenir lorsque nous intégrons des sources de données hétérogènes. Dans ce qui suit, nous présentons une classification détaillée permettant d'illustrer les différents conflits rencontrés lors du *matching* entre les schémas EXS.

4.1 Conflit sémantique

Le conflit sémantique réside lorsque la même entité est interprétée différemment dans les schémas EXS. Dans notre approche, il peut figurer sous les formes suivantes :

4.1.1 Conflit de nom

Le conflit de nom [Kim, 1991], *(noté C.N)* se produit lorsqu'une même entité (concept, relation, propriété) est représentée de deux manières différentes (synonymie) ou lorsque le même nom est utilisé pour des entités différentes (homonymie). Nous devons définir le degré de similarité sémantique entre une paire d'entités. Dans ce cas, nous nous basons sur WordNet [Fellbaum, 1998] afin de définir les liens sémantiques entre ces éléments.

4.1.2 Conflit de structure

La même entité dans le schéma EXS, soit un concept, une propriété ou une relation, peut prendre différentes formes de représentation. C'est-à-dire qu'il est décrit par différents mécanismes. Le conflit structurel *(noté C.S)* est souvent détecté entre deux concepts ou entre un concept et des propriétés.

4.1.3 Conflit de granularité

Nous comparons ici le niveau de granularité des concepts dans la hiérarchie des schémas EXS. En effet, chaque concept est représenté avec son niveau *(via la métaconnaissance «level»)*, ce qui nous permet de l'identifier facilement et de déterminer sa profondeur structurelle dans l'arborescence. Le conflit de granularité est noté *C.G*. La détection de ce type de conflit, contribue aussi à la détection de conflit structurel *(C.S)*.

4.1.4 Conflit d'agrégation

Dans ce type de conflit *(noté C.A)*, nous nous intéressons à la structure intra-concept, ce qui correspond aux propriétés, relations et surtout les concepts caractérisant le concept englobant. Ce conflit est souvent entre concepts équivalents et leurs différences résident dans leurs formes internes. Ils peuvent avoir des propriétés différentes ou d'autres relations engendrant d'autres concepts, par exemple.

4.2 Conflit logique

Le conflit logique dans les schémas EXS se présente sous deux formes, à savoir :

4.2.1 Conflit de type

Le conflit de type *(noté C.T)* entre les entités des schémas EXS se présente lorsque la même entité dans deux ou plusieurs schémas EXS est représentée par différents types.

4.2.2 Conflit de contrainte

Le conflit de contrainte *(noté C.C)* dans les schémas XML étendus peut prendre plusieurs formes : contraintes de cardinalités, valeurs nulles, valeurs par défaut/obligatoires/optionnelles, etc. Généralement, ce sont les propriétés les plus souvent concernées par ce genre de conflit.

La figure 4, illustre la phase de recherche des correspondances entre deux schémas EXS représentés partiellement et la détection des conflits (sémantiques, logiques). L'étape suivante sera la phase de transformation *(mapping)* entre ces schémas.

5 SEMANTIQUE DES PRIMITIVES DE TRANSFORMATION

L'identification des correspondances entre les différents schémas hétérogènes et la définition des conflits entre les concepts et leurs relations sémantiques doivent être validées par l'expert du domaine. Ceci nous permettra d'appliquer les primitives de transformations entre les entités des schémas (source et destination) [Boyd, 2004], [McBrein, 2002] et donc d'employer des opérateurs de transformation réalisant la tâche de *mapping* proprement dite. Ces opérateurs sont exploités par les primitives de transformation afin de définir les règles de passage XSL entre ces schémas. La méthodologie, telle que nous l'avons définie, repose sur la théorie des catégories et la notion de morphisme [Cousin, 1988]. Les primitives de transformations lors de cette phase correspondent aux morphismes entre deux entités du même domaine ; nous l'adaptons à notre approche et nous appliquons des morphismes afin de réaliser le mapping entre les différents éléments.

5.1 Opérations de transformations

Nous employons six opérateurs de base qui seront appliqués lors des appels aux primitives de transformations.

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

$\exists E_i \in EXS_{source} \ \exists E_j \in EXS_{target} \wedge EXS_{source} \cap EXS_{target} \neq \{E_k\} \mid add\langle E_i ; EXS_{source} \rightarrow EXS_{target}\rangle$

§ **Suppression** ***(delete)*** : permet de supprimer une ou plusieurs entités lors du processus de mises en correspondance *(matching)* entre le schéma source et cible *(EXS_{source} ® EXS_{cible})*. La suppression est l'opérateur inverse de l'ajout.

$\exists E_i \in EXS_{source} \ \exists E_j \in EXS_{target} \mid delete\langle E_i ; EXS_{source} \rightarrow EXS_{target}\rangle$

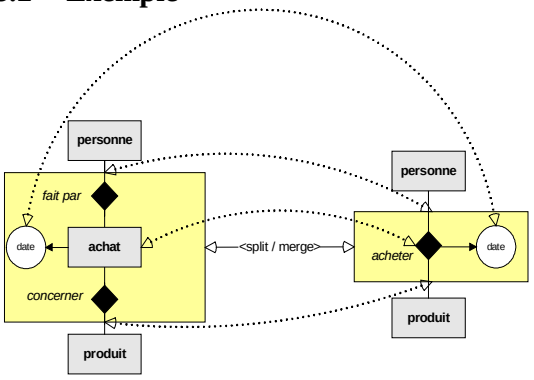
§ **Fusion** ***(merge)*** : permet de fusionner deux entités distinctes en une entité atomique. Généralement, nous appliquons la fusion afin de regrouper des propriétés appartenant au schéma source en un concept dans le schéma cible.

$\exists E_i \in EXS_{source} \ \exists E_j \in EXS_{target} \mid merge\langle E_i \rightarrow E_j ; EXS_{source} \rightarrow EXS_{target}\rangle$

§ **Eclatement** ***(split)*** : permet la décomposition d'une entité du schéma source en un ensemble d'entités équivalentes dans le schéma cible. *Split* est l'opérateur inverse de *merge*.

$\exists E_i \in EXS_{source} \ \exists E_j \in EXS_{target} \mid split\langle E_i \rightarrow E_j ; EXS_{source} \rightarrow EXS_{target}\rangle$

5.2 Exemple



2 relations & 1 concept « 1 relation

(a) mergeá{achat & fait par & concerner}®acheter ; EXS_1®EXS_2ñ

(b) splitáacheter®{achat & fait par & concerner} ; EXS_2®EXS_1ñ

Figure 4 : Exemple d'application d'opérations de transformations

Dans l'exemple précédent (figure 4), nous appliquons l'opération de transformation ***merge***, qui fusionne le concept ***Achat*** et les deux relations ***fait par*** et ***concerner*** (schéma EXS_1) en une seule relation ***(acheter)*** dans le schéma EXS_2 (voir cas (a)). Inversement, nous pouvons exécuter un ***split*** dans l'autre sens ***(EXS_2 à EXS_1)*** (voir cas (b)).

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

6 CONCLUSION

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 structurels et sémantiques, l'hétérogénéité sémantique demeure (deux sources de données peuvent être relatives à un même domaine sans utiliser les mêmes mots ou balises pour décrire des informations similaires ou complémentaires). Notre travail de recherche consiste donc à proposer une formalisation de l'intégration pour diminuer ces types de conflits. Pour cela nous proposons un enrichissement sémantique des schémas XML représentant les schémas sources pour optimiser la recherche des conflits d'intégration afin de préparer la phase des correspondances ***(matching)***. L'apport de notre approche est de donner une représentation simplifiée et assez puissante tout en gardant la sémantique attachée à ces schémas sources. La définition des schémas XML étendus se résume à trois types d'entités, concepts, relations et propriétés, manipulés de manière identique et à six opérations de transformation à appliquer lors de la comparaison, afin de gérer les mises en correspondance après détection de conflits.

Notre travail à venir consiste en l'amélioration du processus de semi-automatisation des mises en correspondance par l'adjonction d'autres métaconnaissances dans le EXS, une meilleure définition des catégories de relations et des primitives de transformations.

BIBLIOGRAPHIE

[Bergamaschi, 1999] Bergamaschi S., Castano S., Vinci M. Semantic integration of semistructured and structured data sources. SIGMOD Record, p.54-59, March (1999).

[Berlin, 2002] Berlin, J., Motro, A.: Database schema matching using machine learning with feature selection. CAiSE, p.452-466, Toronto, Canada (2002).

[Bird, 2000] Bird L., Goodchild A., Halpin T. Object Role Modelling and XML-Schema. Conceptual Modeling-ER 2000, 19th International Conference on Conceptual Modeling, p.309-322, October 9-12, Salt Lake City, Utah, USA (2000).

[Boukottaya, 2004] 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, p.81-90, 2004, Dunedin, New Zealand (2004).

[Boyd, 2004] Boyd, M., Kittivoravitkul, S., Lazanitis C., McBrien, P., Rizopoulos, N.: AutoMed: A BAV Data Integration System for Heterogeneous Data Sources. CaiSE, p.82-97, Riga, Latvia (2004).

[Cousin,1988] Cousin, R.: Apport de la théorie des categories à la representation des connaissances.Thèse de doctorat, Université Pierre et Marie Curie, Paris VI, (1988).

[Delobel, 2003] 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, p.267-298 (2003).

[Do, 2002] Do, H.H., Melnik, S., Rahm, E.: Comparison of Schema Matching Evaluations, Proceedings of the GI-Workshop 'Web and Database', Erfurt (2002).

[Doan, 2003] Doan, A., Domingos, P., Havely, A.: Learning to match the schema of date sources: A multistrategy approach, Machine Learning, p.279-301 (2003).

[EXML, 2004] Web site of e-XMLMedia. *http://www.e-xmlmedia.fr/*.

[Fankhauser, 1998] Fankhauser, P., Gardarin, G., Muntz, J., Tomasic, A.: Experiences in Federated Databases: From IRO-DB to MIRO-DB. Proceedings of the 24th International Conference on VLDB, p.655-658 (1998).

[Fellbaum, 1998] Fellbaum, C. WordNet An Electronic Lexical Database. The MIT Press, (1998).

[Hull, 1995] Hull, R., King, R.: Reference architecture for the intelligent integration of information, (1995).
http://www.isse.gmu.edu/I3_Arch/.

[Kim, 1991] Kim W., Seo J., Classifing schematic and data heterogeneity in multibase systems. IEEE Computer, 24(2), December 1991.

[Lamolle, 2003] Lamolle, M., Mellouli, N.: Intégration de bases de données hétérogènes via XML. EGC'2003, Atelier fouille de données, Lyon, France (2003).

[Lamolle, 2005] 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), Bejaia, Algérie (2005).

[Li, 2000] Li, W.S., Clifton, C.: Semint–a tool for identifying attributes correspondences in heterogeneous databases using neural vetworks, Data Knowledge Engineering, 33(1), (2000).
[Madhavan, 2001] Madhavan, J., Bernstein, P., Rahm, E.: Generic Schema Matching with Cupid. The VLDB Journal, p.49-58, Roma, Italy (2001).
[Madhavan, 2003] Madhavan J., Bernstein, P., Chen, K., Havely, A., Shenoy, P.: Corpus-based Schema Matching. Workshop on Information Integration on the Web, p.59-66 (2003).
[Manolescu, 2000] Manolescu, I., Florescu, D., Kossmann, D., Olteanu, D., Xhumari, F.: Agora–Living with XML and Relational. Proc. of the Intl Conf. on Very Large Databases (VLDB), Cairo (2000).
[McBrien, 2001] McBrien, P.: A Semantic Approach to Integrating XML and Structured Data Sources. In Proceedings of the 13th Intl. Conf. On Advanced Information Systems Engineering, p.330-345 (2001).
[McBrien, 2002] McBrien, P., Poulavassilis, A.: Schema Evolution in Heterogeneous Database Architectures, A Schema Transformation Approach. In Proceedings CAiSE'02, Toronto, Ontario (2002).
[Melnik, 2002] Melnik, S., Molina-Garcia, H., Rahm E.: Similarity Flooding–A Versatile Graph Matching Algorithm, Proceedings of International Conference on Data Engineering, (2002).
[Milo, 1998] Milo T., Zohar, S.: Using Schema Matching to Simplify Heterogeneous Data translation. Proceedings of International Conference on VLDB, New York City, NY, USA (1998).
[Pitoura, 1995] Pitoura, E., Buukhres, O., Elmagarmid, A.: Object Orientation in Multidatabases Systems. ACM Computing Surveys, p.141-195, june 1995.
[Rahm, 2001] Rahm E., Bernstein, P.: A suvery of approaches to automatic schema matching, VLDB journal, p.334-350, Roma, Italy (2001).
[Rahm, 2004] Rahm, E., Do, H.D., Maßmann, S.: Matching large XML schemas, ACM SIGMOD Record, v.33 n.4, Paris, France (2004).
[Reynaud, 2001] Reynaud, C., Sirot, J.P., Vodislav, D.: Semantic integration of XML heterogeneous data sources, Proceedings of the 2001 International DB Engeneering of applications Symposium, july (2001).
[Tranier, 2004] Tranier, J., Baraër, R., Bellahsène, Z., Teisseire, M.: Where's Charlie–Family-Based Heuristics for Peer-to-Peer Schema Integration. IDEAS, p.227-235 (2004).
[W3C, 2000] XML 1.0 (Second Edition), 6 October 2000. W3C Recommendation, *http://www.w3.org/TR/REC-xml-20001006*.
[W3C, 2001] XML Schema, W3C Recommendation: XML Schema Primer, W3 Consortium, available at *http://www.w3.org/TR /xmlschema-0*, (2001).
[W3C, 2002] XSLT 1.0. W3C Recommendation : XSL Transformations XSLT Version 1.0, Available at *http://www.w3.org/TR/xslt*, (2002).