Méthodologie de Matching à large Echelle de schémas XML

Sana Sellami, Aïcha-Nabila Benharkat, Youssef Amghar

LIRIS- Université de Lyon
Bâtiment Blaise Pascal
69621 Villeurbanne
{sana.sellami, nabila.benharkat, youssef.amghar)@insa-lyon.fr

RÉSUMÉ. *Le matching, est une opération importante pour les processus assurant l'interopérabilité de systèmes d'information notamment pour des applications telles que l'échange, l'intégration et la transformation de données. Toutefois, la problématique du matching se pose avec acuité lorsque le nombre et le volume des schémas de données est important. C'est notamment le cas pour les applications e-business avec une très forte orientation internet. Dans ce contexte, nous proposons, dans cet article, une méthodologie de matching à large échelle structurée en trois différentes phases (pré-matching, matching et post-matching), qui vise à optimiser le matching grâce à une phase préalable de prétraitement qui se base sur des techniques d'analyse, de traitement linguistique des éléments des schémas et d'extraction de données. Cette méthodologie est supportée par une plateforme nommée PLASMA (Platform for LArge Scale MAtching) qui a permis de conduire une série d'expérimentations.*

ABSTRACT. *Matching field is becoming a very attractive research topic. One of the challenges of the matching community is to search correspondences between several and voluminous schemas. However, matching these schemas at large scale represents a laborious process. In this paper, we propose a scalable schema matching methodology deployed in three phases (pre-matching, matching and post-matching) that aims at improving schema matching. Our methodology includes a pre-processing phase based on linguistic and mining techniques. Our methodology has been implemented in PLASMA platform (Platform for LArge Scale MAtching) and validated by our experiments.*

MOTS-CLÉS : *Matching, schémas XML, méthodologie, large échelle, décomposition*

KEYWORDS: *Matching, XML schemas, methodology, large scale, decomposition*

1. Introduction

De l'intégration des schémas de bases de données jusqu'à l'alignement d'ontologies, la problématique qui a suscité le plus de points ardus à résoudre cette dernière décennie est la recherche des correspondances (entre schémas de bases de données, schémas ou documents XML ou encore entre ontologies). Mais aujourd'hui, il existe des centaines voire même des milliers de schémas disponibles sur le Web qui nécessitent d'être rassemblés, organisés et mis en correspondance. Cette profusion des données affecte aussi bien les processus de gestion des données, d'évolution des données que de mapping et de matching des données.

Nous nous intéressons plus particulièrement au matching, qui est nécessaire aux processus assurant l'interopérabilité des applications pour lesquelles l'échange, l'intégration et la transformation de données sont des activités majeures. Le matching est par définition un processus qui vise à identifier et découvrir les correspondances sémantiques entre différents formats de données tels que les schémas, les ontologies, interfaces Web,… Plusieurs travaux de matching ont été proposés par (Rahm et *al.*, 2001) et (Shvaiko et *al.*, 2005) comme solution à la problématique de l'intégration des données. La plupart de ces travaux se sont focalisés sur le matching des schémas simples et de petite dimension entre 50-100 composants (éléments, attributs). Cependant, dès que l'on passe à un contexte plus dynamique et à large échelle, plusieurs problèmes se posent. En effet, l'accroissement de la quantité de données entraîne des problèmes d'efficacité des algorithmes en termes de temps d'exécution. De plus, les algorithmes de matching classiques ne résistent pas au passage à l'échelle lorsque les schémas à traiter sont volumineux. Les schémas e-Business par exemple, sont des schémas répartis sur plusieurs fichiers et leur taille (nombre d'éléments) varie d'une centaine à un millier d'éléments.

L'objectif de notre travail est de relever le challenge du matching pour de tels schémas à grande échelle. En particulier, nous proposons dans cet article une méthodologie de matching à large échelle, structurée en trois phases (pré-matching, matching et post-matching), qui vise à optimiser le matching en s'appuyant notamment sur une phase préalable de prétraitement. Ce prétraitement se base sur des techniques d'analyse, de traitement linguistique des éléments des schémas et d'extraction de données. Cette méthodologie est supportée par une plateforme nommée PLASMA (Platform for LArge Scale MAtching) que nous avons développée pour des besoins d'évaluation et d'expérimentation.

Le présent article est organisé comme suit : dans la section 2, nous présentons une étude sur les outils de matching et une synthèse sur les travaux existants. Nous décrivons dans la section 3, la méthodologie de matching à large échelle que nous proposons à travers ses différentes phases. La section 4 présente les expérimentations réalisées et les résultats obtenus. La dernière section conclut le travail effectué et présente les perspectives à venir.

2. Etat de l'art

Il existe dans la littérature plusieurs outils de matching à large échelle réalisant soit un matching « pair-wise » ou un matching « holistique ».

2.1 *Outils de matching pair-wise*

L'approche adoptée par la plupart des travaux sur le matching est l'approche « pair-wise » qui consiste à trouver les correspondances entre paires d'éléments de deux schémas/ontologies. Il existe de nombreuses plateformes intégrant des stratégies permettant d'optimiser les performances du matching à large échelle. Ces stratégies sont basées sur l'utilisation des techniques d'apprentissage non supervisé (e.g clustering) ou encore les logiques de description, les algorithmes statistiques, etc, et consistent pour la plupart à décomposer le problème du matching à large échelle en des petits sous problèmes. Parmi ces plateformes, on distingue :

- COMA++ proposé par (Do et *al*, 2007) est une plateforme de matching intégrant un ensemble de matchers et de stratégies de matching. COMA++ implémente l'approche de *fragmentation* qui consiste à décomposer deux schémas source et cible en des fragments de schémas. Les fragments sont soit des éléments partagés, soit des sous schémas ou même la totalité du schéma. La fragmentation se base donc sur les caractéristiques des schémas.

- Falcon-Ao (Hu et al., 2008) est un outil de matching d'ontologies qui suit une approche de partitionnement en blocs. Cette approche consiste à partitionner les entités de chaque ontologie en un ensemble de petits clusters et de construire des blocs en assignant des sentences RDF à ces clusters. Les blocs des ontologies seront donc mis en correspondance en se basant sur les anchors pré calculés. Le partitionnement est réalisé grâce à l'utilisation d'un algorithme de partitionnement agglomératif hiérarchique inspiré de l'algorithme de clustering ROCK.

- MOM (Modularization based approach) (Wang et *al.*, 2006) est un outil de matching propose la modularisation d'ontologies qui, comme l'approche de partitionnement consiste à décomposer les ontologies en des modules tout en utilisant des techniques telles que les $\mathcal{E}$-connections (Grau et *al.*, 2005). L'epsilon-Connection est une méthode définie comme la combinaison de formalismes logiques. Elle est appropriée pour la combinaison des bases de connaissances et pour le développement d'ontologies modulaires dans le Web.

- Malasco (Matching large scale ontologies) est un outil de matching d'ontologies à large échelle, proposé par (Paulheim, 2008), qui réalise le partitionnement des ontologies et effectue le matching sur des partitions de taille réduite. Le partitionnement est réalisé grâce à l'algorithme « baseline » qui itère sur des structures RDF.

2.2 *Outils de matching holistique*

L'approche holistique permet de réaliser le matching de schémas multiples (des interfaces Web représentant des requêtes sur des bases de données du Web profond) pour trouver les attributs correspondants parmi tous les schémas en une seule fois. Il existe différents outils de matching holistique :

- MGS (Modeling Generation Selection) et DCM (Dual Correlation Mining) sont des implémentations d'approches holistiques pour la découverte d'attributs synonymes et des mappings complexes. MGS (He et *al.*, 2003) est une approche statistique basée sur des observations des attributs les plus fréquents. DCM, proposé par les mêmes auteurs (He et *al.*, 2004), pour remédier au problème de découverte des mappings complexes. Le matching holistique est basé sur la détermination de la cooccurrence des attributs entre les schémas. Cependant, ces approches ne donnent de bons résultats que si une évidence peut être observée. Donc, seuls les attributs qui apparaissent fréquemment peuvent être mis en correspondance. Dans l'une comme dans l'autre approche la présence d'attributs est une entrave à leur bon fonctionnement.

- PSM (Parallel Schema Matching) et HSM (Holistic Schema Matching) (Su et *al.*, 2006) sont basées sur la comparaison d'interfaces Web en les mettant en parallèle et en supprimant les attributs communs qui apparaissent fréquemment afin déterminer les correspondances entre les attributs regroupés (attributs ayant le même concept).

- Wise-Integrator (He et *al.*, 2005) est un outil d'intégration de schémas d'interfaces Web. Il détermine les correspondances entre ces différentes interfaces. Il utilise des techniques de clustering afin d'améliorer la précision du matching.

2.3 Discussion

Suite à notre étude d'état de l'art, nous avons remarqué que les outils de matching pair-wise et holistique partagent le même objectif qui est la détermination du matching mais procèdent de façon différente. Les outils de matching pair-wise prennent en entrée uniquement deux schémas/ontologies source et cible. Ces données sont volumineuses et variées. Les outils de matching holistique, quand à eux, prennent un ensemble de schémas (interfaces Web de petite taille) en entrée et réalisent le matching en une seule fois. Alors que le matching à large échelle consiste à déterminer des correspondances entre un grand nombre de données volumineuses en tenant compte des critères d'efficacité et d'efficience. Par ailleurs, la plupart des outils de matching ne traitent pas les contraintes liées aux schémas lors du parsing et plus précisément les éléments référencés. En effet, les schémas volumineux regroupent un nombre important de ces éléments (référence de type de données et référence de noms). Seuls, les travaux proposés par (Do et *al.*, 2007), (Lu et *al.*, 2005), (Bernstein et *al.*, 2004) et (Madhavan et *al.*, 2001) ont abordé cette question et actuellement le seul outil de matching disponible et traitant les schémas volumineux, à notre connaissance, est COMA++. Cependant,

l'inconvénient majeur reconnu de cet outil est le temps de chargement et d'analyse des schémas qui peut être de l'ordre de plusieurs minutes (voir section « expérimentations »). Enfin, pour optimiser le processus de matching, l'approche la plus utilisée est la décomposition des schémas/ontologies en petits fragments.

Pour contribuer à résoudre ces problèmes, nous proposons une méthodologie de matching à large échelle des schémas XML issus du domaine du E-business. Cette méthodologie combine les deux approches holistique et pair-wise et se base sur une phase préalable de prétraitement permettent l'analyse des schémas en prenant en considération toutes les contraintes d'un schéma XML et particulièrement les éléments référencés, le traitement linguistique et une approche de décomposition des schémas grâce à l'utilisation de techniques d'extraction de données. Cette méthodologie est supportée par une plateforme nommée PLASMA (Platform for LArge Scale MAtching) qui a permis son évaluation lors de nos expérimentations.

3. Méthodologie de matching à large échelle dans PLASMA

Nous présentons dans cette section la méthodologie de matching à large échelle tel qu'elle est mise en œuvre dans PLASMA (figure 1). Contrairement aux plateformes existantes, PLASMA permet le traitement d'un grand nombre de schémas XML volumineux. Elle intègre dans sa méthodologie une approche hybride basée sur la combinaison des modules holistique et pair-wise et se décline en trois phases présentées ci-dessous : Le pré-matching, le matching et le post-matching.

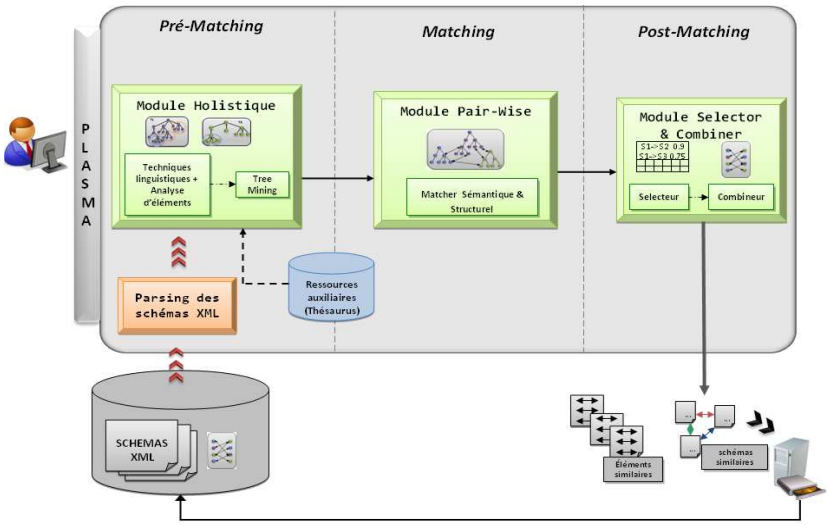


Figure 1. *Méthodologie de matching à large échelle dans la plateforme PLASMA*

3.1 Phase de pré-matching

Cette phase représente la phase plus importante de notre méthodologie. Elle a pour objectif de réaliser un prétraitement sur les schémas XML. Elle prend comme entrée un ensemble de schémas XML volumineux (>100 éléments) qui sont analysés grâce au parseur des schémas, traités linguistiquement grâce à des techniques terminologiques et d'un thésaurus et décomposés grâce à l'utilisation de techniques d'extraction de données.

3.1.1. *Modélisation des schémas XML*

Un schéma XML est modélisé en graphe connecté acyclique où chaque élément ou attribut est représenté par un nœud. Les noms, types et contraintes (occurrences, cardinalités, références, espace de noms, etc) des éléments/attributs représentent les labels des nœuds (voir définition 1).

Définition 1 (arbre XML). Un arbre étiqueté A=(r, N, E, φ) est un graphe connecté acyclique tel que r est le nœud racine, N représente l'ensemble des nœuds (éléments ou attributs), E est l'ensemble des arcs dans l'arbre et φ est une application d'étiquetage φ: $N \rightarrow L$ assignant une étiquette (nom, type, ou contraintes de l'élément) à chaque nœud de l'arbre avec L est l'ensemble des étiquettes des nœuds.

Définition 2 (taille et profondeur de l'arbre). A= (r, N, E, φ) est un graphe connecté acyclique. La *taille* de A notée |A| est le nombre de nœuds dans A. La *profondeur* du nœud N est le nombre d'*ancêtres* de ce noeud. Le nœud racine est de profondeur zéro.

Notations : Soit A un arbre. Soit $u \in N$
Tous les nœuds de A dont *u* est le parent sont définis: Child (*u*) = $\{u_c / (u, u_c) \in E\}$
Les ancêtres de *u* sont définis par: Ancêtres (*u*) = $\{u_c / (u_c, u) \in E^+\}$

3.1.2. *Décomposition de schémas XML*

La décomposition de schémas que nous proposons est inspirée de l'approche holistique car elle prend en entrée un ensemble de schémas et teste cet ensemble en une seule fois. Elle est basée sur les hypothèses et les observations suivantes : a) les schémas à large échelle sont nombreux (>2 schémas) et volumineux (>100 éléments), b) les schémas appartenant au même domaine contiennent des éléments ayant des concepts similaires, et c) les schémas comprennent plusieurs structures répétitives.

Cette décomposition consiste à diviser les schémas en sous schémas tout en identifiant ceux qui sont linguistiquement similaires, en assurant la non perte d'information. Elle utilise une stratégie semblable à celle de la fragmentation proposée par COMA++. Notons toutefois que COMA++ réalise la fragmentation de manière très peu élaborée. En effet tous les fragments utilisés ont déjà été identifiés lors de la phase d'analyse et de transformation des schémas XML. Donc contrairement à COMA++, notre approche permet de réaliser la décomposition au

niveau de plusieurs schémas et de déterminer en même temps les structures fréquentes intra-schémas (appelées partagées dans l'approche de fragmentation) mais également les sous structures communes aux schémas. Nous appliquons pour cela un algorithme de fouille de données afin d'identifier ces sous structures.

Cette approche requiert trois étapes (figure 2) : (1) transformation et optimisation des arbres XML (2) identification et extraction des sous arbres communs; et (3) préparation des sous schémas pour le matching.

Nous allons présenter dans cette partie ces différentes étapes.

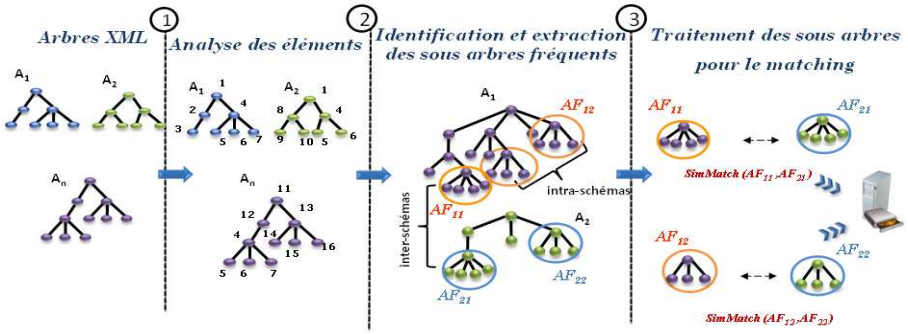


Figure 2. *Décomposition de schémas XML*

3.1.2.1. Prétraitement et traitement linguistique des éléments des schémas

Afin de faciliter l'extraction des sous structures communes, nous transformons les arbres XSD en arbres où les nœuds sont représentés par des entiers. Ce choix permet ainsi d'optimiser considérablement l'espace mémoire qui y aurait été alloué en stockant les noms d'éléments. Lors de ce processus, les éléments similaires linguistiquement seront affectés à la même valeur entière. Pour cela, un traitement linguistique sur les éléments des schémas est nécessaire afin de reconnaitre par exemple les ponctuations, caractères vides, chiffres (e.g PARTY_ID→ <Party, ID>) (tokenization) ou encore les majuscules et minuscules, les abréviations, les acronymes (e.g PO→ PurchaseOrder), les synonymes en utilisant un thésaurus comme WordNet. En effet, ce traitement nous permettra d'identifier les éléments similaires sémantiquement et de les prendre en considération lors de la phase d'extraction des structures communes.

3.1.2.2. Identification et extraction des sous structures communes

Cette étape a pour objectif de déterminer si les arbres issus de la phase de prétraitement partagent des sous structures communes. On distingue deux types de sous structures : les structures intra et inter-schémas. Nous appelons structures Intra- les sous structures fréquentes à l'intérieur d'un même schéma. L'identification de

telles structures joue un rôle important pour la décomposition. Les structures Inter-schémas quant à elles, sont des sous structures communes entre les schémas.

Nous proposons d'utiliser des techniques d'extraction d'arbres fréquents afin d'identifier ces structures communes. Plus précisément, nous avons utilisé l'algorithme proposé par (Termier et *al.*, 2004) pour réaliser cette extraction. Nous allons décrire dans ce qui suit comment ces techniques peuvent être intégrées et utilisées afin de détecter de manière efficace ces sous schémas.

Description du problème : L'extraction d'arbres vise à découvrir de manière automatique les sous arbres qui apparaissent *fréquemment* (la fréquence est calculée en utilisant la notion de support de fréquence noté Support(X,F)) (définition 4) dans un ensemble d'arbres. Le problème d'identification et d'extraction des sous structures communes peut être défini comme suit :

Extraction d'arbres fréquents:
Etant donné un ensemble d'arbres F (appelé aussi forêt) et un seuil minimum σ défini par l'utilisateur, le problème est de déterminer tous les sous arbres *inclus* (définition3) au moins σ fois dans la forêt F.

Définition 3 (Inclusion d'arbres): Soient $A_1 = (r_1, N_1, E_1, \varphi_1)$ un sous arbre étiqueté et $A_2 = (r_2, N_2, E_2, \varphi_2)$ un arbre étiqueté. On dit que A_1 est inclus dans A_2 noté $A_1 \subseteq A_2$ s'il existe un Mapping injectif $\mathcal{M}: N_1 \mapsto N_2$ qui vérifie les règles suivantes :

- ***R1 :*** $\mathcal{M}$ préserve les labels des nœuds tel que $\forall u \in N_1$, $\varphi_1(u) = \varphi_2(\mathcal{M}(u))$ ($\varphi : N \mapsto L$ est une application qui associe à chaque nœud de N un label de L).
- ***R2*** : $\mathcal{M}$ préserve la relation de parenté (a) et d'ancestralité (b) :
  (a) $\forall u, v \in N_1$ si $(u, v) \in E_1$ alors $(\mathcal{M}(u), \mathcal{M}(v)) \in E_2$
  (b) $\forall u, v \in N_1$ si $(u, v) \in E_1$ alors $(\mathcal{M}(u), \mathcal{M}(v)) \in E_2^+$

Définition 4 (support de fréquence). Soit $F = \{T_1, T_2, \ldots T_n\}$ un ensemble d'arbres (ou une forêt).

Le support de fréquence d'un arbre X note Support(X,F) est défini comme suit:

$$\text{Support}(X,F) = \sum_{i=1}^{n} \text{intra-support}(X, T_i)$$

tel que intra-support (X, T_i) est le nombre d'occurrences de X dans T_i

Le support de fréquence prend en considération les deux types de fréquences intra et inter-arbres.

Définition 5 (Arbre fréquent). On dit qu'un arbre X est fréquent dans une forêt F par rapport à un seuil minimum σ si et seulement si $\text{Support}(X,F) \geq \sigma$.

L'ensemble de tous les arbres fréquents dans une forêt est noté AF.

3.1.2.3. Préparation des sous schémas pour le matching

Nous allons décrire dans cette partie comment nous allons préparer les structures fréquentes pour le matching. Les sous structures (sous arbres) déterminées à l'étape précédente seront mises en correspondances lors de la phase de matching. Ces sous structures seront donc considérées comme des schémas à part entière. Cependant, il faudrait tester qu'il n y ait pas de redondances dans le matching de ces sous structures c'est-à-dire qu'on ne refasse pas le traitement sur les mêmes éléments existants dans les mêmes schémas. Nous devons également optimiser le matching en ne le réalisant que sur les sous structures les plus similaires.

Cette préparation de matching se déroule en 2 étapes :

a) Première étape: Sélection des sous structures pour le matching

L'objectif de cette opération est de proposer une stratégie pour la sélection des sous schémas pour le matching. En effet, certaines structures intra schémas, existant à un même niveau dans les schémas, peuvent partager des éléments et cela pourrait engendrer un calcul répétitif de similarités sur certains nœuds. Nous allons donc déterminer les sous arbres fréquents maximaux (définition 7) afin d'éviter un calcul répétitif sur les mêmes nœuds. Notre approche élague tous les sous arbres fréquents minimaux (voir définition 6).

Définition 6 (Sous arbre fréquent minimal). Un sous arbre fréquent est considéré comme sous arbre fréquent minimal (AF_{min}) si et seulement il n'existe aucun sous arbre fréquent inclus dans AF_{min} :

Un sous arbre est dit fréquent minimal (AF_{min}) $\Leftrightarrow \neg\exists$ A**F** $\subseteq$ AF_{min} / **AF** est un sous arbre fréquent fermé.

Définition 7 (Sous arbre fréquent maximal). Un sous arbre fréquent est dit maximal (AF_{max}) si et seulement s'il existe au moins un sous arbre fréquent inclus dans AFmax :

Un sous arbre est dit maximal fréquent (AF_{max}) ssi $\exists$ **AF** fréquent / **AF** $\subseteq AF_{max}$

b) Seconde étape: Détermination des sous structures les plus similaires

Nous allons déterminer l'intersection entre les sous structures fréquentes afin d'identifier celles qui sont les plus liées et de ne réaliser le matching que sur ces sous structures similaires. Ce traitement se fait en deux phases :

i**) L'identification des sous structures communes** possédant des éléments en commun appelés sous structures « liées » (Définition 8)

Nous allons utiliser pour cela un algorithme qui va déterminer les nœuds en commun (N_c) aux arbres fréquents ainsi que les sous arbres associés à ces nœuds (par intersection d'arbres).

Définition 8 (Sous arbre fréquent lié). Un sous arbre fréquent est dit lié (AF_l) s'il partage au moins un nœud en commun avec les autres sous arbres fréquents.

Un sous arbre est dit fréquent lié (AF_l) ssi $\exists$ AF fréquent/ $AF_l \cap AF \neq \emptyset$

ii) La détermination de la similarité entre ces sous structures (Définition 9)

Définition 9 (Similarité des sous arbres fréquents liés). Soient deux sous arbres fréquents liés AF_i et AF_j tel que $AF_i \cap AF_j = N_c$

La similarité entre AF_i et AF_j notée $Sim_{afl}(AF_i, AF_j) = |N_c| / Max(|AF_i|, |AF_j|)$

avec $Sim_{afl}(AF_i, AF_j)$ est comprise entre 0 et 1.

$|N_c|$: représente le nombre de nœuds des AF_i et AF_j en commun

$|AF_i|$ et $|AF_j|$: représentent respectivement les cardinalités de AF_i et AF_j

c) Algorithme de pré matching

Données en entrée :

- $AF = AF_s \cup AF_T$ tels que AF est l'ensemble de tous les sous arbres fréquents.
- AF_s et AF_T représentent respectivement l'ensemble de tous les sous arbres fréquents des arbres sources A_S et des arbres cibles A_T
- $A = A_s \cup A_T$ représente l'ensemble de tous les arbres sources et cibles

Sortie

- AF_{select} représente l'ensemble des sous arbres sélectionnés pour le matching

Déroulement du Prématching

$AF_l = \emptyset$, $AF_{min} = \emptyset$, $AF_{select} = \emptyset$, $AF_{max} = \emptyset$ // *voir Def.8 pour* AF_l

```
1. Sélectionner les sous arbres pour le matching
Pour chaque arbre A vérifier
   Pour  tout couple  (AF_i, AF_j) ∈  AF  Faire
            Si  AF_i ⊆ A  et AF_j ⊆ A  Alors
             AF_min ← Find_min (AF_i, AF_j)et  AF_max = AF – AF_min
                  // Find_min retourne l'ensemble de tous les sous
                  arbres fréquents minimaux AF_min
                   // Voir la définition 6 de AF_min et la définition
                  7 de AF_max
                  //  AF_max représente l'ensemble tous les sous
                  arbres fréquents maximaux
            Fin Si
   Fin Pour
 Fin Pour
2. Déterminer l'ensemble des sous arbres fréquents liés
 Pour chaque sous arbre AF_s∈ A_S et AF_T∈ A_T et
                        {AF_s,AF_T}∈ AF_max Faire
         Sim_afl  ← Calcul_Sim_afl (AF_s, AF_T)
          //voir définition 9
      Si Sim_qfl ≠0  Alors
          AF_l = AF_l  ∪  (AF_s,  AF_T,  Sim_qfl) et
          AF_select = AF_l
         Sinon   AF_select = AF_max
      Fin Si
 Fin Pour
```

3.2. *Phase de matching: Matching pair-wise*

L'objectif de cette phase est de réaliser le matching entre les différentes paires de schémas source et cible de l'ensemble AF_{select} mais aussi les éléments des schémas réduits suite à la décomposition.

3.2.1. *Matching des sous schémas fréquents sélectionnés*

Les paires des sous structures de AF_{select} sont soumises à l'algorithme de matching EXSMAL développé par notre équipe (Chukmol et *al.*, 2005). Cet algorithme réalisait dans sa version antérieure le matching entre des schémas XML sans éléments partagés. Dans notre application, on a étendu EXSMAL pour la prise en considération des éléments partagés. EXSMAL détermine la similarité de base, en prenant en considération la comparaison des types et de la description textuelle, puis de la similarité structurelle qui prend en considération le contexte des éléments (càd les ancêtres, les parents, fils et feuilles).

3.2.2. *Matching des sous schémas réduits*

Les schémas réduits peuvent contenir des éléments similaires linguistiquement. Pour cela, nous allons réaliser le matching avec EXSMAL entre les éléments appartenant aux schémas réduits (càd les éléments ne faisant pas partie des sous structures communes.

3.3. *Phase de Post -matching*

L'objectif de cette phase est d'agréger les similarités spécifiques aux matchers utilisés dans le module pair-wise et les similarités déterminées suite à l'utilisation de ressources auxiliaires (thésaurus), de sélectionner les meilleures correspondances entre les éléments et de combiner ces similarités avec le module ***Selector et Combiner*** afin de déterminer les matchings finaux entre les éléments et les schémas.

3.3.1. *Sélection des similarités*

Cette étape consiste à choisir parmi les correspondances trouvées dans l'étape précédente de matching celles qui sont très plausibles. Le ***Selector*** sélectionne les matchings de la matrice de similarité en fonction d'un seuil en dessous duquel on élimine les correspondances.

3.3.2. Combinaison des similarités

L'objectif du ***Combiner*** est de combiner les similarités, déterminées lors de la phase de sélection (ou filtrage), en une seule valeur de similarité afin de déterminer les correspondances entre les éléments des schémas et finalement entre les schémas. Nous allons tout d'abord déterminer la similarité des sous schémas (ou sous structures) pour déduire la similarité totale des schémas.

Similarité des sous schémas fréquents: La similarité *sim_freqschemas* [1] est déterminée en calculant la moyenne de toutes les valeurs de similarité de tous les

éléments des sous schémas fréquents AF_{Si} *et* AF_{Tj} de l'ensemble AF_{select}. Cette valeur de similarité est donnée par la formule suivante :

$$Sim\_freqschemas\ (AF_{Si}\ AF_{Tj}) = \sum_{i=1}^{|AFSi|} \sum_{j=1}^{|AFTj|} sim\ (e_{AFSi}, e_{AFTj}) / (|AF_{Si}| + |AF_{Tj}|) \quad [1]$$

Avec $sim\ (e_{AFSi}, e_{AFTj})$ [2] représente la similarité des éléments des sous schémas communs :

$$sim\ (e_{AFSi}, e_{AFTj}) = coeff\_ling * sim\_ling\ (e_{AFSi}, e_{AFTj}) + coeff\_struct * sim\_struct\ (e_{AFSi}, e_{AFTj}). \quad [2]$$

avec sim_ling (e_{AFSi}, e_{AFTj}) et sim_struct (e_{AFSi}, e_{AFTj}) représentant respectivement les similarité linguistique et structurelle des éléments.

Similarité des sous schémas réduits: La similarité *sim_reducedschemas* est déterminée de la même manière que *sim_freqschemas.*

Similarité totale des schémas: La similarité totale de deux schémas [3] est la somme de similarités des sous schémas *Sim_freqschemas* et *Sim_reducedschemas* par rapport au nombre total d'éléments ($|S_i| + |T_j|$):

$$Sim\_schemas\ (S_i, T_j) = \sum_{i=1}^{|Si|} \sum_{j=1}^{|Tj|} (Sim\_freqschemas\ (AF_{Si}, AF_{Tj}) + Sim\_reducedschemas\ (reducedS_i, reducedT_j)) / (|S_i| + |T_j|) \quad [3]$$

4. Expérimentations

Nous avons développé et implémenté notre plateforme PLASMA sous Eclipse en Java sous une machine Windows 2.80GHz Intel Pentium avec 2Go Ram. Nous avons réalisé nos expérimentations sur des schémas XML issus du monde réel (XCBL[1] et OAGIS[2]). La figure 3 résume les caractéristiques des schémas utilisés dans nos expérimentations.

Domaine	Nombre de schémas	Taille des schémas (nombre d'éléments)	profondeur
XCBL	40	22-4000	3-22
OAGIS	100	42-680	4-17

Figure 3. *Caractéristiques des schémas*

L'objectif de nos expérimentations est de montrer que notre méthodologie de matching proposée a permis d'optimiser le processus de matching à large échelle sur le plan de la performance notamment en temps de réponse. Nous avons pour cela

[1] http://www.xcbl.org/xcbl35/xsd/schemas.shtml

[2] http://www.openapplications.org/downloads/oagis/oagis90.htm

évalué le temps mis pour le parsing des schémas XML, le temps de décomposition et de matching des schémas.

4.1. *Evaluation du Parsing des schémas XML*

Nous avons évalué le temps de chargement et d'analyse des schémas XML par notre parseur implémenté dans PLASMA et nous l'avons comparé avec celui de COMA++ sur des schémas de taille moyenne : 245 éléments ($\approx$154ko), des schémas grands : 630 éléments ($\approx$330ko) et des schémas très grands : 3800 éléments ($\approx$950ko). Le parsing des schémas dépend du nombre d'éléments et de la taille des fichiers. La figure 4 montre les temps mis par chacun des parseurs pour l'analyse des schémas. Les résultats montrent donc que notre parseur est beaucoup plus performant que celui développé dans COMA++.

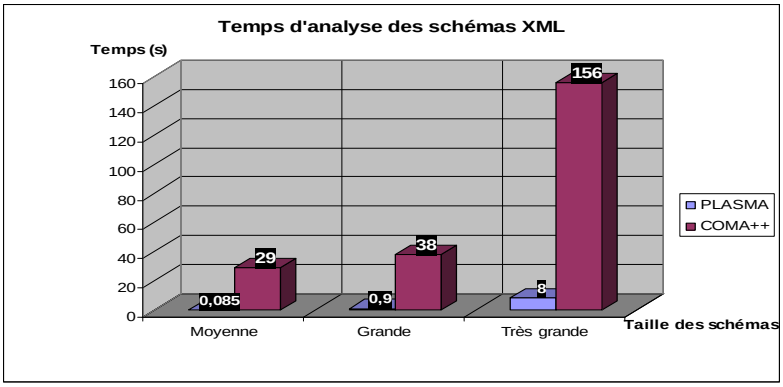


Figure 4. *Temps d'analyse des schémas XML par PLASMA et COMA++*

4.2. *Discussion sur la performance de l'approche de décomposition*

Nos résultats ont montré que la plupart des schémas partagent des structures inter et intra schémas. Nous avons aussi remarqué que la décomposition dépend fortement du domaine des schémas. Les schémas appartenant à un même sous domaine tel que XCBL possèdent un taux important de structures communes inter-schémas. Cependant, les schémas de différents sous domaines tels que XCBL et OAGIS nécessitent un traitement linguistique. Ces schémas partagent des éléments similaires décrivant le même concept mais écrits de manière différente.

Nous avons également évalué la variation du taux de décomposition (définition 10) selon la valeur du seuil σ saisi par l'utilisateur. Les résultats indiquent que plus la valeur du seuil est petite plus la décomposition est importante. Un exemple de cette variation est montré dans les figures 5 (a) et (b).

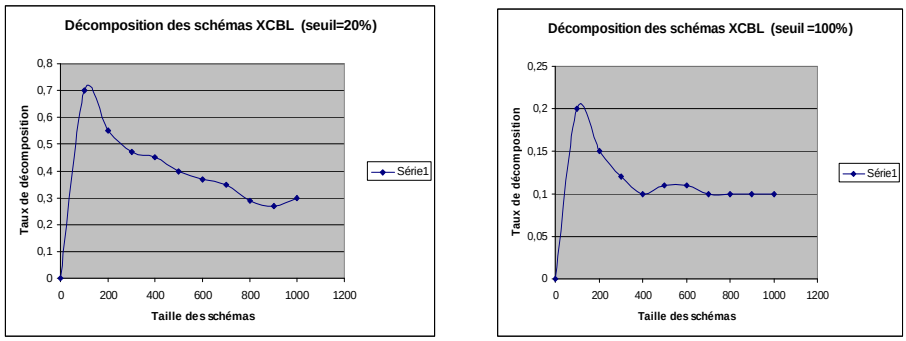
Définition 10 (taux de décomposition d'un schéma). Le taux de décomposition d'un schéma A prend en considération la taille des schémas (nombre d'éléments), la taille des structures fréquentes et le support.

$$D_T = \left(\sum_{i=1}^{n} |AF_i| * \textit{intra-support}\ (AF_i, A) \right) / |A_i|$$

Avec $|A_i|$: représente la taille du schéma d'origine.

$|AF_i|$: représente la taille des sous structures fréquentes.

n est le nombre de sous structures fréquentes.



(a) (b)

Figure 5. *Variation de la décomposition selon le seuil*

Nous avons évalué le temps d'exécution de notre matcher EXSMAL en utilisant la décomposition. Nous avons observé qu'en utilisant notre approche, le matcher est plus efficace. Par exemple, figure 6 illustre le temps d'exécution de deux schémas volumineux OAGIS (ayant 356 et 192 éléments) avant décomposition (AvD) et après décomposition (ApD). Nous remarquons que notre approche améliore les temps d'exécution par rapport à un matching direct des schémas avec un temps d'exécution de l'algorithme d'extraction de quelques secondes (0.007656 s).

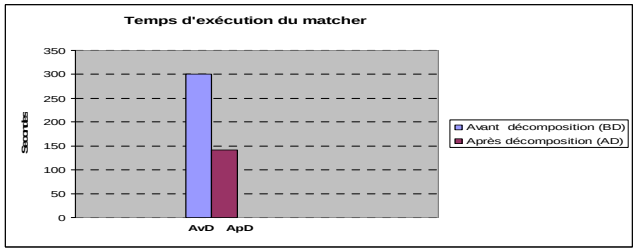


Figure 6. *Temps d'exécution du matcher avant et après la décomposition*

5. Conclusion et perspectives

Nous avons présenté dans cet article une méthodologie de matching à large échelle supportée par la plateforme PLASMA. Cette méthodologie, déployée en trois différentes phases (Pré-Matching, matching et post-matching), prend comme entrée plusieurs schémas XML volumineux, les analyse, détermine leurs similitudes linguistiques, les décompose et réalise le matching des sous schémas décomposés en vue de réduire la complexité du matching. Nous avons implémenté PLASMA avec tous ses composants : un parseur de schémas capable de traiter des schémas issus du monde réel et traitant des éléments référencés très souvent ignorés dans la littérature, des techniques de traitement linguistique, l'algorithme d'extraction , le matcher EXSMAL. Nos expérimentations ont montré que nous avons optimisé le temps du parsing des schémas et que grâce à l'approche de décomposition, nous avons amélioré les temps d'exécution de notre matcher. Comme perspective à notre travail, nous envisageons de développer des métriques de qualité afin d'évaluer les résultats de matching obtenus.

6. Bibliographie

Bernstein P.A., Melnik S., Petropoulos M., Quix C., « Industrial-Strength Schema Matching », *SIGMOD Record* , vol. 33, n° 4, 2004, p. 38-43.

Chukmol U., Rifaieh, R. and Benharkat, A., « EXSMAL: EDI/XML semi-automatic Schema Matching Algorithm », *7th International IEEE Conference on E-Commerce Technology (CEC 2005)*, 19-22 July 2005, München, Germany, p. 422-425.

Do H.H., Rahm, E., Matching large schemas: Approaches and evaluation, Journal of Information Systems, vol. 32, 2007, p. 857-885.

Grau B. C., Parsia, B., Sirin, E., Kalyanpur, A., « Automatic Partitioning of OWL Ontologies Using $\mathcal{E}$ -Connections », *Proceedings of the 2005 International Workshop on Description Logics* (DL'05), July 26-28, 2005, Edinburgh, Scotland, UK, vol. 147.

He B., Chen-Chan Chang, K.,. « Statistical Schema Matching across Web Query Interfaces ». *Proceedings of the ACM SIGMOD International Conference on Management of Data*, June 9-12, 2003 , San Diego, California, p. 217-228.

He B., Chen-Chan Chang, K., Han, J., « Discovering complex matchings across Web Query Interfaces: A Correlation Mining Approach », *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD*) , August 22-25, 2004, Seattle, Washington USA, p. 148-157.

He H., Meng, W., Yu, C., Wu, Z., « WISE-Integrator : A System for Extracting and Integrating Complex Web Search Interfaces of the Deep Web », *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB 2005)*, Trondheim, Norway, August 30 – September 2, 2005, p. 1314-1317.

Hu W., Qu Y., Cheng G., « Matching large ontologies: A divide-and-conquer approach» *Journal on Data and Knowledge Engineering*, vol.67, n°1, 2008, p.140-160.

Lu J., Wang S., Wang J. ,. « An experiment on the Matching and Reuse of XML Schemas », *Proceedings of the 5th International Conference on Web engineering (ICWE 2005)*, Sydney, Australia, July 27-29, 2005, Vol. 3579, p. 273-284.

Madhavan J., Bernstein P.A., Rahm E., «Generic Schema Matching with Cupid », *Proceedings of 27th International Conference on Very Large Data Bases*, September 11-14, 2001, Roma, Italy, p. 49-58.

Paulheim H., « On Applying Matching Tools to Large-Scale Ontologies », *Proceedings of the 3rd International Workshop on Ontology Matching (OM-2008) Collocated with the 7th International Semantic Web Conference (ISWC-2008)*, Karlsruhe, Germany, October 26, 2008, vol. 431.

Rahm E., Bernstein, P.A., « A survey of approaches to automatic schema matching », *International Journal on Very Large Data Bases*, vol. 10, n°4, 2001, p. 334-350.

Shvaiko P., Euzenat J., « A Survey of Schema-based Matching approaches ». *Journal on Data Semantics IV*, vol. 3730, 2005, p.146-171.

Stuckenschmidt H., Klein, M., « Structure-based Partitioning of large concept hierarchies », *Proceedings of the 3rd International Semantic Web Conference (ISWC 2004)*, Hiroshima, Japan, November 7-11, 2004, vol. 3298, p. 289-303.

Su W., Wang, J., Lochovsky, F., « Holistic Schema Matching for Web Query Interface », *Proceedings of the 10th International Conference on Extending Database Technology (EDBT 2006)*, Munich, Germany, March 26-31, 2006, vol. 3896, p. 77-94.

Termier A., Rousset M-A., Sebag M., « DRYADE: a new approach for discovering closed frequent trees in heterogeneous tree databases», Proceedings *of the 4th IEEE International Conference on Data Mining (ICDM 2004)*, 1-4 November 2004, Brighton, UK, p. 543-546.

Wang Z., Wang Y., Zhang S., Shen G., Du T., « Matching Large Scale Ontology Effectively», *Proceedings of the First Asian Semantic Web Conference (ASWC 2006)*, Beijing, China, September 3-7, 2006, p. 99-106.