# Mapping discovery for XML data integration

Zoubida Kedad, Xiaohui Xue

Laboratoire PRiSM, Université de Versailles
45 avenue des Etats-Unis, 78035 Versailles, France
{Zoubida.kedad, Xiaohui.Xue}@prism.uvsq.fr

**Abstract.** The interoperability of heterogeneous data sources is an important issue in many applications such as mediation systems or web-based systems. In these systems, each data source exports a schema and each application defines a target schema representing its needs. The way instances of the target schema are derived from the sources is described through mappings. Generating such mappings is a difficult task, especially when the schemas are semi structured. In this paper, we propose an approach for mapping generation in an XML context; the basic idea is to decompose the target schema into subtrees and to find mappings, called partial mappings, for each of them; the mappings for the whole target schema are then produced by combining the partial mappings and checking that the structure of the target schema is preserved. We also present a tool supporting our approach and some experimental results.

## 1 Introduction

A broad variety of data is available on the Web in distinct heterogeneous sources. The exchange and integration of these data sources is an important issue in many applications such as mediation systems or web-based systems.

In these systems, each data source has a schema (called source schema) that presents its data to the outside world. Applications needs are represented by target schemas. The way instances of the target schema are derived from instances of the source schemas is described through mappings. One example of systems using these mappings is mediation systems, where the target schema is called mediation schema and the mappings are called mediation queries. The user queries are expressed over the mediation schema and rewritten in terms of the source schemas using the mappings.

Defining mappings is a difficult task which requires a deep understanding not only of the semantics of the source schemas, but also the semantic links between the sources and the target schema. The complexity of this task increases when the number of data sources is high. The amount of required knowledge makes the manual definition of the mappings extremely difficult for a human designer. When the target schema and the source schemas are in XML, the definition of the mappings is more complex because of the hierarchical nature of the data.

In [7], we have proposed a general framework for mapping generation. In this paper, we present the algorithms for automatic mapping generation and a tool to support this task. We consider that the target and source schemas are described in XML

Schema, and we assume that a set of correspondences is provided. These correspondences relate elements of a source and elements of the target schema and express that these elements represent the same concept. Our tool produces a set of mappings, corresponding to different ways to derive instances of the target schema from instances of the sources. The generated mappings can be expressed in a standard language, such as XQuery or XSLT.

Due to the semi-structured nature of XML sources, it is extremely difficult to directly define mappings for the whole target schema. The basic idea of our approach is (i) firstly to decompose the target schema into a set of subtrees, called **target subtrees**; (ii) then to find the different ways, called **partial mappings**, to define each target subtree from the source schemas; (iii) and finally to combine the partial mappings to generate the mappings for the whole schema, called **target mappings**.

The paper is organized as follows. In Section 2, we give some basic assumptions and preliminary definitions. Section 3 presents the decomposition of the target schema. Section 4 and Section 5 detail the determination of the partial mappings and the generation of the target mappings respectively. Section 6 gives some experimental results obtained by our system. Some related works are presented in Section 7 and Section 8 concludes the paper.

## 2 Preliminaries

In this section we present the underlying assumptions of our approach: the representation of the target and the source schemas, and the correspondences between the schemas.

### 2.1 Representation of Target and Source Schemas

We consider source schemas and target schema expressed using XML Schema. Figure 1 shows two source schemas and a target schema representing information about books in a library. To avoid confusions, in the rest of the paper, each node will be suffixed by the name of its schema: $AuthorId_{s1}$ will refer to the node AuthorId in S1 while $ISBN_{s2}$ will refer to the node ISBN in S2. Every node in the tree may be either a text node (e.g. $AuthorId_{s1}$), that is, a node containing only text, or an internal node (e.g. $Chapter_{s1}$). The leaf nodes of the tree are always text nodes.

The cardinality of every node is characterized by the attributes minOccur and maxOccur, representing respectively the minimum and maximum number of instances for this node in the tree with respect to its parent. Each node is monovalued (maxOccurs = 1) or multivalued (maxOccurs > 1); it is also optional (minOccurs = 0) or mandatory (minOccurs > 0). In Figure 1, the symbol '+' represents a multivalued and mandatory node (e.g. $Book_{s2}$); the symbol '*' represents a multivalued and optional node (e.g. $Book_{ts}$); and the symbol '?' represents a monovalued and optional node (e.g. $Abstract_{ts}$). A node without symbol is monovalued and mandatory (e.g. $Id_{s1}$).

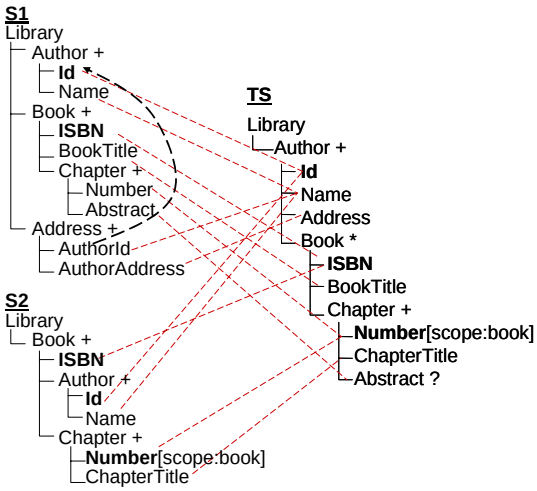



**Figure 1.** Schemas and correspondences

Keys are defined either in the whole schema or only in a subtree of the schema. In the first case, the key is absolute. In the second case, the key is relative and its scope is an antecessor of the identified node, except the root. In Figure 1, the nodes written in bold represent keys. If the name of the key node is followed by a bracket, then the key is a relative key and its scope is the node between brackets (e.g. $Number_{s2}$ is a relative key and its scope is $Book_{s2}$), otherwise it is an absolute key (e.g. $ISBN_{s1}$). A schema may also contain references; each one is a set of text nodes referencing another set of text nodes defined as a key. In our example, $AuthorId_{s1}$ references $Id_{s1}$, and this is represented by an arrow in Figure 1.

### 2.2 Semantic Correspondences

We suppose that a set of semantic correspondences is provided between each source schema and the target schema. The definition of these correspondences is an important issue and several approaches have been proposed to solve this problem [4][5][12].

In our work, we consider two kinds of correspondences: 1-1 and 1-n. A 1-1 correspondence relates a target node n with a source node n', and states that the two nodes represent the same concept. This correspondence is denoted n ≅ n' (e.g. $Id_{s1} \cong Id_{ts}$, $Number_{s2} \cong Number_{ts}$). In Figure 1, dotted lines represent correspondences. A transformation function may be applied to the source node. For example, a correspondence can be specified to relate a target node *PriceInEuro* to a source node *PriceInDollar*; if the exchange rate is 1€ = 0,797$, such correspondence is denoted *PriceInEuro* ≅ 0.797**PriceInDollar*.

A 1-n correspondence relates a target node n to a set of source nodes combined by the mean of a transformation function. For example, a target node *Name* represents the same concept as the concatenation of two source nodes *FirstName* and *LastName*. This correspondence is denoted *Name* ≅ concat(*FirstName, LastName*).

More generally, we consider the correspondences relating a target node n and a set of source nodes n1, …, nk combined using a function f. Such correspondences are denoted n ≅ f(n1, .., nk). For simplicity, in this paper we will restrict ourselves to 1-1 correspondences.

We use the same notation to represent correspondences between sets of nodes. There is a correspondence between two sets of nodes s1 and s2 if (i) both s1 and s2 contain the same number of nodes (ii) and for each node n1 in s1 there is exactly one node n2 in s2 such that n1 ≅ n2, and vice versa. The correspondence between the two sets s1 and s2 is denoted s1 ≅ s2 (e.g. {$ISBN_{s1}$, $BookTitle_{s1}$} ≅ {$ISBN_{ts}$, $BookTitle_{ts}$}).

Correspondences between two source schemas are derived though their correspondences with the target schema. Given two source nodes n and n' in S and S' respectively, the correspondence n ≅ n' holds if there is a node n" in the target schema such that n" ≅ n and n" ≅ n'. Some correspondences may also be provided between the source schemas; they will be used in our approach for mapping generation.

## 3 Decomposing the Target Schema

To handle the complexity of mapping definition, we decompose the target schema into a set of subtrees, called target subtrees; we will first find mappings for each target subtree then combine these mappings to generate the mappings for the whole schema, called target mappings.

Given a target schema, each target subtree t is a subtree of the target schema satisfying the following conditions:

- the root r of the subtree is either a multivalued node or the root of the target schema;
- all the other nodes in t are descendents of r and are monovalued;
- there is at least one text node in t (t may contain a single node).

This decomposition of the target schema gives several subtrees in which every node is monovalued. The mapping generation problem for the target schema is decomposed into two steps: finding mappings for every target subtree, then combining these mappings. Since a target subtree contains only monovalued nodes except the root, finding a mapping for this subtree consists in finding some equivalent nodes in the sources that satisfy the cardinalities constraints regardless their hierarchical organization. The hierarchical structure of the different target subtrees is checked during the second step.

Our target schema given in Figure 1 has three target subtrees shown on the right side of Figure 2: t1 is composed of the multivalued node $Author_{ts}$ and its three monovalued children $Id_{ts}$, $Name_{ts}$ and $Address_{ts}$; t2 is composed of $Book_{ts}$ and its two monovalued children $ISBN_{ts}$ and $BookTitle_{ts}$; and t3 is composed of $Chapter_{ts}$, $Number_{ts}$, $ChapterTitle_{ts}$ and $Abstract_{ts}$. The root $Library_{ts}$ doesn't belong to any target subtree.

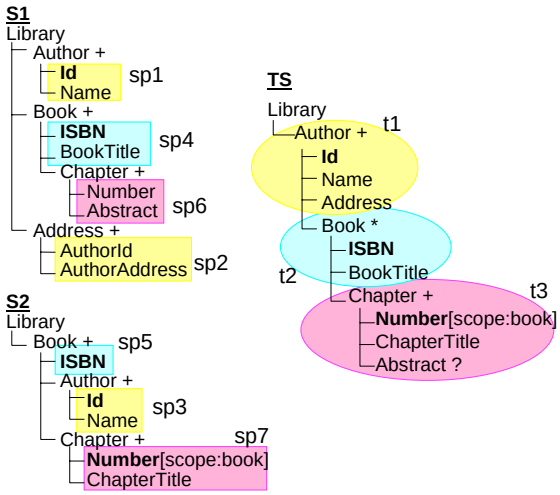



**Figure 2**. Target subtrees and source parts

Given two target subtrees t and t' such that the root of t' is a child of a node in t, we say that t is the parent of t' and t' is a child of t (e.g. in Figure 2, t2 is the child of t1 and the parent of t3).

A target subtree can be either mandatory or optional in a target schema. Consider a target schema with the root R and a subtree t of this schema with the root r. If t has a parent subtree t' with the root node r', we say that t is mandatory if all the nodes on the path from r to r' (except r') are mandatory. If t has no parent subtree, it is mandatory if all the nodes on the path from r to R are mandatory. In all the other cases, t is optional. In our example, t1 and t3 are mandatory and t2 is optional.

## 4 Determining Partial Mappings

Each partial mapping represents a way to derive instances of a target subtree from the instances of the source schemas. The partial mappings of a given target subtree are determined independently from the other subtrees in three steps: (i) identifying the parts of the sources (called **source parts**) that are relevant for the considered target subtree; (ii) searching the joins to combine these source parts; (iii) and determining the partial mappings from the source parts and the joins between them. Each target subtree may have several partial mappings with different semantics. In the rest of this section, we will describe these three steps.

### 4.1 Identifying Source Parts

A source part of a given target subtree is a set of text nodes in the source schemas that can contribute to derive instances for this target subtree.

Before defining source parts, we first present an extended definition of node cardinality. In XML Schema, the cardinality of a node is given with respect to the parent node: a node is multivalued or monovalued with respect to its parent. We generalize this definition to any pair of nodes.

**Def. 1. Extended definition of Cardinality.** Given two nodes n and n' in a schema and their first common antecessor m, n is monovalued with respect to n' if every node on the path from m to n (except m) is monovalued. Otherwise, n is multivalued with respect to n'.

According to the definition, $ISBN_{s1}$ is monovalued with respect to $BookTitle_{s1}$: their common antecessor is $Book_{s1}$ and the only node on the path from $Book_{s1}$ to $ISBN_{s1}$ (except $Book_{s1}$) is $ISBN_{s1}$, which is monovalued. Similarly, $BookTitle_{s1}$ is monovalued with respect to $ISBN_{s1}$. $Number_{s1}$ is multivalued with respect to $ISBN_{s1}$ because their common antecessor is $Book_{s1}$ and the path from $Book_{s1}$ to $Number_{s1}$ contains $Chapter_{s1}$ which is multivalued. On the contrary, $ISBN_{s1}$ is monovalued with respect to $Number_{s1}$.

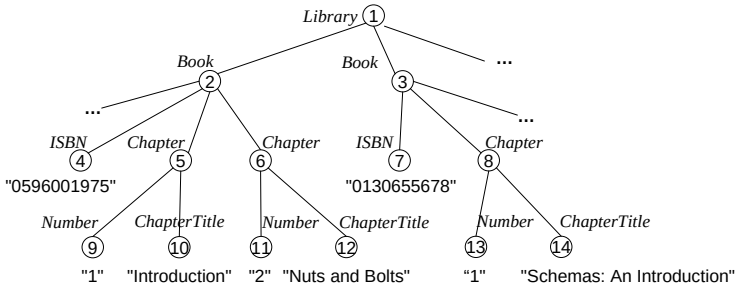



**Figure 3.** An example of instances for the source S2

Note that this extended definition of cardinality is different from the definition of functional dependency. Consider the nodes $ChapterTitle_{s2}$ and $Number_{s2}$ in S2. $ChapterTitle_{s2}$ is monovalued with respect to $Number_{s2}$. However, the functional dependency $Number_{s2} \rightarrow ChapterTitle_{s2}$ doesn't hold as we can see in Figure 3: two different instances of chapter number may have the same value, but associated with different titles; in fact, there are several titles for a given chapter number, one for each book.

Given a target subtree t, a source part sp for t in the source schema S is a set of text nodes that satisfies the following conditions:

- there is a set of text nodes c in t such that $c \cong sp$;
- there is at least one node n in sp such that the other nodes in sp are monovalued with respect to n;
- except c, there is no set of text nodes c' in S such that $sp \subseteq c'$ and c' satisfies the two above conditions.

Given a target subtree t, every source node involved in a correspondence with the nodes of t is found in at least one source part for t. If no source part is found for a target subtree, this means that there is no correspondent node in the sources for any of the nodes of this target subtree.

Consider the target subtree t1 having the text nodes $Id_{ts}$, $Name_{ts}$ and $Address_{ts}$. These nodes have the corresponding nodes $Id_{s1}$, $Name_{s1}$, $AuthorId_{s1}$ and $AuthorAddress_{s1}$ in S1. In the set {$Id_{s1}$, $Name_{s1}$}, both $Id_{s1}$ and $Name_{s1}$ are monovalued with respect to the other; this set is therefore a source part for t1. In {$AuthorId_{s1}$, $AuthorAddress_{s1}$}, both $AuthorId_{s1}$ and $AuthorAddress_{s1}$ are monovalued with respect to the other; this set is therefore a source part for t1. {$Id_{s1}$} is not a source part because it is a subset of {$Id_{s1}$, $Name_{s1}$}. {$AuthorId_{s1}$, $AuthorAddress_{s1}$, $Id_{s1}$} is not a source part also because $Id_{s1}$ is multivalued with respect to both $AuthorId_{s1}$ and $AuthorAddress_{s1}$ and both $AuthorId_{s1}$ and $AuthorAddress_{s1}$ are multivalued with respect to $Id_{s1}$.

The source parts for the target subtrees of our running example are shown on the left side of Figure 2. The subtree t1 has two source parts sp1 and sp2 in S1 and one source part sp3 in S2; t2 has two source parts sp4 and sp5 in S1 and S2 respectively; and t3 has two source parts sp6 and sp7.

### 4.2 Identifying Join Operations

The joins between source parts are identified using keys and key references. There are two distinct cases: the two source parts either belong to the same source schema or to different ones.

Given two source parts sp and sp' in the same source schema, a join is possible if there are two sets of text nodes c and c' in the schema such that:

- c is a key and c' references c;
- there is a node n in c such that every node in sp is monovalued with respect to n;
- there is a node n' in c' such that every node in sp' is monovalued with respect to n'.

In this case, a join is possible between sp and sp' with the join predicate c = c'; it is denoted j[c = c'](sp, sp'). For example, the join j[$Id_{s1}$ = $AuthorId_{s1}$](sp1, sp2) is possible between sp1 and sp2 since $AuthorId_{s1}$ is a reference on $Id_{s1}$.

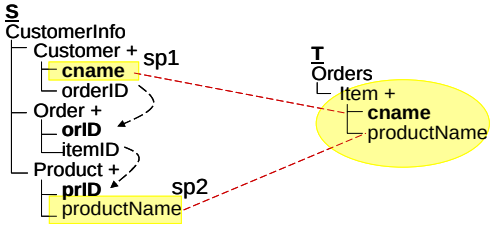



**Figure 4.** Relating two source parts through several references

This definition can be generalized by considering a sequence of references from c' to c instead of a single one. Consider the example shown in Figure 4. In the source S, two source parts sp1 and sp2 correspond to the single subtree of the target schema and no join is possible between them using the previous rule because no reference relates them directly. However, they are related through the two references: $orderID_s$ referencing $orID_s$ and $itemID_s$ referencing $prID_s$. A join is therefore possible and it is denoted j[$orderID_{s1} = orID_{s1}$, $itemID_{s1} = prID_{s1}$](sp1, sp2).

A join can also be possible between sources parts of different schemas. Consider two source parts sp and sp' in the source schemas S and S' respectively. Given a set of text nodes c in S and a set of text nodes c' in S', a join can be applied to sp and sp' with the predicate c = c' if the following conditions hold:

- c ≅ c';
- either c or c' is an absolute key in its schema;
- there is a node n in c such that every node in sp is monovalued with respect to n;
- there is a node n' in c' such that every node in sp' is monovalued with respect to n'.

In our example, the join j[$Id_{s1} = Id_{s2}$](sp1, sp3) is possible between sp1 and sp2 because both $Id_{s1}$ and $Id_{s2}$ are defined as absolute keys. The join between sp6 and sp7 with the predicate $Number_{s1} = Number_{s2}$ is not possible because neither $Number_{s1}$ nor $Number_{s2}$ is defined as an absolute key. However, we know that the combination {$Number_{s2}$, $ISBN_{s2}$} is unique in the whole schema because the scope of $Number_{s2}$ is $Book_{s2}$ which has the absolute key $ISBN_{s2}$. We have therefore to consider the combination {$Number_{s2}$, $ISBN_{s2}$} as an absolute key and use it instead of $Number_{s2}$. In fact, each time a relative key is found, it is combined with other key nodes to get an absolute key if possible. Figure 5 shows all the possible joins in our example.

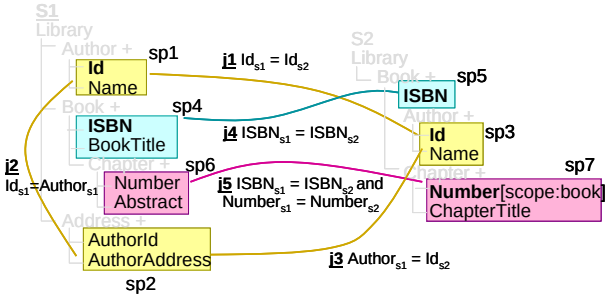



**Figure 5.** Join operations

In our approach, we consider that a join is possible in a limited number of cases; we do not therefore generate all the possible joins but only a subset of them. For example, a join involving two different sources is considered as possible only if the join predicate involves an absolute key. We could also have considered that a join is possible each time a correspondence is found between two sets of nodes, regardless the key definitions. But in our opinion, the semantics of this operation is not clear and

we therefore do not consider these joins. Consequently, only a subset of all the possible target mappings is generated in our approach.

### 4.3 Defining Partial Mappings from the Source Parts and the Joins

The partial mappings of a target subtree are determined using the corresponding source parts and the joins between them.

The source parts and the joins corresponding to a given target subtree are represented by a graph called join graph where every node is a source part and every edge between two source parts is a join between them; the edges are numbered and labeled with the join predicate.

Given the join graph G for a target subtree t, each partial mapping for t, denoted pm, is defined as a connected acyclic sub-graph of G such that for every mandatory text node n in t, there is at least a node n' in one of its source parts such that $n \cong n'$.

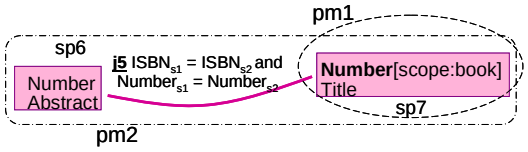



**Figure 6.** Join graph of t3

Considering the subtree t3 of Figure 2, the corresponding join graph is shown in Figure 6. It contains two source parts sp6, sp7 and the join j5. In this graph, there are two partial mappings: pm1 containing a single source part sp7 and pm2 containing sp6 and sp7 related by j5. Both are connected acyclic sub-graphs of the join graph and both produce instances for the mandatory text nodes $Number_{ts}$ and $ChapterTitle_{ts}$ in t3; pm1 does not produce instances for the optional node $Abstract_{ts}$; pm2 joins the two source parts; it may produce fewer chapters than pm1 but more information for every chapter (its abstract).

For simplicity, in the rest of the paper, we refer to a partial mapping by the source part name if it contains a single source part, or by the names of the corresponding joins if it contains more than one source part. In our example, pm1 and pm2 are denoted {sp7} and {j5} respectively.

The algorithm for partial mapping determination is given in Figure 7. It is a recursive algorithm that takes as input one target subtree (st) and the corresponding join graph G(SP, J) where SP represents the set of nodes (the source parts) and J the set of edges (the possible joins). The algorithm produces the set of all the partial mappings (PM) for st; each partial mapping in PM is represented by the corresponding sub-graph.

```
Partial_Mapping_Determination(G(SP, J), st, PM)
    Begin
        PM := ∅;
        for each source part sp in SP:
            J' := ∅;
            SP' := {sp};
            Build_Partial_Mapping (G'(SP', J'), G(SP, J), st, PM);
    return (PM);
    End

Build_Partial_Mapping (G'(SP', J'), G(SP, J), st, PM)
    Begin
        if mandatory_text_nodes(SP', st) //returns true if SP' contains all the mandatory text nodes in st
        then
            PM := PM ∪ {G'(SP', J')};
            //adds a new partial mapping represented by the graph G' to set PM
        for each join j between the source parts sp and sp' such that sp' ∈ SP' and sp ∉ SP'
            SP'' := SP' ∪ {sp};
            J'' := J' ∪ {j};
            if G''(SP'', J'') ∉ PM
            // adding the edge representing the join j to the subgraph G' does'nt give an element of PM
            then
            Build_Partial_Mapping (G''(SP'', J''), G(SP, J), st, PM);
    End
```

**Figure 7.** The algorithm of partial mapping determination

## 5 Generating Target Mappings

The mappings for the whole target schema, called target mappings, are defined using the partial mappings. To perform this task, **candidate mappings** are first generated by combining the partial mappings of the different target subtrees. Then the parent-child relations between the target subtrees are checked to produce target mappings.

A candidate mapping cm for the target schema TS is a set of partial mappings such that:

- there is at most one partial mapping for each target subtree in TS;
- for each mandatory target subtree t having no parent subtree, there is one partial mapping for t;
- for each mandatory subtree t having the parent subtree t', if there is a partial mapping for t' then there is also a partial mapping for t, and vice versa.

Consider the following partial mappings in our example: pm3 = {j2} and pm4 = {j1, j2} for t1; pm5 = {j4} for t2; and pm2 = {j5} for t3. Since t2 is optional and its child t3 is mandatory, each candidate mapping denoted cmi either contains no partial mapping for both t2 and t3 such as cm1 = {pm3} and cm2 = {pm4}, or contains a partial mapping for both t2 and t3 such as cm3 = {pm3, pm5, pm2}and cm4 = {pm4, pm5, pm2}.

The algorithm for candidate mapping generation is given in Figure 8. This algorithm takes as input the target schema TS and the sets of partial mappings PM1, ..., PMn corresponding respectively to the subtrees t1, ..., tn in TS. The algorithm performs a top-down browsing of the subtrees in TS and generates the set of candidate mappings CM.

```
Candidate_Mapping_Generation(TS, PM1, …, PMn, CM)
  Begin
    CM := ∅; // each element of CM is a set of partial mappings
    for each target subtree ti in get-mandatory-top-subtrees(TS)
      // get-mandatory-top-subtrees(TS) returns the subtrees in TS that are mandatory and has not parent subtrees
      if PMi == ∅
      then return (∅);
      //if a mandatory top subtree has no partial mapping, then the target schema has no target mapping
      if CM == ∅
      then
        for each partial mappings pm in PMi
          CM := CM ∪ {pm};
      else
        for each set S in CM
          for each partial mapping pm in PMi
            S' := S ∪ {pm};
            CM := CM ∪ {S'};
          CM := CM - {S};

    for each target subtree ti in TS not in get-mandatory-top-subtrees(TS) from top to down:
      if (top(ti)) // returns true if ti has not parent subtree
        for each set S in CM
          for each partial mapping pm in PMi
      else
        for each set S  in CM
          if contains_parent_mapping(ti, S)
          //returns true if the set S contains a partial mapping for the parent subtree of ti
          then
            for every pm in PMi
              S' := S ∪ {pm};
              CM := CM ∪ {S'};
          if (mandatory(ti)) then CM := CM - {S};
    return (CM);
  End
```

**Figure 8**. The algorithm of candidate mapping generation

Target mappings are derived from the candidate mappings that satisfy the parent-child relations between the target subtrees. Consider a target subtree t, its parent subtree t' and their respective partial mappings pm and pm'; pm and pm' preserve the parent-child relation between t and t' if the following conditions hold:

- there is a source part sp in pm and a source part sp' in pm' which are in the same source;
- there is either a node in sp with respect to which all the nodes in sp' are monovalued; or a node in sp' with respect to which all the nodes in sp are monovalued.

If there is a node in sp with respect to which all the nodes in sp' are monovalued, then for every instance of sp we can find the corresponding instance of sp', and for every instance of sp' we can find the corresponding instances of sp. The parent-child relation is therefore satisfied.

For the target schema of our example, there are two parent-child relations to check: one between t1 and t2 and the other between t2 and t3.

Consider the candidate mapping cm4 = {pm4, pm5, pm2}. The parent-child relation between t1 and t2 is satisfied in cm4 because every node in sp5 (involved in pm5) is monovalued with respect to both $Id_{s2}$ and $Name_{s2}$ in sp3 (involved in pm4). The parent-child relation between t2 and t3 is also satisfied because every node in sp5 is monovalued with respect to both $Number_{s2}$ and $ChapterTitle_{s2}$ in sp7 (in pm2). Therefore, cm4 is a target mapping for TS.

```
<Library>{
    for $au in distinct(S1/Library/Author/Id, S1/Library/Address/AuthorId, S2/Library/Book/Author/Id)
    for $sp1 in S1/Library/Author
    for $sp2 in S1/Library/Address
    for $sp3 in S2/Library/Book/Author
    where $sp1/Id=$sp2/AuthorId and $sp1/Id=$sp3/Id and $sp1/Id=$au
    return <Author>{
            <Id>{data($sp1/Id)}</Id>,
            <name>{data($sp1/Name)}</name>,
            <Address>{data($sp2/AuthorAddress)}</Address>
            for $b in distinct(S2/Library/Book/ISBN, S1/Library/Author/Book/ISBN)
            for $sp4 in S1/Library/Author/Book
            for $sp5 in S2/Library/Book[Author/Id = sp3/Id]
            where $sp4/ISBN = $sp5/ISBN and $sp4/ISBN = $b
            return <Book>{
                    <ISBN>{data($sp4/ISBN)}</ISBN>,
                    <BookTitle>{data($sp4/title)}</BookTitle>
                    for $c in distinct(S1/Library/Author/Book/Chapter/Number, S2/Library/Book/Chapter/Number)
                    for $sp6 in S1/Library/Author/Book/Chapter
                    for $sp7 in S2/Library/Book[ISBN=sp5/ISBN]/Chapter
                    where $sp6/Number = $sp7/Number and $sp7/Number = $c
                    return <Chapter>{
                            <Number>{data($sp6/ISBN)}</Number>,
                            <ChapterTitle>{data($sp7/ChapterTitle)}</ChapterTitle>,
                            <Abstract>{data($sp6/Abstract)}</Abstract>
                    }</Chapter>
            }</Book>
    }</Author>
}</Library>
```

**Figure 9.** An XQuery target mapping

The candidate mappings cm1 and cm2 are also target mappings because both contain a single partial mapping. The candidate mapping cm3 does not lead to a target mapping because the parent-child relation between t1 and t2 is not satisfied.

Other target mappings can be derived by applying set-based operations like Union, Intersection and Difference to two or more mappings. For example the union of cm1 and cm4 is a new target mapping that takes the union of pm4 and pm3 for t1, pm5 for t2 and pm2 for t3.

Each target mapping is an abstract query that can be translated into a specific query language such as XQuery or XSLT. To translate a target mapping into XQuery, each partial mapping is translated into a FWR (For-Where-Return) expression. For each target subtree t and its parent t', the FWR expression of t is nested in the FWR expression of t'. A grouping operation is added for every key in the target schema. For example, Figure 9 gives the translation to XQuery of the target mapping cm4.

## 6 Experimental Results

We implemented a system [8] in Java and we have run five scenarios to evaluate its performance. Table 1 summarizes the main characteristics of these scenarios, such as the number of nodes in the target schema, the number of data sources and the number of nodes for each one, the number of correspondences between the sources and the target schema, and the number of the key definitions in the sources.

**Table 1**. Characterizing the scenarios

| Scenarios | Target schema | | | Correspondences | Source schemas | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Depth | Nodes | Text nodes | | Schemas | Nodes | Text nodes | Keys | Refs |
| Mediagrid | 6 | 18 | 12 | 22 | 3 | 1674 | 825 | 412 | 413 |
| Library1 | 5 | 18 | 14 | 26 | 6 | 56 | 30 | 9 | 1 |
| Library2 | 5 | 18 | 14 | 30 | 6 | 62 | 35 | 10 | 1 |
| ABC1 | 7 | 47 | 36 | 1300 | 50 | 1650 | 1434 | 0 | 0 |
| ABC2 | 7 | 47 | 36 | 1300 | 50 | 1650 | 1434 | 58 | 0 |

The first scenario is from the Mediagrid project[1] which proposes a mediation framework for a transparent access to biological data sources; it considers three biological sources SGD, GOLD, SMD and a target schema built by domain experts. The Library1 scenario contains six source schemas. The Library2 scenario is similar to Library1 but the overlap between the sources is more important (more correspondences are defined for the same number of text nodes in the target schema). The ABC1 and ABC2 scenarios contain 50 source schemas. They are similar, except that the ABC2 scenario contains 58 key definitions while ABC1 contain no key definitions.

We have run these different scenarios on a PC-compatible machine, with a 2.8G Hz P4 CPU and 516MB RAM, running Windows XP and JRE1.4.1. Each experiment is repeated five times and the average of the five is used as the measurement.

**Table 2.** Measuring the scenarios

| Scenarios | Execution time (s) | | | |
|---|---|---|---|---|
| | Load | Target Schema Decomposition | Partial Mapping Determination | Target Mapping Generation |
| Mediagrid | 1.44 | 0.001 | 0.02 | 0.002 |
| Library1 | 0.44 | 0.001 | 0.067 | 0.095 |
| Library2 | 0.046 | 0.001 | 0.105 | 0.25 |
| ABC1 | 0.98 | 0.001 | 0.06 | 1.997 |
| ABC2 | 1.03 | 0.001 | 316 | 27 |

[1] Supported by the French Ministry of Research through the ACI Grid program, www-lsr.imag.fr/mediagrid/

The time needed for the main steps of our approach using the different scenarios are shown in Table 2. The loading time indicates the time to read the schemas and the correspondences into our internal representation. As expected, it is correlated to the size of the schemas and the number of their correspondences.

The target schema decomposition time indicates the time to decompose the target schema into target subtrees. We can see that the time needed to perform the task is negligible.

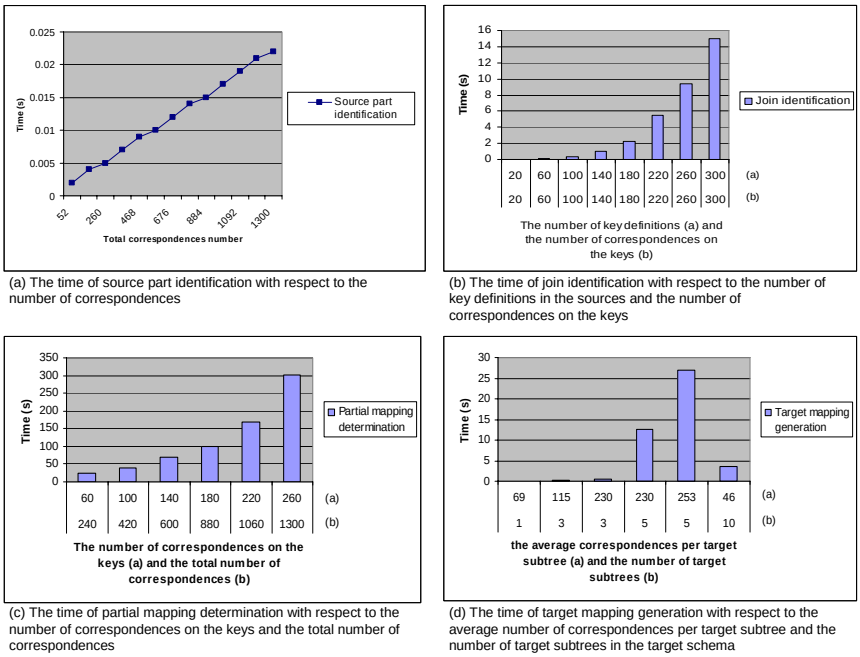
The partial mapping determination (pmd) time is proportional to the number of correspondences between target nodes and source nodes and the key and key references in the sources. The pmd time for Library1 which has 26 correspondences is smaller than the one of Library2 which has 30 correspondences; the two scenarios have the same number of sources and the same target schema. The pmd time for the ABC2 scenario which has 58 keys is largely greater than the one of the ABC1 scenario. This is because the number of keys of the ABC2 scenario makes the join graph very complex.

The target mapping generation (tmg) time indicates the time to find all the candidate mappings and to generate the target mappings. The tmg time is greater in ABC2 than in the other scenarios because in ABC2, most of the target subtrees have a lot of partial mappings (about 150), which leads to much more combinations to consider.

Some evaluations for the partial mapping determination and the target mapping generation are shown in Figure 10. The pmd time is decomposed into source part identification time, join identification time and partial mapping determination time. Figure 10 (a) shows the source part identification time with respect to the number of the semantic correspondences between the target schema and the source schemas. The measures are done using the ABC2 scenario and considering 52 to 1300 correspondences. This task is proportional to the number of correspondences and its time is almost negligible (about only 0.022 second for 1300 correspondences).

Figure 10 (b) shows the time of join identification with respect to both the number of key definitions and the number of correspondences on the keys. We have considered the ABC2 scenario and we have successively increased both the number of key definitions and the number of correspondences involving keys. The time needed to perform this task is influenced by the two parameters. With 300 key definitions and 300 correspondences on the keys (which represents a complex case), the time for the join identification is about 15 seconds.

The time required for the determination of partial mappings for a given target subtree depends on the size of the corresponding join graph. Figure 10 (c) shows the time for partial mapping determination with respect to both the total number of correspondences and the number of correspondences for the keys. We have successively increased the values of these two parameters from 60 correspondences for the keys and 240 total correspondences to 260 correspondences for the keys and 1300 total correspondences. The other parameters of the scenarios used in this experiment are the same as the ABC2 scenario. We can see in the graph that a scenario having 880 correspondences among which 180 correspondences involving source keys takes about 100 seconds for the partial mapping determination.




(a) The time of source part identification with respect to the number of correspondences

(b) The time of join identification with respect to the number of key definitions in the sources and the number of correspondences on the keys

(c) The time of partial mapping determination with respect to the number of correspondences on the keys and the total number of correspondences

(d) The time of target mapping generation with respect to the average number of correspondences per target subtree and the number of target subtrees in the target schema

**Figure 10.** Evaluating the time for the different steps of mapping generation

The target mapping generation depends on the number of partial mappings and the structure of the target schema, that is, the number of parent-child relations between the target subtrees. Figure 10 (d) shows the time required for this task with respect to the number of the target subtrees and the average correspondences per subtree. We have increased both parameters using the same sources as for the ABC2 scenario. For example, in the case of 5 target subtrees and 230 correspondences per subtree, it takes about 12 seconds for generating the target mappings; note that this case is a complex one, since the scenario contains 50 sources, 1300 correspondences and 58 key definitions. The complexity of this process is exponential with respect to the number of partial mappings. It is possible to reduce this complexity using some quality criteria (for example, selecting the partial mappings that use the sources having a high confidence factor) or some heuristics (for example, selecting the partial mappings using a high number of sources).

## 7 Related Works

Several approaches [1][6][10] have been proposed to generate mappings when the target and the source schemas are expressed using the relational model. The approach presented in [1][6] generates a set of mappings from a set of source schemas using

linguistic correspondences between target attributes and source attributes expressing that these elements represent the same concept. The work presented in this paper is inspired by this approach and also uses correspondences to define the mappings.

The approach presented in [10] generates a set of mappings from one source schema using a set of pre-defined value correspondences which specify how a target attribute is generated from one or more source attributes. In our work we also assume that correspondences are provided but we consider several source schemas.

Unlike the previously presented approaches ([1][6][10]), where the schemas are relational ones, we consider that either the target schema or the data sources are described in XML schema. In the case of XML sources, the complexity of mapping generation increases: we have to find instances for nodes of the tree representing the target schema, but also to preserve its structure.

An approach is proposed in [11] for generating mappings from one source schema to a target schema when these schemas are in XML Schema. In [14], a query rewriting algorithm which uses these mappings is proposed for integrating data sources. In our approach, the mappings are defined for a set of data sources; the mappings generated in our approach express the way instances of different schemas are combined to form instances of the target schema.

Other approaches have been proposed [2], [13], and [15] to generate mappings from several source schemas. These approaches comprise two steps: (i) the definition of rules to restructure each source schema according to the structure of the target schema; (ii) and the generation of mappings from these restructured schemas. In these approaches, source schemas must be restructurable with respect to the target schema in order to use them for mapping definition. In our approach, we do not impose such constraint, because some mapping may exist even is the restructuring of a source schema is not possible.

## 8 Conclusion

In this paper, we have presented algorithms for automatically generating mappings; we have implemented a system to support this task and presented some experimental results. This system produces a set of mappings for a target schema considering a set of source schemas and a set of correspondences; each target mapping has a different semantics.

Since the result of our system is a set of target mappings, one interesting perspective is to take advantage of these multiple semantics; the system should be able to select the mapping that most fits the needs of a specific user, using some preferences or some quality criteria. To achieve this goal, our system is already integrated in a platform that also provides tools for evaluating the quality of mappings considering user preferences [9]. Another perspective of our work is the maintenance of the mappings: if some changes occur in the data sources or in the target schema, some of the mappings may become inconsistent; the problem is therefore to detect the inconsistent mappings and to propagate the changes into the mapping definitions.

## References


1. Bouzeghoub, M., Farias Lóscio, B., Kedad, Z., Salgado, A.-C.: Managing the evolution of mappings. Proc. of the 11th. Int. Conf. on Cooperative Information Systems (CoopIS'03), Catania, Italy (2003) 22-37
2. Claypool, K. T., Rundensteiner, E. A.: Gangam: A Transformation Modeling Framework. Proc. of Eighth Int. Conf. on Database Systems for Advanced Applications (DASFAA'03), Kyoto, Japan (2003) 47-54
3. Collet C., Belhajjame K., Bernot G., Bruno G., Bobineau C., Finance B., Jouanot F., Kedad Z., Laurent D., Vargas-Solar G., Tahi F., Vu T.-T., Xue X.: Towards a target system framework for transparent access to largely distributed sources. Proc of the Int. Conf. on Semantics of a Networked World Semantics for Grid Databases (IC-SNW'04), Paris, France (2004) 65-78
4. Dhamankar, R., Lee, Y., Doan, A., Halevy, A. Y., Domingos, P.: iMAP: Discovering Complex Mappings between Database Schemas. Proc. of Int. Conf. ACM SIGMOD (SIGMOD'04), Paris, France (2004) 383-394
5. Do, H.H., Rahm, E.: COMA - A System for Flexible Combination of Schema Matching Approaches. Proc. of the 28th Int. Conf. on Very Large Data Bases (VLDB'02), Hong Kong, China (2002) 610-621
6. Kedad, Z.; Bouzeghoub, M.: Discovering View Expressions from a Multi-Source Information System. Proc. of the 4th. Int. Conf. on Cooperative Information Systems (CoopIS'99), Edinburgh, Scotland (1999) 57-68
7. Kedad, Z., Xue, X.: Mapping Generation for XML Data Sources: a General Framework. Proc. of the Int. Workshop on Challenges in Web Information Retrieval and Integration (WIRI'05), in conjunction with the 21st Int. Conf. on Data Engineering (ICDE'05), Tokyo, Japan (2005)
8. Kostadinov, D., Peralta, V., Soukane, A., Xue, X. (Demonstration) : Système adaptif à l'aide de la génération de requêtes de médiation. Proc. of 20th Conf. of Bases de données avancées (BDA'04) ,Montpellier, France (2004) 351-355
9. Kostadinov, D., Peralta, V., Soukane, A., Xue, X.: Intégration de données hétérogènes basée sur la qualité. Proc. of INFORSID 2005 (Inforsid'05), Grenoble, France (2005) 471-486
10. Miller, R.J., Haas, L. M., Hernández, M. A.: Schema Mapping as Query Discovery. Proc. of the 26th Int. Conf. on Very Large Data Bases (VLDB'00), Cairo, Egypt (2000) 77-88
11. Popa L., Velegrakis Y., Miller R.J., Hernandez M.A., Fagin R.: Translating web data. Proc. of the 28th Int. Conf. on Very Large Data Bases (VLDB'02), Hong Kong, China (2002) 598-609
12. E. Rahm, P. A. Bernstein, A survey of approaches to automatic schema matching. Proc. of the 27th Int. Conf. on Very Large Data Bases (VLDB'01), Roma, Italy (2001) 334-350
13. Yang, X., Lee, M. L., Ling, T. W.: Resolving structural conflicts in the integration of XML schemas: a semantic approach. Proc. of 22nd Int. Conf. on Conceptual Modeling (ER'03), Chicago (2003) 520-533
14. Yu, C., Popa, L.: Constraint-based XML query rewriting for data integration. Proc. of Int. Conf. ACM SIGMOD (SIGMOD'04), Paris, France(2004) 371-382
15. Zamboulis, L., Poulovassilis, A.: XML data integration by Graph Restructuring. Proc. of the 21st Annual British National Conf. on Databases (BNCOD21), Edinburgh (2004) 57-71