Mapping generation for XML data sources: a general framework

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 inter-operability of multiple autonomous and heterogeneous data sources is an important issue in many applications such as mediation systems, data-warehouses, or web-based systems. These systems provide a view, called a target schema, on the top of the data sources. Mappings are defined for describing the way instances of the target schema are derived from instances of the data sources. The generation of such mappings is a difficult problem, especially when the target schema and the source schemas are in XML format.

In this paper, we propose a framework to automatically find the mappings for a target schema given the source schemas and a set of semantic correspondences. In our framework, the target schema is decomposed into subtrees. Mappings are first determined for each subtree, and then combined to generate the mapping for the whole target schema. The generated mappings are expressed in a standard language, such as XQuery or XSLT.

1. Introduction

The inter-operability of several autonomous and heterogeneous data sources is an important issue in many applications such as mediation systems, data-warehouses, or web-based systems. The goal of these systems is to provide a uniform view on the top of the data sources.

In these systems, each data source has a schema (called source schema) that presents its data to the outside world. The applications using the system define a target schema that represents their needs. There are mainly two kinds of links established between each source schema and the target schema: semantic correspondences and mappings. The semantic correspondences between elements of the target schema and elements of the source schemas express that these elements represent the same concept. The definition of the correspondences between two schemas has been the focus of several works, such as [5][12]. Beside these correspondences, mappings are expressions describing the way instances of the target schema are derived from instances of the sources. The mappings are defined using the correspondences existing between the schemas.

The definition of the mappings to transform data from one representation to another one is known as data exchange, data translation or data migration [11]. When the mappings are defined between a target schema and one source schema, the definition of the mappings consists mainly in restructuring the data from one presentation to another. A mapping defined between a target schema and several source schemas is more complex: mapping definition must not only transform the source data from one structure to another but also combine the elements of different sources.

The definition of these complex mappings for multiple source schemas is a complicated process which requires a deep understanding of the data sources and their semantics. The complexity of this process increases when the number of data sources is high. In this case, the amount of knowledge required to manually write the mappings makes this task very difficult for a human designer. If the target schema and the source schemas are in XML format, the definition of the mappings becomes more complex because of the hierarchical nature of the data. In this paper, we propose a framework for the automatic generation of mappings between a target schema and a set of source schemas given a set of correspondences.

The paper is organized as follows. Section 2 presents the related works. In section 3, we describe some basic assumptions. Section 4 gives an overview of our framework. Sections 5 to 8 describe the components of the framework: the identification of the relevant schemas (section 5), the decomposition of the target schema (section 6), the determination of partial mappings (section 7), and the generation of the target mappings from the partial mappings (section 8). Some concluding remarks are given in section 9.

This work is supported by the French Ministry of Research through the ACI Grid program.

2. Related works

The mappings between a target schema and one or several source schemas are expressions describing the way instances of the target schema are derived from instances of the data sources. These mappings are used for rewriting user queries expressed on the target schema in terms of the source schemas.

Several approaches [2][8][9][10][14] have been proposed to generate mappings when the target and the source schemas are expressed using the relational model. The approach presented in [9][10][14] 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. The approach presented in [2][8] 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.

In the case of XML sources, the complexity of mapping generation increases: we must not only find instances for each node of the tree representing the target schema, but also preserve its tree structure.

An approach is proposed in [11] for generating mappings from one source schema to a target schema when these schemas are in XML format. In [16], a query rewriting algorithm which uses these mappings is proposed for integrating data sources.

Other approaches have been proposed [3][15][17] 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 derive mappings from them.

In this paper, we are interested in generating mappings for a target schema from a set of source schemas and given a set of correspondences. The target schema and the source schemas are in XML format. The generated mappings are defined in a standard language, such as XQuery or XSLT and they satisfy the constraints defined in the target schema.

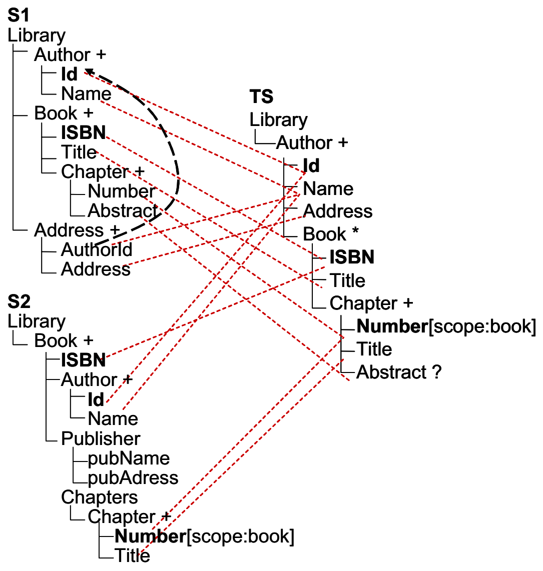

Figure 1. Target and source schemas

3. Preliminaries

In this section, we present some basic assumptions and definitions used in our framework. We consider a simplified version of XML Schema. Each schema is represented by a tree. Figure 1 shows two source schemas (S1 and S2) and a target schema (TS) representing information about books in a library. To avoid confusions, in the rest of the paper, each node in a source will be suffixed with the name of its schema: the node $AuthorId_{s1}$ will refer to the node *AuthorId* in S1 while the node $Author_{s2}$ will refer to the node *Author* 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. $Author_{s1}$), that is, a node used only to contain other nodes. The leaf nodes of the tree are always text nodes.

The cardinality of every node is characterized by the two values minOccur and maxOccur, representing respectively the minimum and maximum number of instances for this node in the tree with respect to its parent. A node may be monovalued (maxOccurs=1), or multivalued (maxOccurs>1). It may also be optional (minOccurs=0) or mandatory (minOccurs>0). In the example of figure 1, the symbol '+' for a node means that this node is multivalued and mandatory (e.g. $Book_{s2}$); the symbol '*' means that the node is

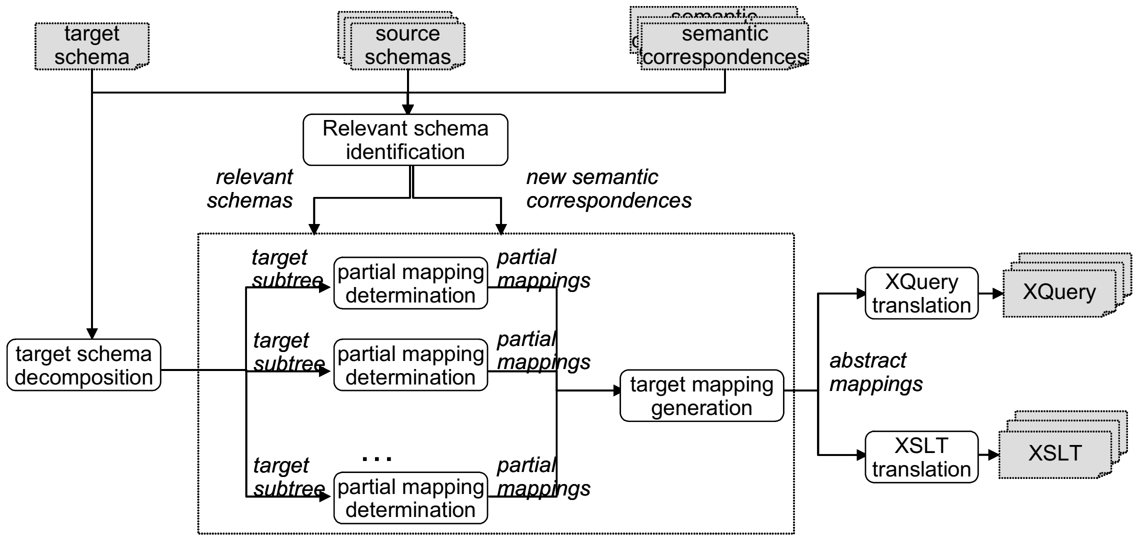

Figure 2. The general framework for mapping generation

multivalued and optional (e.g. *Book*$_{ts}$); the symbol '?' means that the node is monovalued and optional (e.g. *Abstract*$_{ts}$). A node without a symbol is monovalued and mandatory (e.g. *Id*$_{s1}$).

Every multivalued node n may have a key, which is either defined in the whole schema or only in a subtree of the schema. In the first case, its scope is the root of the schema and the key is said to be absolute. In the second case, its scope is an ascendant of n, different from the root, and the key is said to be relative. In the example of figure 1, the nodes written in bold represent keys. If the name of the key node is followed by a bracket, this means that the key is a relative key, and its scope is the node between brackets, otherwise it is an absolute key. For example, *Number*$_{s2}$ is a relative key for *Chapter*$_{s2}$ and its scope is *Book*$_{s2}$, while *ISBN*$_{s1}$ is an absolute key. A schema may also contain references, represented by a set of text nodes referencing another set of text nodes defined as a key. In our example, each arrow represents a reference. For example, *AuthorId*$_{s1}$ is defined as a reference on *Id*$_{s1}$.

We suppose that a set of correspondences is provided between each source and the target schema. Each correspondence relates a source node n with a node in the target schema n' and states that the two nodes represent the same concept. This correspondence is denoted n ≅ n'. In figure 1, dotted lines between pairs of nodes in different schemas represent correspondences (e.g. *Id*$_{s1}$ ≅ *Id*$_{ts}$).

If two source nodes n1 and n2 and a target node n are such that n1 ≅ n and n2 ≅ n, then we consider that a correspondence is also valid between n1 and n2; it is denoted n1 ≅ n2. We also consider correspondences between two sets of text nodes. Given two sets of nodes s1 and s2, there is a correspondence between s1 and s2 if (i) both s1 and s2 contain the same number of nodes; (ii) 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. {*AuthorId*$_{s1}$, *Address*$_{s1}$} ≅ {*Id*$_{ts}$, *Address*$_{ts}$}; {*Id*$_{s1}$, *Name*$_{s1}$} ≅ {*Id*$_{s2}$, *Name*$_{s2}$}).

4. The general framework

This section presents our framework for the automatic generation of XML mappings (figure 2).

The inputs or our framework are a target schema, a set of source schemas and a set of semantic correspondences between elements of the target schema and elements of the source schemas. They are show on the top of figure 2. The outputs are a set of mappings representing different ways to derive the instances of the target schema from the instances of the source schemas. These mappings are expressed in XQuery or XSLT and they are shown on the left side of figure 2. The framework has the following components.

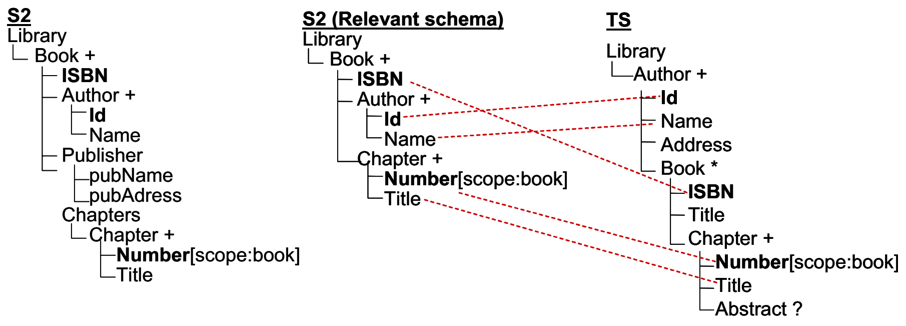

Figure 3. Relevant Schemas

- **Relevant schema identification.** The goal of relevant schema identification is to determine for a given source schema, the portion that is relevant with respect to the target schema. The output of this component is a schema (called **relevant schema**) that represents the relevant portion of the source; this component also produces a set of semantic correspondences representing the matching elements between the target schema and the relevant schema. The identification of the relevant schema is performed because some source schemas may have only a small portion that is relevant with respect to the target schema, and each time this information is needed, the whole schema must be browsed. The identification of the relevant schema allows getting smaller sized representations.
- **Target schema decomposition.** The goal of this component is to decompose the target schema into a set of subtrees (called **target subtrees**). Due to the semi-structured nature of XML documents, it is extremely difficult to directly define the mappings for the whole target schema. We will therefore search for a way to derive instances of each target subtree, and then define the mapping for the whole schema.
- **Partial mapping determination.** Given a target subtree, this component produces as output a set of partial mappings, each of them representing a way to derive instances of the target subtree from instances of the source schemas. The partial mapping determination is performed independently from the other target subtrees. Every partial mapping satisfies the constraints defined in the associated target subtree.
- **Target mapping generation.** This component derives the mappings for the whole target schema by assembling the partial mappings of the different target subtrees. The output of this component is a set of target mappings, each one representing a different semantic. All the target mappings satisfy the cardinality constraints and the hierarchical relations existing between the target subtrees. The generated mappings are abstract queries, independent of the external query language.
- **XQuery and XSLT translations.** The goal of these two components is to translate the abstract mappings into respectively XQuery and XSLT for using them in different contexts.

In the remaining of the paper we will give a description for each component of the framework.

5. Relevant schema identification

The goal of relevant schema identification is to determine and to extract for each source schema, the portion that is relevant with respect to the target schema; its output is a relevant schema.

For a given source schema, the relevant schema is generated along with a set of semantic correspondences that relate elements of the relevant schema with elements of the target schema. If we consider the source S2 of figure 1, its relevant schema is shown in figure 3. This figure also shows the correspondences between the relevant schema and the target schema. These correspondences are derived using the correspondences initially provided between elements of the considered source schema and elements of the target schema.

For identifying the relevant portion of a source schema, we must firstly identify the nodes in this schema that are useful for generating mappings. We call these nodes relevant nodes. There are four types of relevant nodes:

- Obviously, all the nodes which are involved in a correspondence with a node in the target schema are relevant nodes (e.g. *ISBN*$_{s2}$).

- All the multivalued nodes having a descendent which is a relevant node are also relevant nodes (e.g. *Author*$_{s2}$). They are useful for keeping the semantics of the source schema. For example, if we do not keep the node *Author*$_{s2}$ in the relevant schema, we cannot find the corresponding instance of *Name*$_{s2}$ for every instance of *Id*$_{s2}$.
- Each node defined as a key of a relevant node or defined as a reference on a relevant node is also a relevant node;
- The root of the source schema is a relevant node if there is at least one relevant node in this schema. This root guarantees that the extracted relevant schema is a tree (e.g. *Library*$_{s2}$).

The relevant schema for a given source schema contains all the relevant nodes. If two relevant nodes n and n' are such that n is an ascendant of n' and if there is no relevant node which is a descendent of n' and an ascendant of n, then there will be an edge from n to n' in the relevant schema. The relevant schema corresponding to the source S2 of figure 1 is given in figure 3. In S2, *Chapters*$_{s2}$ is not a relevant node. There is therefore an edge from *Book*$_{s2}$ to *Chapter*$_{s2}$ in the corresponding relevant schema. In the source schema S1, all the nodes are relevant. S1 and the corresponding relevant schema are therefore identical.

The correspondences between the relevant schema and the target schema are produced from the correspondences between the source schema and the target schema by replacing every source node by its corresponding node in the relevant schema.

The algorithm for identifying a relevant schema takes as input a source schema and the set of its correspondences with the target schema. The generation is done through an in-width exploration of the source schema from the leaves to the root.

The relevant schema identification module allows reducing the size of the source representation that will be processed by the other modules without losing semantics. In our framework, this step is optional. For a given source schema, we can use either the relevant schema or the initial source schema. For simplicity, we will refer to the description of a source as source schema in both cases.

6. Target schema decomposition

This component decomposes the target schema into subtrees, called **target subtrees**. Given a target schema, each target subtree t is such that:

- the root r of the subtree is either a multivalued node or the root of the target schema;
- all the other nodes in t that are descendents of r and are monovalued;
- for any pair of nodes n1 and n2 in t, if there is an edge from n1 to n2 in the target schema, then this edge is also in t;
- there is at least one text node in t.

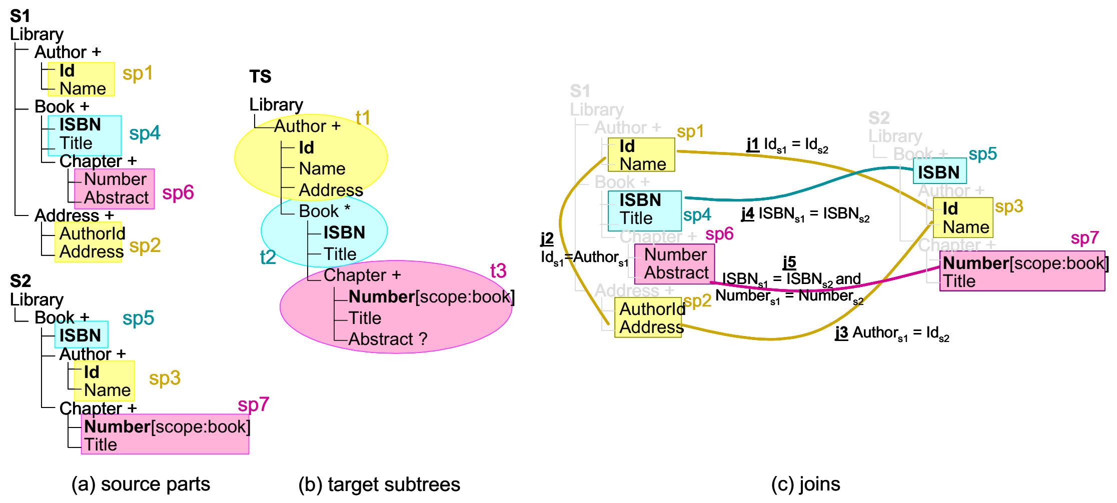

Figure 4. Target subtrees, source parts and join operations

There are three target subtrees for the target schema of figure 1; they are shown in figure 4 (b): t1 is the subtree composed of the multivalued node *Author*$_{ts}$ and its three monovalued children *Id*$_{ts}$, *Name*$_{ts}$ and *Address*$_{ts}$; t2 is the subtree composed of *Book*$_{ts}$ and its two monovalued children *ISBN*$_{ts}$ and *Title*$_{ts}$; and t3 is the subtree composed of *Chapter*$_{ts}$, *Number*$_{ts}$, *Title*$_{ts}$ and *Abstract*$_{ts}$. The node *Library*$_{ts}$ is not in a target subtree since it has no descendent which is a monovalued text node. We can notice that every node in t1, t2 or t3 is monovalued, except the root.

There is a hierarchical relation between the different target subtrees. Given two target subtrees t and t' in the same schema such that the root of t' is the child of one of the nodes of t, t is the parent of t' and t' is a child of t. For example, considering the target subtrees in figure 4 (b), t2 is the child of t1 and the parent of t3. We also consider that a target subtree can be optional or mandatory with respect to its parent subtree. It is optional (resp. mandatory) if its root is optional (resp. mandatory). In our example, t1 is mandatory and t2 is optional.

7. Partial mapping determination

Once the target subtrees have been defined over the target schema, the partial mappings are determined for every target subtree. Each partial mapping represents a way to derive instances of a target subtree from instances of the source schemas, and it also satisfies the constraints defined in the target schema. For a given target subtree, there may be several partial mappings, corresponding to different ways to derive instances for the considered subtree.

The partial mappings determination of each target subtree is performed independently from the others. The building blocks of this process are shown in figure 5: the **source identification** process determines the parts of the source schemas (called **source parts**) that are relevant with respect to each target subtree; the **join identification** process searches the joins that can be used to combine these source parts; the **partial mapping determination** process derives all the partial mappings from the source parts and the join operations between them.

Before defining the notion of parts in a source schema, we first present an extended definition of 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 in the same schema. Given two nodes n and n' in a schema, n is multivalued (resp. monovalued) with respect to n' if there may be several instances (resp. only one instance) of n for each instance of n'. If we consider the path from n to n', the cardinality of n with respect to n' can be derived form the cardinalities associated to the edges composing the path. In the example given figure 4, Id_{S1} is multivalued with respect to $ISBN_{s1}$ and $ISBN_{s1}$ is monovalued with respect to $Number_{s1}$.

Source parts identification. For a given target subtree, we identify all the source parts that are relevant, independently from the other target subtrees. Given a target subtree t, each source part for t, denoted sp, is a set of text nodes in a source schema S that satisfies the following conditions:

- There is a correspondence between each node in sp and a single distinct text node in t;
- there is at least one node n in sp such that the other nodes in sp are monovalued with respect to n;
- there is no other set of text nodes c in S such that sp $\subseteq$ c and such that c satisfies the two above conditions.

Consider the three target subtrees of figure 4 (b); t1 has two source parts in S1: sp1 = {Id_{s1}, $Name_{s1}$}, and sp2 = {$AuthorId_{s1}$, $AuthorAddress_{s1}$}. It has also one source part in S2: sp3 = {Id_{s2}, $Name_{s2}$}; t2 has two source parts sp4={$ISBN_{s1}$, $Title_{s1}$} and sp5={$ISBN_{s2}$} in S1 and S2 respectively; t3 has two source parts sp6={$Number_{s1}$, $Abstract_{s1}$} and sp7={$Number_{s2}$, $Title_{s2}$} in S1 and S2 respectively. These source parts are shown in figure 4 (a).

Join identification. The semantic of the join used in our framework is the one proposed in most of the existing algebras [7][13][1][6]. It takes as input two collections and a predicate, and produces a new collection consisting of the concatenation of pairs of nodes satisfying the predicate. Beside the join, our framework also use set-based operations like Union, Intersection and Difference (cf. section 8).

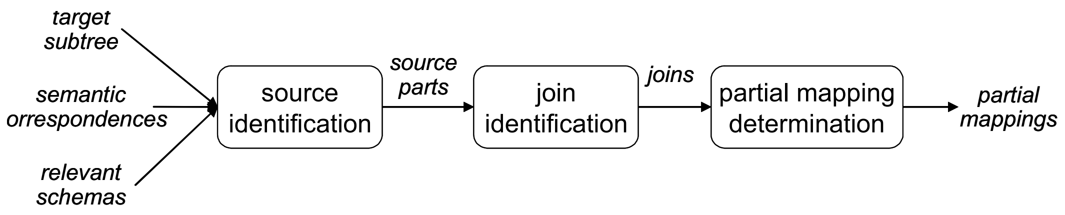

Figure 5. The building blocks of the partial mappings determination

Once the source parts are defined for a given target subtree, we search for the possible joins between these sources parts. These joins are identified using the key definitions. There are two distinct cases: the two source parts may belong to the same source schema or to different ones. If the two source parts sp and sp' are in the same source schema, a join is possible between them if there are two sets of text nodes c and c' in the schema such that:

- c is defined as a key and c' is defined as a reference on c;
- the intersection between c and sp is not empty and the intersection between c' and sp' is not empty.

The join between two source parts sp and sp' with the join predicate c=c' is denoted j[c=c'](sp, sp'). For instance, sp1 and sp2 are in the same schema (figure 4), $AuthorId_{s1}$ is a reference on Id_{s1}; therefore, there is a join between these two parts j[Id_{s1}=$AuthorId_{s1}$](sp1, sp2).

If the two source parts belong to different schemas, the joins are defined as follows: consider two source parts sp and sp' in different source schemas; given a set of text nodes c having a non-empty intersection with sp and another set of text nodes c' having a non-empty intersection with sp', a join is candidate between sp and sp' with the predicate c=c' if the following conditions hold:

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

For example, a join operation 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.

According to the above rule, we cannot find the join j[$Number_{s1}$=$Number_{s2}$](sp6, sp7) because $Number_{s2}$ is defined as a relative key. However, we know that the combination {$Number_{s2}$, $ISBN_{s2}$} is unique in the whole schema because the scope of $Number_{s2}$ is $Book_{s2}$ and $ISBN_{s2}$ is its absolute key. Our rule is therefore extended 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 in a schema during the determination of the possible joins, we search for a combination of two or more keys (with at least one absolute key in the combination) to be used in the predicate of the join.

All the join operations identified in our running example are shown in figure 4 (c). In the figure, every join operation between two source parts is represented by an edge between the source parts; the edges are numbered and labeled with the join predicate.

Partial mappings determination. Each partial mapping corresponding to a given target subtree represents a way to derive its instances from the sources. For each target subtree, the partial mappings are determined from the corresponding source parts and the joins between them.

For each target part, the source parts and the join operations between them are represented by a graph J(V, E), called **join graph**. V is the set of source parts and E is the set of join operations between the source parts. For example, the join graph of t3 is shown in figure 6; it contains two source parts sp6 and sp7 and one join between them.

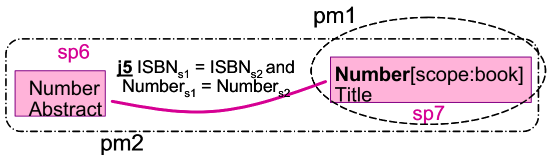
pm1
sp6
j5 $ISBN_{s1}$ = $ISBN_{s2}$ and $Number_{s1}$ = $Number_{s2}$
Number
Abstract
Number[scope:book]
Title
sp7
pm2

Figure 6. Join graph and partial mappings for t3

For a given target subtree, several partial mappings may be determined. Each one preserves the cardinality constraints defined in the target subtree and represents a different semantic. Given a target part t and its join graph J(V, E), a partial mapping, denoted pm, is defined as a connected sub-graph of J(V, E) such that for every mandatory text node n in t, there is at least one node n' in a source part of pm such that n ≅ n'.

In the example given in figure 6, there are two partial mappings for t3: pm1 and pm2; pm1 contains only the source part sp7; pm2 contains source parts sp6 and sp7 and the join between them. Notice that pm1 does not allow deriving instances for $Abstract_{ts}$; this is not a problem since $Abstract_{ts}$ is defined as an optional node in the target schema.

A partial mapping is denoted either by the set of joins contained in the corresponding graph when this graph contains more than one source part or by the name of the source part when the graph contains a single source part. In our example, pm1 and pm2 are denoted {sp7} and {j5} respectively.

8. Target mappings generation

Once the partial mappings for every target subtree are determined, the target mappings are generated by combining the partial mappings of the different target subtrees. The constraints existing between the target subtrees are preserved by each target mapping. For example, in the target schema of figure 4 (b), there is a parent-child relation between t1 and t2. Each target mapping must preserve this information.

A set of candidate mappings are generated by combining partial mappings. These candidate mappings are then checked to see if the parent-child relations existing between the target subtrees are preserved. Consider the set of target parts t1, t2, ..., tn in a target schema TS and their partial mappings, a candidate mapping cm for TS is a set of partial mappings such that:

- there is at most one partial mapping for each target subtree;
- there is exactly one partial mapping for each target subtree t such that all the target subtrees between t (including t) and the root of TS are mandatory;
- for each mandatory target subtree t and its parent target subtree t' in TS, if there is a partial mapping for t in cm then there is also a partial mapping for t' and *vice versa*.

In the graph given in figure 4 (c), several partial mappings can be determined; consider the following ones: the two partial mappings pm3 = {sp1} and pm4 = {j3} for t1; the partial mapping pm5 = {j4} for t2; and the partial mapping pm2 = {j5} for t3. There are four candidate mappings cm1 = {pm3}, cm2 = {pm4}, cm3 = {pm3, pm5, pm2}, cm4 = {pm4, pm5, pm2}. Notice that the candidate mappings cm1 and cm2 do not contain a partial mapping for t2 and t3 because t2 is optional. Each candidate mapping that contains a partial mapping for t2 must also contain a candidate mapping for t3, because t3 is a mandatory subtree of t2; this is the case for the mappings cm3 and cm4. Each candidate mapping that contains a partial mapping for t3 must also contain a partial mapping for t2.

For each candidate mapping, parent-child relations between the target subtrees must be checked. Consider the candidate mapping cm4 = {pm4, pm5, pm2}; each instance returned by pm4 contains an instance of sp3 (because pm4 has a join involving sp3), and each instance returned by pm5 contains an instance of sp5. The instances of sp5 that correspond to each instance of sp3 can be found using their hierarchical relation in S2; we can therefore find which instances of pm4 corresponds to each instance of pm5. We can also check that the same holds for pm5 and pm2 and consequently the parent-child relations in the target schema are preserved by candidate mapping cm4.

On the contrary, cm3 = {pm3, pm5, pm2} does not preserve the parent-child relation between t1 and t2 since there is no source part in pm3 and pm5 that allows to derive this information. Consequently, cm3 is not a valid mapping.

New mappings can be derived by applying set-based operations like Union, Intersection and Difference to two or more mappings. For example, if we apply a union to the mappings cm3={pm3, pm5, pm2} and cm2={pm4}, the result of the union is a new mapping which can derive instances of t1 from the union of pm3 and pm4, instances of t2 from pm5, and instances of t3 from pm2.

The resulting mappings are abstract queries that are independent from any query language. The two modules **XQuery translation** and **XSLT translation** translate these mappings into respectively XQuery and XSLT. This translation is done by instantiating a query pattern for every partial mapping.

9. Conclusion

In this paper, we have presented a framework to automatically generate complex mappings for XML data transformation and integration. The mappings are generated for a target schema, given a set of source schemas and the semantic correspondences between the target schema and the source schemas.

This work is done within the MediaGRID project [4], which proposes a mediation framework for a transparent access to biological data sources. Our framework is used to generate mappings for a target schema constructed by domain experts independently of the sources, and the semantic correspondences with the source schemas are given.

Some questions remain open: choosing the mappings that most fit the needs of a given user; the use of quality criteria can be useful for this purpose; another open problem is how to maintain the generated mappings consistent in case of changes in the target schema or the source schemas; the mapping generation process can also support other meta data and be improved by adding some data cleaning facilities to improve the quality of the generated mappings.

References

[1] D. Beech, A. Malhotra, M. Rys, "A formal data model and algebra for XML", Communication to the W3C, 1999.

[2] M. Bouzeghoub, B. Farias Lóscio, Z. Kedad, A.-C. Salgado, "Managing the evolution of mediation queries". Proc. of the 11th. Int. Conf. on Cooperative Information Systems (CoopIS'2003), 2003.

[3] K. T. Claypool, E. A. Rundensteiner, "Sangam: A Transformation Modeling Framework", Proc. of Eighth International Conference on Database Systems for Advanced Applications (DASFAA'03), Kyoto, 2003.

[4] C. Collet, K. Belhajjame, G. Bernot, G. Bruno, C. Bobineau, B. Finance, F. Jouanot, Z. Kedad, D. Laurent, G. Vargas-Solar, F. Tahi, T.-T. Vu, X. Xue, “Towards a target system framework for transparent access to largely distributed sources”. Proc of the International Conference on Semantics of a Networked World Semantics for Grid Databases (IC-SNW’2004), 2004.
[5] R. Dhamankar Y. Lee, A. Doan, A. Y. Halevy, P. Domingos, “iMAP: Discovering Complex Mappings between Database Schemas”, Proc. of International conference ACM SIGMOD, SIGMOD’04, pp. 383-394.
[6] M.-F. Fernandez, J. Siméon, P. Wadler: “An Algebra for XML Query”, Foundations of Software Technology and Theoretical Computer Science (FSTTCS’00), 2000.
[7] L. Galanis, E. Viglas, D.-J. DeWitt, J.-F. Naughton, D. Maier: “Following the Paths of XML Data: An Algebraic Framework for XML Query Evaluation”, Technical Report, University of Wisconsin, Madison 2001.
[8] Z. Kedad,; M. Bouzeghoub,: “Discovering View Expressions from a Multi-Source Information System”. Proc. of the 4th. Int. Conf. on Cooperative Information Systems (CoopIS’1999), 1999.
[9] R. J. Miller, L.M. Haas, M.A. Hernández: “Schema Mapping as Query Discovery”. Proc. of the 26th International Conference on Very Large Data Bases (VLDB’00), 2000.
[10] R.J. Miller, M.A. Hernández, L.M. Haas, L.L. Yan, C. T. Howard Ho, R. Fagin, L. Popa: “The Clio Project: Managing Heterogeneity”, SIGMOD Record, 2001.
[11] L. Popa, Y. Velegrakis, R.J. Miller, M. A. Hernandez, R. Fagin: “Translating web data”, Proc. of the 28th International Conference on Very Large Data Bases (VLDB’02), 2002.
[12] E. Rahm, P. A. Bernstein, “A survey of approaches to automatic schema matching”. Proc. of the 27th International Conference on Very Large Data Bases (VLDB’01), pp 334-350.
[13] S. D. Viglas, L. Galanis, D. J. DeWitt, D. Maier, J. F. Naughton: “Putting XML Query Algebras into Context”, Technical Report, University of Wisconsin, 2002.
[14] L. L. Yan, R. J. Miller, L. M. Haas, R. Fagin: “Data-Driven Understanding and Refinement of Schema Mappings”, Proc. Of International conference of ACM SIGMOD 2001 (SIGMOD’04). 2001.
[15] X. Yang, M. L. Lee, T. W. Ling, “Resolving structural conflicts in the integration of XML schemas: a semantic approach”, Proc. of 22nd International Conference on Conceptual Modeling (ER’03), Chicago, Illinois, 2003.
[16] C. Yu, L. Popa, “Constraint-based XML query rewriting for data integration”, Proc. of international conference ACM SIGMOD, SIGMOD’04, Paris, 2004, pp. 371-382.
[17] L. Zamboulis, A. Poulovassilis, “XML data integration by Graph Restructuring”, Proc. of the 21st Annual British National Conference on Databases, BNCOD21, Edinburgh, 2004, pp 57-71.