# Constructing Extensible XQuery Mappings

Gang Qian
Dept. of Computer Science and Engineering
Southeast University, Nanjing 210096, China
qiangang@seu.edu.cn

Yisheng Dong
Dept. of Computer Science and Engineering
Southeast University, Nanjing 210096, China
ysdong@seu.edu.cn

## ABSTRACT

Constructing and maintaining semantic mappings are necessary but troublesome in data sharing systems. While most current work focuses on seeking automated techniques to solve this problem, this paper proposes a combination model for constructing extensible mappings between XML schemas. In our model, complex global mappings are constructed by first defining simple atomic mappings for each target schema element, and then combining them using a few basic operators. At the same time, we provide automated support for constructing such combined mappings.

## Categories and Subject Descriptors

H.2.3 [**Database Management**]: Languages – *Query languages*; H.2.5 [**Database Management**]: Heterogeneous Databases.

## General Terms

Management, Languages

## Keywords

XQuery, Mapping, Extensibility, Automated support.

## 1. INTRODUCTION

Schema mapping is one of the underlying components of data sharing systems. As is known, constructing and maintaining such mappings are labor-intensive and error-prone processes. We limit our attention to the XML model and the mappings expressed in XQuery (called XQuery mapping), though our discussion is also applicable to other data models.

While recent research on *schema matching* [4], *mapping discovery* [2, 3] and *mapping adaptation* [5] has made exciting progress towards *semi*-automating these processes, the mapping self is still represented as naive expressions, e.g., XQuery clauses in our context, which is troublesome for the user (administrator) to deal with. For example, in dynamic environment like the Web, as schemas evolve, the mappings may need to be frequently modified and maintained manually. Additionally, as complicated large schemas become prevalent on the Web, it may be more feasible to start with some simple local mappings, and then glue them together to formulate complex ones. Hence we believe that a suitable mapping model would be able to alleviate the burden on the user, for cases out of the capabilities of the above techniques. In fact, this is usual in practice.

We propose a combination model for constructing extensible XQuery mappings between XML schemas. In our model, a global mapping is composed of a set of simple atomic mappings, which are combined by a few of basic operators. With these operators (e.g., *Nest*, *Join* or *Merge*), two mappings (say $M_1$ and $M_2$) are connected to a combined one, say $M_{1,2}$. Here *extensibility* means that the resulting mapping $M_{1,2}$ can be combined again with others, possibly using another combination operator, and it is also possible to reset the operator in $M_{1,2}$, or recover $M_1$ and $M_2$ from it. Consequently, the complex global schema mappings can be incrementally constructed, starting with the simple ones, and continuously applying the combination operators. To maintain them, it only needs to adjust the corresponding parts, e.g., the atomic mappings affected by schema evolving, while other parts are reused. At the same time, based on the previous works on schema matching and mapping discovery, we present automated support for constructing such combined mappings.

## 2. MAPPING COMBINATION

Atomic mapping has the following general form, where SP is a simple path with no branching predicates, and $SP_1$() denotes that $SP_1$ must start at a schema root, while $SP_k(\$v_{k-1})$ indicates that in the FOR clause $SP_k$ is relative to variable $\$v_{k-1}$.

```
for $v1 in SP1(), ……, $vn in SPn($vn-1)
where φ($v1, ……, $vi)
return () | SPn+1($vj) | <e></e>
```

We refer to $\$v_i$ as the *F-variable* of the atomic mapping, and $\$v_n$ as its *primary F-variable* (PFV). In the above formulation, the optional WHERE clause defines a filter $\varphi$, and the RETURN clause indicates that the atomic mapping may be *empty*, *copy*, or *construct* type, which respectively returns empty sequence, copies of XML fragment or new constructed elements, e.g., instances of *e*. Samples of atomic mappings are given below.

```
Mbook(): for $n in doc("S1")//novel
         return <book></book>
Mtitle:  for $t in doc("S1")//novel/title
         return $t
```

Compared to global mapping, atomic mapping is easier to be formulated, since each atomic mapping is defined separately, and the context of the target schema element is ignored. Then, with the combination operators given below, the separately defined atomic mappings are connected and a combined one is obtained, where the source elements are semantically related (by connection condition), and the returned instances are structurally nested.

Let $M_1$ and $M_2$ be atomic mappings, and $M_1$ be *construct* type. Figure 1 shows the combination rules of the *Nest*, *Join* and *Merge* operators, which respectively connect $M_1$ and $M_2$, and generate the combined mapping $M_{1,2}$. Here *exp* corresponds to the return expression of $M_2$, and $\psi$ is an expression w.r.t. the *F-variables* of $M_1$ and $M_2$, which represents the *connection condition* of relating the atomic mappings. Semantically, the *Nest* operator captures the outer-join relationship between $M_1$ and $M_2$. For each binding tuple

```
for $v_{1,1}$ in $SP_{1,1}$(), ……, $v_{1,n}$ in $SP_{1,n}$($v_{1,n-1}$)
where $\varphi_1$
return <e>
   for $v_{2,1}$ in $SP_{2,1}$(), ……, $v_{2,m}$ in $SP_{2,m}$($v_{2,m-1}$)
   where $\varphi_2$ and $\psi$
   return exp </e>
```

```
for $v_{1,1}$ in $SP_{1,1}$(), ……, $v_{1,n}$ in $SP_{1,n}$($v_{1,n-1}$)
let $v:= for $v_{2,1}$ in $SP_{2,1}$(), ……, $v_{2,m}$ in $SP_{2,m}$($v_{2,m-1}$)
          where $\varphi_2$ and $\psi$
          return exp
where $\varphi_1$ and $\sigma$
return <e>{$v}</e>
```

```
for $v_{1,1}$ in $SP_{1,1}$(), ……, $v_{1,n}$ in $SP_{1,n}$($v_{1,n-1}$)
for $v_{2,1}$ in $SP_{2,1}$(), ……, $v_{2,m}$ in $SP_{2,m}$($v_{2,m-1}$)
where $\varphi_1$ and $\varphi_2$ and $\psi$
return <e> exp </e>
```

**Figure 1. The *Nest*, *Join* and *Merge* operators**

of $M_1$, the resulting combined mapping $M_{1,2}$ will returns a new instance of $e$, whether $\psi$ holds or not. In the combination rule of the *Join* operator, $\sigma$ is an expression w.r.t. the variable `$v`. For example, $\sigma$ may be `count($v)`>0. In this case, the *Join* operator represents a full join relationship between $M_1$ and $M_2$. Lastly, the *Merge* operator has an analogy with the product relationship.

Note that for the *Nest* and the *Join* operator, the PFV of $M_1$ forms the PFV of $M_{1,2}$, while for the *Merge* operator, the PFVs of $M_{1,2}$ are the union of the ones of $M_1$ and $M_2$. Recursively, the combined mapping $M_{1,2}$ may be combined with other mappings (atomic or combined). We omit the details from the paper and simply give the following example for illustration.

```
for $n in doc(“S1”)//novel
 return <book>
     for $n1 in doc(“S1”)//novel, $t in $n1/title
     where $n=$n1
     return $t
     for $a in doc(“S1”)//authors/author
     where $n/aid=$a/id
     return <author></author>
     </book>
```

The above combined mapping is generated by applying twice the *Nest* operator, first combining $M_{\texttt{book()}}$ and $M_{\texttt{title}}$ with the connection condition `$n=$n1`, second combining the resulting mapping of the first step with $M_{\texttt{author()}}$, another atomic mapping as highlighted above.

## 3. CONSTRUCTION

Based on the works on schema matching and mapping discovery, we also provide automated support for constructing the combined mapping. Let $e(e_1, …, e_n)$ be a target schema element $e$ nesting $e_i$ ($1 \le i \le n$). Our task is to generate the combined mapping for $e(e_1, …, e_n)$, which is reduced to build the atomic mappings $M_{\mathrm{e}}$ and $M_{\mathrm{e}i}$, choose the combination operator of and discover the conditions of connecting $M_{\mathrm{e}}$ and $M_{\mathrm{e}i}$.

Atomic mapping may be built in terms of the results of schema matching, which produces semantic correspondences (matches) between elements of schemas. For example, the atomic mapping $M_{\texttt{book()}}$ in Section 2 may be derived from the match between the elements `book` and `novel`. For our mapping model, we need not to require that the produced matches should be desired, since the combined mapping is extensible and maintainable.

The combination operator may be determined by the cardinality constraints of $e_i$. For example, if $e_i$ is optional and multiple, the *Nest* operator may be applicable; if $e_i$ is mandatory and unique, then the *Merge* operator may be applicable. Generally, the feasible operators are determined also by factors such as the F-variables of $M_{\mathrm{e}i}$, and the connection condition $\psi$.

Connection condition $\psi$ can be heuristically discovered from the semantic relationships between the source schema elements. As presented in [2, 3, 5], such relationships are captured by the *structural*, *user* and *logical associations*, which respectively describe a set of associated schema elements. Let $a$ be the source schema element specifying the PFV of $M_{\mathrm{e}}$, and $a_i$ be the element specifying the PFV of $M_{\mathrm{e}i}$. If $a$ and $a_i$ are in a structural association, then $\psi$ may be formulated in terms of the common path of the elements $a$ and $a_i$. Otherwise, if they are in a user association, then $\psi$ may be formulated with the path assigned by the user. Lastly, if they are neither in a structural nor in a user, but in a logical association, then $\psi$ may be formulated in terms of the referential path between the schema elements $a$ and $a_i$. For example, the condition, `$n/aid=$a/id`, of connecting $M_{\texttt{book()}}$ and $M_{\texttt{author()}}$ is derived from the logical relationship between the elements `novel` and `author` (see Section 2). Note that the discovered conditions may be multiple, and the user is expected to make right decision in the process.

## 4. RELATED WORK

Schemas and semantic relationships between schema elements are mainly focused on by current work on schema matching [4] and mapping discovery [2, 3], and which mappings are affected by schema evolution is the interest of mapping adaptation [5]. In contrast, our work gives more attention to mapping self and its constitution. We consider mappings as first class of citizens, and provide combination operators to connect them. The same idea is also proposed in [1] to solve the problem of management of meta data. Yet the subjects there are matches between schemas. What we dealt with in this paper are mappings, which are semantically richer and have more complex formulation.

## 5. CONCLUSION

In this paper, we have presented the combination model for constructing extensible mappings between XML schemas. From the simple atomic mappings, a complex global mapping is easy to be constructed step by step by applying the combination operators. Additionally, the constructed combined mappings are maintain-able and adapt well to dynamic environment like the Web.

## 6. REFERENCES

[1] P. A. Bernstein. Applying Model Management to Classical Meta Data Problems. In *Proc. of CIDR*, Pages 209-220, 2003.

[2] R. Miller, L. Haas, and M. Hernández. Schema Mapping as Query Discovery. In *Proc. of VLDB*, Pages 77–88, 2000.

[3] L. Popa, Y. Velegrakis, R Miller, M. A. Hernandez, and R. Fagin. Translating Web Data. In *Proc. of VLDB*, Pages 598–609, 2002.

[4] E. Rahm and P.A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4): 334–350, 2001.

[5] Y. Velegrakis, R. J. Miller, and L. Popa. Preserving mapping consistency under schema changes. *The VLDB Journal*, 13(3): 274-293, 2004