Journal of Software Engineering and Applications, 2015, 8, 62-71
Published Online February 2015 in SciRes. http://www.scirp.org/journal/jsea
http://dx.doi.org/10.4236/jsea.2015.82008

Scientific Research Publishing

Semantic Enrichment of XML Schema to Transform Association Relationships in ODL Schema

Doha Malki, Mohamed Bahaj

Department of Mathematics and Computer Science, University Hassan 1st, Settat, Morocco
Email: doha.malki@uhp.ac.ma, mohamedbahaj@gmail.com

Received 24 January 2015; accepted 13 February 2015; published 15 February 2015

Abstract

This paper presents an approach for transforming an XML schema we enriched in ODL (Object Definition Language) schemas. It is possible to realize the concepts of ODL in a model of XML schema, we propose to introduce an enrichment concretizing these concepts in the XML Schema models. We chose oriented object database as a target database because there are many common characteristics between XML and object-oriented model, thus the mapping from XML data into object-oriented databases is more interesting. Also the object-oriented data bases have become very widespread and acceptable and they offer an evolutionary approach, so we agree that it is time to develop a translation between XML and OO databases. The purpose of this article is to automate transformation process of an XML schema to an ODL database. Our work focuses on preserving semantics transformation of association relationships and we describe set of rules to create ODL classes from an enriched XML schema. The experimental study shows that the approach is feasible, and results are the same, the source database is transformed into target one without loss of data.

Keywords

XML Schemas, ODL, Mapping, Association, OODB

1. Introduction

Extensible Markup Language, a met language that allows users to define their own customized markup languages, is characterized by its flexibility and extensibility. Due to all its qualities, it's considered as hot topic for describing and interchanging data through internet between different systems.

How to cite this paper: Malki, D. and Bahaj, M. (2015) Semantic enrichment of XML schema to Transform Association relationships in ODL schema. *Journal of Software Engineering and Applications*, **8**, 62-71.
http://dx.doi.org/10.4236/jsea.2015.82008

The migration of database appears today very interesting and promotes organizations to move towards new technology. Since information is valuable resources for organizations, the mapping process must be submitted before any shift to a new technology [1]. Furthermore, the characteristics of the XML Schema standard [W3C, 2008] are supported by the standard ODMG 3.0 [2], and query languages are more powerful, which encourage to attempt to migrate existing database into new environment.

Database migration is a process wherein all the components of a source database are converted to their equivalents in the environment of the target one.

ODL is designed to support semantic constructs of ODMG object model. It is used to define the pattern of a compatible ODMG database independently of any programming language.

In this article, we present techniques for enriching the XML Schema. Our goal is to introduce the richness of ODL formalism to facilitate the migration of XML Schemas to ODL schema. The introduction of ODL formalism in the XML schema is obtained by extension. To do this, we proceed in several stages:

- Defining the concepts supported by XML Schema;
- Defining XML Schema extensions to take into account all the specificities of ODL objects. These extensions exploit the extension mechanisms supported by XML Schema to remain compliant with the W3C standard.

The content of this document provides a brief introduction to XML Schema and ODL. The rest of the paper is organized as follows. In Section 2, we review some closest related work. Section 3 presents XML Schema conceptual enrichment, we will explain how XML and ODL implement association relationship; several rules to transform an XML schema enriched in ODL schema focusing on transforming association relationship are described in Section 4. In Section 5, we present the processing steps and an example for each type of association relationship is given. Section 6 presents evaluation of our approach by comparing the results of queries; finally Section 7 concludes the paper.

2. Related Work

There are many works that explain the mapping from XML to object-oriented databases, In [3], it discuss the modeling of XML and the need for transformation. A number of generic transformation rules of the conceptual model OO to the XML schema are presented, accentuated on transforming inheritance and aggregation relationships.

Most existing work focuses on a method that was designed to map an object database into an XML database for the interoperability of databases. Schema translation process is supplied with a UML class model [4]. In this paper, set of rules to translate a simple database schema specified in ODL into XML Schema are presented focusing on transforming association relationships (1:M) and (M:M),

In [5], the paper covers XML modeling and the need for transformation. It presents a number of XML schema transformation steps to ORDB, focusing on the transformation of association relations. Different types of these conceptual relationships (one-to-one, one-to-many and many-to-many) and their transformations are mainly discussed.

In [6], it address the mapping of the contents of an existing object-oriented database into XML using object graph; the reverse process is also proposed to store XML data in object-oriented database. In this work, the author use object graph for the transformation, but it does not cover all possible types of relationships.

3. Conceptual Enrichment of XML Schema

Associations allow complete modeling object states. The ODMG support bidirectional binary associations, cardinality (1: 1), (1: N) or (N: M). An association from A to B defines two opposite paths crossing, A- > B and B- > A. Each path must be defined in the ODL object type source by a relationship keyword.

A class in ODL is specified using the class keyword, an attribute is specified using the attribute keyword and relationship is specified using the relationship keyword. Although it is possible to materialize these concepts implicit in a XML schema, we propose to introduce an enrichment embodying these concepts explicitly in the XML Schema.

To do this, we use the extension mechanism advocated by XML Schema, this addition used to include the specific subjects of ODL and to highlight relationships between concepts. In our mapping process, these new concepts help to preserve the semantic of relationships.

3.1. ODL

A Class in ODL is defined as follows:

class <name>
(extent <names> key <attribute>…
{
<list of elements = attributes, relationships, methods>
};

A relationship between a class C1 and class C2 is defined by attributes in both classes of types according to the cardinality of the relationship [7]:

- One-to-one: an attribute of type C2 * is included in C1 and another one of type C1 * is included in C2.
- One-to-many: an attribute of type collection < C2 *> (a set or a bag) is contained in C1 and C1 contains an attribute of type C2*.
- Many-to-many: C1 contains an attribute of type collection< C2 *> and *vice-versa*.

A relationship must be specified in both directions. In ODL, the inverse keyword is used to designate the relationship in the opposite direction. For example, if we delete an object of C1, links with all C2 objects will be automatically dereferenced, also all objects C2 links back to the C1 objects will be dereferenced. If we relate an object of C1 to a set of C2 objects, the reverse links will automatically be created. In other words, it will create a link to each of C2 objects to the C1 object.

3.2. XML Schema

XML Schema represents integrity constraints using the XPath expression [8]. It is possible to specify constraints that correspond to unique values, primary keys and relationships in ODBS.

The tags unique is used to define unique, key is to define primary key, and key ref for key reference which defined by refer attribute to specify attribute or element corresponds to the key element or unique specified. The XPath expression selector defines the domain of a constraint, and the field XPath defines the elements or attributes that represent the constraint.

For two complex types CT_1 and CT_2, participating in association relationship (see **Figure 1**), XML schema is defined as follows:

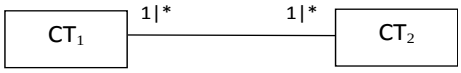

Figure 1. Example of association relationship.

```
<xsd:element name="CT1">
<xsd:complexType>
     <xsd:sequence>
     <xsd:element name="EL1" type="xsd:EL1_type"/>
     ...
     </xsd:sequence>
     <xsd:attribute name="attr1" type="xsd:att1_type" use="required"/>
</xsd:complexType>
</xsd:element>
<xsd:element name=" CT2">
<xsd:complexType>
     <xsd:sequence>
     <xsd:element name="EL2" type="xsd:EL2_type"/>
     ...
     </xsd:sequence>
     <xsd:attribute name="attr2" type="xsd:att2_type" use="required"/>
</xsd:complexType>
</xsd:element>
```

```
<xsd:key name="CT1_K">
<xsd: selector xpath=".//E1"/>
<xsd: field xpath="attr1"/>
</xsd:key>
<xsd:key name="CT2_K">
<xsd: selector xpath="// CT2"/>
<xsd: field xpath="attr2"/>
</xsd:key>
< xsd: keyref name="CT1_CT2_Ref" refer=" CT1_K">
< xsd: selector xpath = "CT2"/>
< xsd: field xpath="@attr1"/>
</keyref>
< xsd: keyref name="CT2_CT1_Ref" refer="CT1_K">
< xsd: selector xpath = "CT1"/>
< xsd: field xpath="@attr1"/>
</keyref>
```

3.3. Enrichment of Semantics in the XML Schema

In this section, we will conceptually enrich XML Schema for the purpose of establishing correspondences between two technologies: XML and ODL. These connections allow us to specify mappings between the two schemas.

The semantic enrichment of an XML schema involves the extraction of its data semantics, to be enriched and converted into a CDM. To do this, we have applied the approach in [9] to enrich semantically XML schema. The process starts by extracting the basic metadata information about an existing XML schema, including relation types and attribute|element properties (*i.e.*, names, types, occurrence, required or not), and keys (K), keyrefs (KR). We assume that data dependencies are represented by keys and keyrefs. As for each keyref tag, there is a reference to a key of a complex type, which can be considered as a value reference.

We extend the semantics of the XML schema above as follows:

We add an element in both of the complex types (CT_1 and CT_2) that we called “elementrole” (ele_rol) whose name expresses the relationship role, its type is the same as the key element of the other complex type, and its cardinality is the same expressed in the relationship, e.g. we add in the element CT_2, an element “ele_rol” with the same cardinality near CT_2 (*) as:

```
<xsd:element name="ele_rol" type="xsd:attr1_type" maxoccurs="unbounded"/>.
```

4. Rules of Mapping from Enriched XML Schema to ODL Schema

Now we present the mapping rules between XML Schema elements and ODL, including concepts describing dependency relationships.

Rule 1: An XML element (<xsd: element>) with complex structure or a global complex type element (<xsd: complex Type>) are transformed into a class in ODL with the same name.

Rule 2: Simple XML elements <xsd: element>, with basic data types <xsd: type> data type (string, short, date, float, etc.), which is enclosed by an <xsd: sequence>, must be converted into attribute in the class ODL resulting from rule 1. XML attributes <xsd: attribute > are also converted into attribute in the corresponding class, with the same name and the same type, except the “elementrole”, it will be transformed in a relationship included in the corresponding class. XML attributes < xsd: minOccurs > and < xsd: maxOccurs > carried by the element can carry on associations (see rule 4).

Rule 3: Each field XPath in the element key is transformed into key attribute of the corresponding class.

Rule 4: Referring to the CDM, the relationship will be as follow:

```
Relationship set|bag|<CTname_referred_to> elan inverse CTname_referred_to:: fiels_xpath_of_
Ctname_referred_to.
```

Depending on the cardinality we add set or bag or nothing.

Definition of CDM: The CDM is defined as a set of complex types: CDM: = {CT|CT:= ⟨ctn, AEcdm, Rel⟩}, where each complex type CT has a name ctn. Each CT has a set of elements|attributes AEcdm, and a set of relationships Rel.

Attributes (A|Ecdm): A complex type CT has a set of elements|attributes AEcdm. AEcdm:= {ela|ela: = ⟨elan, t, tag⟩}, where each element | attribute *ela* has a name elan, data type t and a tag, which classifies ela as a non-key "NK", a key "K", or a relationship as R.

Relationships (Rel): A complex type CT has a set of relationships Rel. Each relationship rel ∈ Rel between CT_1 and complex type CT_2 is defined in CT_1 to represent an association. Rel: = {rel|rel: = ⟨CTn_referred_to, Occ, F_xpath_of_CTn_referred_to⟩}, where CTn_referred_to is the name of CT_2, Occ:= minOcurs. maxOccurs is the cardinality constraint of rel from the CT_1 side, and F_xpath_of_CTn_referred_to denotes the *elementrole* name representing the inverse relationship from the CT_2 side.

Since we are focusing on association relationship, we don't discuss the relation type.

5. Application Mapping of Association Relationship from an Enriched XML Schema to ODL

We presented in the previous section a specification of mappings of basic elements of XML schema. In this section, we will apply these mappings to XML Schema enriched. An association expresses a bidirectional semantic connection between two types. Each instance sharing a kind of relationship with others, it could be of any type as: one-to-one, one-to-many or many-to-many. By default, an association is navigable in both directions [10].

Association verbal active: specifies the reading direction of the main association; roles: specifies the function of a type for a given association; cardinality: specifies the number of instances that participate in a relationship.

5.1. One-to-One Association: (Rarely Applied in Practice)

In this section we use an example of (1:1) relationship between professor and class, we assume that each professor teaches at most one class and *vice-versa*. Keep in mind that this kind of relationship is not very common (see **Figure 2**).

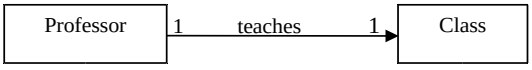

Figure 2. One-to-one association.

The steps below explain how to transform the one-to-one association relationship from enriched XML Schema to ODL.

Enriched XML Schema for one-to-one relationship:

```
<xsd :element name = "professor">
< xsd :complexType>
< xsd :attribute name = "professorId" type = "xsd :string" use = "required" />
< xsd :sequence>
....
< xsd :element name="teaches" type= " xsd :string" minOccurs="0" maxOccurs="1">
</ xsd :sequence>
</ xsd :complexType>
</ xsd :element>

< xsd :element name = "class">
< xsd :attribute name = "classId" type =" xsd :string" use = "required"/>
<xsd :sequence>
....
< xsd :element name="teachedby" type= " xsd :string" minOccurs="0" maxOccurs="1">
< /xsd :sequence>
```

```
</xsd :element>
<xsd:key name="professor_K">
<xsd:selector xpath=".//professor"/>
<xsd:field xpath="@professorId"/>
</xsd:key>
<xsd:key name="class_K">
<xsd:selector xpath=".//class"/>
<xsd:field xpath="@classId"/>
</xsd:key>
<xsd:keyref name="professorRefclass" refer="classK">
<xsd:selector xpath=".//professor"/>
<xsd:field xpath="teaches"/>
</xsd:keyref>
<xsd:keyref name="classRefprofessor" refer="professorK">
<xsd:selector xpath=".//class"/>
<xsd:field xpath="teachedby"/>
</xsd:keyref>
```

5.2. One-to-Many Association

Let's consider the example below: a department may have one or many employees. But an employee works in only one department (see **Figure 3**).

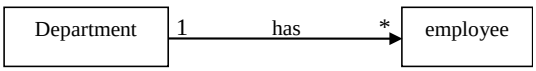

Figure 3. One-to-many Association.

Enriched XML Schema for one-to-many relationship:

```
<xsd:element name="Department">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="departmentId" type="xsd:string"/>
....
<xsd:element name="has" type="xsd:string" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:element name="employee">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="employeeid" type="xsd:string"/>
....
<xsd:element name="worksin" type="xsd:string" maxOccurs="1"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:key name="department_K">
<xsd:selector xpath=".//department"/>
<xsd:field xpath="departmentId"/>
</xsd:key>
<xsd:key name="employee_K">
 <xsd:selector xpath=".//employee"/>
```

```
 <xsd:field xpath="employeeid"/>
</xsd:key>
<xsd:keyref name="departmentRefemployee" refer="employee_K">
 <xsd:selector xpath=".//department"/>
 <xsd:field xpath="has"/>
</xsd:keyref>
<xsd:keyref name="employeeRefdepartment" refer="department_K">
 <xsd:selector xpath=".//employee"/>
 <xsd:field xpath="worksin"/>
</xsd:keyref>
```

5.3. Many-to-Many Relationship

In the example below: a student is teached by one or more professors. The same professor teaches lots of students (see **Figure 4**).

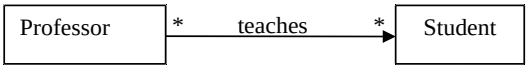

Figure 4. Many-to-many Association.

Enriched XML Schema for many-to-many relationship:

```
<xsd:element name="professor">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="professorId" type="xsd:string"/>
....
<xsd:element name="teaches" type="xsd:string" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:element name="student">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="studentId" type="xsd:string"/>
....
<xsd:element name="teachedby" type="xsd:string" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:key name="professor_K">
<xsd:selector xpath=".//professor "/>
<xsd:field xpath="professouId"/>
</xsd:key>
<xsd:key name=" student_K">
<xsd:selector xpath=".// student "/>
<xsd:field xpath="studentId"/>
</xsd:key>
<xsd:keyref name=" professorRefstudent " refer="student_K">
<xsd:selector xpath=".// professor "/>
<xsd:field xpath="teaches"/>
</xsd:keyref>
<xsd:keyref name=" studentRefprofessor " refer=" professor_K">
```

```
<xsd:selector xpath=".// student "/>
<xsd:field xpath="teachedby"/>
</xsd:keyref>
```

5.4. Generation Canonical Data Model (CDM) of XML Schema

Let's consider the XML schema shown in the example in Section 5.2 above, the corresponding CDM is as **Table 1**.

Table 1. CDM of XML schema for one-to-many association relationship.

CTn	ela			Rel		
	Elan	T	Tag	CTn_referred_to	Occ	F_xpath_of_CTn_referred_to
Department	departmentId	string	K			
	Name	string	NK			
	has	string	R	employee	0.. unbounded	worksin
Employee	employeeid	string	K			
	empName	string	NK			
	DOB	date	NK			
	worksin	string	R	Department	0..1	has

5.5. Algorithm for Schema Translation

Figure 5 shows the algorithm map XML_ODL for mapping XML schema into ODL schema. The algorithm reads each XML complex type one by one and maps it to ODL. In line 4, the complex type is mapped to a class, the algorithm maps all its elements to attributes of the class and forms the relationship with other classes. Specifically, if the relationship is one-to-one, in line 13 the algorithm adds the relationship in the corresponding class as follow:

```
Relationship < CTn_referred_to > elan inverse CTn_referred_to ::
F_xpath_of_CTn_referred_to;
```

If the relationship is one-to-many, in line 17 the algorithm add the relationship as:

```
Relationship set|bag < CTn_referred_to > elan inverse CTn_referred_to ::
F_xpath_of_CTn_referred_to;
```

Algorithm mapXML_ODL
Input:
cdm : CDM

```
1.  for (complex type CTn  in the CDM cdm  CTn∈cdm)
2.  {
3.  // map CTn into a class with the same name
4.  Procedure map_CT_Class(CTn)
5.  if ela.tag <> R  then
6.  {
7.  // map all elements | attribute to  attributes with the same name and data type in class CTn
8.  Procedure map_elan_t(ela.elan, ela.t)
9.  if ela.tag==K then
10. // Mention elan as a key
11. Procedure key(ela.elan)
12. }
13. Else  if Rel.Occ == 0..1
14. // map elementrole to relationship as one
15. //add relationships
16. Procedure add_rel_one  (Rel. CTn_referred_to, ela.elan,Rel. F_xpath_of_CTn_referred_to)
17. Else  Rel.Occ == unbounded  then
18. // map elementrole to relationship as many
19. //add relationships as set|bag
20. Procedure add_rel_may (Rel. CTn_referred_to, ela.elan,Rel. F_xpath_of_CTn_referred_to)
21. }
22. }
```

Figure 5. An algorithm for mapping XML schema into ODL.

The output ODL schema of one-to-many association relationship example is shown as follow (see **Figure 6**).

```
Class department
(extent departments key departmentId )
{ attribute string departmentId;
  attribute string Name;
  relationship set<employee> has inverse employee::worksin;
};
Class employee
(extent employees key employeeid )
{ attribute string employeeid;
attribute string empName;
attribute date DOB;
relationship <department> worksin inverse department::has;
};
```

Figure 6. Sample output ODL schema of one-to-many association relationship example.

6. Experimental Study

To demonstrate the validity of our method, a prototype has been developed, realizing the algorithm above. The algorithm was implemented using Java and EyeDB. To evaluate our approach, we examined the differences between source XML schema and the ODL schema generated by the prototype; we test the query results provided by OQL in EyeDB, and XQuery in stylus studios. Queries returned the same results. The source XML database is transformed into target object database ODL without loss of data.

This section presents two queries applied on the XML schema shown in Section 5.3 and the equivalent ODL generated by the prototype. **Table 2** shows the description and the result of each query.

Table 2. The description and the result of each query.

Description	OQL	XQuery	Result
List the departments and the number of employees in each department	Select D. Name, headCount: count (select E.employeeid from Employee E where E.worksin = D. departmentId) from Departments D, dept. has emp group by Name: D. Name;	for $dept in //Department let $headCount:= count(//Employee[worksin=$dept/ departmentId]) return <Department> {$deptartment/Name} <HeadCount>{$headCount}</HeadCount> </Department>	Accounting 2 Administration 3 Finance 2
Find the names of all employees in Accounting	Select Employee JOIN Department ON Employee. worksin = Department.departmentId WHERE Department.Name =' Accounting'	let $dept: = //Department [Name ='Accounting']/departmentId return//Employee[worksin = $dept]/empName	Smith Scott

7. Conclusion

In this article, we present a method of translation from XML schema into ODL schema, focusing on mapping association relationships; we extend the semantic of XML schema, our proposed method describes a process from the conceptual model to the implementation in the classes. With this method, the results preserve the semantics specified in the conceptual level, either to XML or ODL; our future work will be the development of a better mapping taking into account the concepts that we have not discussed in this paper, such as inheritance relationship.

References

[1] Alhajj, R. and Polat, F. (2001) Reengineering Relational Databases to Object-Oriented: Constructing the Class Hierarchy and Migrating the Data. *Proceedings of the* 8*th Working Conference on Reverse Engineering* (*WCRE*'01), Stuttgart, 2-5 October 2001, 335-344. http://dx.doi.org/10.1109/WCRE.2001.957840

[2] Cattell, R.G.G. and Barry, D.K., Eds. (2000) The Object Data Standard: ODMG 3.0. Morgan Kaufmann Publishers Inc., San Francisco.

[3] Xiao, R.G., Dillon, T.S., Chang, E. and Feng, L. (2001) Modeling and Transformation of Object-Oriented Conceptual Models into XML Schema. *Proceedings of* 12*th International Conference*, *DEXA* 2001, Munich, 3-5 September 2001, 795-804.

[4] de Sousa, A.F., Pereira, J.L. and Carvalho, J. (2002) From ODL Schemas to XML-SCHEMA Schemas: A First Set of Transformation Rules. *Proceedings of the Baltic Conference*, *Baltic DB&IS* 2002, Volume 1, 281-296.

[5] Widjaya, N.D., Taniar, D., Rahayu, J.W. and Pardede, E. (2003) Association Relationship Transformation of XML Schemas to Object-Relational Databases. *Revised Papers of* 3*rd International Workshop*, *IICS* 2003, Leipzig, 19-21 June 2003, 251-262.

[6] Naser, T., Alhajj, R. and Ridley, M.J. (2009) Two-Way Mapping between Object-Oriented Databases and XML. *Informatica*, **33**, 297-308.

[7] SYSRA (2007) EyeDB Object Definition Language Version 2.8.8. SYSRA, Yerres.

[8] Berglund, A., Boag, S., Chamberlin, D., Fernndez, M.F., Kay, M., Robie, J. and Simon, J. (2007) XML Path Language (XPath) 2.0 W3C Recommendation 23 January 2007. http://www.w3.org/TR/xpath20/

[9] Maatuk, A., Akhtar Ali, M. and Rossiter, N. (2010) Converting Relational Databases into Object-relational Databases. *Journal of Object Technology*, **9**, 145-161. http://www.jot.fm/issues/issue_2010_03/article3.pdf

[10] Hoffer, J.A., Prescott, M.B. and McFadden, F.R. (2005) Modern Database Management. 7th Edition, Prentice Hall, Upper Saddle River.

Scientific Research Publishing (SCIRP) is one of the largest Open Access journal publishers. It is currently publishing more than 200 open access, online, peer-reviewed journals covering a wide range of academic disciplines. SCIRP serves the worldwide academic communities and contributes to the progress and application of science with its publication.

Other selected journals from SCIRP are listed as below. Submit your manuscript to us via either submit@scirp.org or Online Submission Portal.