From XML Schema to ODL Schema: Aggregation and Composition transformation

Doha Malki[1], Mohamed Bahaj[2] and Ilyas Cherti[3]

[1] Department of Mathematics and Computer Science, University Hassan 1st, Faculty of Sciences and Technology Settat, 2600, Morocco

[2] Department of Mathematics and Computer Science, University Hassan 1st, Faculty of Sciences and Technology Settat, 2600, Morocco

[3] Department of Mathematics and Computer Science, University Hassan 1st, Faculty of Sciences and Technology Settat, 2600, Morocco

Abstract

This paper presents an approach for transforming an existing XML schema in ODL (Object Definition Language) schemas, We chose oriented object database as a target database because there are many common characteristics between XML and object-oriented model, and we desire to have a large number of object-oriented features, (e.g. user-defined data types, inheritance, substitutability, etc.) and a permanent storage of data. 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, they offer an evolutionary approach, so we agree that it is time to develop a translation between XML and OO databases.
Our work is focused on preserving Semantics transformation of the aggregation and composition relationships, we describe set of rules and pseudo code has been developed to create ODL classes from existing XML schema, the experimental show 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; aggregation; composition; OODB;*

1. Introduction

Recently XML is became the most dominant standard used as new format of representing and exchanging data on the world, it is able to run in the database, this increasing use of XML technology implies an essential requirement for managing XML documents and retrieving data , storing XML data in object oriented databases (OODB) seems a solution, this implies the need to describe the schemas written in XML in the Object Oriented schemas without disfiguring the structure as well as semantic constraints from the source to the target database.

We believe that, Object Oriented databases are more and more accepted. The Object Database Management Group (ODMG) standard, has become more mature [1], so we can assume that the object-oriented DBMS (OO DBMS) are willing to store XML data, the XML document must be structured, and once the XML data is stored, we can query the database.

We chose the object model proposed by Object Database Management Group (ODMG) , we will use the language of object definition (ODL) to define and query the target database.

The objective of this work is transforming XML Schema structures to OODB schema based on the ODMG 3.0 standard (Cattell et al., 2000), and preserving the structure as well as semantic constraints of the source XML schema in the target OO schema and to take the strong points of OO features, focusing on aggregation and composition relationships .

1.1 Related Work

There are several researches to map an XML schema to object relational database, in contrast, the mapping of XML schemas in object-oriented database, has not received much attention.

[2] Describes an XML storage system done for an OO/OR DBMS. The work proposes an algorithm for mapping and storing XML documents in an OO/OR database. But it does not discuss the mapping of different types of relationships notably aggregation and composition relationships.

[3] Proposes the mapping of the OO Conceptual Model into the XML Schema. This work has included collection for aggregation relationship.

[4] Addresses 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.

Number of transformation steps from the XML schema to the ORDB are describing in [5], to preserve the collection.
The conversion of Relational to E/R to XML is described In [6], as the mapping from XML to object-oriented databases is concerned, which describes the reconstruction of the semantic model, in the form of ER model from the logical schema, then the conversion of XML document. However, many-to-many (M: N) relationships and servants are not considered properly,

The work described in [7], is about the mapping from XML to OODB, generates an object oriented database schema from DTDs, stores it into the object-oriented database and processes XML queries; it mainly concentrate on representing the semi-structural part of XML data by inheritance.

[8] Describes rules of transforming a simple ODL database schema into an XML schema, but relationships are not defined.

This document aims to define rules to transform an XML Schema and ODL Schema, focusing on aggregation relationship.

The rest of the paper is organized as follows: background and needful terminology is presented in Section 2, Section 3 describes how existing XML schemas can be transformed in a ODL schemas, and Section 4 concludes the paper.

2. Background and Needful Terminology

2.1 XML Schema: A brief review

Extensible Markup Language is a meta-markup language made up of a set of tags to define and describe the contextual meaning of data [9,10,11]. XML Schema is an XML-based alternative to DTD, it describes the structure of an XML document. The XML Schema language is also referred to as XML Schema Definition (XSD), It is written in XML and offers several important elements including : xsd:element, xsd:attribute, xsd:complexType,….
An XML Schema defines:

- The element: xsd:element is used for defining an element. the cardinality of an element is explicated by “minOccurs” and “maxOccurs”[2].
- The element xsd:attribute is used for defining an attribute. [2].
- The element xsd:complexType is used for defining the type of an element having subelements or attributes[2].

```
<xsd:element name="CT">
<xsd:complexType>
   <xsd:sequence>
   <xsd:element name="EL1" type="xsd:EL1_type"/>
   ...
   </xsd:sequence>
   <xsd:attribute name="attr1" type="xsd:att_type"
use="required"/>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

2.2 Object Definition Language (ODL) Characteristics

The Object Definition Language is a specification language used to define the specifications of object types that conform to the ODMG Object Model [12]. ODL is used to support all semantic constructs of the ODMG Object Model. ODL is not intended to be a full programming language. It is a definition language for object specification. Relational Database management systems have traditionally provided facilities that support data definition through Data Definition Language--DDL) and Data Manipulation (through Data Manipulation Language--DML). ODL defines only the signatures of operations and does not address the definition of methods that implement those operations [12].

A simple example of the class definition of CL is (keywords in bold):

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

- **Extent** is used to define all instanced objects for the interface;
- **Key** is used to specify the attribute or attributes whose values uniquely identify an instance of a class;
- **Relationship** is used to specify a relationship between two classes, names of relationships handle roles.

2.3 Aggregation and Composition: A brief review

An Aggregation relationship is a binary association specifying a whole-part relationship [13], it is an asymmetric association, which expresses a strong coupling and a relationship of subordination. At the same time, an associate member instance can be related to multiple instances of other classes (the associate element can be shared) in cases where there’s a part-of relationship between Complex type CT1 (whole) and Complex type CT2 (part) it doesn’t imply that CT1 owns CT2 or that there is a parent-child relationship between the two.

A composition relationship is strong aggregation, at the same time, a component instance can be bound to a single aggregate, "Composite objects" are instances of classes composed.(CT1 owns CT2)

- Aggregation: Since the “part” complex type can be used inside another “whole” complex type (aggregation : shared association) both the “whole” and the “part” are defined as complex types, inside the “whole” complex type, we define an element of the “part” complex type, with maxoccurs constraint[5].
- Composition: the “part” complex type is used inside one “whole” complex type, at the same time (composition: non shareable association). The “part” component is defined as a complex type inside the “Whole” type element, to prevent another complex type to use the particular “part”, and to be sure that any aggregated complex type is destroyed when the composite is destroyed [5].

Table -1: general syntax of aggregation refered to [5]

<**xsd:complexType** name = " PART_Type" > ... </xsd:complexType> <**xsd:complexType** name = " WHOLE_Type" > <xsd:element name = "PART_Name" type = "**xsd:PART_Type**" **maxOccurs= _ unbounded_** /> ... </xsd:complexType>

Table -2: general syntax of composition refered to [5]

<xsd:complexType name = " WHOLE_Type" > ... <xsd:element name = " PART_Name" maxOccurs="unbounded"/>> <xsd:complexType> ... </xsd:complexType> </xsd:complexType>

3 Transforming Aggregation from XML Schema into ODL Classes

The XML schema to ODL schema conversion implies mapping all the XML elements/attributes, relationships into their corresponding in ODL classes.
In this section, we present set of rules to transform an existing XML schema with aggregation | composition relationships to ODL schema;

3.1. Rules of transformation:

Rule1: the root element is transformed into the name of database.
Rule 2: the element that is included in an anonymous Complex Type, with the choice element, is transformed into a higher level class in the ODL schema with the same name.

Rule 3: elements with built-in data types; which are included in the sequence element; are translated into attributes of the class results from rule2, with the same type (string, decimal, integer, Boolean, date...) otherwise we can specify another data type.

Rule 4: elements whose types are complex types; which are not included in the sequence element and with maxoccurs constraint; are transformed into an attribute in the class of top level outcome from rule 2.

Aggregation is mapped onto an attribute in the aggregating class in ODL[16] with single valued (if 0..1 or 1..1) or collection valued (if 0..* or 1..*).the collection types allowed in ODL are: SET, BAG, LIST, and ARRAY) [17]:
Rule 5: mapping aggregation depends on the multiplicity that the "part" complex type CT2 participate in the relationship[14].

- if "maxoccurs =1", means the "part" complex type participate with 1 occurrence, then "part CT2" is included as attribute on the corresponding class to represent the relationship: **C= (CT_2, X',Y', (CT_1)).**
- When "maxoccurs =m" (m is known), implies that CT_2 participate in the aggregation with "m" occurrences , then the aggregation is mapped to an array of CT2. **C= (CT_1, X,Y, array((CT_2),m)) .**
- When "maxoccurs = unbounded", adequate transformation is to define a bag or set of "part CT_2" to the other class in the relationship, **C=(CT_1,X,Y, bag, set(CT_2))** .

Rule 6: transforming composition:

The "part" complex type is transformed into a structure in the class "whole",

- If the "part" component of the composition is single, we can use the single row, The inner complex type in the XML Schema is
  also mapped as an attribute of the class as struct datatype, as follow: **attribute Struct CT_2 {$type_1$ att_1, $type_2$ att_2} ct_2;**
- If the "whole" component can have more than one "part" component of the same type, we use set-valued attributes, we map the outer complex type as class in ODL and the inner complex type as the row attribute. To preserve the multiple feature we implement the row as a collection with this syntax : **attribute Set|Array<Struct CT_2 {type1 att_1, $type_2$ att_2}> ct_2;**

3.2. Aggregation XML schema:

```
<xsd:schema
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="Customerassociation"/>
 <xsd:complexType>
          <xsd:sequence>
           <xsd:element name="identification"
           type="xsd:string"/>
           <xsd:element name="description"
           type="xsd:string" />
           <xsd:element maxOccurs="15" name="Customer"
           type="Customer_type" />
          </xsd:sequence>
 </xsd:complexType>
 <xsd:element name="Customer"/>
 <xsd:complexType>
          <xsd:sequence>
           <xsd:element name="customerName"
           type="xsd:string"/>
           <xsd:element name="phone" type="xsd:integer" />
          </xsd:sequence>
          <xsd:attribute name="customerId"
type="xsd:integer" use="required" />
 </xsd:complexType>
<key name=" Customer_PK " >
<xsd:selector xpath=".//Customer"/>
<xsd:field xpath="@customerId"/>
```

```
</key>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

3.3. Composition XML schema :

```
<xsd:schema
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="Purchase_order"/>
<xsd:complexType>
        <xsd:sequence>
        <xsd:element name = "shipping" type
        ="xsd:string"/>
        <xsd:element name = "toCity" type ="xsd:string"/>
        <xsd:element name = "toStreet" type
        ="xsd:string"/>
        <xsd:element name = "toZip" type ="xsd:integer"/>
        <xsd:element name = "Orderlineitem"
        maxOccurs="unbounded"/>
        <xsd:complexType>
        <xsd:attribute name = "line" type = "xsd:integer"
        use = "required"/>
        <xsd:element name = "quantity" type
        ="xsd:integer"/>
        </xsd:complexType>
        </xsd:sequence>
        <xsd:attribute name = "order" type ="xsd:integer"
        use="required"/>
</xsd:complexType>
<key   name=" Pusrchase_order_PK">
 <xsd:selector xpath=".// Pusrchase_order "/>
 <xsd:field xpath="@order"/>
</key>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

Related to the XML schema, we summarize the complex type informations in the following table: (see table4)

AE_XML*(complextype(CTn), element/attribute(AE), type, occurrence(OCC), key, relationship(rel)p, CT_2(dirCT)).*
Where : CT_n: name of complex type; AE: set of attributes and elements of CT_n; type: is the type of AE; occ: the multiplicity that CT_2 participate in relationship with CT1, rel: type of relationship between CT_1 and CT_2, dirCT is the name of CT_2

Table4: AE_XML : a list of all elements and attributes in the xml schema aggregation and composition

Complextype (CT_n)	***Attribute/Element (AE)***	***Type***	***Occurrence (occ)***	***key***	***Relationship (rel)***	***CT2 (dirCT)***
Customerassociation	identification description Customer	string string Customer	1..15		AG	Customer
Customer	customerId customerName phone	integer string integer		PK		
Puschase_order	order shipping toCity toStreet toZip Orderlineitem	integer string string string integer Orderlineitem	unbounded	PK	CM	Orderlineitem
Orderlineitem	line quantity	integer integer		PK		

3.4 Algorithm for transforming XML schema in an ODL schema

```
Algorithm XML_ODL (ae_xml: AE_XML) return
ODLschema
Foreach complextype CT in AE_XML  do
        Foreach relationship rel in AE_XML do
        If rel= 'AG' then
        // create Class dirCT
        Procedure create_class(dirCT)
        // create Class CT
        Procedure create_class(CT)
                If occ= unbounded  then
                // Add an attribute dirCT as a set or bag
                addAttribute dirCT SET|BAG <dirCT>;
                Else if occ= m then
                // Add an attribute dirCT as a array[m]
                addAttribute dirCT ARRAY (dirCT, m);
                If occ= 1 then
                // Add an dirCT as an attribute
                        addAttribute (dirCT)
                        end if
        Else if rel = 'CM' then
        // create Class CT
        Procedure create_class(CT)
                If occ=1 then
                //add dirCT as struct in CT
                addStruct direct
                else
        //add dirCT as set|array (struct) in CT
        Add set|array struct (dirCT)
        End if
End Foreach
End Foreach
End Algorithm
```

Thus, if the transformation rules and the algorithm described above for the previous XML schema are applied, we should get the ODL Schema shown in Table 5,6

Table 5: The ODL classes corresponding to the XML schema aggregation

```
class  Customer{
( extent Customers key customer_id)
attribute integer customer_id;
attribute string customer_name;
attribute string phone ;
}
class  Customer_assoc{
( extent Customer_assocs)
attribute string  Identification;
attribute string Description;
attribute array <Customer 15>  Customer;
}
```

Table 6:The ODL classes corresponding to the XML schema composition

```
class  Puschase_order {
( extent Puschase_orders key order)
attribute integer order;
attribute string shipping;
attribute string toCity ;
attribute string toStreet ;
attribute integer toZip ;
attribute     strust     set<Orderlineitem
{integer line; integer quantity}> orlnitem
;
}
```

4. Experimental Study

To evaluate our approach we test the query results provided by OQL in eyedb , and XQUERY in stylus studios. Table 7 shows the description of queries, queries return the same results. The source XML database is transformed into target Object database ODL without loss of data.

Table 7 : Results of the queries

Description	OQL	Xquery	Result
Find the name of all Customers of the customer_association Identified by "ASS1" Ordered by name of customer	Select customer: c.name, From customer_associations ca, ca.customers c Where ca.identification = "ASS1" Order by c.name asc;	for $ca in doc('customer.xml')/NewDataSet/Customerassociation , $id in $ca/identification , $c in $ca/Customer where $ca/identification='ASS1' order by $c/customerName return <customer> {$c} </customer>	12 Dupont 147852369 10 Scott 123456789 11 Smith 987654321
The first customer name of the customer_association identified by "ASS1"	Fisrt(select(customer:c.name) From customer_ associations ca, ca.customers c Where ca.identification = "ASS1"	for $ca in doc('customer.xml')/NewDataSet/Customerassociation, $id in $ca/identification , $c in $ca/Customer[1] where $ca/identification='ASS1' return <customer> {$c} </customer>	10 Scott 123456789
Compute the number of all Customer of customer_association	Select Cust_ASS : ca.identification, number: count(c.name) from customer_associations ca, ca.customers c Group by ca.identification	for $x in doc('customer.xml')/NewDataSet/Customerassociation return {$x/identification } {number=count($x/Customer)}	ASS1 3 ASS2 2 ASS3 1
Find the identification of all Customer associations	Select customer_associations: ca.identification, From customer_associations ca	for $b in doc("customer.xml")//Customerassociation, $id in $b/identification return <result> {$id} </result>	ASS1 ASS2 ASS3

5. CONCLUSIONS

In this article, we described a translation from XML schema into ODL schema, focusing on mapping aggregation|composition relationships; we have shown that the semantic in the aggregation|composition of XML data using can be preserved in the implementation using ODL. We have a prototype to realize the solution, and we have evaluated it by comparing query results the results of queries are the same.

We proposed the use of collection types allows by the ODL for aggregation|composition, in the mapping of the aggregation|composition we distinguish different cases according to the multiplicity of "part" participating on the relationship.

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 on developing a better mapping taking into account the definition ODL (OBJECT DEFINITION LANGUAGE), allowing to establish correspondences between concepts that were not taken into account such as relationship one_to_many, many_to_many, inheritance, besides the addition of these concepts allows to specifically identifies semantic links between elements and to provide information regarding the life cycle thereof.

REFERENCES

[1] Cattel, R. G. G. and Barry, D., "The Object Data Standard: ODMG 3.0, Morgan Kaufmann", 2000.
[2] Woo-Shin Han, Ki-Hoon Lee, Byung Suk Lee: "An XML Storage System for Object-Oriented /Object-Relational DBMSs", in Journal of Object Technology, vol. 2, no. 3, May-June 2003, pp. 113-126
[3] Xiou, R., Dillon, T.S., Chang, E., and, Feng, L. "Modeling and Transformation of Object-Oriented Conceptual Models into XML Schema", DEXA 2001, Springer-Verlag, 2001, 795-804.
[4] Taher Naser, Reda Alhajj, Mick J. Ridley "Two-Way Mapping between Object-Oriented Databases and XML". Informatica 33 (2009) 297–308.
[5] Eric Pardede, J.Wenny Rahayu, David Taniar "On Using Collection for Aggregation and Association Relationships in XML Object-Relational Storage", SAC'04 March 14-17, 2004, Nicosia, Cyprus.
[6] J. Fong, F. Pang, and C. Bloor, "Converting Relational Database into XML Document," Proc. of the International Workshop on Electronic Business Hubs, pp61-65, Sep. 2001.
[7] T.-S. Chung, S. Park, S.-Y. Han, and H.-J. Kim. "Extracting Object-Oriented Database Schemas from XML DTDs Using Inheritance," Proc. of the International Conference on Electronic Commerce and Web Technologies, pp.49-59, 2001.
[8] Artur Afonso de Sousa; José Luís Pereira, and João Carvalho, "From ODL Schemas to XML-SCHEMA Schemas: A First Set of Transformation Rules", Proc. of the XXII International Conference of the Chilean Computer Science Society (SCCC'02),
[9] Biron, P. and Malhotra, A.: "XML Schema Part 2: Datatypes", May 2001 (available from `http://www .w3.org/ TR/ xmlschema-2`).
[10]Fallside, D.: XML Schema Part 0: Primer, May 2001 (available from http://www.w3.org/TR/xmlschema-0).
[11]Thompson, H. et al.: XML Schema Part 1: Structures, May 2001 (available from http://www.w3.org/TR/ xmlschema-1).
[12]Roderic Geoffrey Galton Cattell,Douglas K. Barry, "The Object Data Standard: ODMG 3.0"
[13]UML Semantics version 1.1, available from Rational's web-site: http://www.rational.com/uml , September 1997.
[14]M.F. Golobisky, A. Vecchietti/Proc. of Argentine Symposium on Software Engineering (2005) 65-79 65, "Mapping UML Class Diagrams into Object-RelationalSchemas",
[15]M. Naci Akkøk "From ODL/OO-DBMS Design to Relational Designs & Object-Relational Database Systems (ORDBS)" 25/2-2003, 20/2-2004 and 14/2-2005.

Doha Malki she is phd student in the Department of Mathematics and computer sciences, Faculty of Sciences & Technologies of Settat, University Hassan 1st, Settat, Morocco. her area of interest includes Databases and semantic web.

Mohamed Bahaj He is a full Professor in Department of Mathematics and Computer Sciences from the University Hassan 1st Faculty of Sciences & Technology Settat Morocco. He is co chairs of International Conference on Software Engineering, Databases and Expert Systems (SEDEXS'12) , NASCASE'11. He has published over 60 peer-reviewed papers. His research interests are intelligents systems, Ontologies Engineering, Partial and differential equations, Numerical Analysis and scientific computing.

Ilyas Cherti He is a full Professor in Department of Mathematics and Computer Sciences from the University Hassan 1st Faculty of Sciences & Technology Settat Morocco. His research interests are intelligents systems, partial and differential equations. He has organized various national and international events.