Journal of Theoretical and Applied Information Technology
20th November 2015. Vol.81. No.2
© 2005 - 2015 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

RDB2XSD: AUTOMATIC SCHEMA MAPPING FROM RDB INTO XML

[1]LARBI ALAOUI, [2]OUSSAMA EL HAJJAMY, [3]MOHAMED BAHAJ
[1]International University of Rabat, 11100 Sala Al Jadida, Morocco
[2, 3]University Hassan I, FSTS Settat, Morocco
E-mail: [1]larbi.alaoui@hotmail.de, [2]elhajjamyoussama@gmail.com, [3]mohamedbahaj@gmail.com

ABSTRACT

Extensible Markup Language (XML) is nowadays one of the most important standard media used for exchanging data on the internet. Massive data is still however treated, transferred and stored using relational database systems (RDBs). Therefore, there is a need for an integrated method that deals with database migration from RDB schema to XML schema. In this paper we provide and develop a new solution called RDB2XSD that migrates the conceptual schema of RDB into XSD through a MA (multidimensional array) model. This solution takes an existing RDB as input and extracts its metadata with as much constraints as possible, creates the MA model to capture the semantics of the relational database and applies our mapping algorithm to generate the hierarchical XSD schema. For the implementation of our approach we developed a tool based on java and tested it using Oracle and Mysql databases. Our experimental results based on this tool show that our mapping strategy is feasible and efficient.

Keywords: *XML, XSD, Relational Database RDB, Schema Conversion, MA Model*

1. INTRODUCTION

The use of XML schema is rapidly increasing to represent structured and unstructured data in the Internet. However, very large volumes of data are always stocked in relational databases. So, in order to exchange data between relational database (RDB) and XML, a translation algorithm is necessary.

Currently, there are two options recommended by theW3C for defining an XML schema. One is the Document Type Definition (DTD) and the other is the XML Schema (XSD). We choose XML Schema because:

- it has a powerful set of types and constraints which leads to a better translation;
- it provides us with a more flexible and powerful mechanism through "key" and "keyref" constructs;
- and with XSD we are able to model composite, multi-valued attributes and complex cardinality constraints.

Our aim in this paper is to tackle the problem of translation of relational database schema models to XML schema models. As it is detailed in section 2 the existing works in this sense do not provide a complete solution, and so far there still be no effective proposals that could be considered as a standard method that preserves the whole original structure and constraints of the relational database.

For a complete and efficient translation we provide a mapping strategy that takes into account several issues in order to preserve all details related to the relational structure of a relational database so that all its static and semantic information will be reflected by the resulting XML schema. In order to achieve such a complete mapping our approach first extracts the metadata of the considered database, generates the multi dimensional model (MA model) to capture the semantics of the source RDB and apply our algorithm to build the XML structure. Our mapping algorithm uses a set of transformation rules that we give according to a categorization of the types of relations and the types of the constraints we are dealing with in a relational database following some ideas we gave in our previous works [1-2] that are related to mapping RDB to OWL (Ontology Web Language). To validate our solution we have developed a prototype that implements this algorithm and tested its effectiveness using concrete examples.

The rest of the paper is organized as follows. In section 2 we review the existing RDB to XML transformation works. Needful terminology and several rules to convert a relational database

© 2005 - 2015 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

schema into XML schema along with a corresponding categorization of the RDB relations and attributes are given in Section 3. Section 4 discusses the methods for extracting semantics using the MA model and provides our mapping algorithm based on the list of rules. It also presents a result of the performance test of our developed mapping tool. Section 5 concludes this paper.

2. RELATED WORK

The conversion from relational database to XML has recently received significant attention and become an active research domain. Various algorithms have been developed to reflect information about relational database using transformations into XML documents.

The first works associating RDB with XML were either XML views based or DTDs based. The XML views based methods consisted in presenting XML views of RDB data without providing any schema for the structure of such views. Users should therefore have a better knowledge of what the obtained views represent, in order to be able to query such views. Among these works we cite Silkroute in [6] that aims at publishing relational data as XML views using a transformation language RXL. Users can then issue queries against these views. In this sense we can also mention the works in [3-4], [15], [18-19] and [20].

The DTDs based methods dealt with the mapping of RDB schemas into DTD schemas providing the users with conceptual structures of the considered RDB relations. These were the starting point for the upcoming transformations that map RDB schemas into XSD schemas. It is however preferable for the reasons mentioned in the previous section to have an XSD schema representation of the RDB schema rather than a DTD one.
Among the RDB to DTD transformations we cite the work in [11] that mainly considers the static constraints on attributes and do not handle the functional dependencies between relations. Another RDB to DTD mapping technique is given in [8] and consists in joining normalized relations into tables that are mapped into DOMs that are then integrated into a user specified XML document trees which are converted into XML DTDs. As DTD-based translation algorithms we also mention Nesting-based Translation (NeT) and Constraints-based Translation (CoT) algorithms [12-14]. However, NeT does not support any referential integrity constraint. COT considers the structural part of RDB schema such as cardinality and only a restrained semantic part such as foreign key constraint. In [17] an algorithm NeT-FD is also proposed for an RDB to DTD mapping that takes into account the functional dependencies and keys.

To come up with solutions to the limitations of DTD some mapping techniques have appeared to transform RDB schemas into XSD (XML schema). The work in |7] gave an approach in this sense that does not handles semantic details and uses a transformation into Extended Entity Relationship model and an XSD graph as intermediary steps. In [10] VP-T (Values Pattern-based translation) and QP-T (Query Pattern-based translation) algorithms have been proposed to resolve the problem of CoT. However, both have a critical restriction that cannot extract a semantic relationship between column titles. Another approach is given in [5] but uses intermediary adjacency matrix and oriented graph. In the same sense transformation approaches were proposed in [16] and [21] but they do not handle all details and they respectively use a reference graph and an ER model as intermediary steps. Also a so-called holistic transformation algorithm is proposed in [22] to transform relational database into a nested XML schema without building a reference graph. This solution classifies relations into three categories (base relation, single related relation and multi related relation according to the number of foreign keys in the relation tables) and gives transformation rules to map these categories. However, marking dominant relations for circular relations and dominant participant relations for multi-related relations based on queried data can provide different XML schema results when an update of the data is performed on the source relational database. Therefore, this solution cannot guarantee an exact XML document creation. Another technique was presented in [9] where the authors consider the case where referential integrity constraints are not included in the RDB schema due the designer's fault or old and poor documentation and extract such constraints from users' queries.

All the aforementioned XSD based transformation present limitations in treating various important RDB elements related to the art of either relations or attributes such as composite keys, composite foreign keys, self referenced relations and cyclic relations. In the following section we give a more concise and complete categorization of RDB relations that reflects all associated static and semantic details. This categorization will be the basis of our mapping algorithm we are presenting in section 4. We

© 2005 - 2015 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

assume that all relations in the RDB schema are at least in 3NF.

3. RDB TO XML SCHEMA MAPPING RULES

In this section we give a complete list of rules for building the XML schema from the RDB source. To this end we consider relevant categorizations related to the various constraints in a relational database. The first categorization aims at classifying the relations in the database into four categories based on different types of the foreign keys. This classification is as follows:

- NormalRel(R): R is a relation with no foreign keys;
- PKAndFKRel(R): the primary key of R also acts as a foreign key;
- OneFKRel(R): R is a relation with one foreign key;
- MoreThanOneFKRel(R): R is a relation with more than one foreign key.

Then, to preserve the semantics contained in the database source we take into account all the integrity constraints, such as primary keys, foreign keys, not null and unique characteristics:

- NormalAttr(A, R): A is an attribute in relation R that is not part of a primary or foreign key and that is not declared as unique or as not null;
- PK(x, R): x is a single or composite primary key of the relation R;
- FK(x, R, y, S): x is a single or composite foreign key in relation R that references y in relation S;
- Unique(x, R): x is declared as a unique attribute;
- NotNull(x, R): x is declared as not null attribute.

Finally we capture all circular relations in the database source and find a way to convert it to a hierarchical XML schema. Circular relationships are divided into two categories:

- SelfRefRelation(R): the relation R has a foreign key x referencing itself and we denote it by SelfRefAttribute(R, x).
- CyclicRel: A cyclic relation is defined as a set of relations R_1, ..., R_n ($n > 1$), where R_i is referenced by R_{i+1} ($1 \leq i \leq n$) and Rn is referenced by R_1.

With all these categorizations we are now ready to give the associated mapping rules.

3.1. Mapping relations

Based on the categorization of relations we gave in the previous section with respect to their types and to the various related constraints we are now able to list in the following the associated mapping rules for the transformation of RDB schema into XML schema.

Rule 1. XML schema root element: To ensure the XML Schema has a single root, we need to prepare our XSD schema by creating a root element. The root element is created using the name of the database source.

```
<xsd:schema
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="targetNamespaceURI"
xmlns="targetNamespaceURI"
elementFormDefault="qualified">
 <xsd:element name="DatabaseName">
  <xsd:complexType>
   <xsd:sequence>
    <!--mapped relational schema is here -->
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
</xsd>
```

Rule 2. NormalRel(R): For every normal relation R, we create an element named R as a child of the root element.

```
<xsd:element name="R">
 <xsd:complexType>
  <!-- details of attributes in R -->
 </xsd:complexType>
</xsd:element>
```

Rule 3. OneFKRel(R): If a relation S with one foreign key references another relation R, then the generated element from S must be a sub-element of the generated element from R. In this case we have a 0:n relationship. So we add the minOccur="0" and the maxOccurs="unbounded" constraints to the relation S (maxOccurs = "unbounded" indicates the element S may appear more than once).

```
<xsd:element name="R">
 <xsd:complexType>
  <xsd:sequence>
   < xsd:element name="S" minOccurs="0"
      maxOccurs="unbounded"/>
  <xsd:sequence>
 </xsd:complexType>
</xsd:element>
```

Rule 4. PKAndFKRel(R): If the primary key of a relation S is at the same time a foreign key that is

© 2005 - 2015 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

referencing a field in another relation R, then the generated element from S must be a sub-element of the generated element from R. In this case we have a 0:1 relationship, so we add the minOccur="0" and the maxOccurs="1" to relation S.

```
<xsd:element name="R">
 <xsd:complexType>
  <xsd:sequence>
   < xsd:element name="S" minOccurs="0"
       maxOccurs="1"/>
   <xsd:sequence>
 </xsd:complexType>
</xsd:element>
```

Rule 5. MoreThanOneFKRel(R): If a relation R with more than one foreign key and reference R_1… R_n relations with (n>1), then, to preserve the integrity constraints, we add "keyRef" element for each foreign key attribute in R as follow:

```
<xsd:element name="R">
 <xsd:complexType>
  <xsd:attribute name="FK1" type="xsd:TypeOfFK1" />
   ……...
  <xsd:attribute name="FKn" type="xsd:TypeOfFKn" />
 </xsd:complexType>
</xsd:element>

<xsd:keyref name="R_Ref_R1" refer="R1_y1">
  <xsd:selector xpath="R"/>
  <xsd:field xpath="FK1"/>
</xsd:keyref>
  ……
<xsd:keyref name="R_Ref_Rn" refer="Rn_yn">
  <xsd:selector xpath="R"/>
  <xsd:field xpath="FKn"/>
</xsd:keyref>
```

3.2. Mapping attributes

Rule 6. NormalAttr(x, R): For each normal attribute x in relation R, we create an "attribute" element with the XSD type corresponding to the type of the field in the RDB.

```
<xsd:attribute name="x" type="xsd:TypeOfx" />
```

Rule 7. PK(x, R): A primary key is transformed into a "key" element with a selector to select the XPath of its relation.
To ensure the uniqueness of the key element name we propose to give for each of them a name obtained by concatenating the name of the Relation and the primary key value corresponding to the converted record.

```
<xsd:key name="R_x">
  <xsd:selector xpath="R"/>
  <xsd:field xpath="x"/>
</xsd:key>
```

Rule 8. FK(x, R, y, S): To capture the reference relationship between two relations, a foreign key is converted to a "keyRef" element. Note that the foreign key of a OneFKRel(R) is not converted to a "keyRef" element.

```
<xsd:keyref name="R_Ref_S" refer="S_y">
  <xsd:selector xpath="R"/>
  <xsd:field xpath="x"/>
</xsd:keyref>
```

3.3. Mapping Constraints

Rule 9. Unique(x, R): For each attribute declared as UNIQUE we create a "unique" element with a selector to select the XPath of the element and a field to specify the attribute that must be unique.

```
<xsd:unique name="UniqueR">
  <xsd:selector xpath="R"/>
  <xsd:field xpath="x"/>
</xsd:unique>
```

Rule 10. NotNull(x, R): If the attribute is declared as NOT NULL, we add use="required" into the mapped attribute element.

```
<xsd:attribute name="x" type="xsd:TypeOfx"
    use="required"/>
```

Rule 11. For attributes with the special constraints Length, CHECK VALUES or CHECK IN we treat them as follows:

- To limit the length of a value in an attribute we can use the xsd:maxLength.

```
<xsd:simpleType name="LimitedString">
  <xsd:restriction base="xsd:string">
    <xsd:maxLength value="100" />
  </xsd:restriction>
</xsd:simpleType>
<!--So we declare the attribute as follows-->
<xsd:attribute name="A"  type="LimitedString" />
```

- CHECK VALUE x: denotes all values that x can take. In this case we use the facets xsd:minInclusive, xsd:maxInclusive, xsd:minExclusive or xsd:maxExclusive.

```
<xsd:element name="CheckValue">
 <xsd:simpleType>
  <xsd:restriction base="xsd:integer">
   <xsd:minInclusive value="0"/>
   <xsd:maxInclusive value="120"/>
```

© 2005 - 2015 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

```
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
<!--So we declare the attribute as follows-->
<xsd:attribute name="A"  type="CheckValue " />
```

- CHECK IN constraint on a column allows only certain values for this column: In this case we use the facet xsd:enumeration.

```
<xsd:simpleType name="CheckIn">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="A1"/>
    <xsd:enumeration value="A2"/>
     --------------------------
    <xsd:enumeration value="An"/>
  </xsd:restriction>
</xsd:simpleType>
<!--So we declare the attribute as follows-->
<xsd:attribute name="A"  type="CheckIn" />
```

3.4. Mapping Circular Relations

Rule 12. SelfRefRelation(R): In this case we consider the SelfRefAttribute(R, x) as a normal attribute, we apply the relation mapping rules to convert R (rule 1-5) and we add the following element:

```
<xsd: element name=”R” type="typeR"/>
```

For example, consider the following Relation "Author" with "NameChefProj" as foreign key referencing "NameAuthor" in the same relation "Author":

Author(NameAuthor, TitlePaper, #NameChefProj)

The corresponding transformation rule is:

```
<xsd:element name="Author" type="typeAutor"/>
<xsd:complexType name="typeAutor">
  <xsd:sequence>
    <xsd:element name=" Author " type="xsd:string"/>
  </xsd:sequence>
  <!-- details of attributes in Author -->
</xsd:complexType>
```

Rule 13. CyclicRel: In this case it is important, in this cyclic relation, to make at least one element (or a reference to an element) as optional. Otherwise an infinite loop will occur and an error will be thrown while validating the XML schema.
For example, for a cyclic relationship (R1→R2→ R3→R1) we get the following transformation:

```
<xsd:element name="R1">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="R2"  minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="R2">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="R3"  minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="R3">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="R1" minOccurs="0" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

4. OUR METHODOLOGY FOR MAPPING

Our approach aims to define a correspondence between the RDB and XML schema using a multidimensional array model (MA) to build the XML structure. Our approach consists of three separate phases, as shown in figure 1. The first phase extracts tables, fields, relationships and metadata (MTRDB) from the relational database using java database connectivity (JDBC) components. In the second phase a multidimensional array model is generated to facilitate the migration process. Once the MA model is created we apply our algorithm based on the list of rules to create the equivalent XML schema.

Journal of Theoretical and Applied Information Technology
20th November 2015. Vol.81. No.2
© 2005 - 2015 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

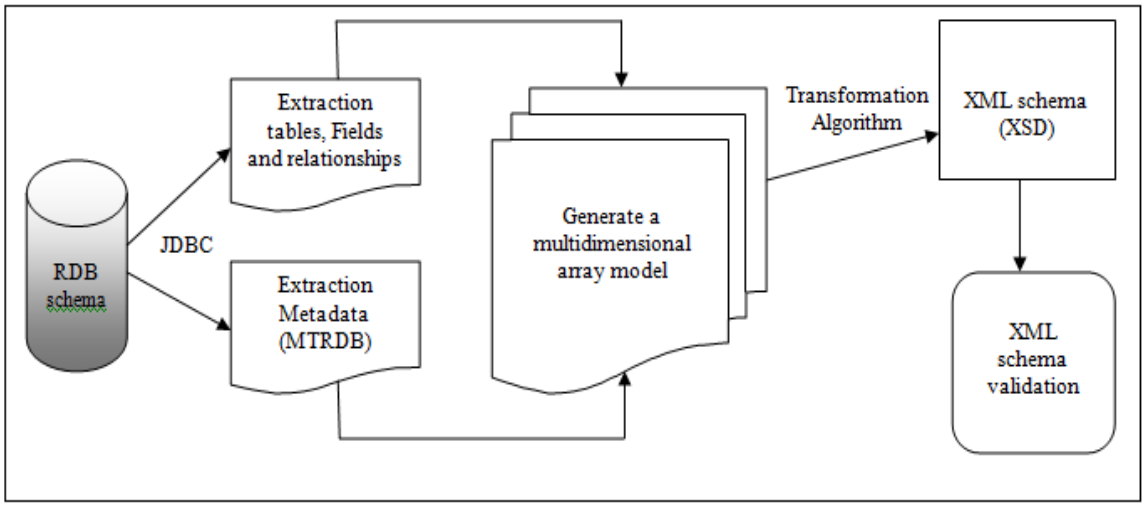

Figure 1: Relational Database Schema Overview

4.1. Extraction MetaData of RDB schema

Our process starts by extracting the metadata from the relational database including fields and relations, by using Java Database Connectivity (JDBC) components.

$MTRDB = \{R_N, R_{Ref}, R_{Refby}, R_{Type}, Nbr_{FK}, Type, F\}$

R_N: The relation name
R_{Ref}: All relation referenced by R_N
R_{Refby}: All relations that reference R_N
Nbr_{FK}: Number of foreign key in R_N
Type: (PFK) if the primary key of R also acts as a foreign key, (SelfR) if R is a SelfRefRel and (Simple) else

F: List of the fields of the relation RN

$F= \{F_N, F_T, F_{Key}, F_U, F_{Null}\}$

F_N: The field name
F_T: The field type
F_{Key}: (PK) if the field is a primary key, (FK) foreign key, (PFK) if act as both PK and FK or (CFK) if foreign key in a cyclic relation that references another field in the same cyclic relation
F_U: (Uq) for unique constraint
F_{Null}: (N) for Not null constraints

4.2. The multidimensional Array model of the RDB schema

The next step of our mapping approach consists in generating the MA model based on classification of elements extracted from MTRDB to facilitate the migration process.

MA model is a set of array elements that defines the list of relations taken from RDB schema with the necessary metadata for our mapping algorithm.

To illustrate the MA model we will use the following examples of relational database schema. Underlined attributes represent primary keys. Attributes endowed with a "#" represent foreign keys. The first example is given without any circular relation (SelfRefRel or CyclicRel) and its equivalent MA model is represented in Table 1:

Communication(idCom, CName)
Country(idCountry, CountName)
City(idCity, CityName, #idCountry)
Company(idCompany, CompName, #idCity)
University(idUniversity, UnivName)
Author(idAuthor, name, #idUniversity, #idCity)
Professor(#idProfessor, Grade)
Address(idAddress, Address)
PrivateAddress(#idPAddress, Type)
Student(#idStudent, Age, #idAddress)
Department(idDept, DeptName, #idChefDept)
Paper(idPaper, PaperTitle, Year)
WritePaper(#idPaper, #idAuthor)

Journal of Theoretical and Applied Information Technology
20th November 2015. Vol.81. No.2
© 2005 - 2015 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 1: Representation Of MA Model For Example 1

R_N	R_{Ref}	R_{RefBy}	NbrFK	Type	Fields				
					F_N	F_T	F_{Key}	F_U	F_{Null}
Communication			0	Simple	idCom	Int	PK	U	N
					CName	VarChar			N
Country		City	0	Simple	idCountry	Int	PK	U	N
					CountName	VarChar		U	N
City	Country	Author	1	Simple	idCity	Int	PK	U	N
		Company			CityName	VarChar			N
					idCountry	Int	FK		N
Company	City		1	Simple	idCompany	Int	PK	U	N
					CompName	VarChar			N
					idCity	Int	FK		N
University		Author	0	Simple	idUniversity	Int	PK	U	N
					UnivName	VarChar			N
Author	City	Department	2	Simple	idAuthor	Int	PK	U	N
		Professor			Name	VarChar			N
	Univer-sity	Student			idUniversity	Int	FK		
		WritePaper			idCity	Int	FK		N
Professor	Author		0	PFK	idProfessor	Int	PFK	U	N
					Grade	VarChar			
Address		Student	0	Simple	idAddress	Int	PK	U	N
		PrivateAdress			Address	VarChar		U	N
PrivateAdress	Address		1	PFK	idPAddress	Int	PFK	U	N
					Type	VarChar			N
Student	Address		2	PFK	idStudent	Int	PFK	U	N
	Author				Age	Int			N
					idAddress	Int	FK		N
Department	Author		1	Simple	idDept	Int	PK	U	N
					DeptName	VarChar			N
					idChefDept	Int	FK		N
Paper		WritePaper	0	Simple	idPaper	Int	PK	U	N
					PaperTitle	VarChar			
					Year	Date			
WritePaper	Author		2	PFK	idPaper	Int	PFK		N
	Paper				idAuthor	Int	PFK		N

In the second example we illustrate how to represent a SelfRefRel in MA model.

Employee(idEmp, nameEmp, Job, #Chef, #idDept)

The attribute "Chef" is a foreign key that reference idEmp in the same table. In this case we present "Chef" as a normal attribute and we put "SelfR" in the type Column of our MA model (Table 2).

Table 2: Representation Of MA Model For Example 2

R_N	R_{Ref}	R_{RefBy}	NbrFK	Type	Fields				
					F_N	F_T	F_{Key}	F_U	F_{Null}
Employee	Departement	Employee	1	SelfR	idEmployee	Int	PK	U	N
					nameEmp	VarChar			N
					Job	VarChar			
					Chef	Int			
					idDept	Int	FK		

© 2005 - 2015 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Finally we illustrate a MA model representation for a CyclicRel example:

City(idCity, NameCity, #idCountry)
Country(idCountry, NameCountry, #idUniversity)
University(idUniversity, NameUniversity, #idCity)

To resolve and extract all cyclic relations in the database source we use the MappingCircularRelation() algorithm in our previous work done in [1]. This algorithm uses a recursive function to detect if there is any cyclic relation in RDB schema and produces a list of cyclic relations as output.

We put "CFK" in FKey column for every foreign key in a cyclic relation that references another field in the same cyclic relation.

Table 3: Representation Of MA Model For Cyclic Relationship Example

R_N	R_{Ref}	R_{RefBy}	NbrFK	Type	Fields				
					F_N	F_T	F_{Key}	F_U	F_{Null}
City	Country	University	1	Simple	idCity	Int	PK	U	N
					NameCity	VarChar			
					idCountry	Int	CFK		
Country	University	City	1	Simple	idCountry	Int	PK	U	N
					NameCountry	VarChar			
					idUniversity	Int	CFK		
University	City	Country	1	Simple	idUniversity	Int	PK	U	N
					NameUniversity	VarChar			
					idCity	Int	CFK		

4.3. Mapping Algorithm

In this section, we present our algorithm for the automatic construction of XML schema from a relational database. This algorithm takes into consideration all the aforementioned conversion rules.

Given a MA model as input, the algorithm captures all relations types in order to assemble the mapped XML schema into a reasonable tree pattern

MapRelation() - Algorithm for mapping relations

Input: The MA model
Output: The corresponding XML schema

Begin
Step 1: Apply rule 1: Create the XML schema root element

Step 2: For each R_N (R) in MA model loop
  If (R_{RefBy}(R) = null and Nbr_{FK}(R) = 0) then
    Apply rule 2: create element as a child of the root element
    add R to MapRelationList
    MapAttribute of R
  End If
End loop

Step 3: For each R_N(R) in MA model loop
  If (R_{RefBy}(R) != null and Nbr_{FK}(R) = 0) then
    create element R as a child of the root element
    add R to MapRelationList
    MapAttribute of R
    Step 3-1: For each R_N(R_i') in MA model loop // R_i' is a R_{RefBy}(R)
      If (Nbr_{FK}(R_i') = 1 and Type(R_i') = Simple) then

© 2005 - 2015 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

```
                                Apply rule 3: create element Ri' as a child of R with minOccurs = 0
                                        and maxOccurs = unbounded
                                add Ri' to MapRelationList
                                MapAttribute of Ri'
                                If (RRefBy(Ri') != null) then
                                   Apply step 3-1 for all RRefBy(Ri')
                                End If
                              Else If ( NbrFK(Ri' = 1 and Type(Ri') = PFK ) then
                                 Apply rule 4: create element Ri' as a child of R with minOccurs = 0
                                        and maxOccurs = 1
                                  add Ri' to MapRelationList
                                  MapAttribute of Ri'
                                  If (RRefBy (Ri') != null) then
                                    Apply step 3-1 for all RRefBy (Ri')
                                  End If
                               Else If(NbrFK(Ri') = 1 and Type(Ri') = SelfR) then
                                  Apply rule 12 for self referenced relation
                                  MapAttribute of Ri'
                                  If (RRefBy (Ri') != null) then
                                     Apply step 3-1 for all RRefBy (Ri')
                                  End If
                               Else if(NbrFK(Ri') > 1) then
                                   Add Ri' FKMoreThanOneList
                               End if
                           End loop
                    End If
             End loop

Step 4: For each R in FKMoreThanOneList loop
             create element R as a child of the root element
             add R to MapRelationList
             MapAttribute of R
             For each RN(Si) in MA model loop      // Si is a RRef(R)
                If(Type(Si) =PFK) then
                    Apply rule 5: create element R_ Si as a child of R with minOccurs = 1
                                  and maxOccurs = 1
                Else
                    Apply rule 5: create element R_ Si as a child of R with minOccurs = 0
                                  and maxOccurs = unbounded
                End If
             End loop
             For each RN(Ri') in MA model loop      // Ri'  is a RRefBy(R)
                 Apply step 3-1 for all RRefBy(Ri')
             End loop
End
```

The algorithm used by "MapRelation()" for mapping attributes is as follows:

© 2005 - 2015 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

MapAttribute() - Algorithm for mapping attributes and constraints

```
Input: Relation R
Output: The corresponding XML schema

Begin
    For each Fields in R loop
      If (F_Null = N) then
          Apply rule 10: Create attribute with use="required"
      Else
          Apply rule 6: Create a normal attribute
      End If

      If (F_Key = PK) then
          Apply rule 7: add a xsd:key element
      Else if (F_Key = FK) then
          Apply rule 8: add a xsd:keyRef element
      Else if (F_Key =CFK) then
          Apply rule 13 for a cyclic relation
      End If

      If (F_U = U) then
          Apply rule 9: add a xsd:Unique element
      End If
End
```

4.4. Implementation and validation

To demonstrate the effectiveness and validity of our approach a tool has been developed (figure 2 & 3). This tool takes as input an RDB, then extracts its MTRDB, creates the corresponding MA model and applies our algorithm to create the resulting XML schema document.

To develop our prototype, we used Java as a programming language, and to store the data and metadata we used Mysql DBMS which contains system tables that define the structure of the database (including names of tables, columns, constraints ...). Our implementation can however also work with any other relational database system. We used the JDBC-API to establish the connection with the database. This API allows full access to relational database metadata and quickly retrieves a description of the tables and constraints of the database from data dictionaries.

For the example of a relational database schema considered above Fig. 2 and Fig. 3 at the end of the paper show the obtained XML schema and the conversion of the cyclic relationship of Table 3.

Journal of Theoretical and Applied Information Technology
20th November 2015. Vol.81. No.2
© 2005 - 2015 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

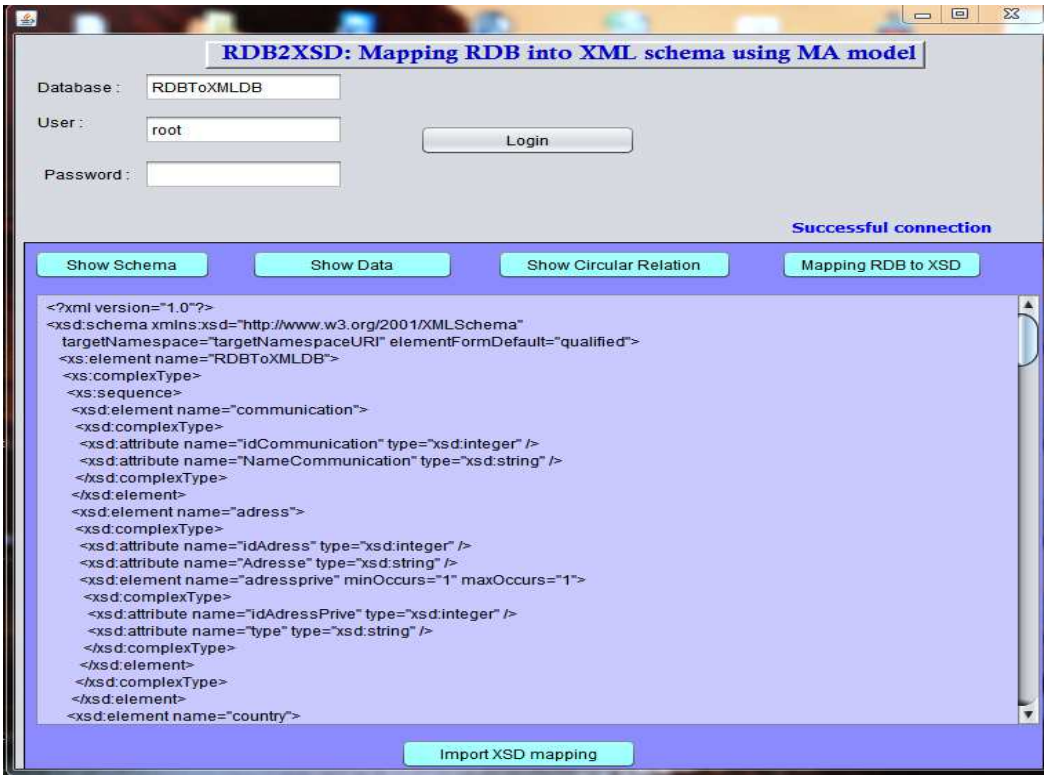

Figure 2: Mapping result of RDB schema

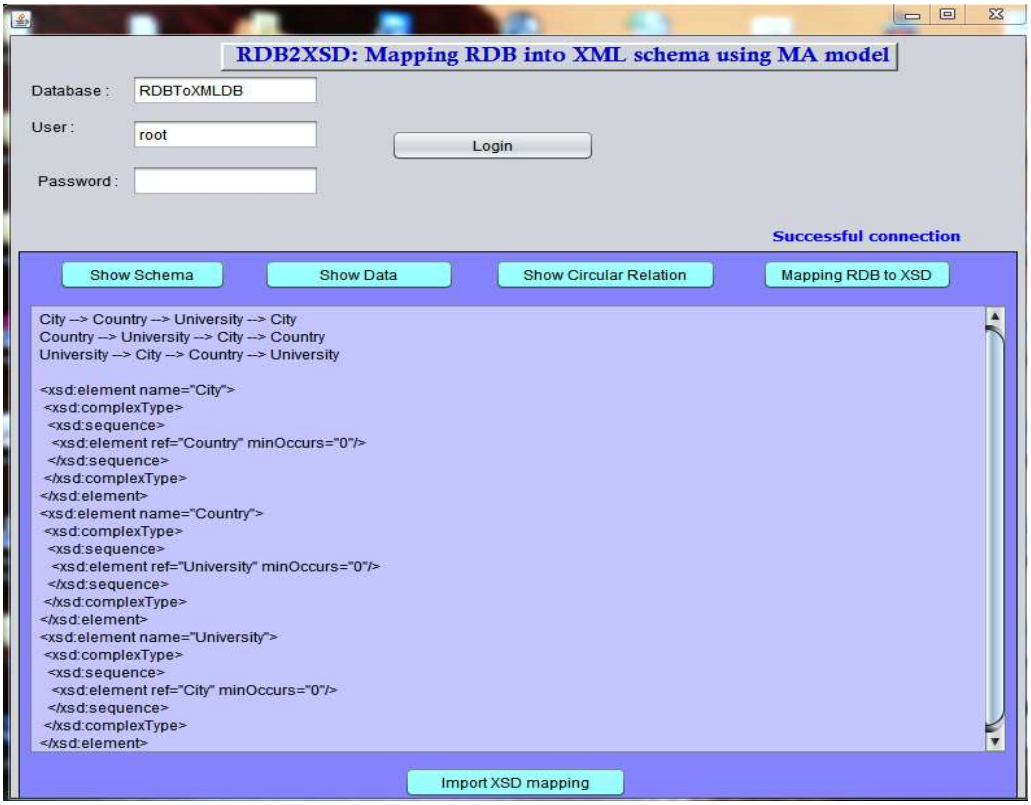

Figure 3: Mapping result of cyclic relation

© 2005 - 2015 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

5. CONCLUSION

In this paper, we have presented a new mapping process for converting relational database schema into XML schema. This process handles the mapping of the static and semantic constraints based on a well chosen categorization of the relations in the starting relational database. This categorization takes into accounts various aspects with respect to the referential integrity properties and to the various constraints on attributes. The mapping process first extracts the metadata from the RDB source, then a multidimensional array model (MA model) is generated automatically to capture the categorization structural designs and comes up with a complete and well structured hierarchical XML schema reflects all details in the initial relational database schema. The results obtained from our implementation prove the accuracy and performance of our mapping strategy.

REFRENCES:

[1] L. Alaoui, O. EL Hajjamy, and M. Bahaj, "RDB2OWL2: Schema and Data Conversion from RDB into OWL2," International Journal of Engineering Research & Technology (IJERT), vol. 3, Issue. 11, November 2014

[2] L. Alaoui, O. EL Hajjamy, and M. Bahaj, "Automatic Mapping of Relational Databases to OWL Antology," Int. J. Engineering & Research Technology IJERT, Vol. 3, Issue 4 (April, 2014)

[3] C. Baru, "XViews: XML Views of Relational Schemas." In Proceedings of DEXA Workshop, 1999, pp. 700–705

[4] M. Carey, J. Kiernan, J. Shanmugasundaram, E. Shekita, and S. Subramanian, "XPERANTO: Middleware for Publishing Object-Relational Data as XML Documents." In Proceedings of VLDB, 2000, pp. 646–648

[5] A. Duta, K. Barker, and R. Alhajj, "Converting relationships to XML nested structures"Journal of information and organizational sciences, Volume 28, Numberl- 2 (2004)

[6] M. Fernandez, W. Tan and D. Suciu, SilkRoute, Trading between Relations and XML, Computer Networks, Vol. 33 (Elsevier Science, 2000), pp. 723–745

[7] J. Fong, S. K. Cheung, "Translating relational schema into XML schema definition with data semantic preservation and XSD graph" Information and Software Technology, Volume 47, Issue 7, 15 May 2005, Pages 437–462

[8] J. Fong, H.K. Wong, Z. Cheng, "Converting relational database into XML documents with DOM" Information and Software Technology, Volume 45, Issue 6, 15 April 2003, Pages 335–355

[9] J. Kim, D. Jeong, and D. K. Baik, "A Translation Algorithm for Effective RDB-to-XML Schema Conversion Considering Referential Integrity Information," Journal of Information Science and Engineering 25, 137-166, 2009

[10] J. Kim, D. Jeong, and D. K. Baik, "An algorithm for automatic inference of referential integrities during translation from relational database to XML schema," in Proceedings of the International Conference on Computational Intelligence and Security, LNCS 3802, 2005, pp. 725-730.

[11] C. Kleiner and U. Lipeck, "Automatic generation of XML DTDs from conceptual database schemas", GI Jahrestagung (1) 2001, pp.396-405

[12] D. Lee, M. Mani, F. Chiu, W. W. Chu, "Schema conversion methods between XML and relational models" in Knowledge Transformation for the Semantic Web, pp. 1–17 (2003)

[13] D. Lee, M. Mani, F. Chiu, W. W. Chu, "Net & cot: translating relational schemas to XML schemas using semantic constraints" in CIKM CIKM'02, November 4–9, 2002, McLean, Virginia, USA (2002)

[14] D. Lee, M. Mani, F. Chiu, and W. Chu, "Nesting-Based Relational-to-XML Schema Translation." In Proceedings of the WebDB, 2001, pp. 61–66

[15] C. Liu, M. Vincent, J. Liu, and M. Guo, "A Virtual XML Database Engine for Relational Databases." In Proceedings of XSYM, 2003, pp. 37–51

[16] Ch. Liu, W. Vincent and J. Liu, "Constraint Preserving Transformation from Relational Schema to XML Schema" , World Wide Web: Internet and Web Information Systems, 9, 93–110, 2006

[17] T. Lv, P. Yan, "Schema Conversion from Relation to XML with Semantic Constraints", Fourth International Conference on Fuzzy Systems and Knowledge Discovery, 2007 - FSKD 2007. (Vol. 4), pp. 619 – 623

[18] J. Shanmugasundaram, J. Kiernan, E. Shekita, C. Fan, and J. Funderburk, "Querying XML Views of Relational Data." In Proceedings of VLDB, 2001, pp. 261–270

© 2005 - 2015 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[19] J. Shanmugasundaram, E. Shekita, R. Barr, M. Carey, B. Lindsay, H. Pira-hesh, and B. Reinwald, "Efficiently Publishing Relational Data as XML Documents." In Proceedings of VLDB, 2000, pp. 65–76

[20] V. Turau, "A framework for automatic generation of web-based data entry applications based on XML," in Proceedings of ACM Symposium on Applied Computing, 2002, pp. 1121-1126

[21] C.Wang, A. Lo, R. Alhajj, K. Barker, "Reverse engineering based approach for transferring lagacy relational databases into xml" in Proceedings of the 6th International Conference on Enterprise lnformation Systems, lCEIS2004, Porto, Portugal, April 4 -17, 2004

[22] R. Zhou, Ch. Liu, and J. Li, "Holistic Constraint-Preserving Transformation from Relational Schema into XML Schema", DASFAA 2008, LNCS 4947, pp. 4–18, 2008