Journal of Theoretical and Applied Information Technology
15th November 2016. Vol.93. No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

CONVERSION OF AN XML SCHEMA TO OBJECT RELATIONAL DATABASES USING A CANONICAL DATA MODEL

[1]ADIL JOUNAIDI, [2]DOHA MALKI, [3]MOHAMED BAHAJ, [4]ILIAS CHERTI

[1]Phd, Department of Mathematics and Computer Sciences, University Hassan I Settat, Morocco
[2]Phd, Department of Mathematics and Computer Sciences, University Hassan I Settat, Morocco
[3]Prof., Department of Mathematics and Computer Sciences, University Hassan I Settat, Morocco
[4]Prof., Department of Mathematics and Computer Sciences, University Hassan I Settat, Morocco

E-mail: [1]jounaidiadil@gmail.com, [2]doha.malki@uhp.ac.ma, [3]mohamedbahaj@gmail.com, [4]iliascherti@gmail.com

ABSTRACT

To describe or define the content of an XML file there is two main ways: either use a DTD or an XML Schema also called an XSD file. This last is the most recommended, however it lacks the object-oriented aspect. And also all the XML data flowing in the web are badly managed. So our paper came to resolve this problem by converting automatically XML Schemas to an Object-Relational Databases (ORDBs), which will allow us to manage XML data easily through SQL query. To do that we're going to use the CDM (canonical data modeling) by giving an algorithm that's going to conserve all type of relationships such as association, composition, aggregation and inheritance…, our prototype algorithm will be able to extract the schema metadata from the XML file and convert them into information represented as classes and relationships that are mentioned before. Simplicity and efficiency, those are the strength of our approach to come up with a system that generates a set of User-Defined-Types (UDTs) and a set of typed tables.

Keywords: *XML Schema Definition (XSD), Object Relational Database (ORDB), Canonical Data Modeling (CDM), User-Defined-Types (UDTs), Document Type Definition (DTD).*

1. INTRODUCTION

XML (eXtensible Markup Language) is emerging as a standard format of data and documents on the Internet [1], that's why every XML document has to be stored to be reused later in other applications, however, for effective development of enterprise applications in different environments, XML needs to have databases to store all data. Therefore, it is certain to use ways needed to describe the XML schema formats in the object-relational data base (ORDB).

Several studies have proposed solutions to map a DTD to a relational database [2, 3], which made us lose the wealth of object-oriented.

This reason motivated us to propose the method of transforming an XML schema to an object relational database by classifying all Complex Types of XML schema file in a CDM in order to preserve its structure and semantics and also the highlight of the OO modeling.

The purpose of this article, is to automate transformation process of any XML schema file to an object-relational database, to do this, we proceed in three steps, the first one is to determine all the relationships between the XML Schema Complex Types, the second step is to classify all Complex Types in a CDM to facilitate their transformation, and the last one is to present their corresponding in ORDB.

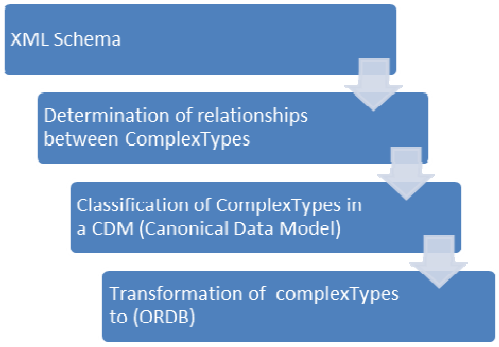

Figure 1: Mapping setps from XML schema to ORDB

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The rest of the paper is as follows. Section 2 provides an overview of related works, Section 3 provides an overview of the Semantic Enrichment of XML Schema, Section 4 describes the generation of CDM from XML schema, Section 5 proposes the translation of CDM into Object-Relational Database as a detailed algorithm for converting XML schema, Section 6 validates the approach, section 7 demonstrates the validity of our method, and finally section 8 concludes the paper.

2. RELATED WORK

There are several works that describe the mapping from relational databases to XML [9], this paper describes a way to map a Relational Data Base to different targets using a CDM, and our paper discusses a way from XML schema to ORDB using a CDM.

[4] This approach allows storing and retrieving any XML in a relational database fixed without taking in consideration neither the XML schema nor the DTD, this approach does not require the extension of the database, on the other hand it does not use the wealth of object-oriented modeling of ORDB.

[5] This paper proposes an approach for transforming a DTD to a relational database, based on the DTD Graph which is a node tree of the XML file and translating XML queries to SQL queries; although this approach takes into account the schema of the XML file, however, it does not take into account the different types of relationships (aggregation, composition, ...) between the Graph DTD node.

[6] This work presents the mapping of an XML schema to OO / OR, it does not propose the transformation of all types of relationships such as aggregation.

[7] Provides an overview of the mapping of the OO conceptual model to XML schema, this work proposes the mapping of the aggregation relationship, it does not cover all relationships as association.

[8] The paper proposes a method of mapping an XML schema to an object relational database in two stages, the first is used to transform an XML schema to a conceptual model OO, and the second is used to transform the conceptual model to an object relational database (ORDB), however, this work does not list all types of relationships such as inheritance,

In addition to this work [8], we offer our own classification method to classify all Complex Types of an XML schema in a CDM (Canonical Data Model) to simplify the automation of the transformation to an object-relational database (ORDB).

These related works don't offer solutions that will allow to convert all types of relationships between an XSD complex types but only some of them.

3. SEMANTIC ENRICHMENT OF XML SCHEMA

To enrich the semantic of an XML Schema we have to extract its data semantics, to be enriched and converted into a CDM. To do this, we have applied the approach in [9] to enrich semantically our XML schema. The process starts by extracting the basic information about an existing XML Schema, including relation types and attribute properties (i.e., attribute names, types, occurrence, required or not). 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. It is preferable that the process is applied to an XML schema well Formed, and validated. The next step is to identify the CDM constructs based on a classification of complex types, attributes and relationships, and then the CDM structure is generated.

3.1 Definition OF CDM

A canonical data model (CDM) defines the relevant entities for a specific domain, their attributes, their associations and their semantics. As a reference model, the CDM defines the associations, and the types of attributes, it is a method to extend and exchange the schema. The CDM is a data reference model that is designed to allow the sharing of information and data to reuse.

Our CDM is defined as a set of complex types CDM: = {CT | CT: = [ctn, cls, EAcdm, RLcdm, REFcdm]}, where each ComplexType C has a name ctn, has a classification cls, a set of elements and attributes EAcdm (Elements|Attributs cdm), a set of relationship RLcdm (ReLations cdm), and finally keys and keyrefs.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

3.1.1 Classification (cls)

We applied the approach in [8] to classify the ComplexTypes, this approach offers five classes:

a) Shareable and Existence-Independent Aggregation complex type (SEIA)

If a Complex Type can be shareable with others complex Types and its existence is independent of all of them we can classify this complex type as a (SEIA): "cls=SEIA".

b) Non-Shareable and Existence-dependent Aggregation complex type (NSEDA)

If a Complex Type that cannot be shareable with other Complex Types, and its existence depends to that of the others, this complex type is classified as "cls=NSEDA".

c) Association 1 :N complex type (A1N)

In this case, if a complex Type contains a reference which can be implemented inside another Complex Type, as its element with maxOccurs "unbounded" , therefore it is classified as "cls=A1N".

d) Association M :M complex type (AMM)

In the XML Schema for many-to-many association relationship, each types in the association has maxOccurs = « unbounded ». Each element will be linked to another element by using the attribute name that refers to another element ID. In this case we classify the complex type as (AMM).

e) Inheritance complex type (INHER)

If a complex type extends an existing complexType element, it classified as (INHER) in the CDM.

3.1.2 Elements|Attributs cdm (EAcdm)

Each complexType element has a set of attributes and EAcdm:= {a | a: = [Ele, Type, MinO/MaxO, Use]}, where each Element|Attribute belongs to a class that we presented at the beginning of this paragraph Cls = {SEIA, NSEDA, A1N, AMN, INHER}, Ele is the name of the element or attribute, type is the type of the element, MinO/MaxO is the minimum/maximum of occurrence, Use is to say that this element or attribute is mandatory or not.

3.1.3 Relations cdm (RLcdm)

Each complexType has a set of relationships with other complexTypes, each relationship rl ? RLcdm between complexTypes C1 and C2 is defined in C1, and represents an association, aggregation, composition, or inheritance. RLcdm = {rl | rl: = [RlType, DirC]}, where RlType is the type of relationship and DirC, is the name of the complexType C2.

3.1.4 Keys and Keyrefs (REFcdm)

Data dependencies are represented by keys and KeyRefs as for each keyref tag there is a reference to a key of a complex Type. REFcdm= {keys:= [k, kr]}.

4. GENERATION OF CDM FROM XML SCHEMA

The CDM presents the starting point for the rest of the migration process which, in the end, generates the target schema.

Let consider the following example (see Figure 2), in this example, we describe an XML Schema which models the purchase of order administration in a business company.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns: xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Customer_type">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="customerName" type="xsd:string"/>
        <xsd:element name="street" type="xsd:string"/>
        <xsd:element name="city" type="xsd:string"/>
```

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

```
            <xsd:element name="zipCode" type="xsd:integer"/>
            <xsd:element name="phone" type="xsd:integer"/>
            <xsd:element name="order" type="xsd:integer" maxOccurs="unbounded"/>
          </xsd:sequence>
          <xsd:attribute name="customerId" type="xsd:integer" use="required"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Customer_Association_Type">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="identification" type="xsd:string"/>
        <xsd:element name="description" type="xsd:string"/>
        <xsd:element name="percentage" type="xsd:integer"/>
        <xsd:element name="Customer" type="xsd:Customer_type" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Person_Type">
    <xsd:complexType>
      <xsd:complexcontent>
        <xsd:extension base="Customer_Type">
          <xsd:attribute name="personId" type="xsd:integer" use="required"/>
          <xsd:element name="discount" type="xsd:integer"/>
        </xsd:extension>
      </xsd:complexcontent>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Company_Type">
    <xsd:complexType>
      <xsd:complexcontent>
        <xsd:extension base="Customer_Type"/>
          <xsd:element name="type" type="xsd:string"/>
          <xsd:element name="taxes" type="xsd:integer"/>
        </xsd:complexcontent>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Purchase_Order_Type">
    <xsd:complexType>
      <xsd:sequence>
```

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

```
        <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:element name="Orderlineitem_Type">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="quantity" type="xsd:integer"/>
              <xsd:element name="productId" type="xsd:integer" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="line" type="xsd:ID" use="required"/>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
      <xsd:attribute name="order" type="xsd:integer" use="required"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="PRODUCTS_Type">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="description" type="xsd:string"/>
        <xsd:attribute name="price" type="xsd:decimal"/>
      </xsd:sequence>
      <xsd:attribute name="ProductId" type="xsd:ID" use="required"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Store_Type">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="capacity" type="xsd:integer"/>
        <xsd:element name="street" type="xsd:string"/>
        <xsd:element name="city" type="xsd:string"/>
        <xsd:element name="zipCode" type="xsd: integer"/>
        <xsd:element name="Stock_Type" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="quantity" type="xsd: integer"/>
              <xsd:element name="date" type="xsd:string"/>
```

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

```
            </xsd:sequence>
            <xsd:attribute name="productId" type="xsd:integer"/>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
      <xsd:attribute name="location" type="xsd:ID" use="required"/>
    </xsd:complexType>
  </xsd:element>
  <keyref name="PRODUCTS_productId_Ref" refer="PRODUCTS_productId">
    <selector xpath="STORE/STOCK">
      <field xpath="@productId"/>
    </selector>
  </keyref>
  <keyref name="ORDERLINEITEM_ProductId_Ref" refer="ORDERLINEITEM_ProductId">
    <selector xpath="PRODUCTS"/>
    <field xpath="@productId"/>
  </keyref>
  <key name="PRODUCTS_productId">
    <selector xpath="PRODUCTS"/>
    <field xpath="@productId"/>
  </key>
  <keyref name="PURCHASE_ORDER_order_Ref" refer="PURCHASE_ORDER_order">
    <selector xpath="CUSTOMER"/>
    <field xpath="@order"/>
  </keyref>
  <key name="CUSTOMER_customerId">
    <selector xpath="CUSTOMER"/>
    <field xpath="@customerId"/>
  </key> </xsd:schema>
```

Figure 2: XML Schema to be converted to ORDB

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Now we generate the CDM of the XML Schema described in the example above (see Table 1)

Table 1: CDM generated from the XML Schema to be converted to ORDB

cn	cls	eacdm				rlcdm		k/kr	
		ele	typ	mino/max	use	rltype	dirc	key	keyr
customer	seia	customerid	xsd:id		required			k	
		customernam	xsd:strin						
		orderid	xsd:integ	unbounded		asso	purchase_ord		kr
customer_association		identification	xsd:strin						
		description	xsd:strin						
		customer	customer	unbounded		aggr	customer		
purchase_order	a1n	orderid	xsd:id		required			k	
		shipping	xsd:strin						
orderlineitem	nseda	line	xsd:integ			comp	purchase_ord	k	
		productid	xsd:integ			comp	purchase_ord		
		quantity	xsd:integ			comp	purchase_ord		
products		productid	xsd:id		required			k	
		description	xsd:strin						
		price	xsd:deci						
store		location	xsd:id		required			k	
		capacity	xsd:integ						
stock	amn	productid	xsd:integ			asso	store		kr
		quantity	xsd:integ			asso	store		
person	inher	personid	xsd:id		required	inherb	customer	k	
		discount	xsd:integ			inherb	customer		
company	inher	type	xsd:strin			inherb	customer		
		taxes	xsd:integer			inherbv	customer		

The Complex Types are classified, their Elements, and relationships between different Complex Types are identified, and their cardinalities are determined. All these are stored in the CDM above.

5. TRANSLATING CDM INTO OBJECT-RELATIONAL DATABASE

In this section we present an algorithm that converts the CDM to ORDB (see figure 3).

```
public void buildOrdb(ArrayList<Cdm> cdm){

    // Initialization of SQL code
    Table t;

    for (Cdm c : cdm) {

        // Test on the name of the class in CDM
        switch (c.cls.toUpperCase()) {

           case "SEIA":
        // Creating a new UDT
        t = new Table(c.cn, "UDT", null);

        // Adding attributes of UDT
        for (EAcdm ea : c.a) {
           // Creating attributes of UDT
           Attribut attribut = ea.getAttribute();
            // Add attribute to UDT
           t.att.add(attribut);
           t.att.add(key);
        }
        // Add UDT
        tt.add(t);

        break;
```

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

```
            case "AMN":
            case "NSEDA":

          // reset of attribut
          Attribut amr = new Attribut(c.cn, null,
true, false);
          // name of super table NSEDA
          String tableNameNseda = "";
          // Add list of attributes ROW
          for (EAcdm ea : c.a) {
             tableNameNseda = ea.rl.dirC;
             // Creating attributes of UDT
             Attribut attribut = ea.getAttribute();
             // add attributes to MULTISET ROW
             amr.attMultisetRow.add(attribut);
          }
          // link attribute to the table
          this.addMultisetRow(amr,
tableNameNseda, tt);
          break;

            case "A1N":

          // Create new UDT
          t = new Table(c.cn, "TABLE", null);
          // Ajout des attribut d'UDT
          for (EAcdm ea : c.a) {
             // Create attribute of UDT
             Attribut attribut = ea.getAttribute();
             // add attribute to UDT
             t.att.add(attribut);
                    t.att.add(key);
          }
          // Add UDT
          tt.add(t);
          break;

            case "INHER":

          // reset super table name
          String tableExtendsParam = "";

          // Create new UDT
          t = new Table(c.cn, "TABLE", null);

          // Add attribute to UDT
          for (EAcdm ea : c.a) {
             tableExtendsParam = ea.rl.dirC;
       // Create attribute of UDT
             Attribut attribut = ea.getAttribute();
             // Add attribut to UDT
             t.att.add(attribut);
                    t.att.add(key);
          }
          // retrieve super table name
          t.tableExtends = tableExtendsParam;
          // Add UDT
          tt.add(t);
          break;

            default:

          // Create new UDT
          t = new Table(c.cn, "TABLE", null);
          // Add attribute of UDT
          for (EAcdm ea : c.a) {
             // Create attribute of UDT
             Attribut attribut = ea.getAttribute();
             // add attribute to UDT
             t.att.add(attribut);
                    t.att.add(key);
          }
          // Add UDT
          tt.add(t);
          break;
          }
       }
       this.getCodeSql();
    }
```

Figure 3: Algorithm to convert the XML Schema to ORDB

6. VALIDATION

After classifying the different Complex Types, we ran our algorithm that automates the transformation to ORDB which gave us the following SQL script (see figure 4)

```
CREATE TYPE CUSTOMER AS OBJECT
(       customerId NUMBER CONSTRAINT
customerId_pk PRIMARY KEY,
        customerName VARCHAR(250),
        orderId MUTISET(NUMBER)),
        NOT FINAL;


CREATE                                  TABLE
CUSTOMER_ASSOCIATION
(       identification VARCHAR(250),
        description VARCHAR(250),
        CUSTOMER MUTISET(CUSTOMER));
```

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

```
CREATE TABLE PURCHASE_ORDER
( orderId NUMBER CONSTRAINT orderId_pk PRIMARY KEY,
    shipping VARCHAR(250),
    ORDERLINEITEM MULTISET
    (ROW(line NUMBER,
        productId NUMBER,
        quantity NUMBER)));

CREATE TABLE PRODUCTS
( productId NUMBER CONSTRAINT productId_pk PRIMARY KEY,
    description VARCHAR(250),
    price DECIMAL(10,2));

CREATE TABLE STORE
( location NUMBER CONSTRAINT location_pk PRIMARY KEY,
    capacity NUMBER,
    STOCK MULTISET
    (ROW(productId NUMBER,
quantity NUMBER)));

CREATE TABLE PERSON UNDER CUSTOMER
( personId NUMBER CONSTRAINT personId_pk PRIMARY KEY,
    discount NUMBER);

CREATE TABLE COMPANY UNDER CUSTOMER
( type VARCHAR(250),
    taxes NUMBER);
```

Figure 4: Results of CDM generation

7. 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 Oracle. To evaluate our approach we examined the differences between source XML schema and the ORDB schema generated by the prototype, we test the query results provided by SQL in Oracle, and XQUERY in stylus studios. Queries returned the same results. The source XML database is transformed into target Object Relational database ORDB without loss of data.

This section presents queries applied on the XML schema shown in section (4) and the equivalent in SQL generated by the prototype. Table below shows the description, and the result of each query.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Tableau 2: Experimental Study

Description	SQL	Xquery	Result
Find the name of all Customers of the customer_association Identified by "ASS1" Ordered by name of customer	Select customerId, customerName, From customer c Where c.customer_Id in (select ca.c.customerId from customer_ associations ca) And 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 10 Scott 11 Smith
The first customer name of the customer_association identified by "ASS1"	SELECT TOP 1 * FROM customer_ associations ca, 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 count(costumer) from customer_associations ca, 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

After demonstrating the validity of our method technically, we can say that every XSD file can be transformed to an ORDB, after that we count on applying this method on applications that run on the web such as heterogeneous applications that use the XML files as a way of communication.

We mention as application examples the pharmacy and the medicine that use XML files to exchange data, where we can apply our method to create an exchanged data backup in an ORDB in order to use it in case of a communication failure between two applications.

8. CONCLUSION

Our paper gave an automatic way to convert an XML schema into an ORDB using a canonical data modeling CDM this method will help to store scattered web documents in a databases so users can manage them more easily.

Our method to do that is by creating a CDM from an XML schema and we use it as an input to a java program, and this last generates an SQL script. The java program was coded to respect all the content of the xml schema and the different relationships. At the end we applied some queries to the XML schema and the SQL generated by the program, these tests proved what we said earlier, and also we noticed a resemblance between our data in the OR and the one in the CDM and of course the XML schema.

In the future we count on proposing an XSD transformation method into OWL files for the applications using ontologies.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

REFRENCES:

[1] World Wide Web Consortium. 1998. Extensible Markup Language (XML) 1.0. http://www.w3.org TR/1998/REC-xml-19980210.W3C Recommendation 10-February-1998.

[2] Jayavel Shanmugasundaram, Eugene J. Shekita, Jerry Kiernan, Rajasekar Krishnamurthy, Stratis Viglas, Jeffrey F. Naughton, Igor Tatarinov: "A General Technique for Querying XML Documents using a Relational Database System" SIGMOD Record 30(3): 20-26 (2001)

[3] Pensri Amornsinlaphachai, Nick Rossiter and M. Akhtar Ali : "Storing linked XML documents in object-relational DBMS"CIT. Journal of computing and information technology ISSN 1330-1136

[4] Fayed F. M. Ghaleb, Sameh S. Daoud, Ahmad M. Hasnah, Jihad Mohamad Alja'am, Samir A. El-Seoud and Hosam F. El-Sofany: "A DOM-Based Approach of Storage and Retrieval of XML Documents Using Relational Databases". In Proceedings of the International Conference on Interactive Computer Aided Learning – ICL2006, Villach, Austria, September 27–29, 2006.

[5] Jayavel Shanmugasundaram, Eugene J. Shekita, Jerry Kiernan, Rajasekar Krishnamurthy, Stratis Viglas, Jeffrey F. Naughton, Igor Tatarinov: "A General Technique for Querying XML Documents using a Relational Database System" SIGMOD Record 30(3): 20-26 (2001)

[6] Han, W-S., Lee, K-H. and Lee, B.S."An XML Storage System for Object-Oriented/Object-Relational DBMSs", Journal of Object Technology 2(1), 2003, 113-126

[7] 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

[8] Eric Pardede, J.Wenny Rahayu, David Taniar "On Using Collection for Aggregation and Association Relationships in XML Object-Relational Storage", 2004 ACM Symposium on Applied Computing, 2004.

[9] Maatuk, A., Ali, M. A. and Rossiter, N.: Semantic Enrichment: The First Phase of Relational Database Migration. In CIS2E '08, 6pp, Bridgeport, USA, 2008.