World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:9, No:7, 2015

Schema and Data Migration of a Relational Database RDB to the Extensible Markup Language XML

Alae El Alami, Mohamed Bahaj

***Abstract*—**This article discusses the passage of RDB to XML documents (schema and data) based on metadata and semantic enrichment, which makes the RDB under flattened shape and is enriched by the object concept. The integration and exploitation of the object concept in the XML uses a syntax allowing for the verification of the conformity of the document XML during the creation. The information extracted from the RDB is therefore analyzed and filtered in order to adjust according to the structure of the XML files and the associated object model. Those implemented in the XML document through a SQL query are built dynamically. A prototype was implemented to realize automatic migration, and so proves the effectiveness of this particular approach.

***Keywords*—**RDB, XML, DTD, semantic enrichment.

I. Introduction

A relational database is a set of data stored in supports, organized as records, stored in tables, and is based on the principle of relation. XML (Extensible Markup Language) is a standard responsible for describing the structure of a text file and is considered as a database that stocks the data in the form of files, allowing for the structuralization of information.

Several problems have arisen with the relational model, such as the lack of Declaration of complex objects, data redundancy, the use of a rigid structure; meet a lot of problems during its evolution and its simplicity deprives it of several concepts. The relational model is limited hence the birth of this migration approach to XML.

XML began to emerge because it offers the ability to store files in a robust form. It is a standard that follows a certain syntax that is based on the principle of tags which can be nested, providing a hierarchical structure, thus benefitting from the exploitation of several concepts [9].

The XML is considered as a flexible and heterogeneous bridge that allows the various languages to consider it from a functional point of view as a connection that favors the research, the exchange, and the storage of the information of every type.

XML preserves the separation of content from presentation by taking the data that applications use and storing it separately from the presentation, regardless of the presentation of media used. XML is an open format that can be read by many applications. XML can be used on the client side and the server side and is supported by a number of languages and platforms.

The approach of the migration is based on the elaboration of a model, which plays the core role based on the principle of semantic enrichment. This approach discusses the transition from relational databases to XML files with their corresponding DTD.

Alae El Alami is a Phd in the Faculty of Science and Technology / Department of Mathematics and Computer Sciences, University Hassan I Settat, Morocco (e-mail: elalamialae@gmail.com).

Mohamed Bahaj is Professor the Faculty of Science and Technology / Department of Mathematics and Computer Sciences, University Hassan I Settat, Morocco (e-mail: mohamedbahaj@gmail.com).

II. Related Work

An approach of migration of the RDB towards the XML file is based on three conventional methods for the translation: user-specific translation methods, structural methods translation, and semantic translation methods. Each step requires the intervention of the human factor for a semi-automatic translation. The entire migration is based on the principle of cardinality to the make the correspondence between tables [1].

A certain approach shows the emergence of XML as a standard of exchange of data in the World Wide Web. This requires a migration of RDB to XML files through the EER (extended entity relationship) that is based on progressive procedures and corresponds between the tables of the relational databases to their flattened XML equivalents [2]. Another approach discusses the migration of relational databases to XML documents through a cross-Platform DOM while preserving the constraints of the RDB. The migration process starts with the creation of the XML scheme and then establishes the data conversion. To achieve this migration it is necessary to de-normalize the standardized relations in adjoining tables according to the data dependency constraints [3].

An approach discusses the conversion of RDB to XML files using the principle of reverse engineering. This extracts the ER model of RDB through an existing approach [5] which is based on the principle of identifier and allows for the specification of the primary keys and the foreign keys to determine the cardinality between tables. The comprehensive approach is based on four stages. The first is responsible for translating the ER model to the XML schema. The second step is to trace the responsible entities to sub-elements that correspond to complex-type. The next step creates a root for each element and inserts them into the ER model. The final step performs the mapping of keys and key referencing between the relation [4].

There are some tools that provide the ability to generate XML documents arbitrarily, by which the user support queries the RDB based on an extension of the SQL language. This allows the extraction of the RDB data through SQL-X [7]. The

tool enters within the framework of exchange between the data and the applications [6].

A fresh approach discusses the migration of databases concerning other models that is based on a meta-model. The meta-model acts as an intermediary between the conceptual model towards the object-relational model and vice versa. Along with the principle of semantic enrichment, one is able to realize the schema migration and data migration to the target database [12], [13]. Another approach discusses publishing XML files from a basic source relational database. Based primarily on the use of a conversion specification language between the relational model and XML, this results in the completion of the requirements that establishes the mechanism of conversion of the flat information into a hierarchical structure [8].

III. Semantic Enrichment

The semantic enrichment aims at enriching metadata and structuring data in such a way as to make the database flattened, and therefore adaptable for several models. The first stage of the migration is the semantic enrichment which is considered as an extended model of the relational model [10] defined by a set of class: meta-model:= {C | C: = ("Cn", degree, "Cls", a, contributor)}.

"Cn"=the name of the class.

Degree = first degree (the tables that contain PK) | 2nd degree (the tables that contain FK without PK).

"Cls"=aggregation, association, inheritance, simple class (the class that does not belong to the other classifications).

Contributor=class list.

A=attribute:={a | a:= (an, t, tag, l, n, d)} (An: name of the attribute, T:type of the attribute, Tag: primary key(PK) | foreign key(FK), L: length of the attribute, N: if the attribute takes the parameter null, D:the default value of the attribute).

The model schematizes the conception and the structure of the target physical model which allows to understand the navigation between three layers of abstraction, the first layer is the views that present the result that with which the user interacts, the second layer is the conceptual model that describes and determines the logical structure of the information system (IS) with the elements and their respective properties, and the third layer is the physical schema which describes the implementation of the database in memory at the level of storage structure and access methods. These parameters are obtained through responsible objects to query the database in order to obtain information about the physical structure of the database, container objects of general information about the database ("DatabaseMetaData"), containers information objects on a table or query and a type or property of a column ("ResultSet", "ResultSetMetaData"). Each corresponds to an access point data, and corresponds to a step of data access. Every object used requires the use of several methods to obtain accurate and detailed information about the database that is accessible through JDBC on the DB itself or on queries excluding data.

The relational database below serves as an example for this particular method until the end of the migration.

kids

kno	kname	sexe	pno
34	badr	m	d543
23	sarah	f	d543
21	jeff	m	g234

proj

prno	pname	description
1	Payment Management	integration of a module in an erp open source
2	tramway casa	realization of management complette Tramway casa

works_on

pno	prno
d543	1
f552	2
e234	1

employ

pno	salary	grade
d543	9000	engineer
g234	12000	director
f552	7000	commercial

trainee

pno	level	type
e234	master	hiring

dept

dno	dname
1	computer
2	commercial
3	after-sales service

person

pno	pname	bdate	adress	dno	pnosup
d543	alae	15/03/1987	residence ibn sina appt 3	1	g234
e234	fouad	03/01/1987	rayhan imm 4 appt 5	2	d543
g234	azar	24/04/1984	lotissemnt 34 rue des far appt 6	1	null
f552	jean	28/05/1975	rue la fayette residence bmo imm majid appt 9	3	d543

Fig. 1 Relational database of departure

Each part of the model has a specific role dedicated to the correspondence between the conceptual model, the physical model and the target database. The meta-model acts as intermediary between the two databases (source → target).

- The classification makes the attribution of the principle object for every table of the RDB to realize an abstracted

representation from a physical or virtual existence.

- The contribution lists all tables that interact with the table whose reference is made to eliminate the adjoining criteria. This is done when the data of associated tables are combined in queries and based on the shift by reference and modeling the conceptual model in a tabular form. The contribution also plays a role in the detection of the classification and improvement of the database by specifying the tag of some attributes if it is a foreign key, in order to apply data integrity.
- The detection of the object of various principles is made with metadata and data from the RDB.
- The detection of inheritance is done through the base of a data dictionary that includes a set of names. Each name indexes a set of synonyms or a list of matching words, which are dependent on the mother-child relationship according to standard naming rules. A verification step is necessary to eliminate the problem of non-standardized databases. If there is a key match between the super class and subclass then a legacy is extracted; if not, we proceed as if it is a simple class.
- The detection of the aggregation is composed of two parts. Firstly, when a class ("aggregate") interacts with a single class ("aggregate of"), with the absence of dependence between the two classes to be able to keep a history during the destruction of the object which is in collaboration with the aggregated class. Then secondly when there's an aggregation hierarchy, the direction of communication is to be down to a single path ("aggregate of" to "aggregate"), resulting in the detection of the foreign keys in the hierarchical classes of the class ("aggregate of").
- The detection of the composition is when a class ("component") interacts with a single class ("composite"). If there is a dependency between the two classes that prevents the referenced object to be removed, the dependence is bound to the physical schema.
- The detection of associations is due to the absence of the primary keys or the existence of a composite key consisting of foreign keys that reflect the authenticity of the recording during the transition to the physical model.
- The detection of reflexive associations that interact in the same table is detected by the occurrence of a foreign key. The latter is not in any other table as the primary key. This results in a self-contribution.
- The Detection of simple classes is done in the terminal stage through the absence of belonging of the other previously mentioned classifications.

These catches of functional object oriented specifications are assigned to a semantic enrichment discovery mechanism implemented in an application with several modules where each module is assigned to a given task.

TABLE I
ALGORITHM DETECTING OBJECT CONCEPT

```
Connection to the relational database (RDB);
// getting metadata from connection with the comprehensive information
about the database as a whole.
    “databaseMetadata dmd = connection.getMetadata();”
      //retrieving information
      “resultSet tables = dmd.getTables();”
“seti=0 ;”
   while( “tables.next()”)
      “incrementNbrTable;”
        “tab[i]”==(cast from object to string)
“tables.getObject(columnLabel);”
      “resultat = dmd.getColumns(connection.getCatalog(),tab[i]);”
   “resultset clefs”  =
“dmd.getPrimaryKeys(connection.getCatalog(),tab[i]);”
        “rsmd = resultat.getMetadata();”
        if((“clefs.absolute(1))==”true)
          “columnName = clefs.getString("column_name");”
          “increment j;”
        end if
        “increment i ;”
        “nbrAttribut=0;”
        while (“resultat.next()”)
        “incrementNbrAttribut;”
        “object val1 =resultat.getObject(columnLabel);”
        end while
        “attrTotal=attrTotal+nbrAttribut;”
        “incrementNumTab;”
   end while
   while(“resultat1.next()”){
              “string col1= rsmd.getColumnName(column_name);”
        “object val1 = resultat1.getobject(column_name);”
        “attribut[attributCount][0]=val1;”
        “string val6;”
   if((“val1.equals(nomColonne1”))&&(( “ndm[gh][2])!="association""))
          “val6="pk";”
          else “val6="";”
end if    “attribut[attributCount][2]=val6;”
          “string col2 = rsmd.getColumnName(type_name);”
          “object val2 = resultat1.getobject(type_name);”
          “attribut[attributCount][1]=val2;”

          “string col3 = rsmd.getColumnName(column_size);”
          “object val3 = resultat1.getobject(column_size);”
          “attribut[attributCount][3]=val3;”

          “string col4 = rsmd.getcolumnName(is_nullable);”
          “object val4 = resultat1.getobject(is_nullable);”
          “attribut[attributCount][4]=val4;”

          “string col5 = rsmd.getColumnName(column_def);”
          “object val5 = resultat1.getobject(column_def);”
          “attribut[attributCount][5]=val5;”

          “incrementAttributCount ;”
        end while
“#inheritanceDetection()”
“#agregationDetection()”
“#compositionDetection()”
“#associationDetection()”
“#reflexiveAssociationDetection()”
```

IV. Creating XML Documents

XML is a very flexible data encoding language derived from SGML and consists of text and structural information. XML is written in markup language in respect to the XML specification, and is intended to describe and store data.

XML documents begin with the statement that indicates the version. The encoding declaration then identifies the encoding used that represents the characters appearing in the document. The documents must comply with rules of writing, must have a single root element, the tags must strictly be closed, case sensitive, and nested elements.

Our approach of migration of the RDB towards the XML is based on the exploitation of the object model using the principle of inheritance, compositions and aggregations, and has the ability to exploit concretely or abstractly objects or to make a reference to other objects. From the meta-model, we realize the creation of XML files.

For each derived class meta-model such as "C.Classification": = (simple || association) we create an XML file whose name is defined in the meta-model "C.Cn" and has a header and a root element that will collate the all records. Two tags specifying the "C.Attribut.An" element of the meta-model delimit each record.

For the classes of the meta-model such as "C.Classification": = ("inherBy" && "Inherts"), we create XML files with names defined in the meta-model with the root element and required tags, making its kinship correspondence between the "superClass" and "subClass" by eliminating the join problem in the relational model by using identifiers (ID) and references (IDREF or IDREFS) which respectively contain keys or references, realized and verified in their respective DTDs.

To create aggregations which their classification "C.Classification": = aggregation we create a simple XML file with a header and a root element, and achieves a referencing link in the file that cooperate with the aggregate file with referencing constraints, to determine the semantic correspondence indicating that an object is part of another object and not leading to its destruction.

For the composition of which "C.Classification":= composition && "C.Contributor"= "C1.Cn" such as "C.Cn"→"C1.Cn", is created within the XML file schematically by the occurrence of a new element in the XML file which models the semantic expressed by "component, consisted" which causes the destruction of the contents during the destruction of the container.

V. Defining the XML Document Structure

In order to respect the standards and basic rules of XML, it is necessary to use files, which verify that the XML document is in accordance with a precise syntax, and that it respects the declaration of elements, attributes, notations, and entities. Our approach opts for DTD (Document Type Definition).

From the meta-model, we realize the creation of the DTDs with the declaration of elements of permitted and required attributes, and entities. The declaration of elements follows the following form: <!ELEMENT name "type_element">. Each element therefore defines the elements that must contain the operator that determines the number of occurrences depending on the parameter "C.Attribut.N". The specifications of the types change according to the parameters of the meta-model "C.Attribut.Type" && "C.Attribut.N" in order to determine the elements to use, the information-seeking to operate, and the ability to not contain any data element (ANY, EMPTY, #PCDATA) along with the possibility to use the combined elements.

The declaration of the attributes presents a target element (the allowed name, the type, and the default value). According to the meta-model and the parameter "C.Attribut.Tag" we specify the type ID to identify an element considered unique, and from the parameters "C.Classificiation" && "C.Contributor" we realize the referencing by IDREF towards the element that contains the type ID attribute with the same value of the attribute, with the possibility of achieving a multiple referencing by IDREFS. The specification of default values is realized by the keyword #FIXED by the parameter "C.Attribut.D" that declares that it is fixed and not modifiable. The declaration of DTDs for compositions is made within the DTD that checks the class and that enters into collaboration with the composed class "C1.Cn" as "C.Classification":= composition && "C.Contributor"= "C1.Cn" and "C.Cn"→"C1.Cn".

VI. Data Migration

The correspondence between the relational model and XML model is done through the meta-model. The meta-model is based on a set of treatments to achieve the adequate allocation of each element of relational tables to the corresponding XML files that operates the object model.

The information extracted from the RDB through a SQL query is built dynamically, and is analyzed and filtered as to automatically adjust depending on the structure of the created XML files and the corresponding DTD that operates the object concept.

The data retrieved from the relational database must match, and in order to comply with the new database, a procedure will be dedicated to the integration of data to XML files. This procedure adapts dynamically to a specification from the meta-model based on mutable objects [11]. The specifications will be: the classification of the tables drawn of the RDB, the classes entering in collaboration with the referring object class, the degree of the class, and the sub parameters of the parameter attribute (the tag and the null).

For single classes and associations such as "C.Classification": = simple || "C.Classification": = association, the data selection process is built around a complete selection from the relational table. The result of the selection is injected to each element of the corresponding XML file between the tags. For classes that specify the mother-daughter inheritance such as "C.Classification":= "inherBy" && "C.Contributor"= "C1.Cn" && "C1.Classification":= "Inherts", the injection procedure is applied to the super class and then performs the dependency

link via references. For aggregation classes the insertion procedure performs injection and maintains the semantics via references such as "C.Classification":= aggregation && "C.Contributor"="C1.Cn" REF "C1.Attribut.Tag":= PK.

For class composition is located within the composite class "C.Classification":=composition && "C.Contributor"= "C1.Cn" such as "C1.Attribut.Tag":= PK → "C.Attribut.Tag":= PK, through a SQL query using a selection by a field built dynamically.

TABLE II
RESPONSIBLE METHOD FOR THE SELECTIVE QUERY

```
public String[][] "selectAllByChamp" (String "tableName", String "champ", String "name") {
    String "req"= "SELECT * FROM " + "tableName"+" where "+ champ +"="+name+"";
    try {
        "int Type = ResultSet.TYPE_SCROLL_INSENSITIVE;"
        "int Mode = ResultSet.CONCUR_UPDATABLE;"
        "Statement sql=db.createStatement(type,mode);"
        "ResultSet rs = sql.executeQuery(req);"
        "ResultSetMetaData rsm = rs.getMetaData();"
        "int Columns = rsm.getColumnCount();"
        String data[][];
        try {
            "rs.last();"
        } catch (Exception e) {
            "System.err.println("ERROR rs.last()");"
        }
        "int rows = rs.getRow() + 1;"
        "data = new String[rows][columns];"
        for ("int i = 1; i<=columns; i++") {
            "data[0][i-1] = rsm.getColumnName(i);"
        }
        "int row = 1;"
        "rs.beforeFirst();"
        while ("rs.next()") {
            for ("int i=1; i<=columns; i++") {
                "data[row][i-1] = rs.getString(i);"
            }
            "row++;"
        }
        "returndata;"
    }
    catch (Exception e) {
        "e.printStackTrace();"
        "return null;"
    }
}
```

Fig. 2 represents the comprehensive approach of the migration of a RDB towards XML files (schema and data) exploiting metadata, and a set of processing to extract the various object concepts towards a XML model.

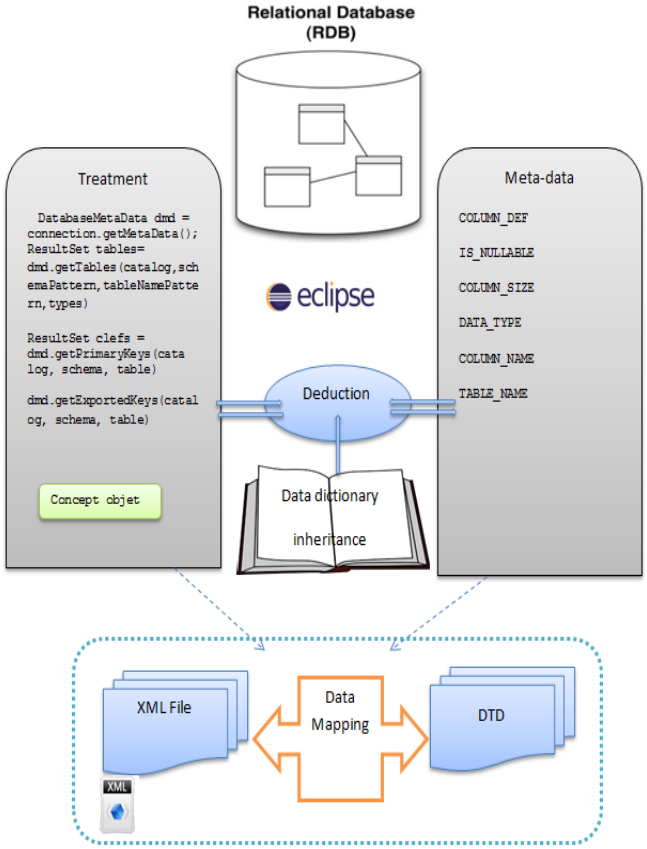

Fig. 2 Graphical representation of the RDB migration approach to XML

VII. CONCLUSION

This paper realizes the migration of RDB to XML files using metadata and semantic enrichment by exploiting the meta-model that plays the core role in this approach. From this, we achieved the creation of XML documents and their corresponding DTD, and proceeded to migrate data from the RDB to the target file. A prototype was made proving the effectiveness of this approach.

In a forthcoming work, we will implement a schematron [14], which is capable of expressing constraints in such a way as other languages cannot, and is able to perform the structure validation of the XML files by assertions instead of validation grammar.

REFERENCES

[1] Jinhyung Kim, Dongwon Jeong, Doo-Kwon 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).

[2] Kanagaraj S, Dr Sunitha Abburu. Converting Relational Database Into Xml Document. IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 2, No 1, March 2012 ISSN (Online): 1694-0814.

[3] J. Fong, H.K. Wong, Z. Cheng. Converting relational database into XML documents with DOM. information and Software Technology 45 (2003) 335–355.

[4] C. Wang, A. Lo, R. Alhajj, and K. Barker. "Converting Legacy Relational Database into XML Database through Reverse Engineering". Proc. of ICEIS,2004.

[5] R. Alhajj, "Extracting the Extended Entity-Relationship Model from a legacy Relational Database, "Information Systems, Vol.28, No.6,pp.597-618, 2003.
[6] Orsini, R., Pagotto, M. (2001). Visual sql-x: A graphical tool for producing xml documents from relational databases. In Poster proceedings of the international world wide web conference. Hong Kong.
[7] R. Orsini, "A preliminary proposal for SQL-X: A Language to Extract XML Documents from Relational Databases", SEBD 2000, L'Aquila, June 2000.
[8] Mihai Stancu. From relational databases to xml documents: efficient alternatives for publishing. International Journal of Digital Information and Wireless Communications (IJDIWC) 1(2): 545-553The Society of Digital Information and Wireless Communications, 2011(ISSN 2225-658X).
[9] Gordana Pavlovic Lazetic. Native Xml Databases Vs. Relational Databases In Dealing With Xml Documents. Kragujevac J. Math. 30(2007) 181-199.
[10] Alae El Alami, Mohamed Bahaj; Migration of the Relational Data Base (RDB) to the Object Relational Data Base (ORDB); World Academy of Science, Engineering and Technology International Journal of Computer, Information Science and Engineering Vol:8 No:1, 2014.
[11] Mohamed Bahaj, Alae El Alami, The Migration Of Data From A Relational Database (Rdb) To An Object Relational (Ordb) Database. Journal of Theoretical and Applied Information Technology 20th December 2013. Vol. 58 No.2
[12] Alae El Alami, Mohamed Bahaj; The Road to a Full Migration of Relational Database (RDB) to Object Relational Database (ORDB): Semantic Enrichment, Target Schema, Data Mapping; International Journal of Advanced Information Science and Technology (IJAIST) ISSN: 2319:2682 Vol.30, No.30, October 2014.
[13] Alae El Alami, Mohamed Bahaj; The Migration of a Conceptual Object Model COM (Conceptual Data Model CDM, Unified Modeling Language UML class diagram ...) to the Object Relational Database ORDB; MAGNT Research Report (ISSN. 1444-8939) Vol.2 (4). PP: 318-32.
[14] Van der Vlist, Eric. Schematron. " O'Reilly Media, Inc.", 2007.