ISSN No. 0976-5697

Volume 2, No. 4, July-August 2011

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Canonical Approach for Data transformation from traditional databases to XML

Vandana Dabass
M.Tech. (C.E)
PDMCE, bahadurgarh,
Haryana, India
vandanadabass@gmail.com

Abstract: XML and traditional databases both are considered as the active research areas for computer industry as internet and technology penetrate more in the software market. XML and relational databases are two most essential mechanisms for storing and transforming the data. In this paper we are going to present a canonical approach for data transformation from traditional databases to XML databases in which with the help of canonical data model traditional databases such as relational databases will be transformed to XML database in order to preserve the semantics of both.

Keywords: XML, relational databases, canonical data model, data transformation, database technology

I. INTRODUCTION

XML and traditional (RDB) databases are two of the most important mechanisms for storing and transferring data. A reliable and flexible way of moving data between them is very desirable goal. The way data is stored in each method is very different which makes the transformation process difficult. XML is getting increasingly popular for data exchange and storing for the web applications because of its portability and ease of data exchange features. It is useful when applications must communicate with other applications or integrate information from several other applications. XML documents are self-contained i.e. they contain the data as well its presentation format.To successfully move data from one model to the other a way of describing the schema is needed for this purpose a canonical approach is used in which CDM (canonical data model) is used as a conceptual model. Canonical data model is a federated collection of local metamodels including the definition of the common semantics and the format transformation rules.

II. MOTIVATION

Many organizations have stored their data in RDBs and aspire to take advantage of databases that have emerged more recently. Instead of discarding existing RDBs or building non-relational applications on top of them, it is generally preferable and beneficial to convert existing relational data into a new environment. However, the question is: which of the new databases is most appropriate to move to? So there is a need for a method that deals with database transformation from RDB to XML in order to provide an opportunity for exploration, experimentation and comparison among alternative database technologies. The method should assist in evaluating and choosing the most appropriate target database to adopt for non-relational applications to be developed according to the required functionality, performance and suitability. This could help further increase the acceptance of such newer and richer databases among enterprises and practitioners. An XML document represents the data along with the metadata. In a relational model data and the metadata exist at different places. XML data is portable while a data in relational model is not.

III. BACKGROUND

One of the challenges in relational to XML transformations that schema overhead is as big a factor as data redundancy [8]. Thus, the ability to avoid encoding data items has a double benefit the data item itself is not encoded and its accompanying tags are not encoded. Further, the transformation should be easily expressible and preferably require minimal user input. The common weakness with most approaches is that the relational model is mapped to a different data model before migration to XML and this procedure requires human involvement. Once the user has built the intermediate model, they have limited impact on the final result as the mapping is performed by transformation rules. Note this work is not about XML normalization [1]. The assumption is that the relational schema has already been normalized, and the algorithm must only ensure redundancies do not get re-introduced during transformation. Four categorizes of transformation methods:

A. Flat Transformation
B. Query-based Transformation
C. Model-based Transformation
D. Dependency-based Transformation

IV. XML DATABASES VS TRADITIONAL DATABSES

An XML database is a data persistence software system that allows data to be stored in XML format. This data can then be queried, exported and serialized into the desired format.XML solves many problems by providing a standard format for data interchange, some challenges remain. In the real world, applications need reliable services to store, retrieve, and manipulate data. These services were traditionally offered by relational databases. The relational database technology has matured over the last 30 years, and

it is well-known for its impressive SQL query performance, unequaled reliability and scalability, strong management and security, and legendary concurrency through locking and caching. So it would seem to be natural to use relational databases to persist and manipulate XML documents[3]. Well, the problem is that relational and hierarchical representations of data are very different. In the relational model, the data is stored in rows of two-dimensional tables where the physical order of rows is insignificant. XML, on the other hand, is a highly hierarchical model where the order of elements *is* significant, and the relationship among elements is described in a given document. Using a relational model to express a hierarchy of elements in a complex XML document is a non-trivial task. Therefore, some software vendors decided to implement pure XML databases designed to efficiently handle the hierarchical model. Unfortunately, the native XML databases don't provide maturity, scalability, and concurrency of the relational databases yet. Another approach adopted by other software vendors is to programmatically process the XML documents and map their hierarchy into a relational database.
Two major classes of XML database exist:

A. *XML-enabled:*

These map all XML to a traditional database (such as a relational database), accepting XML as input and rendering XML as output. This term implies that the database does the conversion itself (as opposed to relying on middleware) [2].

B. *Native XML (NXD):*

The internal model of such databases depends on XML and uses XML documents as the fundamental unit of storage, which are, however, not necessarily stored in the form of text files.

While traditional (relational) databases are the most popular mechanism for storing the data as compared to xml databases in which information is stored in the form of tables and accessed with the help of queries. The relational data model, introduced represents a database as a collection of relations (i.e., tables of values); hence the name relational database. Later, the ER model, which is currently used as the main conceptual model, was proposed for graphically structuring a relational model. Extensions to this model, have been proposed in the '80s and '90s because of its widespread use in practice. Data are structured and stored in RDBs in two dimensional tables. The relational model focuses on tuple-oriented information and primitive data types. Each table consists of a number of rows, called tuples, each of which consists of a collection of related values.

V. CANONICAL DATA MODEL

A. *Brief Overview of CDM*

Canonical data model is said to be the Meta model which have all the information required by everyone. It is a superset rather than a subset in which a point-to-point connection requires.

B.

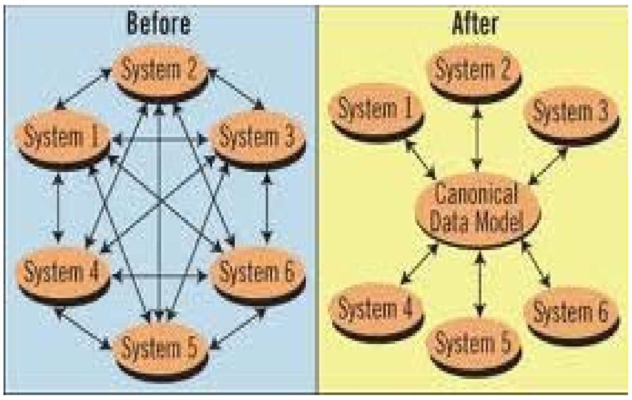

Figure 1 CDM before and after

The CDM is a source of valuable semantics giving an enriched and well organized data model, which can be converted flexibly into any of the target database. Besides taking into account the characteristics of the target model, the CDM retains all data semantics that could be extracted from an RDB and the integrity constraints imposed on it. Moreover, it acts as a key mediator for converting existing RDB data into target databases based on the structure and the concepts of the target models. The CDM facilitates the reallocation of attribute values in an RDB to the appropriate values in a target database. Based on the CDM definition, target attributes that represent relationships among classes are materialized into references or changed into other domains.

VI. NEED FOR DATA TRANSFORMATION

The transformation of RDBs into relatively newer database XML has been motivated by the dominance of traditional RDBs in the marketplace and their limitations in supporting complex structures and user-defined data types provided by these new technologies. The problem is how to effectively transform an existing RDB as a source into the newer databases as target, and what is the best way to enrich the semantics and constraints of the RDB in order to appropriately capture the characteristics of this target. Canonical approach takes an existing RDB as an input, enriches its metadata representation with required semantics, and constructs an enhanced relational schema representation (RSR). Based on the RSR, a canonical data model (CDM) is generated, which captures the essential characteristics of the target data models, for the purpose of transformation. Due to the heterogeneity of the target model, it is believed that it's necessary to develop a CDM to bridge the semantic gap among them and to facilitate the migration process. The CDM is designed to preserve the integrity constraints and data semantics of the RDB so as to fit in with the target database characteristics. This canonical approach preserves the structure and semantics of an existing RDB to generate XML schemas, and effectively converts existing RDB data into target database without redundancy or loss of data.

A. *Phase 1 (Transformation from Realtional to CDM):*

In this first phase relational database are transformed into the more generalized form that is Canonical data model.

B. *Phase 2 (Transformation from CDM to XML):*

In this second phase the acquired cdm is converted into desired xml database. All the data transformation from relational to xml database is done with the help of canonical data model.

VII. PROS AND CONS OF XML

XML is self-describing in nature: An XML document represents the data along with the metadata. In a relational model data and the metadata exist at different places.XML data is portable while a data in relational model is not.For example to exchange data between MS-Access and Oracle we need to do some data transformations even both of them are based on relational model [1].

Adaptable to Changes: The format of an XML document is not rigid we can easily add new tags as per our requirements.However doing the same thing in relational model is not possible because of the restrictions imposed by the relational model constraints [5] .

Good for storing tree or graph based structure: XML's hierarchical structure makes it possible for us to store tree or graph structured data in XML documents. Databases using relational model do not allow us to store hierarchical data. XML is considered to be a database in a weak sense and has the following limitations as compared to conventional database systems:

A. Efficient access to data due to parsing and conversion
B. Efficient Storage Indexes
C. Security
D. Multi-user access
E. Triggers
F. Transaction Management
G. Data Integrity
H. Queries across multiple documents

Given these limitations XML documents work good in an environment consisting of small amounts of data, few users, and modest performance requirements[7].

VIII. EXPERIMENTALS RESULTS

This canonical approach was implemented using the Java 1.5.0 software development kit installed on a computer with CPU Pentium IV 3.2 GHz and RAM 2 GB, operating under Windows XP Professional. The Java database connectivity (JDBC) API has been utilized to establish a connection with MYSQL, which holds the input RDB(s) to be transformed.
Input data in form of RDB is given in figure 2:

```
-- Table structure for table `emp`
DROP TABLE IF EXISTS `emp`;
CREATE TABLE `emp` (
`FirstName` char(20) DEFAULT NULL,
`SecondName` char(20) DEFAULT NULL,
`EmpID` int(11) NOT NULL DEFAULT '0',
`hiredate` date DEFAULT NULL,
`EmailID` char(20) DEFAULT NULL,
`Dept` char(20) DEFAULT NULL,
`designation` char(20) DEFAULT NULL,
PRIMARY KEY (`EmpID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Figure 2. Input data

Now with the help of CDM this data is converted into XML representation. XML have its own schema for representing the data so this table "emp" is converted in to XML format as given in the figure 3.with this example a small fragment of database is converted in to xml with the help of canonical data model.

```
    <xs:element name="emp">
     <xs:complexType>
      <xs:sequence>
     <xs:element name="FirstName" nillable="true"
minOccurs="0">
        <xs:simpleType>
         <xs:restriction base="xs:string">
          <xs:maxLength value="20"/>
         </xs:restriction>
        </xs:simpleType>
       </xs:element>
       <xs:element name="SecondName" nillable="true"
minOccurs="0">
        <xs:simpleType>
         <xs:restriction base="xs:string">
          <xs:maxLength value="20"/>
         </xs:restriction>
        </xs:simpleType>
       </xs:element>
       <xs:element name="EmpID" default="0">
        <xs:simpleType>
         <xs:restriction base="xs:int">
          <xs:minInclusive value="-2147483648"/>
          <xs:maxInclusive value="2147483647"/>
         </xs:restriction>
        </xs:simpleType>
       </xs:element>
       <xs:elementname="hiredate"nillable="true"
minOccurs="0">
        <xs:simpleType>
         <xs:restriction base="xs:date">
          <xs:minInclusive value="1000-01-01"/>
          <xs:maxInclusive value="9999-12-31"/>
         </xs:restriction>
        </xs:simpleType>
       </xs:element>
       <xs:elementname="EmailID"nillable="true"
minOccurs="0">
        <xs:simpleType>
         <xs:restriction base="xs:string">
          <xs:maxLength value="20"/>
         </xs:restriction>
        </xs:simpleType>
       </xs:element>
       <xs:elementname="Dept"nillable="true"
minOccurs="0">
        <xs:simpleType>
         <xs:restriction base="xs:string">
          <xs:maxLength value="20"/>
         </xs:restriction>
        </xs:simpleType>
       </xs:element>
       <xs:elementname="designation"nillable="true"
minOccurs="0">
        <xs:simpleType>
         <xs:restriction base="xs:string">
```

```
                <xs:maxLength value="20"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:element>
          <xs:elementref="salary"minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
      <xs:key name="emp_PrimaryKey_1">
        <xs:selector xpath="."/>
        <xs:field xpath="EmpID"/>
      </xs:key>
      <xs:keyrefname="salary_ForeignKey_1"
refer="emp_PrimaryKey_1">
        <xs:selector xpath="salary"/>
        <xs:field xpath="EmpID"/>
      </xs:keyref>
     </xs:element>
```

Figure 3. Output data after applying canonical approach

IX. CONCLUSION

As from the above discussion it is concluded that with the help of Canonical data model input traditional mechanism of storing that is RDB can be easily transformed into newer technique that is XML database.

X. REFERENCES

[1] Arenas, M. and Libkin, L., “A Normal Form for XML Documents”, Proceedings of ACM PODS 2002, pages 85-96.

[2] Banerjee, S., Krishnamurthy, V., Krishaprasad M., and Murthy, R., “Oracle 8i – The XML Enabled Data Management System”, Proceedings of ICDE 2000.

[3] Bird, L., Goodchild, A., and Halpin, T., “Object Role Modelling and XML-Schema”, Proceedings of ER 2000, pages 309-322.

[4] Bohannon, P., Freire, J., Roy, P., and Simeon, J., “From XML Schema to Relations: A Cost-Based Approach to XML Storage”, Proceedings of ICDE 2002, pages 64-76.

[5] Du, W., Lee, M., and Ling, T., “XML Structures for Relational Data”, Proceedings of WISE 2001, pages 151-160.

[6] Edmonds, J., “Optimum Branchings”,Journal of Research of the National Bureau of Standards, 71B:233-240, 1967.

[7] Embley, D.,and Mok, W., “Developing XML Documents with Guaranteed ‘Good’ Properties”, Proceedings of ER 2001, pages 426-441.

[8] Fernandez, M., Kadiyska, Y., Suciu, D., Morishima, A., and Tan, W., “SilkRoute: A Framework for Publishing Relational Data in XML”, 27(4):438-493, 2002