Advances in Information Mining, ISSN: 0975–3265, Volume 1, Issue 2, 2009, pp-11-13

Representing object oriented database using XML-DTD

Chitra Desai *, Manisha Patil and Sahebrao Shinde
Department of MCA, MIT Engineering College, Aurangabad, chitrag_desai@yahoo.com
Department of Computer Science, University of Pune, manishap6@gmail.com
Department of Computer Science, University of Pune, sns110@gmail.com

Abstract- The ability to handle complex data has made object oriented database gain a wide attention in the field of scientific research and engineering. Using object-oriented database for scientific data in it has several characteristics that should be considered for storing and retrieving it for future use. As this information needs to be exchanged and shared, XML in itself is emerging as a standard for information exchange over heterogeneous systems. The object oriented database structure designed using UML standard can be transformed into XML DTD. The XML DTD can further be used for obtaining DTD graph, which is further, used for extracting object oriented database schema by applying set of rules for class definition. This paper takes an example XML schema and transforms it into relational database schema. As object oriented database can be looked for more complex type of data handling compared to relational database, the XML schema is transformed to XML DTD and is used for representing object oriented database schema.
Keywords- XML Schema, Relational database, XML DTD, DTD graph, Object Oriented Database

Introduction

Many applications are built today as loosely coupled, distributed systems where individual components (often called services) are combined together. Since many of these components will be reused for other applications, the architecture needs to be flexible enough to allow individual components to join or leave the heterogeneous conglomerate of services and components and to change their internal design and data models without jeopardizing the whole architecture. XML [1], has emerged as a standard for information exchange and has therefore gained wide attention in database communities to extract information from XML seen as data model. However, database have schema, which are used to constraint what information can be stored in the database and to constraint the data types of the stored information. In contrast by default, XML document can be created without any associated schema: an element then may have any sub element or attribute. While such freedom may occasionally be acceptable given the self-describing nature of the data format, it is not generally useful when XML documents must be processed automatically as part of an application, or even when large amount of related data are to be formatted in XML. A document oriented schema mechanism; the document type definition (DTD) is therefore used. The Document Type Definition (DTD) is an optimal part of an XML document. The main purpose of a DTD is much like that of a schema: to constrain and type the information present in the document. However, the DTD does not in fact contain types in the sense of basic types like integer or string. Instead, it only constraints the appearance of sub elements and attributes within an element. The DTD is primarily a list of rules for what patterns of sub elements appear within an element.T he DTD definition can further be used to extract Object Oriented database or Relational database schema. Various existing tools mainly focus on extracting relational database schema from existing XML schema. However, the mapping is not an easy one because the data model of an XML document is fundamentally different from that of a relational database. Especially the structure of an XML document is hierarchical and the XML elements may be nested and repeated. Object oriented database represents the latest addition in the toolbox of modern software developers. The impetus of research on object-oriented database has come from its increasing use in design and implementation of artificial intelligence (AI) and computer aided design (CAD) systems. With its rich data model, it is believed that object oriented database have great potential. An OODBMS is the result of combining object oriented programming principles with data management principles. Due to the popularity of the object oriented modeling approach in recent years, an object oriented information model is often constructed to represent the static structure and dynamic behavior of the information and the processes in a construction project and expressed by UML (Unified Modeling Language) [2], a popular tool for object oriented modeling. Moreover, extensible Markup Language (XML) has become a defacto standard for information sharing and exchange in recent years. Therefore, there is a need to define an XML schema based upon the UML object oriented information model to further facilitate information sharing. Converting a UML to XML requires a very detailed and strict way for how to convert object-oriented design into standard compliant UML class diagrams, into physical XML representation. This can be achieved as shown in

[3, 4]. This paper takes an example XML schema (parts.xsd file) and transforms it into relational database schema. As object oriented database can be looked for more complex type of data handling compared to relational database, the XML schema is transformed to XML DTD and is used for representing object oriented database schema.

Generating Relational Database Schema from XML Schema

XML provides many of the things found in databases, such as storage (e.g., XML documents), schemas (e.g., DTDs, and XML schemas), query languages (e.g., XQuery, XPath, XQL), programming interfaces (e.g., SAX, DOM, JDOM), and more. However, XML lacks many important features that are supported by the real databases, such as efficient storage, indexes, security, transactions and data integrity, multi-user access, triggers, queries across multiple documents, etc. [7] Therefore, databases are still the top choice for managing information in a enterprise or government agency. Although several kinds of database technologies are currently available in the market (e.g., object-oriented database, object-relational database, etc.), RDB technology remains the most popular one employed in the construction industry. Although several RDB providers have provided common data import/export tools to allow for data transformation between the XML documents and the RDB tables, the capabilities of these tools are still quite limited. Recently some commercial software has also provided tools for transforming information in XML documents into RDB. However, most of them can only transform simple XML documents. That is, if the XML documents have a nested data structure and association, most of these tools are still incapable of making a complete transformation. Also additional overhead is required to map the traditional E-R diagrams to the XML schema.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!--W3C Schema generated by XMLSpy v2006 rel. 3 U (http://www.altova.com)-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xs:import namespace="http://www.w3.org/XML/1998/namespace"/>
    <xs:element name="Parts">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="Title" minOccurs="0"/>
                <xs:element ref="Part" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="Title">
        <xs:complexType mixed="true"/>
    </xs:element>
    <xs:element name="Part">
        <xs:complexType>
            <xs:sequence maxOccurs="unbounded">
                <xs:element ref="item"/>
                <xs:element ref="manufacturer"/>
                <xs:element ref="model"/>
                <xs:element ref="cost"/>
            </xs:sequence>
            <xs:attribute name="type">
                <xs:simpleType>
                    <xs:restriction base="xs:NMTOKEN">
                        <xs:enumeration value="airplane"/>
                        <xs:enumeration value="computer"/>
                        <xs:enumeration value="auto"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
        </xs:complexType>
    </xs:element>
    <xs:element name="item">
        <xs:complexType mixed="true"/>
    </xs:element>
    <xs:element name="manufacturer">
        <xs:complexType mixed="true"/>
    </xs:element>
    <xs:element name="Model">
        <xs:complexType mixed="true"/>
    </xs:element>
    <xs:element name="Title"/>
    <xs:element name="cost"/>
    <xs:element name="model"/>
</xs:schema>
```

Figure 1. XML Schema for Parts (parts.xsd file)

In addition, several XML-to-Relational transformation algorithms and mapping tools have been proposed in the literature. For example, Bourret et al. [8] introduced an XML-RDB mapping language to specify transformation rules for generating an RDB schema from an existing XML DTD (Document Type Definition).

```
CREATE TABLE [Part] (
   [type] varchar (255),
    [item] varchar (255) NOT NULL ,
    [manufacturer] varchar (255) NOT NULL ,
    [model] varchar (255) NOT NULL ,
    [cost] varchar (255) NOT NULL
);
CREATE TABLE [Parts] (
   [Title] varchar (255) NOT NULL
);
```

Figure 2. DB structure from parts.xsd (source database used Microsoft Access)

They also proposed a lightweight, DBMS- and platform independent load/extract utility to facilitate the data transfer between XML documents and relational databases. Lee and Chu [9] proposes a method where the hidden semantic constraints in DTD are systematically found and translated into relational data models. However, most of them deal with XML DTD, instead of XML Schema, which is more flexible and offers more supports for data types. If XML DTD is to be used the object-oriented database schema becomes a more useful as it can handle complex data plus it can be easily transformed from the available UML design. UML design can be easily transformed into XML DTD, which- can be further use for obtaining object oriented database schema.

Generating Object Oriented Database Schema from XML DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT Parts (Title?,Part*) >
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Part (item,manufacturer,model,cost)+>
<!ATTLIST   Part type (computer|auto|airplane) #IMPLIED>
<!ELEMENT item (#PCDATA)>
<!ELEMENT manufacturer (#PCDATA)>
<!ELEMENT Model (#PCDATA)>
```

Figure 3. parts.dtd file contents

In a DTD file each declaration is in the form of a regular expression for the sub elements of an element. The | operator specifies "or" while the + operator specifies "one or more". The * operator is used to specify "Zero or more", while the? operator is used to specify an optional element (that is "zero or one"). Here the first line says that an element Parts has the Title and Part sub elements, here the Title is optional and Part may have zero or more occurrence. The keyword "#PCDATA" in line 2 indicates text data; it derives its name, historically from "parsed character data". Two other special type declarations are empty, which says that the element has no

contents, and any, which says that there is no constraint on the sub elements of the element; that is, any elements, even those not mentioned in the DTD, can occur as sub elements of element. The absence of declaration for an element is equivalent to explicitly declaring the type as any. In line 4, attributes of Part can be of type computer, auto or airplane. Attributes must have a type declaration and a default declaration. The default declaration can consist of a default value for the attribute or #REQUIRED, meaning that a value must be specified for the attribute in each element, or #IMPLIED, meaning that no default value has been provided. Here for Part, attributes are defined implied. While attempting to represent an object oriented database schema from the XML DTD one should take care to see: Here each element definition creates one class, these classes should therefore be inline into one class. Inheritance aspect should be taken into consideration by first creating a DTD graph. This will help reconstructing classes using inheritance. Once we have the DTD file with us the next task is to obtain the DTD graph. The DTD graph for the figure 3 is as shown in figure 4.

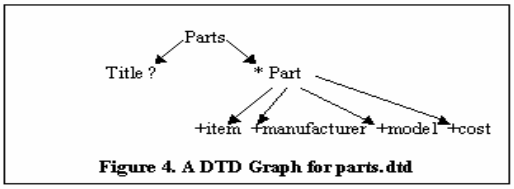

Figure 4. A DTD Graph for parts.dtd

To decide which classes to create from the DTD graph the following five rules [5] are applied Classes are created for element nodes that have an in degree of zero. Elements below '*' or a "+' node are made into separate class. Nodes with in degree of one are inline. Among mutually recursive elements all having in degree one, one of them is made into a separate relation. Element nodes having an in degree greater than one are made into separate relation. Once we decide which classes are created, we construct an object-oriented schema. An object-oriented schema is created for the DTD graph in figure 4 as shown in figure 5.

```
class Part public type tuple(type:string, item:string,
manufacturer: string, model: string, cost: string)
class Parts public type tuple(Title: string)
```

Figure 5. Object Oriented Database Schema for DTD graph in Figure 4.

Conclusion

In this paper we have shown how XML Schema can be used for representing relational database. However, due to the limitations of the relational database compared to object oriented database and the complexity associated with transforming XML schemas to relational database a more stress on representing object oriented database schema from XML DTD is stressed. As XML DTD can easily be obtained from UML design it can be further used for generating DTD graph which in turn can be used for object oriented database schema extraction. Thus this will help in easy information exchange and sharing complex data over heterogeneous systems.

References

[1] Pardi W. J. *XML in Action, Microsoft Press, Washington*, 1999.

[2] Fowler M. and Scott K. *UML Distilled, 2nd Edition, Addison Wesley, Boston*, 2000.

[3] Michel Huvka. *Aw, SCHUCS! An approach to creating XML Schema from UML class Diagrams, Control and Design Systems, Pasedana* 2000.

[4] Atole C.S., Gawali B. W., Kale K.V., Mehrotra S.C. *National Conference on Issues and Trends in Wireless Networking . Patiala, Punjab Dec* 2004.

[5] Tae-Sun Chung, Sangwon Park, Sang-Young Han, Hyoung-Joo Kim. *Second International Conference, EC-Web 2001 Munich,Germany, September* 4-6, 2001.

[6] Bourret R. *XML and Databases. from web page at http://www.rpbourret.com/xml/XMLAndDatabases.htm, February*, 2002.

[7] Bourret R., Bornhövd C. and Buchmann A. *Proceedings of the Second International Workshop on Advance Issues of E-Commerce and Webbased Webbased Information Systems (WECWIS 2000), San Jose, California, U.S.A., June* 8- 9, 2000.

[8] Lee D. and Chu W. W. (2001) *Data Knowledge Engineering*, 39(1), 3-25.

