Journal of Southeast University (English Edition) Vol.19 No.2 June 2003 ISSN 1003—7985

A method for publishing relational schema into DTD

Liang Zuopeng Wang Xiaoling Xu Lizhen Dong Yisheng

(Department of Computer Science and Engineering, Southeast University, Nanjing 210096, China)

Abstract: This paper focuses on exporting relational data into extensible markup language (XML). First, the characteristics of both relational schemas represented by E-R diagrams and XML document type definitions (DTDs) are analyzed. Secondly, the corresponding mapping rules are proposed. At last an algorithm based on edge tables is presented. There are two key points in the algorithm. One is that the edge table is used to store the information of the relational dictionary, and this brings about the efficiency of the algorithm. The other is that structural information can be obtained from the resulting DTDs and other applications can optimize their query processes using the structural information.

Key words: XML; DTD; relational database; XML schema

Extensible markup language (XML)[1] is fast emerging as the dominant standard for representing and exchanging data on the Internet. An XML document consists of nested element structures, starting with a root element. Element data can be in the form of attributes or sub-elements. In order to be exchanged conveniently, XML documents may conform to some kind of document type definitions (DTDs)[2], which are essentially schemas for XML documents. In fact, there are already several industry proposals to standardize DTD. Despite the excitement surrounding XML, it is important to note that most operational business data, even for new web-based applications, continue to be stored in relational database systems. This is unlikely to change in the foreseeable future because of the reliability, scalability, tools, and performance associated with relational database systems. Consequently, if XML is to fulfill its potential, some mechanism is needed to publish existing relational data in the form of XML documents.

There are already many works addressing this problem, for example, the research work of Refs.[3 – 5] concentrates on the publishing problem and addresses several of the key difficulties in the problem. However, in real application circumstances the data in a relational database is huge and applications usually use only a very small fraction of it. It is more important to transform the relational schemas into DTDs. Consequently other applications can query and exchange the exact part of data more efficiently. In this paper, we address the problem of publishing relational schemas into DTDs. First, we analyze both the relational schema and DTD in detail. Then we discuss how a relational schema can be mapped into a DTD. Since we use an E-R diagram (ERD) and a DTD graph as intermediate transforming representations, our mapping rules are simple and straightforward. At last, an algorithm based on an edge table is presented.

The rest of this paper is organized as follows. In section 1, we analyze the characteristics of the relational schemas represented in ERDs and those of DTDs. In section 2, several transformation rules are presented and in section 3 we put forward an efficient algorithm. Section 4 concludes this paper and proposes future work.

1 Relational Schema and DTD

A relational schema or just schema, written as $R(A_1, A_2, \cdots, A_n)$, is a relation name and a list of attribute names. The schema describes the organization of the possible relations. An individual relation is said to be an instance of a relational schema.

An example of a relational schema is shown as follows.

- Customer (id: integer, name: varchar (20));
- Account (id: varchar (20), custid: integer, acctnum: integer);
- PurchOrder (id: integer, custid: integer, acctid: varchar(20), date: varchar (10));
- Item (id: integer, poid: integer, desc: varchar (10));
- Payment (id: integer, poid: integer, desc: varchar (10)).

The customer table records the information of customers while the account table and the purchorder table hold the account information and the purchase information of customers, respectively. The two tables

Received 2002-08-29.

Biographies: Liang Zuopeng (1973—), male, graduate; Dong Yisheng (corresponding author), male, professor, ysdong@seu.edu.cn.

are associated with the customer table through foreign keys. The item table and payment table are associated with purchorder table in the same way.

Relational schemas are usually represented by ERDs. All the entities, attributes, domains, primary keys, foreign keys, constraints, relationships, also notes and other physical and logical data, can be laid-out in a transparent order. Using the graphically well-arranged ERD, database structures can be easily created and maintained. The corresponding ERD of the relational schema shown as an example is presented in Fig.1.

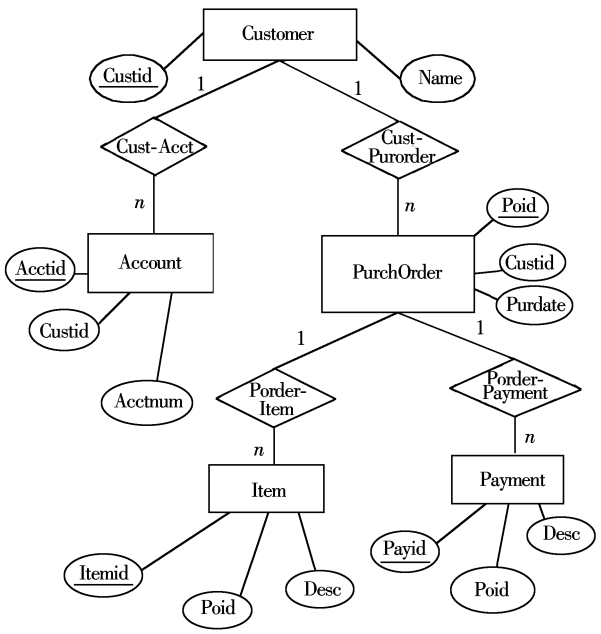

Fig.1 The E-R diagram describing relations

- An entity is represented by a rectangle labeled "entity";
- An attribute is represented by an ellipse labeled "attribute";
- A relation is represented by a rhombus labeled "relation" and the cardinality constraints are labeled at the two ends of the edge;
- A primary attribute is labeled with an underline.

XML documents may have DTDs to define their structure and constraints on them. DTD allows users to specify the set of tags, the order of tags, and attributes associated with each tag. A well-formed XML document that conforms to its DTD is called valid. DTDs are not mandatory for XML documents. However, if XML documents conform to a DTD, it is convenient for applications to manage and search the XML documents. So we expect that XML documents conform to some DTDs. It is just like other data management systems, DTDs describe a schema for a set of XML documents. The function of DTD is similar to that of relational schema in relational databases or class definition in object-oriented systems. If an XML document conforms to a DTD, then it can be checked valid or not by a sparser. A DTD can be declared inline in your XML document, or as an external reference.

We define a DTD graph to describe the logical and structural information among the elements in a DTD. Its nodes represent elements, attributes and operators in DTD. Each node representing an individual element appears exactly once in the graph, while other nodes representing attributes and operators appear as many times as they appear in the DTD. The formal definition is as follows.

Definition 1 Graph = (N, E), where N = {Element | Attribute | Operator}, and E denotes directed edges from node to node and Operator = {*|+|?}. The semantic of operators is as follows. "*" means zero or more, "+" means one or more and "?" means zero or one. The DTD graph of the example is shown in Fig.2.

2 Transformation Rules

We can draw a conclusion from the analysis above that both relational schema and DTD are used to offer constraints on data. However, they differ from each other dramatically. The relational schema consists of domains and attributes, and relational data is a set of tuples which are composed of atomic attributes. In contrast, DTDs consist of elements and attributes, which allow set values and nesting relationships between elements. All these differences make the exportation of relational data into XML difficult. It is necessary to find some bridges that help to overcome the gap between relational data and XML. We consider the ERD as a good candidate. It is obvious that if the relational schema is represented with an ERD, the schema can be easily mapped into DTDs. That is, an entity in ERDs corresponds to an element node in DTDs and a relation corresponds to an operator node. Attributes can be mapped directly. The mapping rules are presented as follows:

Rule 1 Transformation of entities: each entity in ERDs will be transformed into an element in DTD graphs with the same name.

Rule 2 Transformation of attributes: each attribute in ERDs will be transformed directly into a required attribute node in DTDs. If the application does not care about the order of elements, it is not necessary to transform the id attribute.

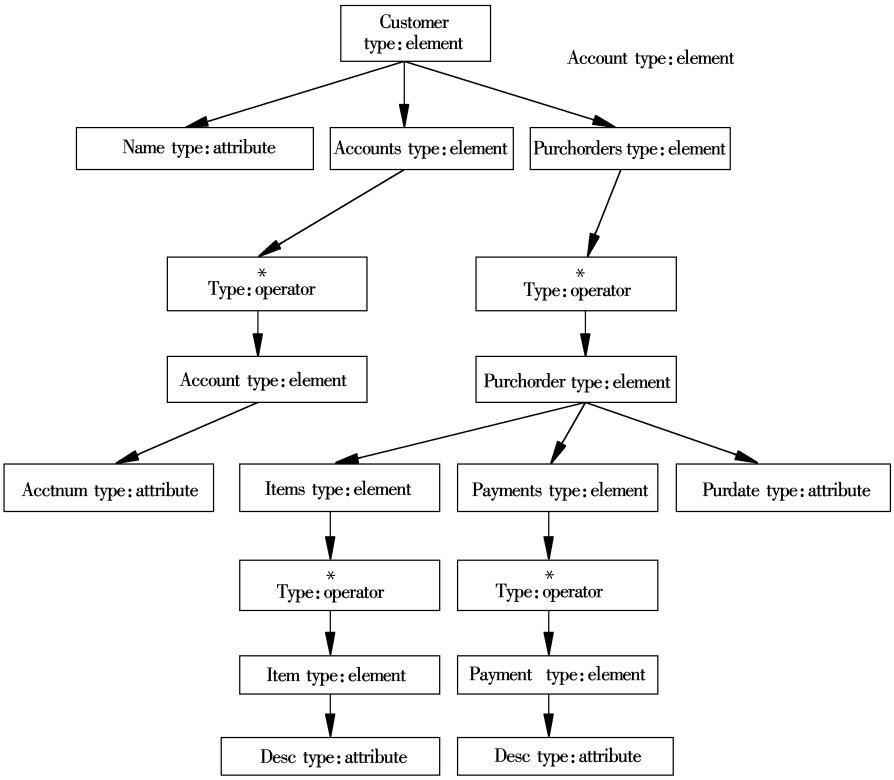

Fig.2 A DTD graph describing a customer relation

Rule 3 Transformation of relations: each relation will be transformed into an operator element node.

3 Transformation Algorithm

We design an algorithm to publish a relational schema into a DTD using the transformation rules mentioned above. We first transform and store the relational schema in an edge table for the purpose of simplicity of implementation. The edge table storing the ERD information of Fig.2 is shown in Tab.1. Id is used to record the order of nodes in ERDs, and Sid is the edge's source node. Similarly, Did is the edge's destination node. The Name and Type domain are used to record the node's name and type, respectively. We define three types of nodes here, they are: Entity, Attribute and Relation.

Tab.1 Edge table storing the customer relation

Id	Sid	Did	Name	Type
1	0	Null	Customer	Entity
2	1	Null	Name	Attribute
3	1	Null	Account	Entity
4	3	1	Accounts	Relation
5	3	Null	AcctNum	Attribute
6	1	Null	PurchOrder	Entity
7	6	1	PurchOrders	Relation
8	6	Null	PurDate	Attribute
9	6	Null	Item	Entity
10	9	6	Items	Relation
11	9	Null	Desc	Attribute
12	6	Null	Payment	Entity
13	12	6	Payments	Relation
14	12	Null	Dese	Attribute

3.1 Algorithm description

Input: edge table storing the information of the relational schema.

Output: a DTD file.

Step 1 Create a text file.

Step 2 Get the string from the input edge table.

Procedures used to get the string are as follows:

① Locate the root record, that is, a record whose Sid domain is 0;

② Find all the entities whose Sid is the Id of the root entity and add them to the queue recording the information of the child entitities of the root record. Transform all the entities using transformation rule 1;

③ Find all the attributes whose Sid is the Id of the root entity and transform them using transformation rule 2;

④ Find all the operators whose Sid is the Id of the root entity and transform them using transformation rule 3;

⑤ Process the child entities iteratively as ②, ③, and ④ till no new child entities are found.

Step 3 Write the string achieved from step 2 into the text file.

Evidently, the time complexity of our algorithm is $O(n)$, where n is the number of nodes comprised in the ERD.

3.2 An example

We apply the algorithm above to transform the relational schema shown in Fig.1. The DTD file customer.dtd obtained from our algorithm is shown as follows.

```
〈!ELEMENT Customer (Accounts, PurchOrders)〉
〈!ATTLIST Customer Name CDATA #REQUIRED〉
〈!ELEMENT Accounts (Account)*〉
〈!ELEMENT Account EMPTY〉
〈!ATTLIST Account AcctNum CDATA #REQUIRED〉
〈!ELEMENT PurchOrders (PurchOrder)*〉
〈!ELEMENT PurchOrder (Items,Payments)〉
〈!ATTLIST PurchOrder PurDate CDATA #REQUIRED〉
〈!ELEMENT Items (Item)*〉
〈!ELEMENT Item EMPTY〉
〈!ATTLIST Item Desc CDATA #REQUIRED〉
〈!ELEMENT Payments (Payment)*〉
〈!ELEMENT Payment EMPTY〉
〈!ATTLIST Payment Desc CDATA #REQUIRED〉
```

4 Conclusion and Future Work

XML is rapidly emerging as the dominant standard for exchanging data on the World Wide Web, making the ability to publish data as XML increasingly important. In this paper, we have studied the characteristics of both relational schema and XML DTD. We proposed transformation rules using ERDs and DTD graphs and presented an algorithm based on these rules. By storing relational schema information in an edge table, the transformation process is efficient and straightforward.

Possibilities for future work include establishing standard DTDs, storing XML data economically, and querying XML data efficiently. We believe that the approach outlined in this paper can be adopted in the optimization of querying XML data with XQuery[6]. Next, we will focus our research on optimizing XQuery over relational data using DTDs achieved from our approach.

References

[1] Bray T, Paoli J, Sperberg-McQueen C. Extensible markup language(XML) 1.0 [EB/OL]. http://w3.org/XML,2002-08-15.

[2] Bosak J. W3C XML specification DTD [EB/OL]. http://w3.org/xml,2002-08-15.

[3] Deutsch A, Fernandez M, Suciu D. Storing semi-structured data with STORED [A]. In: *SIGMOD Conference* [C]. Dallas, Texus, 2000.

[4] Florescu D, Kossman D. Storing and querying XML data using and RDBMS [J]. *IEEE Data Engineering Bulletin*, 1999, **22**(3):27 – 34.

[5] Jayavel Shanmugasundaram. Relational databases for storing XML documents: limitations and opportunities [A]. In: *VLDB Conference* [C]. Edinburgh, Scotland, 1999.

[6] Don Chamberlin, Daniela Florescu, Jonathan Robie, et al. XQuery: A query language for XML W3C working draft [R]. Technical Report WD-xquery-20010215, World Wide Web Consortium, 2001.

一种关系模式的 DTD 发布方法

梁作鹏　王晓玲　徐立臻　董逸生

(东南大学计算机科学与工程系, 南京 210096)

摘　要　提出了一种基于边表的关系模式的 DTD 发布方法. 在分析了关系模式和 DTD 的特征的基础上,给出了它们之间的映射规则. 然后,给出了基于边表的转换算法. 获取关系数据字典中的关系模式信息和基于边表的转换算法是该方法的 2 个关键点. 关系数据模式发布为 DTD 文档后,其他应用就可以利用 DTD 所包含的结构信息对关系数据进行优化查询.

关键词　XML; DTD; 关系数据库; XML 模式

中图分类号　TP39