An Analysis of Element Information in XML Documents

Sungrim Kim[1] and Yong-ik Yoon[2]
[1] Computer Science Department, Dongduk Women's University
23-1 Wolgok-dong, Sungbuk-ku, Seoul, Korea 136-714
Tel. +82-2-940-4589, Fax.: +82-2-940-4194
[2] Multimedia Department, Sookmyung Women's University
Chungpa-Dong 2-Ka, Yongsan-Gu, Seoul, Korea 140-742
e-mail : srkim@dongduk.ac.kr, yiyoon@sookmyung.ac.kr

Abstract: This paper proposes the way to analyse XML docuements according to the element information. XML documents, which are becoming new standard for expressing and exchanging data in the Internet, don't have defined schema. It is not adequate to directly apply XML documents to the existing relational database or object-oriented database query language. Research on how to extract schema for XML documents and query language is going on actively. For users' query, the results could be too many or too less. It is important to give the users adequate results. Our proposed analysis method can be reduced or extended to correspond to the users' query more flexibly.

1. Introduction

XML is becoming a new standard for expressing and exchanging data in the Internet[1]. XML tags describe data themselves and can show documents in various forms.

XML documents are composed of groups of tag elements that show data structure. Even though it doesn't have schema that the other existing database have, it can be said that each document has a structure (DTD: Document Type Definition). The structure of XML data model is different from existing database. It is not proper to apply the existing relational database or object-oriented database query language. As a result, research on how to extract schema for XML documents and query language is going on actively [5, 7, 8, 9].

This paper suggests the way to analyze element information of XML documents. This method of extracting schema reduces or extends the scope of query by applying it to many levelized schema when the query results are too many or too small. Then it can meet the requests of users efficiently.

The structure of this paper is as follows. Chapter 2 deals with research on schema extraction. Chapter 3 deals with theories that support this paper. Chapter 4 shows algorithm of extraction and examples. Chapter 5 explains the way to analyse element information. Chapter 6 concludes the paper.

2. Related Works

There is a way to extract common schema with maximum tree expression according to occurrence frequency of tree expression [10, 11]. At tree expression , schema extraction is done according to the following. The MINISUP(i.e. minimum support) of tree expression *te* is the number of documents that has weaker expression *te* than document *d*. If the MINISUP is greater than the MINISUP that a user defines, it can be said that *te* is frequent. If *te* is high and it is more frequent than the other tree expression, the *te* can be said that it has the "*maximum frequency*". It has the merit of executing similar queries efficiently by making tree expression equation for frequently occurring similar queries and extracting Schema based on it. But it has a weak point that it is hard to find schema for the entire document.

The way to find occurrence frequency pattern has been studied in the area of transaction database time series database and the other database area. Among many methods, the way to find maximum pattern has been suggested by establishing occurrence frequency pattern tree (*FP-tree*) [3]. It has the merit of extracting variety of Schema based on the number of user-defined occurrence frequency. But it can extract special pattern of schema rather than the entire schema and it repeats schema extraction steps every time for the user-defined value for the special pattern and the number of occurrence frequency.

Lore is database management system for XML developed by Stanford University [14]. As XML documents don't have schema that was pre-defined, it is difficult to create meaningful queries if there aren't tags and attribute pattern. Query engine should understand the database structure to perform the queries efficiently. For the feature, Lore provides *DataGuide*. DataGuide shows clear and dynamic arranged structure for XML database and performs the role of database schema and DTD. The user can understand the entire structure of database through DataGuide and create queries. DataGuide can understand the entire schema for all documents. But as data in all documents are expressed, the created schema could be maximized and the searching area in XML documents could be widened.

3. Definitions

3.1 Structure of Element Information

Two graphs are defined in this paper for analysis.

Definition 1 : Data Graph

It defines edge labeled directed graph that describes all data in XML documents as *Data Graph*. Edge is made from root node to lower nodes and the label of edge becomes the name of element.

Definition 2 : Schema Graph

The graph that is created to express the all paths only once using the depth priority exploration method in Data Graph for XML documents is called *Schema Graph*.

3.2 Label path indexing using bitmap indexing

Basically, bitmap indexing uses 0 or 1 to express whether the attribute in tuple has special value or not [12]. The merit of bitmap indexing is that it can increase the performance speed with hardware operation of bitwise-AND, OR, NOT calculation. Because of this merit, it is widely used in opinion determinations system and dataware housing [4, 10].

This paper creates more dynamic Schema Graph with bitmap indexing of label path at XML documents defined as follows.

Definition 3 : *Label Path*

The path from one node to a certain lower node in Data Graph or Schema Graph is defined as Label Path. Edge is existent between nodes in Graph. The label with element name is existent. The middle node that appears in the path from root node and leaf node is expressed as . (dot).

4. Analysis of Element

4.1 Example of DTD and XML documents

The DTD and XML documents are based on movies in IMDB Top 250 films from http://www.imdb.com/ .

The information on movies includes the subject, year of production, director, scenario writer, genre, actor, awards history, families, language, nation, color and keyword

Some XML documents can be made based on DTD in table 4.1. It is shown in Figure 4.1.

```
<!ELEMENT movie (title, year, director+, writer+,
          genre+, cast+, language*, country*, color*,
          keywords* ) >
<!ELEMENT title (#PCDATA) >
<!ELEMENT year (#PCDATA) >
<!ELEMENT director ((lastname, firstname)
                    | fullname)>
<!ELEMENT writer ((lastname, firstname)
                    | fullname)>
<!ELEMENT lastname (#PCDATA) >
<!ELEMENT firstname (#PCDATA) >
<!ELEMENT fullname (#PCDATA) >
<!ELEMENT genre (#PCDATA) >
<!ELEMENT cast (name,role, (award, category)*,
spouse* >
<!ELEMENT name (#PCDATA) >
<!ELEMENT role (#PCDATA) >
<!ELEMENT award (#PCDATA) >
<!ELEMENT category (#PCDATA) >
<!ELEMENT spouse (name, occupation)* >
<!ELEMENT occupation (#PCDATA) >
<!ELEMENT language (#PCDATA) >
<!ELEMENT country (#PCDATA) >
<!ELEMENT color (#PCDATA) >
<!ELEMENT keywords (#PCDATA) >
```

Table 4.1 A DTD Example : movie.dtd

```
<movie>
  <title>Citizen Kane </title><year>1941</year>
  <director>Orson Welles</director>
  <writer>Herman J. Mankiewicz </writer>
  <writer><firstname>Orson</firstname>
        <lastname>Welles</lastname></writer>
  <genre>Drama</genre>
  <cast> <name>Orson Welles</name>
    <role>Charles Foster Kane </role>
    <award>Oscar</award>
    <category>Best Writing, Original
        Screenplay</category>
  <spouse><name>Rita Hayworth </name>
      <occupation>divorced</occupation></spouse>
  </cast>
  <cast><name>Dorothy Comingore</name>
    <role>Susan Alexander Kane </role>
  <spouse><name>Richard Collins (I) </name>
      <occupation>?</occupation></spouse>
  </cast>
  <language>English</language>
  <country>USA</country>
  <color>Black and White</color>
  <keywords>sled </keywords>
  <keywords>dying-words </keywords>
</movie>
```

Figure 4. 1 A Sample XML Document

4.2 Data Graph and Schema Graph

4.2.1 Data Graph and Schema Graph

If the XML document in Figure 4.1 is expressed in Data Graph, it is shown as in Figure 4.2. As defined in 3.1, the DataGraph of Figure 4.2 expresses all elements in XML document. If the XML document in Figure 4.1 is expressed in Schema Graph, it is shown as in Figure 4.3. As defined in 3.1, Schema Graph of Figure 4.3 expresses the all label paths only once.

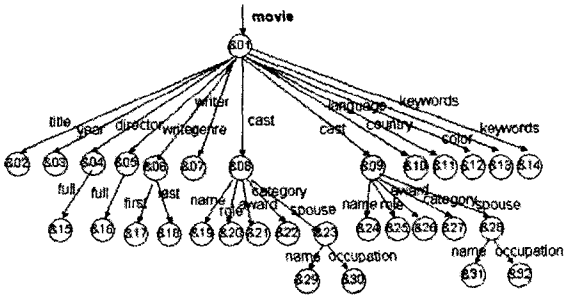

Figure 4.2 Data Graph

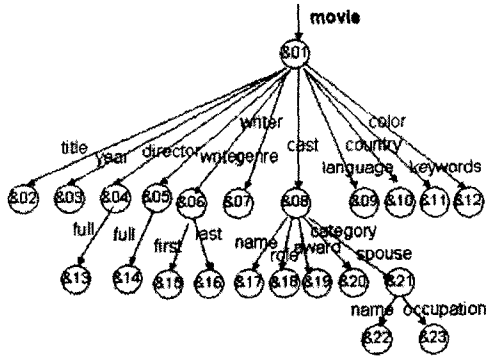

Figure 4.3 Schema Graph

4.3 Schema extraction using bitmap indexing

After creating Schema Graph for XML document and showing whether the label path is existent in XML document or not, it can create variety of Schema Graphs according to the frequency.

4.3.1 Label path

The label path for Schema Graph is added based on depth priority exploration method from root node to leaf node. This method is the same to that of algorithm 4.1.

4.3.2 Schema extraction using label path frequency

If Bitwise-OR calculation is performed for bitmap index for all XML documents. It creates Schema Graph including all label paths. This graph is same to Schema Graph that is created in data graph. The algorithm is shown in algorithm 4.2.

Using the occurrence frequency of each label path, it creates many steps of Schema Graph. Dynamic Schema Graph is to reduce or extend the scope of XML documents dynamically for efficiently performing the users' queries. The algorithm is shown in 4.4. The method to find label path bitmap according to limited value is described in algorithm 4.5.

```
// Algorithm 4.1
MakeLabelPath (treeNode u)
 begin
   path = u.label
   while (u isNot leafNode)
    begin
        v = u.childNode
        path = path + v.label
     u = v
    end
end

// Algorithm 4.2
 CalcBitVector(document d)
   begin
     while (pi isNot EOF)
       if (pi in d) d(pi) = 1
       else    d(pi) = 0
       endif
  end

// Algorithm 4.3
 bitwiseORLabelPath()
 begin
    for each i in path P
        new_path(i) = bitwise_OR( djPi)
end

// Algorithm 4.4
 countLabelPath()
 begin
   for each i in path P
        Freq(i) = count( dj)
End
```

```
// Algorithm 4.5
 FilterFreq(threshold t)
 begin
   while (Freq(i) isNot EOF)
        if (Freq(i) >= t)  New_Freq(i) = 1
        else  New_Freq(i) = 0
        end if
end
```

Algorithms for schema extraction using bitmap indexing

5. Implementation and Experiments

5.1 Implementation and Setup

The environment for testing schema extraction method according to label path occurrence frequency is as follows. The operating system is Windows 2000 and the database is Oracle9i. It uses JDK 1.3.1 and JSP for accomplishment languages. It also used Oracle JDBC thin driver to be compatible to Oracle. The web server is IIS5.0. It uses resin 2.0.1 as JSP engine. It uses Oracle Parser (version 2.0.1.0) as XML parser.

It created and tested XML documents based on data except multimedia data among 100 movies from IMDB Top250films of http://www.imdb.com.

5.2 Experiments

Figure 5.1 shows the result of creation frequency of label path for each element based on 100 XML documents. The frequency of label paths that have family name and calling name of a director (*movie.director.firstname, movie.director.lastname*) was 49 among 100 data tested.

The creation frequency of actor, color, country, genre, keyword, subject and production year was 100 and it appears at all documents.

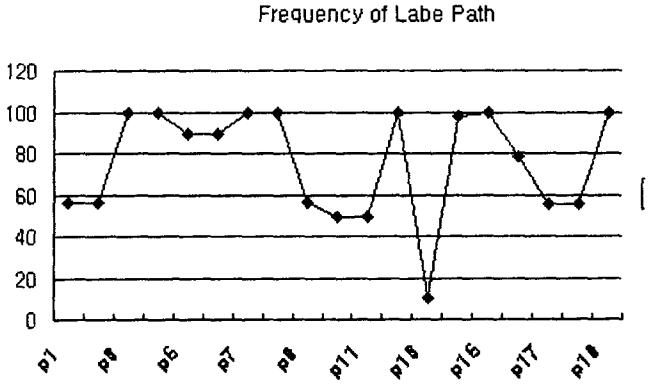

Figure 5.1 Frequency of Label Path

Next, we give limit value for creation frequency of label path. Figure 5.2 and Figure 5.3 show the result of extracted schema and the number of XML documents that can apply to the schema. The limit value increases from 0.05 to 1.0 with interval of 0.05.

In case the limit value is 0.05, the number of elements was 19. But the number of XML documents that include all the 19 elements was 2. In case the limit value is 0.5, the

number of elements included was 17 and the number of documents was 10. In case the limit value is 1, it includes elements that appear at all documents. In this case it includes 8 elements. The number of documents that satisfy it is 100.

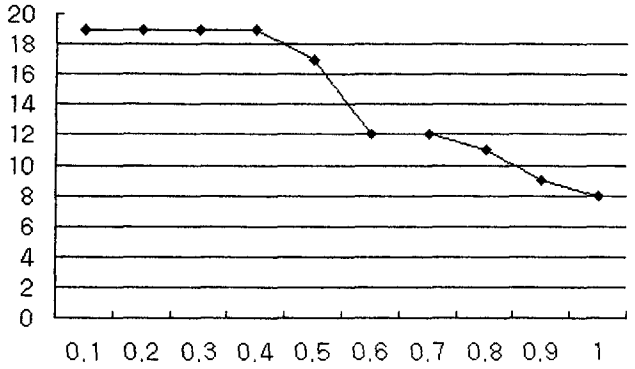

Figure 5.2 Number of elements

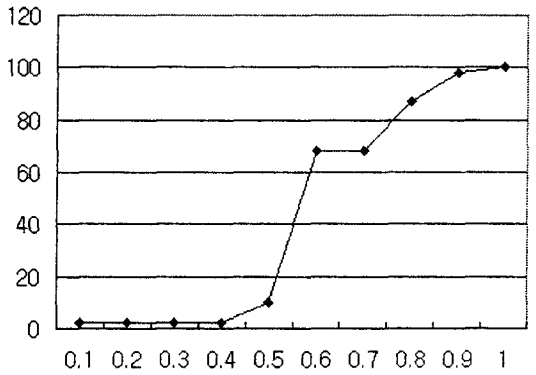

Figure 5.2 Number of XML Documents

6. Conclusion

XML doesn't have pre-defined schema. It has the data and data structure in documents themselves. Therefore it is difficult to directly apply SQL and OQL which are used in the existing RDB or OODB. Research on new queries for XML and Schema extraction for queries processing is active.

This paper suggests the way of schema extraction according to label path occurrence frequency in XML documents. It creates Schema Graph for XML documents that have same DTD and expresses the information of all elements in XML document only once. Then it expresses whether label path of input XML documents is existent or not with bit vector based on Schema Graph and calculates label path occurrence frequency at XML document. It can process the users' queries more efficiently by enabling many steps of Schema extraction according to a certain limited value.

This paper proves that dynamic application is possible to satisfy users' queries by enabling many steps of Schema extraction according to label path occurrence frequency shown in XML documents. If there are so small or so many results for users' queries, it can put limited value to reduce or extend the results for users' queries.

Research on finding more efficient way of Schema extraction shall be needed in the future for analyzing it not only with the shape but also in meaning of data considering the way of Schema extraction using meta data for multimedia documents.

References

[1] Jon Bosak, "XML, Java, and the Future of the Web", http://webreview.com/wr /pub/97/12/19/xml/index.html

[2] Roy Goldman, Jennifer Widom, "DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases", In Proceedings of VLDB, 1997

[3] Jiawei Han, Jian Pei, Yiwen Yin, "Mining Frequent Patterns without Candidate Generation", Proceedings of the 2000 ACM SIGMOD on Management of data, 2000, pp. 1-12

[4] Theodore Johnson, "Performance Measurements of Compressed Bitmap Indices", VLDB 1999, pp. 278-289

[5] Alon Levy, "More on Data Management for XML", University of Washington, May 9th, 1999. http://www.cs.washington.edu/homes/alon/widom-response.html

[6] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, J. Widom, "Lore : A Database Management System for Semistructured Data", SIGMOD Record, 26(3), pp.54-66, September 1997

[7] Jayavel Shanmugasundaran, Kristin Tufte, Gang He, Chun Zhang, David DeWit, Jeffrey Naughton, "Relational Databases for Querying XML Documents: Limitations and Opportunities", Proceedings of the 25th VLDB Conference 1999

[8] Dan Suciu, "Semistructured Data and XML", In Proceedings of International Conference on Foundation of Data Organization, 1998

[9] Jennifer Widom, "Data Management for XML",Working Document, initial draft appeared April 1999, Also IEEE Data Engineering Bulletin, Special Issue on XML, 22(3):44-52, September 1999.

[10] Ke Wang, Huiqing Liu, "Schema Discovery from Semistructured Data", International Conference on Knowledge Discovery and Data Mining, August 1997, pp.271-274

[11] Ke Wang, Huiqing Liu, "Discovering Typical Structures of Documents : A Road Map Approach", The ACM SIGR conference on Research and Development in Information Retrieval, August 1998, pp.146-154

[12] Ming-Chuan Wu,, "Query optimization for selections using bitmaps", Proceedings of the 1999 ACM SIGMOD international conference on Management of data, pp.227-238

[13] J. Yoon, S. Kim, "Schema Extraction for Multimedia XML Document Retrieval", in Proc. of International Database Symposium on Mobile, XML and Post-Relational Databases, Hong Kong, June 2000

[14] S. Abiteboul, D. Quass, J. McHugh, J. Widom and J. Wiener, "The Lorel Query Language for Semistructred Data", International Journal of Digital Libraries, 1(1):66-88, 1997

[15] Serge Abiteboul, Peter Buneman, Dan Suciu, "Data on the Web : From Relations to Semistructured Data and XML", Morgan Kaufmann, 2000