AJCSE
ASIAN JOURNAL OF COMPUTER SCIENCE ENGINEERING

www.ajcse.info

Asian Journal of Computer Science Engineering 2016; 2(1):19-23

RESEARCH ARTICLE

A Study on Mapping Semi-Structured Data to a Structured Data for Inverted Index Compression

[1]B.Usharani*

[1]Dept. of CSE, INDIA

Received on: 02/09/2016, Revised on: 22/12/2016, Accepted on: 23/02/2017

ABSTRACT

Semi-structured data is used for representing the data over the internet. In this paper, an analysis is performed on how to convert XML documents to tuples, translating the semi-structured queries to SQL queries, and converts the results back to XML.

Keywords: Relational, Semi-Structured data, structured data, XML.

INTRODUCTION

Semi-structured is emerged as important because

- There is the need to handle different data sources that cannot be constrained by the fixed schemas.
- There is the need to maintain format for the data exchanges between remote databases.

There are different approaches to store semi-structured data and execute queries on the corresponding semi-structured data.

1. **DBMS**: In this approach, semi-structured data are mapped to tables in the relation schemas, and queries in a semi-structured data are translated into SQL queries.
2. **OODBMS:** In this approach the semi-structured data is mapped with Object-oriented databases. Semi-structured queries are translated to OOqueries.
3. **Special Purpose DBMS**: Special data structures are used to store and to put the queries .Examples are Lore etc.

The key attributes that distinguish semi-structured data over the structured data.

1) Lack of a rigid schema
2) Nested data structures.

The Structured data requires rigid schema and it should defined in advance, semi-structured data does not require any prior information about the definition of a schema in advance. Structured data, represents data in a table, semi-structured data can contain hierarchies of information.

Semi-structured data is usually stored in flat files, but it is difficult to query or update To store and use the Semi-structured data easily, there are two options.

1) Convert semi-structured data to a rigid schema before using it.
2) Store semi-structured data in a relational table. This approach simplifies loading of semi-structured data.

REPRESENTATION OF XML DATA AS A GRAPH

The XML document can be represented as a graph. Each XML element is represented as a node and the node is labeled with an id. Element to the sub-element relationships are represented by the edges in the graph and labeled by the name of the sub-element. To represent the order of sub-elements in the graph, outgoing edges of the graph is recorded. Values of an XML document are recorded as the leaves in the graph. Different approaches are used to store XML data in a structured form i.e in the relational database and discussed in the next section. The XML document is given below.

```
<? xml version="1.0" encoding="UTF-8"?>
<information>
   <tuple>
      <mid>m1 </mid>
      <moviename>          toy          story
(1995)</moviename>
      <generes>comedy drama </generes>
      <url>          http://us.imdb.com/M/title-
exact?Toy%20Story%20</url>
```

```
  </tuple>
   <tuple>
      <mid>m2 </mid>
      <moviename>jumanji (1995) </moviename>
      <generes>animation childrens comedy
</generes>
      <url>http://us.imdb.com/M/title-
exact?Jumanji%20</url>
    </tuple>
…………..
</information>
```

XML Tree for the XML document

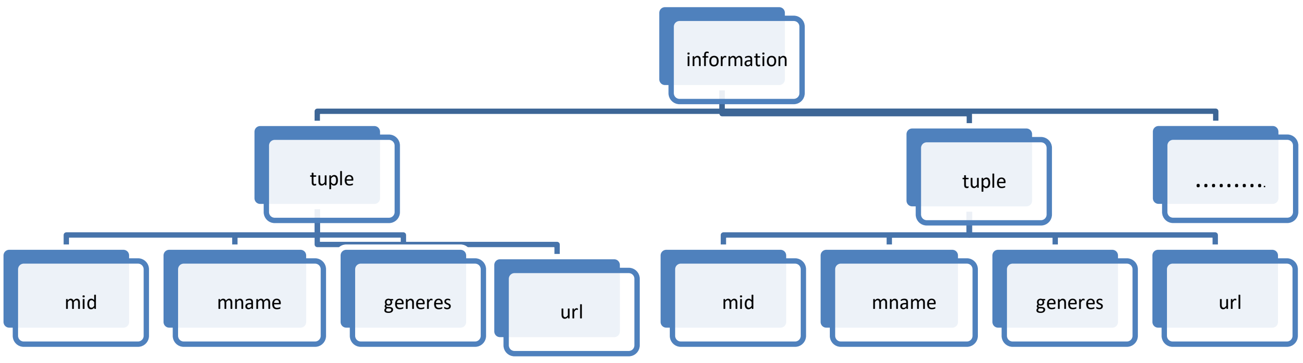

Fig 1: XML Tree Structure representation

STORING XML DOCUMENTS IN A RELATIONAL DATABASE

Different mapping techniques are discussed in this section that can be used to store the XML document in
a relational database.

ASCII Approach

Semi-structured data is stored as a normal text.

But this approach has several major drawbacks.

- The XML files in ASCII format need to be parsed every time when they are accessed.
- The original XML document must be memory-resident during query processing.
- Update operations are very difficult to implement.

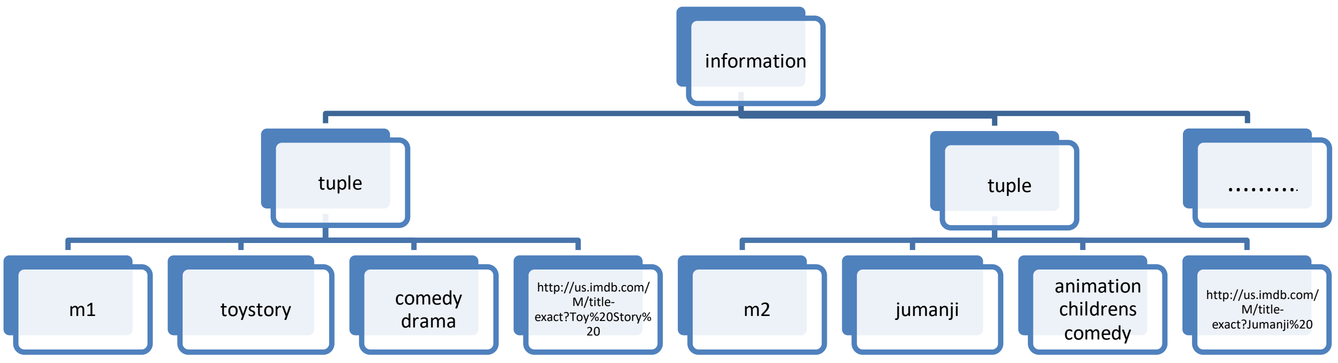
information
tuple
tuple
………
m1
toystory
comedy drama
http://us.imdb.com/M/title-exact?Toy%20Story%20
m2
jumanji
animation childrens comedy
http://us.imdb.com/M/title-exact?Jumanji%20

Figure 2: XML TREE

The Edge Table Approach

The directed graph of an XML file is stored in **Edge table**. Every node in the directed graph is assigned an id. Each tuple in the edge table corresponds to one edge in the directed graph and contains the ids of source and target of the edge, the tag of target element, and an ordinal number that is used to record the order of the children nodes. When an element has only one child, the text is stored with the edge.

The edge table has the following schema:

Edgetable (sourceID, ordinal, name,flag, targetID)

In this appraoch, user queries are written directly in SQL. Path expressions are evaluated by repeatedly joining the edge table with itself, and, eventually, with the value tables. [4,6]

The sourceID column is used for forward traversal.

The name,targetID column is used for backward traversal.

DrawBack:

- This approach is very large, too expensive, and more redundancy.
- The Edge approach is not suitable for heavy queries because joins with the Edge table becomes expensive

Advantage:

- Reduces the no: of tables in the database and the amount of disk space consumed.

SourceID	Ordinal	Name	Flag	TargetID
1	101	mid	String	M1
1	102	mname	String	toystory
1	103	generes	String	comedydrame
1	104	url	String	http://us.imdb.com/M/title-exact?Toy%20Story%20
2	**105**	**Mid**	**String**	**M2**
2	106	mname	String	jumanji
2	107	Generes	String	Childrenanimationcomedy
2	108	url	String	http://us.imdb.com/M/title-exact?Jumanji%20
……	……….	……… …..	……… …….	………………………… ………….

Figure 3: EdgeTable for figure2

The Attribute Table Approach

This approach corresponds to the horizontal mapping of the Edge Table. A attribute table is maintained for each existing label [4,6].

The schema for the attribute table is given below:

Attributetable(sourceID, ordinal, targetID)

Drawback:

- It is so complex to do search and perform the update.
- The attribute approach divides a XML file into many relations and increases the disk space.
- Difficult to write the XML query in SQL.

SourceID	Ordinal	TargetID
1	101	M1
2	105	M2
3	109	M3

Figure 4: Attribute Table for mid for figure2

The Universal Table Approach

This approach generates a single universal table to store all the edges. The universal table matches the result of an outer join of all attribute tables [4, 5, 6, and 7]. The universal table has the separate columns for the entire attribute name that occur in the XML document.

The schema for the universal table is:

Universaltable(sourceID,ordinal(n1),targetID(n1), ordinal(n2),targetID(n2),……..,ordinal(nm),target ID(nm));

Drawback:

- The universal table has fields which are set to null, and it is a deal of redundancy.
- Nobody knows the no: of attributes and the attributes' name in some cases.
- Performs badly for heavy queries.

Source	Ordinal(mid)	Target (mid)	……	Oridinal(generes)	Target(generes)	..	..
1	101	M1		103	comedydrama		
1	101	M2		107	childrenamimationcomedy		
2	105	M3		111	actionthriller		
3	109	M4		115	animation		

Figure 5: Universal Table for figure2

The normalized universal approach

This approach is an alternate of the universal approach [4, 7].The difference is that the multi-valued attributes are stored in separate overflow tables in the normalized universal approach.

The schema for the universal table is:

Universalnormtable(sourceID,ordinal(n1),targetID (n1),ordinal(n2),targetID(n2),……..,ordinal(nm),t argetID(nm));

Overflowtable(sourceID,ordinal,targetID);

Eg:

Source	Ordinal(mid)	Target (mid)	……	Oridinal(generes)	Target(generes)	..	..
1	101	M1		103	comedydrama		
2	105	M2		107	childrenamimationcomedy		
3	109	M3		111	actionthriller		

SourceID	Ordinal	TargetID
1	101	101
1	105	105
2	109	109

Figure 6: Universal normalized Table, Overflow table for figure 2

STORED

This approach uses the concept of the OEM model and the relational database management system to store and manage semi-structured data [2, 3].

Drawback:

- Many fields are set to null, and it leads to redundancy
- Does not handle multi-valued attributes, when ever retrieving the multi-valued attributes in uses join operation which is expensive.
- This approach may not store all data.

MID	MNAME	GENERES	URL

M1	Toystory	comedydrama	http://us.imdb.com/M/title-exact?Toy%20Story%20
M2	jumanji	childrenanimationcomedy	http://us.imdb.com/M/title-exact?Jumanji%20

Figure 7: STORED storage for figure 2

Stores each XML element of the XML file as a separate object [1, 6].

The drawback of this approach is, the file needs to be frequently updated.

Shore

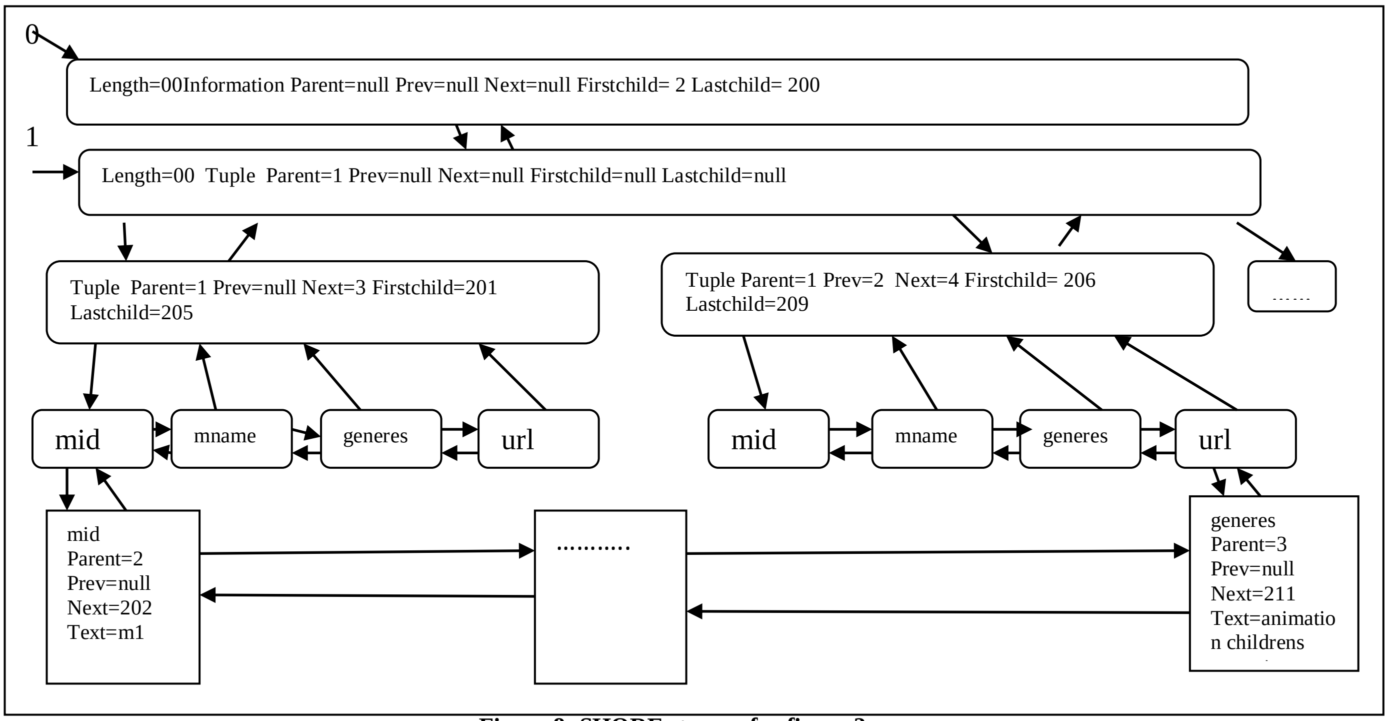
0
Length=00Information Parent=null Prev=null Next=null Firstchild= 2 Lastchild= 200
1
Length=00 Tuple Parent=1 Prev=null Next=null Firstchild=null Lastchild=null
Tuple Parent=1 Prev=null Next=3 Firstchild=201 Lastchild=205
Tuple Parent=1 Prev=2 Next=4 Firstchild= 206 Lastchild=209
……
mid
mname
generes
url
mid
mname
generes
url
mid Parent=2 Prev=null Next=202 Text=m1
………..
generes Parent=3 Prev=null Next=211 Text=animation childrens

Figure 8: SHORE storage for figure 2

B-tree approach

This approach eliminates the drawback of the Shore approach. Every object maintains a key. [1,6].The B-tree automatically manages the disk space in leaves and there is no need to move object when the file size is updated .

Drawback:

But when doing search, we do not know the key-value of the object.

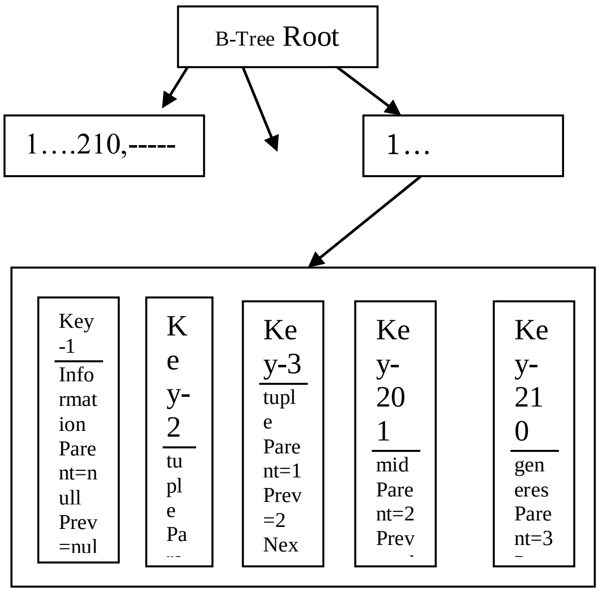

Figure 9: B-TREE storage for figure 2

Inlining approach (DTD to Relational Schema)

Inlining- Put as many sub-elements of an element as possible into a single relation.

Techniques to translate XML DTD to Structured tables are:

- Basic Inlining Technique
- Shared Inlining Technique
- Hybrid Inlining technique

Basic Inlining Technique

- Every DTD element maintains a relation.

Advantages

- Reduces the no: of joins.

Disadvantages

- Generates many relations
- Separate schema for each element.

Shared Inlining Technique

Identify commonly used element nodes and share them by creating separate relations for them. Ensure that an element node is presented in exactly one relation

Problems with Shared

Many joins are required

Advantages

- Reduces the no: of joins for the queries

Disadvantages

- Extra joins are required.

Hybrid Inlining Technique

- Inlines some elements that are shared in the Shared Technique.

Advantages

- Reduces the no: of joins.

Disadvantages

- Requires more sub-queries in SQL.

CONCLUSION

This paper presents a view for storing the information of the XML document in structured databases. In this paper, different mapping schemes are discussed to store XML data in a rigid schema i.e in the structured database. For small datasets, Edge-table approach is the suitable one. In future, the Edge-table approach is implemented for constructing the inverted index.

REFERENCES

1. M. Carey, D. DeWitt, J. Naughton, M. Solomon, et. al, Shoring Up Persistent Applications, Proc. of the 1994 ACM SIGMOD Conference
2. Alin Deutsch, Mary F. Fernandez, and Dan Suciu. Storing Semistructured Data in Relations, ICDT’99
3. Alin Deutsch, Mary F. Fernandez, and Dan Suciu. Storing semistructured data with STORED. In SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadephia, Pennsylvania, USA, pages 431-442, 1999
4. D. Florescu, D. Kossman, A Performance Evaluation of Alternative Mapping Schemes for Storing XML Data in a Relational Database, Rapport de Recherche No. 3680 INRIA, Rocquencourt, France, May 1999
5. Daniela Florescu, Donald Kossmann: Storing and Querying XML Data using an RDMBS. IEEE Data Engineering Bulletin 22(3): 27-34(1999)
6. Feng Tian, David J. DeWitt, Jianjun Chen, Chun Zhang, The Design and Performance Evaluation of Alternative XML Storage Strategies, 1999
7. Jeffrey D. Ullman. Principles of Database and Knowledgebase Systems, Volumes I, II. Computer Science Press, Rockville MD, 1989.