Knowledge discovery from XML Database

Pravin P. Chothe[1]
PG Scholar, ADCET,
Maharashtra, India

Prof. S. V. Patil[2]
Professor, ADCET Ashta,
Maharashtra, India

Prof.S. H. Dinde[3]
Professor, SGI, Atigre,
Maharashtra, India

Abstract—Data Mining is used to extract interesting knowledge from large amount of data. This knowledge can be represented in different forms such as decision tree, decision rules, clusters and so on. Among them mainly association rules can be used as effective tool to discover interesting relations in large amount of data. Extracting knowledge from semi structured documents is a very complicated task, and is going to become more complicated as the amount of digital information available on the Internet grows. Indeed, documents are so big that the data returned as answer to a query may be too large to convey interpretable information about the document. In this paper, we use an approach which is based on Tree-Based Association Rules (TARs): first we have to determine the frequent sub trees in given document and then depending support and confidence generate rules, which provide approximate, intensional knowledge on both the structure and the contents of Extensible Markup Language (XML) documents, and can be stored in XML format as well. This mined knowledge is later used to provide: 1) a concise idea—the gist—of both the structure and the content of the XML document and 2) quick, approximate answers to queries. In this paper, we focus on the second feature. A prototype system and experimental results demonstrate the effectiveness of the approach.

Keywords- XML, approximate query-answering, data mining, intensional information, succinct answers.

I. INTRODUCTION

In recent years, the Extensible markup language is used to represent large amount of data without fixed schema and irregular and incomplete structure. Extensible markup language represent the data in hierarchical model which is suitable to represent large amount of information. XML document can be accessed in two different ways. The firs way is by using keyword-based search and second way is by using query-answering [1]. In the first method we can search the XML document by using keyword and get the result. In second method we get the result by applying query over the XML document.

The first one comes from the tradition of information retrieval [2], where most searches are performed on the textual content of the document. As for Query-answering, as query languages for semi structured data is depend on the document structure to convey its semantics, in order to make query formulation to be effective users must know the structure of document in advance, which is often not the case. When the user provides a query even without knowing the structure of document they fail to get the result which was there but under the different structure. The extraction of intensional information through the use of data mining techniques has been proposed in the literature, both with respect to the relational model [3] and to the XML format [4].

Accessing for the first time a large dataset, gaining some general information about its main structural and semantic characteristics helps investigation on more specific details. Discovering recurrent patterns inside XML (documents) provides high-quality knowledge about the document content: Frequent patterns are in fact intensional information about the data contained in the document itself, that is, they specify the document in terms of a set of properties rather than by means of data.

To represent intensional knowledge in native XML document we mine and store TARs (Tree based Association Rules) TAR represents intensional knowledge in the form of SB => SH, where SB is the body of tree and SH is head tree of the rule and SB is a sub tree of SH. The rule SB=>SH states that, if the tree SB appears in an XML document D, it is likely that the "wider", tree SH also appears in D. The intensional information stored in the TARs provides a valid support in several cases:

1) It allows obtaining and storing implicit knowledge of the documents, useful in many respects:
   (i) To get a concise idea – the gist – of both the structure and the content of an XML document quickly without knowing its internal structure.
   (ii) TARs represent a data guide that helps users to be more effective in query formulation.

2) TARs can be used to query to obtain fast and approximate, answers. This is useful when the original documents are unavailable. In fact, once extracted, TARs can be stored in a (smaller) document and be accessed independently of the dataset they were extracted from.

MOTIVATION

The motivation behind this work can be summarized as follows. Database research field has concentrated on the Extensible Markup Language (XML) is used as flexible hierarchical model which is suitable to represent huge amounts of data without fixed schema.

For query-answering, since query languages for semi structured data depend on the document structure to provide its semantics, therefore to make query formulation more effective users need to know this structure in advance.
When users provide queries without knowing the document structure, they may fail to retrieve knowledge which was there, but under a different structure. This limitation is a crucial problem which did not emerge in the context of relational database management systems.

LITERATURE REVIEW

Association rules describe the co-occurrence of the data items in the large amount of stored data and can be represented as implication of the form A $\Longrightarrow$B where A and B are two separate set of data items. The notion of association rule can be extend which is introduced in the context of relational databases to adapt it to the hierarchical nature of XML documents. Given two trees T =<N_T, E_T, r_T, l_T, c_T> and S =<Ns, Es, rs, ls, cs> S is an induced subtree of T if and only if there exists a mapping Θ=N_S->N_T such that for each node n_i€N_S, $l_T(n_i)=l_S(n_j)$ and $c_T(n_i)=c_S(n_j)$ where $\Theta(n_i)=n_j$. Moreover, S is a rooted subtree of T if and only if S is an induced subtree of T and $r_S=r_T$. Given a tree T =<N_T, E_T, r_T, l_T, c_T>, a sub tree of T, t==<N_t, E_t, r_t, l_t, c_t> and a user-fixed support threshold $s_{min.}$ 1) t is frequent if its support is greater or at least equal to $s_{min..}$ 2) t is maximal if it is frequent and none of its proper supertrees is frequent; and 3) t is closed if none of its proper supertrees has support greater than that of t.
For obtaining XML query answer the System architecture is as shown bellow.

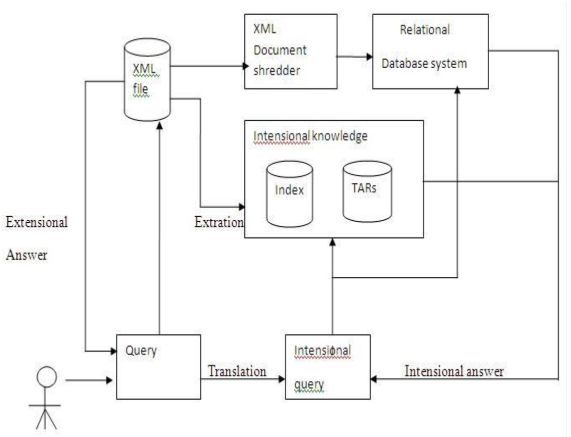


Figure :- System Architecture

Given an XML document, it enables users to extract intensional knowledge and compose traditional queries as well The purpose of this framework is to perform data mining on XML and obtain intensional knowledge. The intensional knowledge is also in the form of XML. This is nothing but rules with supports and confidence. In other words the result of data mining is TARs (Tree-based Association Rules).
Users formulate XQueries over the original data, and queries are automatically translated and executed on the intensional knowledge and relational database. Given an XML document, we extract two types of TARs:

1) A TAR is a structure TAR (sTAR) iff, for each node n contained in SH, cH(n) = $\perp$, that is, no data value is present in sTARs, i.e. they provide Information only on the structure of the document.
2) A TAR, SB=>SH, is an instance TAR(iTAR) iff SH contains at least one node n such that cH(n)$\neq\perp$, that is, iTARs provide information both on the structure and on the data values contained in a document.
TARs provide an approximate view of both the content and the structure of an XML document,
1) sTARs can be used as an approximate Data Guide of the original (document, to help users formulate queries;
2) iTARs can be used to provide intensional, approximate answers to user queries.

TAR EXTRATION

TAR mining is a process composed of two steps:

1. Mining frequent subtrees, that is, subtrees with a support above a user defined threshold, from the XML document.
2. Computing interesting rules, that is, rules with a confidence above a user defined threshold, from the frequent subtrees.

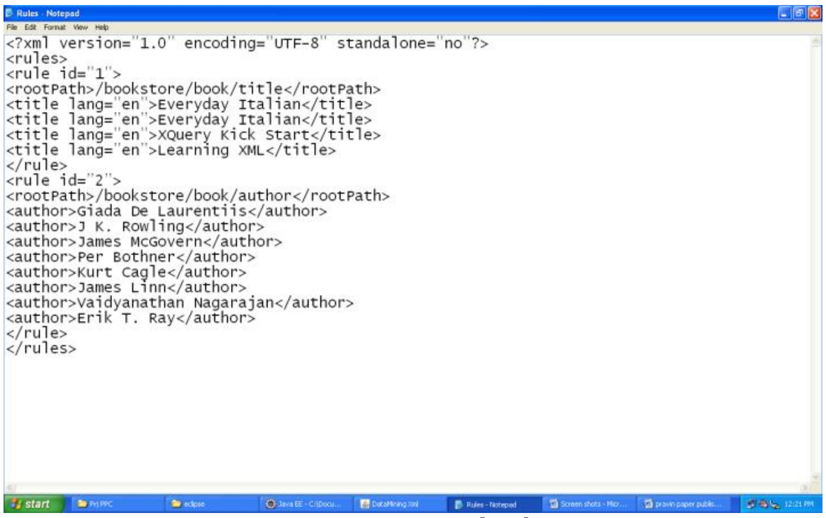
Algorithm 1 finds frequent sub trees and calculates interesting rules. Algorithm 1 represents our extension to a generate frequent subtree-mining algorithm in order to compute interesting TARs. The inputs of Algorithm 1 are the XML document D, the threshold for the support of the frequent sub trees minsupp, and the threshold for the confidence of the rules, minconf.

Algorithm 1. Get-Interesting-Rules (D, minsupp, minconf)

```
1: // frequent subtrees
2: FS = FindFrequentSubtrees (D, minsupp)
3: ruleSet = ∅;
4: For all s ϵ $F_{s}$ do
5:// rule compute from s
6: tempset= Compute-rules(s, minconf)
7:// all rules
8:ruleSet = ruleSet ∪ tempset
9: end for
10: return ruleSet
```

Function 1 Compute-Rules (*s*, *minconf*)

```
1: ruleSet = ∅; blackList = ∅
2: for all cs, subtrees of s do
3: if cs is not a subtree of any element in blackList then
4: conf = supp(s) / supp(cs)
5: if conf ≥ minconf then
6: newRule = {cs, s, conf, supp(s)}
7: ruleSet = ruleSet ∪ {newRule}
8: else
9: blackList = blackList ∪ □cs
10: end if
11: end if
12: end for
13: return ruleSet
```



```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<rules>
<rule id="1">
<rootPath>/bookstore/book/title</rootPath>
<title lang="en">Everyday Italian</title>
<title lang="en">Everyday Italian</title>
<title lang="en">XQuery Kick Start</title>
<title lang="en">Learning XML</title>
</rule>
<rule id="2">
<rootPath>/bookstore/book/author</rootPath>
<author>Giada De Laurentiis</author>
<author>J K. Rowling</author>
<author>James McGovern</author>
<author>Per Bothner</author>
<author>Kurt Cagle</author>
<author>James Linn</author>
<author>Vaidyanathan Nagarajan</author>
<author>Erik T. Ray</author>
</rule>
</rules>
```

Figure1: Generated Rules

Algorithm 1 finds frequent sub trees and then hands each of them over to a function that computes all the possible rules as shown in figure 1. Depending on the number of frequent sub trees and their cardinality, the amount of rules generated by a naïve Compute Rules function may be very high. Given a subtree with n nodes, we could generate 2n-2 rules. The rules obtained from algorithm 1 are written to an XML file. Then indexing is made. Afterwards when XML queries are made, the proposed system uses index and TARs and quickly answers the query.

Finally, the rules in the XML file are sorted on the number of nodes of their antecedent; this is an important feature that is used for optimizing the answering of queries which are containing a count operator. One reasons for using TARs instead of using original document is that processing the original document take more time than TARs. To make the use of this, we introduce indexes on TARs to reduce the access to mined trees - and in general of intensional query answering [5]. In general, path indexes are proposed to quickly answer queries that follow some frequent path template, and are built by indexing only those paths having highly frequent queries. We start from a different perspective: we want to provide quick, and often approximate, answers also to casual queries.

Given a set R of rules, the index associates, with every path p present in at least one rule of R, the references to rules that contain p in SH. An index is an XML document containing a set of trees T1, ...,Tn such that each node n of each tree Ti contains a set of references to the rules containing in SH the path from the root of Ti to n. A TAR-index contains references both to iTARs, sTARs, and is constructed by Algorithm 2.

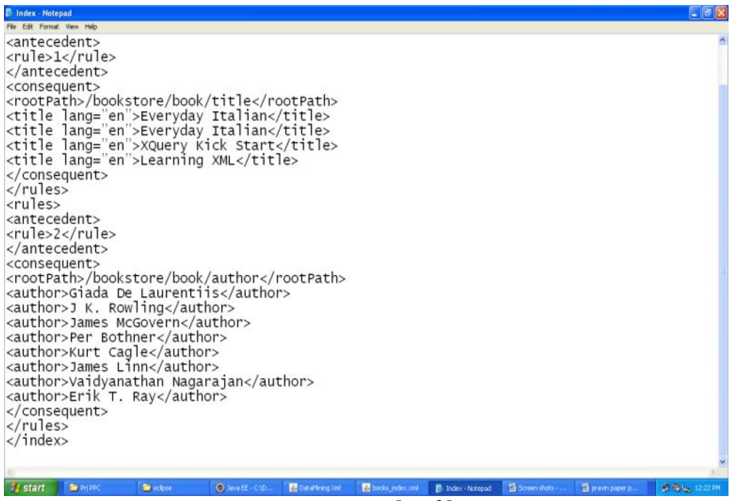
Algorithm 2 Create-Index (D)

```
1: for all Di ∈ D do
2:   for all dj ∈ Di with j∈ {2, 3 . . . n} do
3:     references (root (d1)) = references (root (d1)) ∪
         references (root (dj))
4:     sumChildren (d1, dj)
5:   end for
6: end for
7: return D
```

Function 2 sumChildren (T1, T2)

```
1: for all x ∈ children (root (T2)) do
2:   if □□c ∈ children(root(T1)) | c = x then
3:     references (root(c)) = references (root(c)) ∪ references (root(x))
4:     c = sumChildren(c, x)
5:   else
6:     addChild (root (T1), x)
7:   end if
8: end for
9: return
```

Before applying the algorithm, two sets A and C are constructed which contain the antecedent and consequent trees of all the TARs to be indexed. Each tree Ti in the index is annotated in a way that each node contains the reference to the ID of the rule it comes from; then trees are scanned looking for those that have the same root. For each node n of the resulting tree a set of references is stored, pointing to the TARs that contain the path from the root of the tree to node n. Once the algorithm is applied to all TARs, the result is a set of trees whose nodes contain references to one or more rules and which are stored in an XML file to be queried later on. Index file is as shown in figure 2.



```
<antecedent>
<rule>1</rule>
</antecedent>
<consequent>
<rootPath>/bookstore/book/title</rootPath>
<title lang="en">Everyday Italian</title>
<title lang="en">Everyday Italian</title>
<title lang="en">XQuery Kick Start</title>
<title lang="en">Learning XML</title>
</consequent>
</rules>
<rules>
<antecedent>
<rule>2</rule>
</antecedent>
<consequent>
<rootPath>/bookstore/book/author</rootPath>
<author>Giada De Laurentiis</author>
<author>J K. Rowling</author>
<author>James McGovern</author>
<author>Per Bothner</author>
<author>Kurt Cagle</author>
<author>James Linn</author>
<author>Vaidyanathan Nagarajan</author>
<author>Erik T. Ray</author>
</consequent>
</rules>
</index>
```

Figure2: Index file

INTENSIONAL ANSWERS

iTARs(intensional Tree-based Association Rules) provide an approximate intensional view of the content of an XML document, which is in general more concise than the extensional one because it describes the data in terms of its properties, and because only the properties that are verified by a high number of items are extracted. A user query over the original dataset can be automatically transformed into a query over the extracted iTARs. The answer will be intensional, because, rather than providing the set of data satisfying the query, the system will answer with a set of properties that these data "frequently satisfy", along with support and confidence. There are two major advantages: i) querying iTARs requires less time than querying the original XML document; ii) approximate, intensional answers are in some cases more useful than the extensional ones. The classes of queries that can be managed with our approach have been informally introduced in [6] and further analyzed in the relational database context in [7].They includes the main *retrieval* functionalities of XQuery,

Experiments and Results

The classes of queries that are used in our approach are informally introduced in [8] and thereafter analyzed in the relational database context in [9]. They include the main retrieval functionalities of XQuery, i.e., path expressions, FLOWR expressions, and the COUNT aggregate operator.

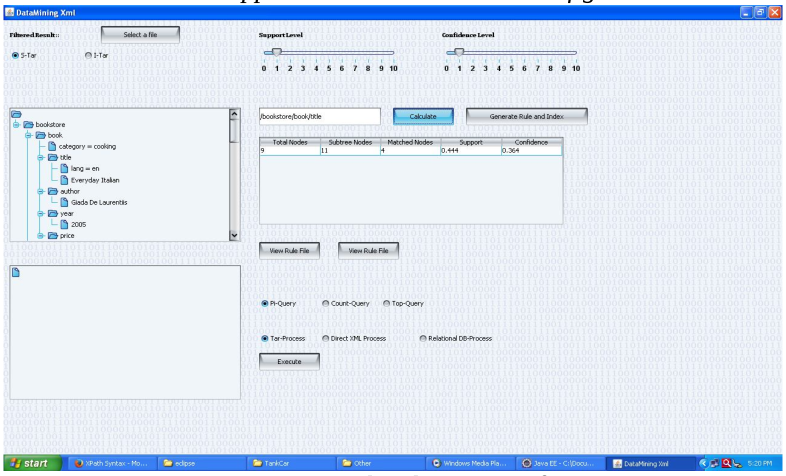
Class 1: σ/π-queries: Used to impose a simple, or complex (containing AND and OR operators), restriction on the value of an attribute or the content of a leaf node, possibly ordering the result. The query imposes some conditions on a node's content and on the content of its descendants, orders the results according to one of them and returns the node itself.

Class 2: count-queries: Used to count the number of elements having a specific content. The query creates a set containing the elements which satisfy the conditions and then returns the number of elements in such set.

Class 3: top-k queries: Used to select the best k answers satisfying a counting and grouping condition. The query counts the occurrences of each distinct value of a variable in a desired set; then orders the variables with respect to their occurrences and returns the most frequent k.

RESULT

The main application GUI is as shown in figure 3.



[✓ S-Tar] I-Tar

bookstore
- book
  - category = cooking
  - title
    - lang = en
    - Everyday Italian
  - author
    - Giada De Laurentiis
  - year
    - 2005
  - price

/bookstore/book/title

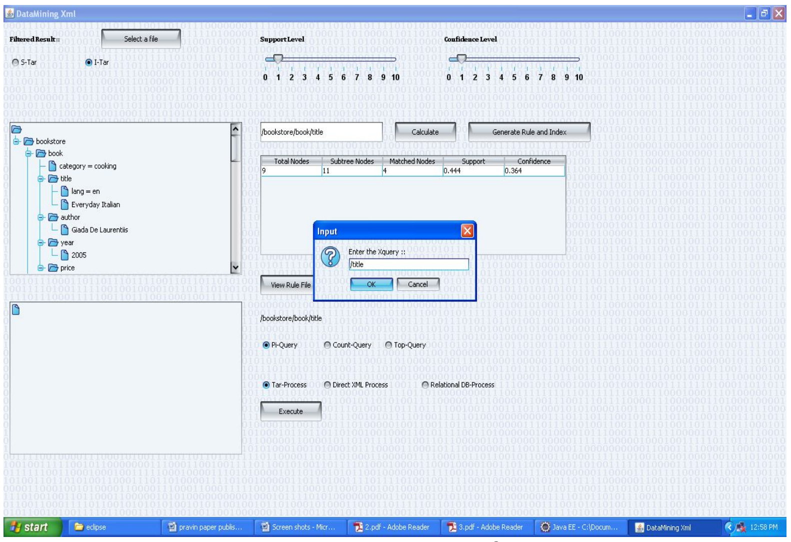
Total Nodes	Subtree Nodes	Matched Nodes	Support	Confidence
9	11	4	0.444	0.364

[✓ Pi-Query] Count-Query Top-Query

[✓ Tar-Process] Direct XML Process Relational DB-Process

Figure 3: Graphical User Interface

The GUI has provision to choose an XML file as input. It also has provision to calculate rules and create index on extracted rules. A graphical area is provided to show the XML file content with graphical tree representation. On clicking the view File button, it extracts TARs from the given XML file and finally extracted rules are saved into the given TAR file. The Execute button invokes a form where user can enter queries. The query interface is shown in figure 4.



S-Tar [✓ I-Tar]

bookstore
- book
  - category = cooking
  - title
    - lang = en
    - Everyday Italian
  - author
    - Giada De Laurentiis
  - year
    - 2005
  - price

/bookstore/book/title

Total Nodes	Subtree Nodes	Matched Nodes	Support	Confidence
9	11	4	0.444	0.364

Enter the Xquery :: /title

/bookstore/book/title

[✓ Pi-Query] Count-Query Top-Query

[✓ Tar-Process] Direct XML Process Relational DB-Process

Figure 4: Query Interface

We have performed three types of experiments. They are based on time required to extract extensional knowledge from XML, time required to get result from extracted TAR,s, time required to get result from database and study of accuracy of in tensional answer.

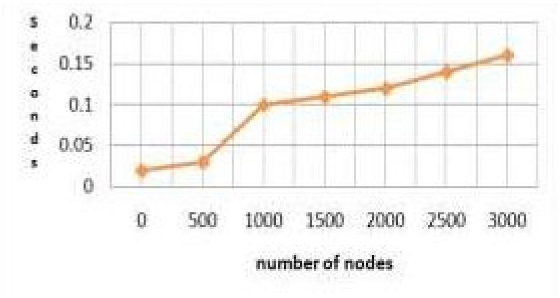
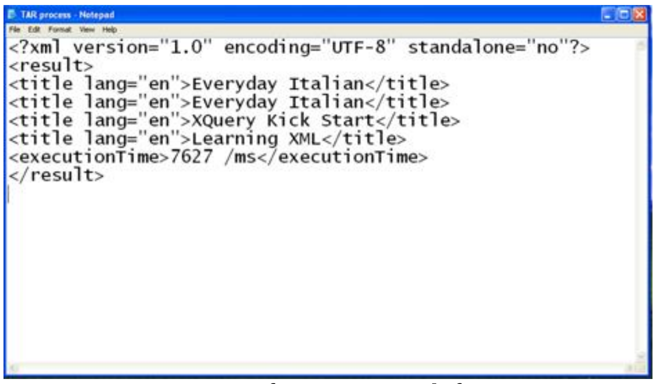


Figure 5: Time for TAR extraction

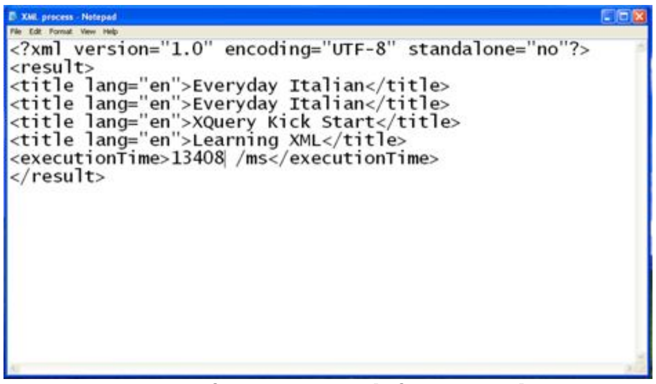
As shown in figure 5 , the TAR extraction time is more when number of nodes in XML document is more. In other words, the time taken to extract TARs is directly proportional to the number of nodes in given XML document.



```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<result>
<title lang="en">Everyday Italian</title>
<title lang="en">Everyday Italian</title>
<title lang="en">XQuery Kick Start</title>
<title lang="en">Learning XML</title>
<executionTime>7627 /ms</executionTime>
</result>
```

Figure 6: Time for extract result form TAR

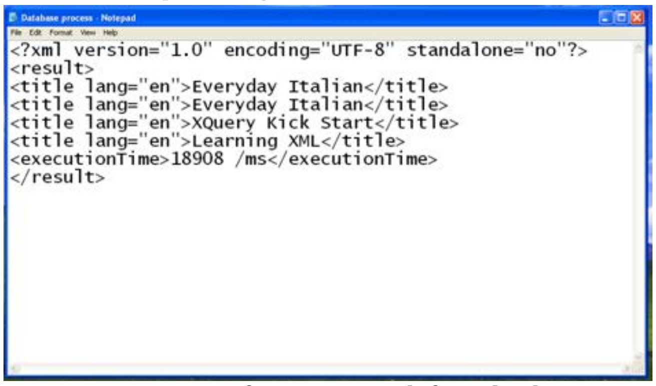
Figure 6 shows the result of simple select query which is executed over the extracted TAR. It show all titles of the book present in the XML file. It also shows the time required to get result in milliseconds.



```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<result>
<title lang="en">Everyday Italian</title>
<title lang="en">Everyday Italian</title>
<title lang="en">XQuery Kick Start</title>
<title lang="en">Learning XML</title>
<executionTime>13408 /ms</executionTime>
</result>
```

Figure 7: Time for extract result form XML document

Figure 7 shows the result of simple select query which is executed over the original XML file. It show all titles of the book present in the XML file. It also shows the time required to get result in milliseconds.



```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<result>
<title lang="en">Everyday Italian</title>
<title lang="en">Everyday Italian</title>
<title lang="en">XQuery Kick Start</title>
<title lang="en">Learning XML</title>
<executionTime>18908 /ms</executionTime>
</result>
```

Figure 8: Time for extract result form database

Figure 8 shows the result of simple select query which is executed over the relational database. It show all titles of the book present in the XML file. It also shows the time required to get result in milliseconds.

CONCLUSION

In this paper we presented a framework for extracting TARs from given XML file so as to support XML queries. The aim of this paper is to mine frequent association rules and store the mined content in XML format, use the TARs to support query Answering or to gain information from XML file. A prototype application is implemented to test the efficiency of the proposed framework and compare the result with the relational data base. The application takes XML file as input and generates TARs and then finally index file that helps in query processing.

REFERENCES

[1] Mirjana Mazuran, Elisa Quintarelli , and Letizia Tanca " Data mining for XML query answering support" IEEE Transaction on knowledge and data engineering, Vol. 24, No.8, August 2012.
[2] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke, " Privacy Preserving Mining of Association Rules," Proc. Eighth ACM Int'l Conf. Knowledge Discovery and Data Mining, pp. 217-228, 2002.
[3] B.Goethals and M. J. Zaki. Advances in frequent Itemset mining implementations: report on FIMI'03. SIGKDD Explorations,6(1):109– 117, 2004.
[4] R. Goldman and J. Widom. Dataguides: Enabling Query formulation and optimization in semistructured Databases. In Proc. of the 23rd Int. Conf. on Very Large *Data Bases*, Pages 436–445, 1997.
[5] K.Wong,J.X Yu, and N. Tang," Answering XML queried Using path based Indexes: A survey, " world wide web, vol. 9, no.3, pp.277-299. 2006.
[6] S.Gasparini and E. uintarelli. Intensional query answering to xquery expressions. In Proc. of the 16th Int. Conf. on Database and Expert Systems Applications, pages 544–553, 2005.
[7] E. Baralis, P. Garza, E. Quintarelli, and L. Tanca. Answering xml queries by means of data summaries. Conf. Database and Expert Systems Applications, pp. 544-553, 2005.
[8] S. Gasparini and E. Quintarelli, "Intensional Query Answering to XQuery Expressions," Proc. 16th Int'l Conf. Database and Expert Systems Applications, pp. 544-553, 2005.
[9] E. Baralis, P. Garza, E. Quintarelli, and L. Tanca, " Answering XML Queries by Means of Data Summaries," ACM Trans. Information Systems, vol. 25, no. 3, p. 10, 2007.