ISSN:0975-9646

Storing, Querying and Validating Fuzzy XML Data in Relational Database

Naresh Kumar. K[#1], Satyanand Reddy. ch[*2], V.E.S. Murthy. N[#3]

[#] *Dept. of CS&SE, Dept. of Mathematics, Andhra University*
Visakhapatnam, India

[*] *Dept. of CS&SE, Andhra University*
Visakhapatnam, India

***Abstract*— Information ambiguity and uncertainty are the major issues in real real-world scenario and for resolving these issues fuzzy data management has been incorporated in different database management systems in different ways. Even though there is much literature on XML-to-relational storage, few of these have given satisfactory solutions to the given problem of storing fuzzy XML data in RDBMs. In this paper, we attempt to present yet another technique for storing and querying fuzzy XML data in relational databases. The content XML document is retrieved using XPath. It views the XML document as a tree and uses one table that stores both the information contained in the nodes as well as the structure of the tree. We convert the XML tree representation into a table form using LOAD XML statement. We use SQL for query purposes. Also we can use crisp as well as fuzzy data for input in queries. We can also validate the XML input data using some unique key. It is a pre-defined key given at beginning of XML code. Further, we also propose a generic technique to convert path expression queries into SQL for processing XML queries.**

***Keywords*— XPath, RDBMs, XML, SQL.**

I. INTRODUCTION

XML [1] serves dual functionalities as markup language and data format. It separates presentation and data thus offering independency and flexibility for content association. Due to this nature of flexibility, data interchanged between two very different systems can use XML as the data format. XML tree-like structure is intuitive, human readable, and easy to understand. With the help of XML schema or DTD, the type and attributes of each tag usable for certain XML document can be well defined [2]. A database is any organized collection of information. There are many traditional database management systems that can represent crisp data using well-understood structures. However, real-world information containing subjective opinions and judgments may contain complex and imprecise data along with crisp data. The representation of such uncertain and complex data, in a database is still a very much research issue. As applications manipulate an increasing volume of XML data, there is a growing need for reliable systems to store and provide efficient access to these data [3]. The use of relational database systems for this purpose has attracted considerable interest. Storing and querying XML data in relational database systems is promising because efficient access methods for relational data have been developed for over thirty years and query planning and optimization in the relational algebra is well-understood [4]. Another important issue is the sharing of information between databases. Because different databases store data in different, and sometimes incompatible, formats, which makes exchanging of information a challenge. In many businesses, data from a large number of heterogeneous databases need to be integrated in connection with data warehousing, or system integration. Many organizations and enterprises establish distributed working environments, where different users need to exchange information based on a common model.

Extensible Mark-up Language (XML) is a proposed solution as a data representation and exchange format through the Internet and different database models have to meet this challenge. The Lore system is one example of a database model which is implemented in XML. Unlike HTML, XML allows the separation of content and presentation, that is, XML documents simply define the data representation and do not deal with the presentation. XML can also be used to represent complex and imprecise data formats in addition to crisp data formats. Not only can XML process complex and hierarchical information, it can also be used for commercial transactions. Therefore XML documents can be used to transfer data between the various Internet applications and different database models.

In order to make use of advantages of XML, information in XML documents must be made available quickly, reliably and in large amounts when necessary, for transactions. As is true for management of other forms of data, management of persistent XML data requires capabilities for data independence, integration, access rights, versions, views, integrity, redundancy, consistency, and recovery standards. Functionality, consistency, restart capability, data security, and recovery tools of a database management system can be utilized by the XML data [26].

Some of the techniques for XML-to-relational storage where internal data are deterministic are sound. However, they lack the strength to cope with complex XML-to-relational storage when the internal data contain imprecise and uncertain information. Currently, many practical applications, e.g., information extraction, natural language processing, data mining, generate imprecise and uncertain data. Moreover, in many of these applications, information is represented in a semi-structured manner J. Liu et al. (XML) [5]. The study of combining XML and imprecision/uncertainty has become an emerging topic for various applications on the Web, and there have been some achievements in this area, including several proposed combination frameworks [6, 7, 8–10, 11, 12, 13, 15]. Some representations of probabilistic data in XML were proposed in previous research papers, such as Abiteboul et al. [6, 7], Nierrmanand Jagadish [13], and Hung et al. [10]. In [14], Hollander and Keulen presented an approach for adapting existing mapping techniques to map unordered probabilistic XML data to probabilistic relational data. Gaurav and Alhajj [8] incorporated fuzzy data in XML documents extending the XML Schema associated to these documents. They defined a mechanism to represent fuzzy data along with deterministic data in XML format by introducing some new constructors. In [15], Turowskia and Weng introduced a formal syntax for important fuzzy data types used to store fuzzy information. They also illustrated how fuzzy information, whose description is based on the DTDs (Document Type Definition), could be exchanged between application systems by means of XML.

Surprisingly, although relational databases have the advantage of accessing and processing deterministic and uncertain data, and fuzzy values [16, 17] have been employed to model and handle imprecise information in relational databases. Although Zadeh introduced the theory of fuzzy sets [18], the study of storing and querying fuzzy XML data in relational databases has only recently started and still merits further attention. Current efforts on fuzzy XML [8, 15] are mainly made on the problems of representing and incorporating fuzzy information in an XML format. Due to the lack of effective strategies for supporting fuzzy XML to-relational storage, the powerful and reliable data management services provided by RDBMs are not fully exploited for fuzzy XML data processing. We therefore need the means to manage fuzzy XML information gathered by a database management system during its entire life, and in particular evaluate queries over such data.

To filling the gap in the research of the fuzzy XML-to-relational storage, it appears Ma and Yan [19] firstly investigated schema mapping from a fuzzy XML model to fuzzy relational databases. Their solution is to decompose fuzzy XML instances into a set of tables based on an XML schema definition (e.g., Document Type Definition, DTD for short). This approach depends on the existence of a schema describing the XML data, which is only suitable for unordered XML data with a well-defined structure in static scenarios (the schemas are not changed). However, it is problematic when no XML schemas are available or XML schemas are dynamic (the schemas that vary over time). On the other hand, it ignores the ordered nature (document order) of XML data, where document order is an inherent property of XML instances and should be preserved when storing fuzzy XML data in relational databases. Moreover, this approach ignores the mapping of fuzzy XML queries, which is considered to be the foundation of implementing fuzzy XML data management in relational databases. The uncertainty of XML data found today as well as the flexibility of representation offered by XML raises challenging issues for storing fuzzy XML data in traditional relational systems. Moreover, the study of the mapping of fuzzy XML queries, especially path expression queries, into their SQL counterparts, is still a blank field. In order to solve the storing and query mapping problems above, in this paper, we study a methodology of storing and querying fuzzy XML data in relational databases. In particular, we present a novel approach to shared fuzzy XML data into relational data. The unique feature of proposed approach is that no schema information is required for our data storage. On this basis, we present a generic approach to translate path expression queries into SQL for processing XML queries.

II. RELATED WORK

Goran Panic *et.al* [20] suggested a process simplifying syntax and execution for fuzzy logic in XML. An application that enabled the use of fuzzy logic with XML data is given. In fact, users were provided a probability to explain the self-willed membership functions, and their calculation has got in real time with the application of the MATLAB software. The suggested usage solution was a backbone for a fore development of fuzzy XML application. Their future work, explaining and executing the fuzzy XQuery interpreter that permitted the application of priorities planned. Also, the concentrated of future work was on enhancing the executed functionalities and the syntax, further enhancement of the performance and finally, the XML structure.

E.J. Thomson Fredrick and Dr. G. Radhamani [21] proposed a basic XML Schema explanation for presenting fuzzy knowledge in XML documents. They have suggested a fuzzy constraint-based structure to accredit the XML document contrary the XML schema. They presented a Trapezoidal and Triangular fuzzy membership functions for executing Fuzzy Constraints. They have explained the mechanism to show fuzzy data along with crisp data in XML Schema. Their technique was able to support uncertainty in schema comparing by exploiting fuzzy constraints. In given paper, they have emphasized only on Domain Integrity constraints using Fuzzy Logic. Because that was an essential control for XML based usage for executing data consistency. They have to do afore research on Entity Integrity Constraints and Referential Integrity Constraints for Native XML Databases. In future, they suggested to do further research on explain Fuzzy Triggers on the basis of XML Schema with Fuzzy Constraints.

Z.M. Ma and Li Yan [22] described a wide exploitation of the Web has outcome in the utility of much amounts of electronic data. Knowledge representation and commutation over the Web become essential, and XML has been the de facto standard. In the other hand, given paper made a new set of data management essential involving XML, such as the essence to store and query XML files. Secondly, fuzzy sets and potentially distributions have been considerably applied to deal with knowledge imprecision and incertitude in real-world usages, and fuzzy database modelling was catching propagate notice for intelligent data processing. In order to manage fuzzy data in XML, given paper investigated the fuzzy XML data modelling. Based on potential distribution theory, they first recognized multiple granularity of data fuzziness in UML and XML. The fuzzy UML data model and fuzzy XML data model that considered all types of fuzziness created. Further, they executed the formal exchange from the fuzzy UML model to the fuzzy XML model, and the formal mapping from the fuzzy XML model to the fuzzy relational databases. It should be noted that the fuzzy extension of XML in given paper only emphasized on XML DTD due to it has traditionally been the most common technique for depicting the framework of XML instance files.

Li Yan [23] described a paper in which they emphasized on fuzzy data with fuzzy data kinds in the fuzzy databases and fuzzy XML. They spotted various fuzzy data types, containing fuzzy basic data types, fuzzy set data types and fuzzy explained data types. For the aim of showing and processing fuzzy data, they defined the declarations of the fuzzy data kinds in the fuzzy OODB model and fuzzy XML Schema, respectively. In future work, they suggested prototypes of fuzzy OODB system and fuzzy XML database system, and then accomplished the evaluation experiment of fuzzy data type.

Alnaar Jiwani *et.al* [24] XML schemas have substituted DTDs as the new way for putting constraints on XML files. They have explained a fuzzy XML schema to show an execution of a fuzzy relational database that permitted for resemble relations and fuzzy sets. They have also given a flat translation algorithm to convert from the fuzzy database execution to a fuzzy XML file that conforms to the given fuzzy XML schema. The given algorithm has been implemented within VIREX and an explaining example has been showed into the paper to explain the power of VIREX in translating fuzzy relational data into fuzzy XML. Currently, they were focusing and working on the given extensions to the suggested technique. The fuzzy database model and the fuzzy XML schema have to be extended to unify other sorts of fuzziness such as fuzzy rules, fuzzy integrity constraints and non -atomic data values. The flat converting methods used to translate from the fuzzy relational database to an XML file ensuring to the given XML schema has to be optimized. Whether nested translation approaches used to a fuzzy relational database was also be focused. For given aspect, they focused to gain from and expand the already functional executed of VIREX for building nested XML without fuzziness. As they have successfully created flat fuzzy XML, they expected the process of building nested fuzzy XML to be an easy extension of the current nested implementation of VIREX.

III. Problem Definition

Information imprecision and uncertainty exist in many real-world applications and for this reason fuzzy data management has been extensively investigated in various database management systems. Currently, introducing native support for XML data in relational database management systems (RDBMs) has attracted considerable interest with a view to leveraging the powerful and reliable data management services provided by RDBMs. Although there is a rich literature on XML-to-relational storage, none of the existing solutions satisfactorily addresses the problem of storing fuzzy XML data in RDBMs. In this paper, we study the methodology of storing and querying fuzzy XML data in relational databases [25].The use of relational database management systems (RDBMSs) to store and query XML data has attracted considerable interest with a view to leveraging their powerful and reliable data management services. Due to the mismatch between the relational and XML data models, it is necessary to first shred and loads the XML data into relational tables, and then translates XML queries over the original data into equivalent SQL queries over the mapped tables. Although there is a rich literature on XML-relational storage, none of the existing solutions addresses all the storage problems in a single framework. Works on mapping strategies often have little or no details about query translation, and proposals for query translation often target a specific mapping strategy. XML-storage solutions provided by RDBMS also have limitations. Notably, they are tied to a specific backend and use proprietary mapping languages, which not only may require a steep learning curve, but often are unable to express certain desirable mappings [3].

IV. Proposed Methodology

In this paper, we are presenting a fuzzy based xml data management system. The given system handles both crisp as well as fuzzy data. Information is unclear sometime. We are presenting a system for storing and querying fuzzy XML data in relational databases. The architecture of the given system is given below:

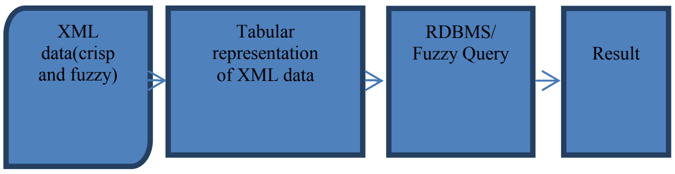


Fig. 1 Architecture of the fuzzy xml database system

A. *XML Query Language*

The XML is an Extensible Mark up Language (XML) has a well-established data format and an increasing amount of information which available in XML form. The XML path language XPath is query language for choosing the nodes from an XML document. XPath is defined by the World Wide Web Consortium (W3C). The XPath language is based on a tree representation of the XML document, and gives the ability to navigate around the tree, selecting nodes by a kind of criteria. XPath is using as a declarative notation and permit the extraction of information through path expressions. The core part of XPath is the path expression, which mostly seem in XML queries. Each expression is shaped of a finite series of steps. A step is shaped of the main elements: an axis specifiers, that suggests the navigation direction within the XML tree representation, a node test, that specifies a node name or, more generally, an expression, which permits the identification of one or more particular nodes or paths in the specified direction, and a predicate, that is an expression of any complexity, which must be satisfied before the preceding node will be matched by an X Path expression. Simplified grammars for XPath are given below:

$q: = n^{+}[/O]$

$n:: = \{/|//\}nodetest[p]$#
$p:: = [f\ [Opconstant]]$#
$f:: = @attribute|nodetest[@attribute]|text()$#
$O:: = @attribute|text()|count()|sum()$#
$Op:: = > | \geq | = | < | \leq | != $
#

In particular, an XPath query is an expression of the form of $n1, n2 \ldots n_k\ [/O]$, which consists of a location path , $n1, n2 \ldots n_k$, and an optional output function O. Each location step n_i is of the form $/a_i::n_i[p_i]$where a_i is an axis, n_i is a node test that specifies the name of elements n_i can match, and p_i is an optional predicate that is specified syntactically using square brackets.

Thus in this step we are converting our xml file into tree representation. Now we generate all the table information for creating table. XPath query extracted the node information and then we have to traverse in such a way so that we can collect all the information for converting given information into table form. The traversing will start from root node called list node and propagate through child node. We are performing traversing from left to right in top to bottom fashion. After traversing all the nodes we will have the group of information which will be helpful for creating table. It will analyse the number of columns, number of rows and content of table. Here we are giving a graphical representation in which all nodes showed the tree representation by which we have extract the information for creation the table. In the given diagram we have taken an example of city. Here we are using crisp as well as fuzzy data. The diagram of above concept is shown below:

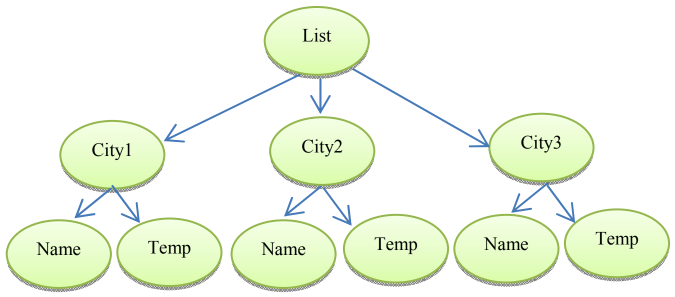


Fig. 2 Tree Representation of Xml Document

In fig. 1, we are showing the tree representation of xml file. This can be achieved by XPath query. This tree representation is converted into table using Load XML statement. Here we are giving the syntax.

LOAD XML [LESS_PRIORITY | CONCURRENT] [LOCAL]
INFILE 'file_name'
[REPLACE | IGNORE]
INTO TABLE [db_name.] tbl_name
[CHARACTER SET charset_name]
[ROWS IDENTIFIED BY '<tagname> ']
[IGNORE number {LINES | ROWS}]
[(column_or_user_var,...)]
[SET col_name = expr,....]
#
#The LOAD XML statement reads data from an XML file into a table. The file_name must be given as a literal string. The tagname in the optional ROWS IDENTIFIED BY clause must also be given as a literal string, and must be surrounded by angle brackets (< and >).
Now we insert the XML data into table using structure query language.

B. *Structured Query Language*

A relational database management system, which is a database management system, is based on the relational model. In the relational model of a database, all data shown in terms of tuples, grouped into relations. Mostly, implementations of the relational model using the structured query language (SQL).

The SQL language divided into many language elements that are
1. CLAUSES, which are constituent components of statements and queries.
2. EXPRESSIONS, which can produce either scalar values or tables consisting of columns and rows of data.
3. PREDICATES, which define conditions that can be measured to SQL three-valued logic (true, false and null) or Boolean truth values, which are used to limit the effects of statements and queries or to convert the program flow.
4. QUERIES, which retrieve the data based on specific criteria. This is the most important element of SQL.
5. STATEMENT, which may have a persistent effect on schemas and data, or which can control the transactions, program flow, etc.

The most commonly operation SQL has the query, which is worked with the declarative SELECT statement. SELECT can get data from one or more tables or an expression.
SELECT is the most complex statements contain in SQL, with few optional keywords and clauses that contains are:
1. The FROM clause which indicates the table from which data is to be retrieved.
2. The WHERE clause includes a similitude predicate, which banned the rows returned by the query.
3. The GROUP BY clause is used to project rows having common values into a smaller set of rows.
4. The ORDER BY clause recognizes which columns used to sort the all resulting data, and in which direction they sorted in the order to (ascending or descending).

C. *Fuzzy XML Data Model*

XML data are well structured, and XML may naturally show imprecise and uncertain information. In this case of XML, when some elements (attributes) are uncertain, we can allies' elements (attributes) with membership degrees to show them and their possibilities. Likely, when values of elements (attributes) are uncertain, that time we can use possibility distributions to express them and their possibilities. Theoretical foundation of the suitability of using the above representations to define fuzzy data and their possibilities is that XML has flexible format and the character of self-definition. Let us interpret what a membership degree helped with an element means, provided that the element can nest contain under other elements, and more than one of these elements may have an associated membership degree. The existential membership degree associated with an element should be the possibility that the state of the world includes this element and the sub-tree rooted at it. For an element with the sub-tree rooted at it, each node in the sub-tree is not treated as independent but dependent upon its root-to-node chain. Each possibility in the source XML document is assigned based on the fact that the parent element is known to exist. This possibility is relative one basis upon the assumption that the possibility the parent element exists is exactly 1.0. In order to numerate the absolute possibility, we can consider the relative possibility in the parent element. In generally, the absolute possibility of an element e can be contained by multiplying the relative possibilities found in the source XML along the path from to its root. Fuzzy XML document can naturally modeled as an ordered node tree. In general, we have six kinds of nodes in a fuzzy tree that are root, elements, attributes, text, fuzzy construct, and possibility attribute, where the root, elements, attributes, and text nodes are deterministic nodes, and fuzzy construct and possibility attribute nodes are fuzzy nodes. A simple example illustrates the six node types. The root node is a virtual node pointing to the root element of an XML document. Elements in an XML document are showing as an element node with an expanded-name. Element nodes have m (m≥0) other elements, attributes or text as its children. Text nodes are string-valued nodes, and they haven't any child nodes. An attribute node has an attribute name and an attribute-value. The fuzzy nodes Dist and Val, constructs, and possibility attribute nodes are used for specifying the possibility (possibility distribution) of a given element existing in the XML document.

D. Fuzzy XML to Relational Storage

In the hierarchical, ordered XML and the flat, unordered relational data models, are not fully accusation, so fuzzy XML to-relational storage is not a straightforward work. In this chapter, we are presenting an XPath mapping technique to store fuzzy XML data in relational databases. In an XML deterministic scenario, we have three main kinds of nodes in an XML data tree, which are element, attribute and text nodes. The element nodes further classified into two kinds they are leaf element and non-leaf element nodes. Note that the root node special non-leaf element node. However, in a fuzzy XML scenario, in addition to element, attribute and text nodes, there are three special nodes, which are possibility attribute, Val construct and Dist construct nodes. As a result, the relational schema constructed from fuzzy XML data trees is clearly different from the schema constructed from deterministic data trees. Fuzzy output will show like this in table form as given below:

TABLE I
Fuzzy Output Shown for Given ID

Property Name	Crisp ID
Very cold City	1
Cold City	2
Warm City	3
Hot City	4

E. XML Input Data Validation

In this section we present XML input data validation through unique key. We assign a unique key to each child node for identifying uniqueness in group. Here we are taking an example of city. We have a group of city as a child node which will be child node of list node. There is more than one city in XML document. We have to distinguish these columns, for this we assign a unique key to each sub tag of list. It will ensure the validation of uniqueness. In the case we will not provide unique key to each child of list node, it will give error at the time of compilation. It will also help when we query for fuzzy output. Uniqueness will be predefined, when we will right the xml file at that time only we will give unique in the tag field when it is defined. Here we are given the xml code, which demonstrate the given concept.

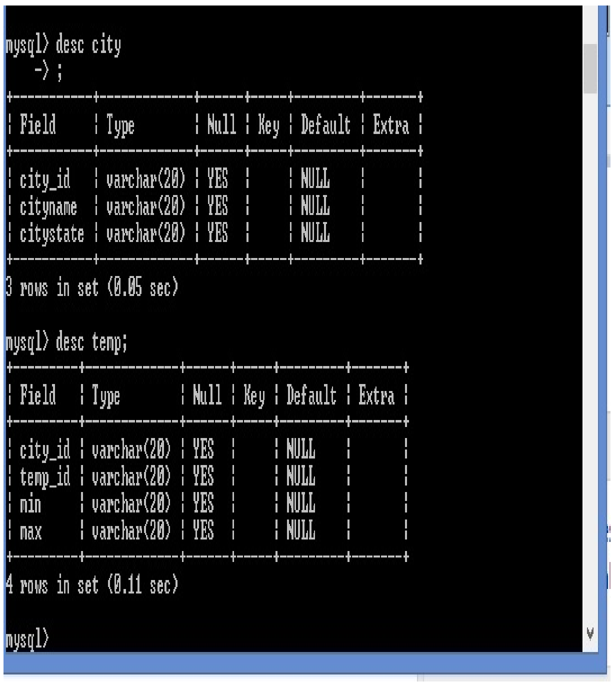
```
< ? Xml version = 1.0? >
< List >
< City city_id = "1" >
< City name > Pune < /City name >
< City state > Maharashtra < /City state >
< Temp temp_id = "1" >
< Linguistic label = “cold” >
< Min > 4 < /Min >
< Max > 10 < /Max >
< /Linguistic label >
< Linguistic label = “warm” >
< Min > 12 < /Min >
< Max > 30 < /Max >
< /Linguistic label >
< Linguistic label = “hot” >
< Min > 32 < /Min >
< Max > 40 < /Max >
< /Linguistic label >
< /Temp >
< /City >
< City city_id = "2" >
< City name > Trivandrum < /City name >
< City state > Kerala < /City state >
< Temp temp_id = "2" >"
< Linguistic label = “cold” >
< Min > 1 < /Min >
< Max > 5 < /Max >
< /Linguistic label >
< Linguistic label = “warm” >
< Min > 8 < /Min >
< Max > 15 < /Max >
< /Linguistic label >
< Linguistic label = “hot” >
< Min > 18 < /Min >
< Max > 25 < /Max >
< /Linguistic label >
< /Temp >
< /City >
< City city_id = "3" >
< City name > Shillong </ City name >
< City state > Meghalaya < /City state >
< Temp temp_id = "3" >
< Linguistic label = “cold” >
< Min > 1 < /Min >
< Max > 6 < /Max >
< /Linguistic label >
< Linguistic label = “warm” >
< Min > 7 < /Min >
< Max > 15 < /Max >
< /Linguistic label >
< Linguistic label = “hot” >
< Min > 16 < /Min >
< Max > 22 < /Max >
< /Linguistic label >
< /Temp >
< /City >
< City city_id = "4" >
< City name > Shimla < /City name >
< City state > Himachal Pradesh < /City state >
< Temp temp_id = "4" >
< Linguistic label = “cold” >
< Min > -5 < /Min >
< Max > 1 < /Max >
< /Linguistic label >
< Linguistic label = “warm” >
< Min > 5 < /Min >
< Max > 10 < /Max >
< /Linguistic label >
< Linguistic label = “hot” >
< Min > 11 < /Min >
< Max > 15 < /Max >
< /Linguistic label >
< /Temp >
< /City >
< City city_id = "5" >
< City name > Hyderabad < /City name >
< City state > Andhra Pradesh < /City state >
< Temp temp_id = "5" >
< Linguistic label = “cold” >
< Min > 10 < /Min >
< Max > 20 < /Max >
< /Linguistic label >
< Linguistic label = “warm” >
< Min > 22 < /Min >
< Max > 30 < /Max >
< /Linguistic label >
< Linguistic label = “hot” >
< Min > 32 < /Min >
< Max > 40 < /Max >
< /Linguistic label >
< /Temp >
< /City >
< City city_id = "6" >
< City name > Nagpur < /City name >
< City state > Maharashtra < /City state >
< Temp temp_id = "6" >
< Linguistic label = “cold” >
< Min > 10 < /Min >
< Max > 20 < /Max >
< /Linguistic label >
< Linguistic label = “warm” >
< Min > 22 < /Min >
< Max > 30 < /Max >
< /Linguistic label >
< Linguistic label = “hot” >
< Min > 32 < /Min >
< Max > 40 < /Max >
< /Linguistic label >
< /Temp >
< /City >
< City city_id = "7" >
< City name > Mysore < /City name >
< City state > Karnataka < /City state >
< Temp temp_id = "7" >
< Linguistic label = “cold” >
< Min > 4 < /Min >
< Max > 10 < /Max >
< /Linguistic label >
< Linguistic label = “warm” >
< Min > 12 < /Min >
< Max > 25 < /Max >
< /Linguistic label >
< Linguistic label = “hot” >
< Min > 26< /Min >
< Max > 35 < /Max >
< /Linguistic label >
< /Temp >
< /City >
< City city_id = "8" >
< City name > Badrinath < /City name >
< City state > Uttarakhand < /City state >
< Temp temp_id = "8" >
< Linguistic label = “cold” >
< Min > -5 < /Min >
< Max > 0 < /Max >
< /Linguistic label >
< Linguistic label = “warm” >
< Min > 1 < /Min >
< Max > 5 < /Max >
< /Linguistic label >
< Linguistic label = “hot” >
< Min > 6 < /Min >
< Max > 10 < /Max >
< /Linguistic label >
< /Temp >
< /City >
< City city_id = "9" >
< City name > Bhopal < /City name >
< City state > Madhya Pradesh < /City state >
< Temp temp_id = "9" >
< Linguistic label = “cold” >
< Min > 10 < /Min >
< Max > 20 < /Max >
< /Linguistic label >
< Linguistic label = “warm” >
< Min > 22 < /Min >
< Max > 30 < /Max >
< /Linguistic label >
< Linguistic label = “hot” >
< Min > 32 < /Min >
< Max > 40 < /Max >
< /Linguistic label >
< /Temp >
< /City >
< City city_id = "10" >
< City name > Goa < /City name >
< City state > Goa < /City state >
< Temp temp_id = "10" >
< Linguistic label = “cold” >
< Min > 4 < /Min >
< Max > 10 < /Max >
< /Linguistic label >
< Linguistic label = “warm” >
< Min > 12 < /Min >
```

< Max > 20 < /Max >
< /Linguistic label >
< Linguistic label = “hot” >
< Min > 22 < /Min >
< Max > 30 < /Max >
< /Linguistic label >
< /Temp >
< /City >
< /List >
< /Xml >

The overall algorithm can be described below:
Step 1: create an xml file which consists of crisp and fuzzy data.
Step 2: cluster the xml data using fuzzy for creating table.
Step 3: convert xml file into table form using LOAD XML.
Step 4: query from obtain table to get desired result.

V. Results And Discussion

In this paper we have extracted the XML information using XPath query language that has been implemented in the working platform of JAVA (NetBean 6.8) and MySQL. We insert the extracted information into table using SQL. We converted the XML file into table form using LOAD XML statement. Here we are showing the fuzzy output in the form of table. Fuzzy XML data are stored in relational database. By converting XML file into table we can easily query and retrieved the data from relational database. Here we are showing the different screenshot for the entire work.



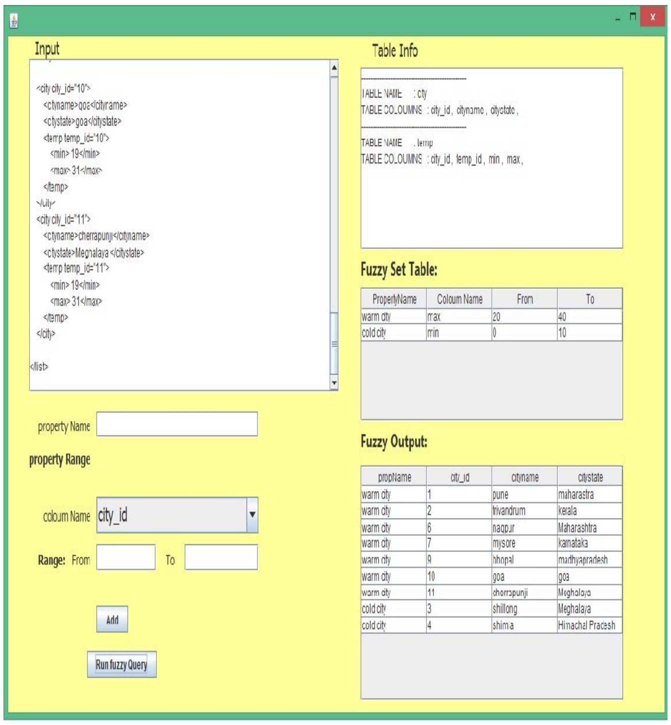
```
mysql> desc city
    -> ;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| city_id   | varchar(20) | YES  |     | NULL    |       |
| cityname  | varchar(20) | YES  |     | NULL    |       |
| citystate | varchar(20) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
3 rows in set (0.05 sec)

mysql> desc temp;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| city_id | varchar(20) | YES  |     | NULL    |       |
| temp_id | varchar(20) | YES  |     | NULL    |       |
| min     | varchar(20) | YES  |     | NULL    |       |
| max     | varchar(20) | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
4 rows in set (0.11 sec)

mysql>
```

Fig. 3 Table Representation of XML File

In the above fig. 3, we extracted the XML information using XPath algorithm to create table. In this section we identified the attribute of table which will be inserted in table using LOAD XML statement.



Input

```
<city city_id="10">
  <cityname>goa</cityname>
  <citystate>goa</citystate>
  <temp temp_id="10">
    <min> 19</min>
    <max> 31</max>
  </temp>
</city>
<city city_id="11">
  <cityname>cherrapunji</cityname>
  <citystate>Meghalaya </citystate>
  <temp temp_id="11">
    <min> 19</min>
    <max> 31</max>
  </temp>
</city>

</list>
```

Table Info

```
-----------------------------------------
TABLE NAME    : city
TABLE COLOUMNS  : city_id , cityname , citystate ,
-----------------------------------------
TABLE NAME    : temp
TABLE COLOUMNS  : city_id , temp_id , min , max ,
```

Fuzzy Set Table:

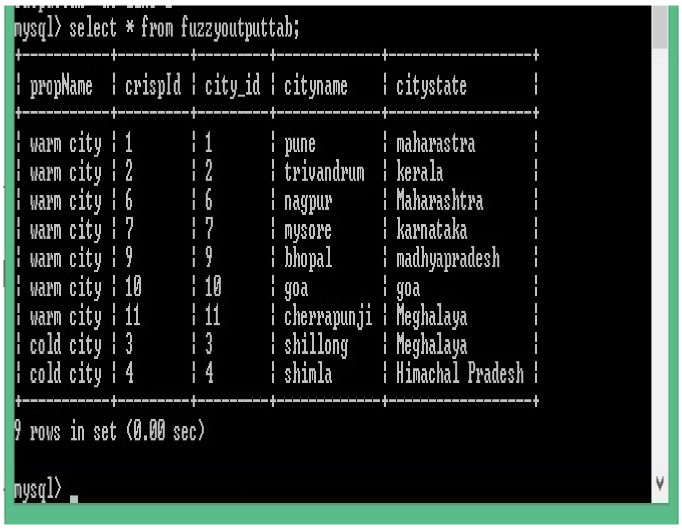
PropertyName	Coloum Name	From	To
warm city	max	20	40
cold city	min	0	10

Fuzzy Output:

propName	city_id	cityname	citystate
warm city	1	pune	maharastra
warm city	2	trivandrum	kerala
warm city	6	nagpur	Maharashtra
warm city	7	mysore	karnataka
warm city	9	bhopal	madhyapradesh
warm city	10	goa	goa
warm city	11	cherrapunji	Meghalaya
cold city	3	shillong	Meghalaya
cold city	4	shimla	Himachal Pradesh

Fig. 4 XML Data in the Form of Table

In the above fig. 4, we created a table using SQL and insert the extracted information using LOAD XML statement.



```
mysql> select * from fuzzyoutputtab;
+-----------+---------+---------+-------------+------------------+
| propName  | crispId | city_id | cityname    | citystate        |
+-----------+---------+---------+-------------+------------------+
| warm city | 1       | 1       | pune        | maharastra       |
| warm city | 2       | 2       | trivandrum  | kerala           |
| warm city | 6       | 6       | nagpur      | Maharashtra      |
| warm city | 7       | 7       | mysore      | karnataka        |
| warm city | 9       | 9       | bhopal      | madhyapradesh    |
| warm city | 10      | 10      | goa         | goa              |
| warm city | 11      | 11      | cherrapunji | Meghalaya        |
| cold city | 3       | 3       | shillong    | Meghalaya        |
| cold city | 4       | 4       | shimla      | Himachal Pradesh |
+-----------+---------+---------+-------------+------------------+
9 rows in set (0.00 sec)

mysql>
```

Fig. 5 Fuzzy Input Output Table

In the above fig. 5, we showed the fuzzy input output table. The fuzzy set table contained the fuzzy input for temperature, which is varying between minimum to maximum. The fuzzy output table is created on the basis of table generated from XML file.

VI. Conclusion

The uncertainty of XML data is the major issue in current time for storing fuzzy XML data in traditional relational systems. In this paper we present a new technique by which we retrieved fuzzy XML data using XPath algorithm. XPath algorithm extract the information stored in XML document. We convert the given extracted information into table form using LOAD XML statement. LOAD XML statement has its own syntax which is used for converting xml file to table form. We insert the xml information into table using SQL. The unique feature of our technique is that no schema information is needed for the data storage. On this basis, we have given a new approach to convert path expressions to SQL queries for processing XML queries. We can query from table using simple SQL query. Here we have presented the input data validation in XML. This can be achieved using unique key. We can easily distinguish the child node of XML document.

There are a number of ways for future research. We are presently working on a prototype giving the feasibility of the technique for large practical applications. As future work, we plan to enhance our mapping approach by taking constraints during the data mapping, developing query translation techniques to translate complex XML queries such as XQuery queries into corresponding SQL statements, and improving our technique by providing query optimization techniques.

REFERENCES

[1] W3C XML Path Language Specification, Latest. *http://www.w3.org/TR/xpath.*

[2] Mikael Fernandus Simalango, "*XML Query Processing and Query Languages": A Survey,* 2007.

[3] Amer-Yahia. S, Du. F and Freire. J, "*A comprehensive solution to the XML-to-relational mapping problem,*" In: Proceedings of WIDM, pp 31–38, 2004.

[4] Weigel. F, Schulz. KU and Meuss. H, "*Exploiting native XML indexing techniques for XML retrieval in relational database systems*", In: Proceedings of WIDM, pp 23–30, 2005.

[5] Senellart. P and Abiteboul. S, "*On the complexity of managing probabilistic XML data,*" In: Proceedings of PODS, pp 283–292, 2007.

[6] Abiteboul. S and Senellart. P, "*Querying and updating probabilistic information in XML,*" In: Proceedings of EDBT, pp 1059–1068, 2006.

[7] Abiteboul. S, Kimelfeld. B, Sagiv. B and Senellart Y, "*On the expressiveness of probabilistic XML models*", VLDB J 18(5):1041–1064, 2009.

[8] Gaurav. A and Alhajj. R, "*Incorporating fuzziness in XML and mapping fuzzy relational data into fuzzy XML*", In: Proceedings of the 2006 ACM symposium on applied computing, pp 456–460, 2006.

[9] Hollander. ES, and van Keulen. M (2010) "*Storing and querying probabilistic XML using a probabilistic relational DBMS*", In: Proceedings of the 4th international workshop on management of uncertain data (MUD 2010), pp 35–49.

[10] Hung. E, Getoor. L and Subrahmanian. V.S (2003) "*PXML: a probabilistic semi-structured data model and algebra",* in: Proceedings of ICDE, pp 467–478.

[11] Liu. J, Ma. ZM and Yan. L (2009) "*Efficient processing of twig pattern matching in fuzzy XML*". In: Proceedings of CIKM, pp 193–204.

[12] *Ma. ZM, Liu. J and Yan. L (2010) Fuzzy data modeling and algebraic operations in XML. Int J IntellSyst 25(9):925–94*

[13] Nierrman. A, and Jagadish. HV (2002) *ProTDB: probabilistic data in XML*. In: Proceedings of VLDB, pp 646–657.

[14] Hollander. ES and van Keulen. M (2010) *Storing and querying probabilistic XML using a probabilistic relational DBMS*. In: Proceedings of the 4th international workshop on management of uncertain data (MUD 2010), pp 35–49.

[15] Turowski. K and Weng. U (2002) *Representing and processing fuzzy information an XML-based approach*. J Knowl-Based Syst 15:67–75.

[16] Valova. I, Milano. G, Bowen. K and Gueorguieva. N (2011) *Bridging the fuzzy, neural and evolutionary paradigms for automatic target recognition.* ApplIntell 35(2):211–225.

[17] Zajaczkowski. J and Verma. B (2012) *Selection and impact of different topologies in multi-layered hierarchical fuzzy systems*. ApplIntell36(3):564–584.

[18] Zadeh. LA (1965) *Fuzzy sets.* Inf Control 8(3):338–353.

[19] Ma. Z.M and Yan. L (2007) *Fuzzy XML data modeling with the UML and relational data models*. Data KnowlEng 63:972–996.

[20] Panic. G, Rackovi. M and Škrbic. S, "*Fuzzy XML with Implementation*", Novi Sad, Serbia, 2012.

[21] Thomson Fredrick E.J. and Radhamani. G, "*Fuzzy Integrity Constraints for Native XML Database"*, International Journal of Computer Science Issues, Vol. 9, Issue 2, No 3, March 2012.

[22] Ma. Z.M, and Yan. L, "*Fuzzy XML data modeling with the UML and relational data models*", Elsevier B.V., 2007.

[23] Yan. L, "*Modeling Fuzzy Data with Fuzzy Data Types in Fuzzy Database and XML Models*", IAJIT, 2007.

[24] Jiwani. A, Ali Mohamed. Y, Spence. K, Lo. A, Özyer. T and Alhajj. R, "*Fuzzy XML Model for Representing Fuzzy Relational databases in Fuzzy XML Format*", 2007.

[25] Liu. J, Ma. Z.M, Feng. X, "*Storing and querying fuzzy XML data in relational databases*", Journal Applied Intelligence archive Volume 39 Issue 2, September 2013.

[26] Üstünkaya. E, "*Fuzzy Querying In Xml Databases*", Approval Of The Graduate School Of Natural And Applied Sciences, 2004.