A GPFCSP-Based Fuzzy XQuery Interpreter

Pannipa Sae Ueng¹, Srdan Škrbić¹, Supaporn Kansomkeat², Apirada Thadadech²
¹Department of Mathematics and Informatics, Faculty of Sciences, University of Novi Sad, Novi Sad, Serbia.
²Department of Computer Science, Faculty of Science, Prince of Songkla University, Songkhla, Thailand.

Abstract—Nowadays XQuery has become the strongest standard for querying XML data. However, most of the real world information is in the form of imprecise, vague, ambiguous, uncertain and incomplete values. That is why there is a need for a flexible query language in which users can formulate queries that arise from their own criteria. In this paper, we propose an implementation of the Fuzzy XQuery - an extension of the XQuery query language based on the fuzzy set theory. In particular, we provide priority, threshold and fuzzy expressions for handling flexible queries. In addition, we have implemented an interpreter for this language by using the GPFCSP concept in Java and eXist-db environment.

Index Terms—Fuzzy XQuery; XQuery Interpreter; XML Database; Fuzzy Set Theory.

I. INTRODUCTION

XQuery language has been proposed as a standard for XML querying. It provides a feature called a FLWOR expression that supports the iteration and binding of variables to intermediate results. The FLWOR is an acronym: FOR, LET, WHERE, ORDER BY, RETURN, which is a powerful and important part of XQuery, similar in some aspects to the SQL query language in relational databases.

In real life, most information is imprecise, vague, ambiguous, uncertain or incomplete. Ideas related to improving query languages that can include such imprecise information in terms of the user’s criteria is therefore natural. However, XQuery does not support the use of this kind of information by itself. In an effort to enrich it in a suitable manner, we have attempted to use the fuzzy set theory to provide a more flexible XQuery language, namely “Fuzzy XQuery”. The Fuzzy XQuery is based on the standard XQuery v.1.0 with an added priority, threshold and fuzzy expressions. The interpreter for Fuzzy XQuery has been developed by using Java programming language and the eXist-db database. We can calculate the global constraint satisfaction degree of the result set with the concept of Generalized Prioritized Fuzzy Constraint Satisfaction Problem (GPFCSP) [1] [2].

This paper is organized as follows. The next section contains the literature review. The third section presents the definition of the GPFCSP, compatibility operation and fuzzy ordering options. The architecture and implementation are shown in the fourth section. The fifth section presents an illustrative example and the last section is the conclusion.

II. RELATED WORKS

In this section we briefly review the main approaches of the flexible query techniques focusing on the application of fuzzy set theory.

Škrbić et al. proposed an extension of SQL with fuzzy capabilities called PFSQL (Prioritized Fuzzy Structured Query Language) [3]. A PFSQL interpreter was implemented using the priority fuzzy logic that is based on the concept of GPFCSP.

Many attempts for fuzzy querying in XML documents have been made in recent years. Campi et al. [4] presented FuzzyXPath, an attempt to enhance the flexibility of XPath. They introduced two fuzzy constraints: CLOSE and SIMILAR applied to specific items within XML documents. Moreover, they also defined two flexible conditions for the flexible matching of path structures: BELOW and NEAR. Goncalves and Tineo [5] extended XQuery with the new xs:truth built-in data type to represent gradual truth degrees and xml:truth attribute of type xs:truth to handle the satisfaction degree in nodes of fuzzy XQuery expressions. Their language extension allowed users to declare fuzzy terms and used them in query expressions. Fredrick and Radhamani [6] illustrated their fuzzy XQuery techniques that allowed users to use linguistic terms based on the user-defined function. After that, in 2010 [7], they extended their earlier work by implementing the GUI tool with VB.net for the automatic generation of XQuery and fuzzy XQuery queries. In 2011 [8], they described the fuzzy XQuery process which used the arithmetic operations on fuzzy sets. Recently in 2012 [9], they defined fuzzy information in XML documents and the fuzzy domain integrity constraints through XML schemas for restricting invalid XML data into the XML database.

Panić et al. [10] implemented a similar approach as presented in this paper. They presented the fuzzy XML and fuzzy XQuery extension which used GPFCSP expressions, priority expressions and threshold expressions. The GPFCSP concept was used to calculate the membership degree in the same way as we did. In addition, they also developed a tool for working with XML, XSD and DTD documents, and fuzzy XQuery extension queries. However, the main difference between Panić’s work and our work is that Panić’s implementation used .NET framework, MATLAB and the Microsoft SQL Server database on a windows based application, whereas our approach used Java programming language to implement the new interpreter that was independent of MATLAB with eXist-db native XML database on web based application.

III. BACKGROUND

A. Generalized Prioritized Fuzzy Constraint Satisfaction Problem (GPFCSP)

Škrbić et al. [1] [2] proposed the concept of GPFCSP for calculating the fuzzy membership degrees of PFSQL in fuzzy relational databases.
Theorem: The following system \((X, D, C', \rho, g, \Lambda, \vee, \neg, \top, \bot)\) where
1. \(X = \{x_i \mid i = 1, 2, \ldots, n\}\) is a set of variables,
2. \(D = \{d_i \mid i = 1, 2, \ldots, n\}\) is a set of domains. Every
domain \(d_i\) is a set that contains possible values of variable \(x_i \in X\),
3. \(C'\) is a set of fuzzy constraints:
   \[
   C' = \left\{ \sum_{i=1}^{m} x_i d_i x \cdot \cdots \cdot x_k d_k \mid 0.1, i = 1, \ldots, m, 1 \leq k_i \leq n \right\}
   \]
where \(R_i^f\) denotes the set of constraint variables,
4. \(\rho: C' \rightarrow [0, \infty]\) is the priority of each constraint,
5. \(g : [0, \infty] \rightarrow [0, 1]\) is the global satisfaction degree,
6. \(\Lambda = \Lambda_l\)
7. \(\vee = \vee_l\)
8. \(\neg = \neg_l\)
9. \(\diamond(x_i, c_i) = S_p(x_i, 1-\rho(c_i)), \rho(c_i)\) represents its priority,
10. \(\forall x\) is a simultaneous valuation \(v_x(x_1, \ldots, x_k), x_i \in d_i\) of
    all variables in \(X\) is a GPFCSP. The global satisfaction
degree of a valuation \(v_x\) for a formula \(F\) is obtained in the following way:
\[
\alpha_F(v_x) = F \left( \bigwedge_{i=1}^{m} (v_x \rho(c_i)) \right) R_i^f \in C'
\]
where \(C'\) is the set of constraints of formula \(F\), \(\rho_{max} =
max\{\rho(R_i^f) \mid R_i^f \in C'\}\).

In a similar way, we use the concept of GPFCSP to
calculate the global satisfaction degree of Fuzzy XQuery
because of the \(\text{where}\) clause in a FLWOR expression
that contains a sequence of constraints connected with logical
operators in the same way as in PFSQL.

B. Compatibility Operation

We can compare the fuzzy values in Fuzzy XQuery queries
using standard notation fuzzyvalue1 = fuzzyvalue2. For example,
\(S(x, y) = \text{triangle}(25, 30, 35)\). However, in order to
allow the use of fuzzy values in Fuzzy XQuery queries, we need
to calculate the compatibility of two fuzzy sets to
measure to what extent one fuzzy set is a subset of some other
fuzzy set.

Definition [2]: Let A and B be two fuzzy sets over universe X.
The measure of compatibility of the set A to the set B is
defined as:
\[
\text{Compatibility value } (C_{AB}) = \frac{P(A \cap B)}{P(A)} \quad (1)
\]
where \(P(A \cap B)\) is the area of intersection between two fuzzy
sets and \(P(A)\) is the area of the fuzzy set A.

In our previous work, we implemented the modules for
calculations of compatibility operations. There are three steps
needed to calculate the compatibility in our approach [11].
Firstly, we define the algorithms to find the coordinates of the
intersection area of two fuzzy sets. Secondly, the size of the
shape of the intersection area is calculated as in Equation (2).
\[
\text{Area} = \frac{\left| x_1 y_1 - x_2 y_2 + x_2 y_3 - x_3 y_2 + \cdots + x_3 y_1 - x_4 y_2 \right|}{2} \quad (2)
\]
Lastly, the compatibility value is calculated using Equation
(1).

C. Fuzzy Ordering

As mentioned before, we have defined fuzzy values in
Fuzzy XQuery queries like fuzzyvalue1 = fuzzyvalue2. However, we can compare two fuzzy sets with the relational
operators: >, >=, <, <= like fuzzyvalue1 > fuzzyvalue2. For
example, \(S(x, y) = \text{triangle}(25, 30, 35)\). In this case, we need to
calculate the fuzzy ordering of two fuzzy sets. One usable
definition of fuzzy ordering was proposed by Bodenhofer
[12]. An ordering of fuzzy sets A and B is generalized as:
\[
A \leq B \iff LTR(A) \supseteq LTR(B) \text{ and } RTL(A) \subseteq RTL(B) \quad (3)
\]

LTR(A) stands for Left-to-Right closure which is the
smallest fuzzy superset of A with a non-decreasing characteristic
function, while RTL(A) stands for Right-to-Left closure which is the smallest fuzzy superset of A with a
non-increasing characteristic function. We have proposed the
algorithms and developed modules for fuzzy ordering
calculations in [13].

IV. IMPLEMENTATION

A. Designing the Fuzzy XQuery Grammar

We recall the extension of the XQuery language in EBNF
(Extended Backus-Naur Form) from [14].

B. Compatibility Operation

Having this listing in mind, we conclude that our approach
extends XQuery in the following points.

- Threshold Expression is an expression with the
  keyword threshold that removes results that have a
  membership degree to the result set that is less than the
  specified threshold value in a Fuzzy XQuery query. If
  there is no threshold expression, we assume that the
  value is 0.
- Priority Expression is an expression with the keyword
  priority that defines the level of influence of the
  corresponding constraints on the result. If the query
does not specify the priority expression, the default
  value is 1.
- Fuzzy Expression is an expression that allows users to
  specify fuzzy numbers in XQuery queries. There are
  five types of fuzzy constants:
  - ‘ling’ (‘QNAME’) means a linguistic label with
    the name given by QNAME.
C. Fuzzy XQuery Execution

We implemented an interpreter that allowed for the execution of the Fuzzy XQuery queries defined above. The execution process is shown in Figure 2.

Let us explain in detail how we execute a Fuzzy XQuery. The system first checks the syntax of the Fuzzy XQuery following the given EBNF grammar. After that, if it is valid, the Fuzzy XQuery is transformed to a standard XQuery by parsing the Fuzzy XQuery, creates an Abstract Syntax Tree (AST) and extracts the fuzzy part from it. Next, the system sends the standard XQuery into the database. When the database returns the result set, the system will interpret this result set again using the GPFCSP concept to calculate the membership degree of every element of the result set. Now we have the results that have a fuzzy membership degree in every element. Then, if the query has a threshold expression, the system will remove the tuples which have the fuzzy membership degree under the threshold value. Finally, we print the output to an XML file.

Now we describe how to calculate the fuzzy membership degrees in detail. After we have some result set that was obtained from a standard XQuery query, we calculate the fuzzy membership degree for every element of the result set. We walk the Fuzzy XQuery tree and find the WHERE node. Next, we traverse the whereclause subtree. If the current node represents a conjunction (AND) or disjunction (OR) node, we calculate the global constraint satisfaction degree (α) by calling the Lukasiewicz triangular norm (Tₐ) function or the Lukasiewicz triangular conorm (Sₐ) function, respectively. However, if the current node is an operator (|=, !|=, <, <=, >, >) we walk its child node and check the type of the child node. If it is “=/,” we get the variable after this node. On the other hand, if it is a fuzzy constant (ling, tri, trap, interval or fs), we read the linguistic variable or offsets after this node. After that, we check the type of the operator. We calculate the membership degree by calling the compatibility operation module when an operator is an equality operator (=) or inequality operator (|=). However, if the operator is a relational operator (<, <=, >, >), we call the fuzzy ordering module. Finally, if there is a child node with a priority expression, we aggregate the obtained value with a priority using the triangular product conorm (Sₚ).

V. ILLUSTRATIVE EXAMPLE

In this section, we illustrate the execution of the process of Fuzzy XQuery query with an example. Suppose that we have a Fuzzy XQuery query as in Listing 1 that retrieves the students who are of young age and their height is more than 150 cm with the priority 0.6 and 0.3, respectively. In addition, we define the threshold value equal to the 0.5 meaning that we want the results that have the global constraint satisfaction degree more than 0.5.

Listing 1: An example of a Fuzzy XQuery query

for $x$ in document("student.xml") where $x$/GPA > 2.75 and $x$/age = #ling('young')# priority 0.6 and $x$/height > #tri(100,150,200)# priority 0.3 Threshold 0.5 return $x$

Let us now describe how to calculate this Fuzzy XQuery. First of all, we transform the Fuzzy XQuery to a standard XQuery by removing the fuzzy expressions, priority expressions and threshold expression as shown in Listing 2.

Listing 2: Transformation a Fuzzy XQuery to a standard XQuery query

for $x$ in document("student.xml") where $x$/GPA > 2.75 return $x$
Second, we get the result set after we send the standard XQuery to the database. Third, we send the results back to the interpreter to calculate the global constraint satisfaction degree by calling CalculateMembershipFunction. In this function, the system will remove the non-fuzzy conditions from the Fuzzy XQuery, which in this example is “$x/GPA > 2.75”, as in Listing 3.

Listing 3: The Fuzzy XQuery after removing the non-fuzzy node

Listing 3: The Fuzzy XQuery after removing the non-fuzzy node

for $x$ in document("student.xml")
where $x$/age = #ling(young) priority 0.6 and
$x$/height > #tri(100,150,200)# priority 0.3
Threshold 0.5 return $x$

We use the concept of GPFCSP (as in the preceding section) to calculate the global constraint satisfaction degree for all the result set in step two by using the Equation (4).

$$\alpha = T_L(S_P(\mu_{R_1}^{f}(v), 1- \rho(R_1')), S_P(\mu_{R_2}^{f}(v), 1- \rho(R_2')))$$

(4)

In the Equation (4), $R_i'$ is the fuzzy constraint i and $\mu_{R_i}^{f}$ is the satisfaction degree of constraint $R_i'$. The priority of each constraint is represented by the function $\rho(R_i')$. The greater value of $\rho(R_i')$ means that the constraint $R_i'$ is more important. In this example, the constraint $R_1' : age$ is more important than the constraint $R_2' : height$ because the priority value of the constraint age is 0.6 but the priority value of the constraint height is 0.3. It is noticeable that we use the $T_L$ because of the conjunction AND in this Fuzzy XQuery. The $S_P$ is used to aggregate with priority.

Let us assume that we have the student data in the XML file as in Listing 4 and the result set from the standard XQuery is shown in Listing 5. It is noticeable that Ana’s GPA is not greater than 2.75. Consequently, the result in Listing 5 does not show Ana’s record.

We calculate the constraint satisfaction degree ($\mu_{R_i}^{f}$) for every constraint and every student as in Table 1.

Table 1

<table>
<thead>
<tr>
<th>Name</th>
<th>$\mu_{R_1}^{f}$</th>
<th>$\mu_{R_2}^{f}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>Peter</td>
<td>0.8</td>
<td>0.5</td>
</tr>
<tr>
<td>Alex</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

In the case of the first constraint age, these degrees are obtained directly as the values of the corresponding membership functions of the young linguistic fuzzy variable at the given point of the age data. Suppose that we define the linguistic value of young in an XML document whose membership function have the left fuzzy shoulder which can be seen in Figure 3. However, with the second constraint height, we calculate $\mu_{R_2}^{f}$ by using the fuzzy ordering modules since the type of the fuzzy constant is tri and the operator is >. If we substitute $\mu(R_2')$ and $\rho(R_2')$ for the first student (John) into the Equation (4), we obtain the following:

$$\alpha_{John} = T_L(S_P(0.1, 0.6), S_P(0.5, 0.3))$$

(5)

Therefore, we obtain the global constraint satisfaction degree of John as follows:

$$\alpha_{John} = T_L(S_P(0.1, 0.6), S_P(0.5, 0.3)) = T_L(0.85) = 0.25$$

(6)
Listing 4: The snippet of student data

```xml
<?xml version= "1.0" encoding="UTF-8"?>
<students>
    <student>
        <name>John</name>
        <GPA>3.5</GPA>
        <age>25</age>
        <height>170</height>
    </student>
    <student>
        <name>Peter</name>
        <GPA>3.0</GPA>
        <age>21</age>
        <height>165</height>
    </student>
    <student>
        <name>Alex</name>
        <GPA>2.8</GPA>
        <age>20</age>
        <height>165</height>
    </student>
</students>
```

The other students are calculated in the same way and are given in Table 2.

```
Figure 3: Membership function of young
```

Table 2

<table>
<thead>
<tr>
<th>Name</th>
<th>α</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>0.25</td>
</tr>
<tr>
<td>Peter</td>
<td>0.73</td>
</tr>
<tr>
<td>Alex</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Finally, because of the threshold value, the system will print the results which have the global constraint satisfaction degree more than 0.5 as shown in Listing 6.

Listing 6: The final result set

```xml
<?xml version= "1.0" encoding="UTF-8"?>
<results>
    <student>
        <name>Peter</name>
        <alpha>0.73</alpha>
    </student>
    <student>
        <name>Alex</name>
        <alpha>1.0</alpha>
    </student>
</results>
```

VI. CONCLUSION

In this paper, we present an approach that uses the fuzzy set theory that can manage the imprecise, vague, ambiguous, uncertain or incomplete data with XML technology. We have proposed extensions for the XQuery query language in order to handle flexible fuzzy queries that provide priority, threshold and fuzzy expressions. Furthermore, we implement an interpreter for this language and web GUI using Java programming language and eXist-db. The GPFCSP concept is used to calculate the global constraint satisfaction degrees. In the future, we plan to test the performance of our application with different case studies and develop a more modern web application using AngularJS.

ACKNOWLEDGEMENT

This work was supported by the budget revenue from Prince of Songkla University and Faculty of Science, Prince of Songkla University, Thailand, through the project no. SCIS570329S: A Fuzzy XML Database System and partially supported by the Ministry of Education and Science of the Republic of Serbia, through the project no. 174023: Intelligent techniques and their integration into the wide-spectrum decision support.

REFERENCES


