https://doi.org/10.48009/2_iis_2009_58-65

A STUDY ON DESIGN AND DEVELOPMENT OF FXDB SYSTEM FOR SEMANTIC SEARCH

Seong-Yong Hong, Korea Advanced Institute of Science and Technology (KAIST), gosyhong@kaist.ac.kr
Hae-Yeon Choi, Savannah State University, choi@savannahstate.edu

ABSTRACT

With rapidly increasing number of web users and web contents on the net, intelligent information system and semantic web services are getting more important. In this paper, we propose FXDB (Fuzzy XML Database) system design techniques for semantic information retrieval and suggest a method to generate semantic information, giving it to users in real time through the web search service. This paper also attempts to apply XML and fuzzy techniques to intelligent information system and web service so that users can interpret metadata in the database semantically and automatically. Thus this is a web service tool whose purpose is to automate the interpretation of metadata of products or services more efficiently and intelligently. As a result, web users can make a better and faster semantic web searching along with its rich semantic information provided.

Keywords: XML, Semantic Information, Fuzzy XML Database, Semantic Search, Semantic Web Service

INTRODUCTION

While the number of web users and web contents has been increasing rapidly over the last decades, new technologies and services for web usage are introduced in various fields. It, however, may not be good enough for the web surfers to simply manipulate more data of the real world objects on the net. The functionality of the web service should be more intelligent and semantic, so that they are more meaningful and informative to its users. With its various use of XML technology in program languages, different data files, documentation, etc., XML-based web services have been expanded into every field. As a result, many researches in various fields have been conducted for standardization of XML usage [1]. XML is being used as a most basic markup language for semantic web and it has also become a foundation of newer semantic web technology such as RDF or RSS [2, 3, 4, 5].

As other internet related hardware and software technologies, a web technology is also in rapid change from Web 1.0 to Web 2.0 and a few years later, we may use a Web 3.0. The web service technology we are using currently is often referred to as Web 2.0. For Web 2.0, it is really important to represent and provide the web users with needed information to make their purchasing decision in more semantic manner. Many different ideas and approaches for running web services that way have been researched in various fields [6]. Rather than just use the web to complete a business transaction between consumer and business on the web, the usefulness of web service, however, should be something that any users can easily understand the results of the search retrieved. That is, providing simple information to its users' web search may not be enough anymore in this information age. In other words, information should contain "semantic information" in itself as we see in Blogs, RSS, and Ajax. However, such kinds of web techniques cannot avoid having a certain limitation; since the web contents are created in users' own hands and are apt to include vague meanings, so that it is difficult to be properly interpreted.

We, therefore, refer to semantic information as information including semantic context around objects – products or services -, which web surfers want to know more about. For example, there is a product whose price can be expressed numerically. The numerical value of $100.00 itself is not yet semantic information. As shown, however, in Figure 1, "*expensive*" is a kind of semantic information. The fact, "*Being expensive*", can deliver more useful meaning to users on the web. The information of numerical value of the product should be transformed into semantic information, resulting from semantic web services. Figure 1 also shows the relationship between metadata and semantic information graphically. In this example, the numeric data but not yet semantic information is being transformed into semantic one to give web user a certain meaning for their decision making process, namely, "*this is somewhat expensive*" to me.

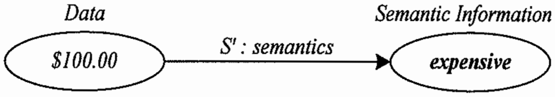


Figure 1. Relationship between Data and Semantic Information

Above example demonstrates well about what this research attempts to explain with semantic web services. First, it extracts metadata, in this example, a price of the product. This metadata can be found in XML documentations representing the product. Second, it saves the extracted information in XML database. Finally it, then, analyzes the information in a framework to decide whether the product is expensive or not to a user; this step also is done automatically. With the three steps, web service users will be provided with semantic information of the desired product.

One of the main focuses of the research is on automatic generation of semantic information. For that end, the concepts of fuzzy set, fuzzy number, and fuzzy logic can be applied. At the same time an algorithm is created so that a web user can extract semantic information from metadata (i.e., the price) of real objects on the web (i.e., some real products or services). This research also designs trigger events, which correspond to insertion of, update of, and deletion of metadata. Lastly it shows some examples for which our model can be applied for the experimentations of the hypothetical objects. The remainder of this paper is organized as follows: Section 2 describes the related works about fuzzy technology and XML database. Section 3 explains the architecture of the FXDB system. In section 4, the algorithm for automatic generation of fuzzy data and the algorithm for getting semantic information using fuzzy data are explained. In section 5, the result of the implementation and experiment of FXDB system is being discussed. Finally, section 6 summarizes the research results and future works.

MOTIVATION AND RELATED WORKS

Metadata, for example, information of a product, includes price, id, size, color, description, texture, shape, etc. To search a desired product or service on the web effectively and efficiently, the description of metadata should be systematic and structuralized. There have been many preceding researches about representation of metadata through XML in the literature. An example of XML document, as shown in the next column of this page, describe metadata of product information. For that purpose, XML-Schema is used to define the form of XML documents for product metadata; that is, a XML-Schema is designed to represent product metadata containing multiple objects. They try to collect diversified and complicated features of the objects. However, for example, XML documents in the XML-schema do not give users enough and useful information in their decision-making as to which product to buy or not to buy among different products. Because they considered either only data of the product itself or simple metadata, so it was hard for them to represent human's semantic information. Thus the approach could not provide web services with converting some data automatically to semantic one, because they did not consider vagueness of a user expression; as a result, they could not support the user to search for something including semantic information. A user expression being semantic is related to the fact that he or she makes a different decision according to the change of the numeric value mapped to the data.

```
<?xml version='1.0' encoding='euc-kr' ?>
<products xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <product>
  <iid>i00001</iid>
  <ref> image01.jpg</ref>
  <subject> man shirt </subject>
  <category> fashion </category>
  <description/>
  <object> <oid>o00001</oid>
    <MIDS>
      <mid>m0001</mid>
      <material> cotton 100% </material>
      <size> large </size>
      <made> <company> samsung </company>
               <nation> korea </nation>
               <brand>SMC</brand>
          <date> 10/15/2007</date> </made>
      <price> $100.00 </price>
    </MIDS>
    <semanticDS> <sdsid>sd0001</sdsid>
      <when>08/28/2007</when>
      <where/>
      <why/>
      <what> shirt</what>
      <who> man </who>
      <how/>
    </semanticDS>
    <featureDS>
      <color> <cid>c0001</cid>
        <r>255</r><g>0</g><b>0</b><h>0</h><s>0</s><i>100</i>
        <color_SD> warm</color_SD>
           <color_text>red</color_text>
      </color>
      <texture><tid>t0001</tid>
        <texture_type>0001000000000000</texture_type>
        <texture_SD>coarse</texture_SD>
           <texture_text>stripes</texture_text>
      </texture>
      <shape><sid>s0001</sid>
        <shape_type>0100000000000000</shape_type>
        <shape_SD>simple</shape_SD>
        <shape_text>rectangle</shape_text>
      </shape>
      <spatial> <spid>sp0001</spid>
        <location>NE</location>
        <spatial_SD>enough</spatial_SD>
        <RO> <NW/> <N/> <NE/> <E/> <SE/> <S/> <SW/> <W/> </RO>
      </spatial>
    </featureDS>
  </object>
 </product>
</products>
```

For instance, say that the price of a certain product A has been fixed to *$100.00* for several years since 2004. Customers thought that this A was expensive in 2004. However, the same customers might think this A was relatively cheap in 2005 and might say "*this is very cheap*" in 2007. Namely, if semantic information about a certain product has been changed, web users would make a different decision. This is a typical example for the change of semantic variable or linguistic variable with static number. Here is another example. The price of the same A was *$100.00* in 2004. Customers thought that this was expensive. In 2005, as the price went down to *$80.00*, customers thought this was cheap. And in 2007, the price of it was *$50.00* so that customers thought "*this is very cheap*". In other words, along with the variation of the price of a certain product, a user may extract different semantic information or evaluation from the product. This is a typical example for the change of semantic variable with variable number. Figure 2 describes the related works about variable number and semantic variable.

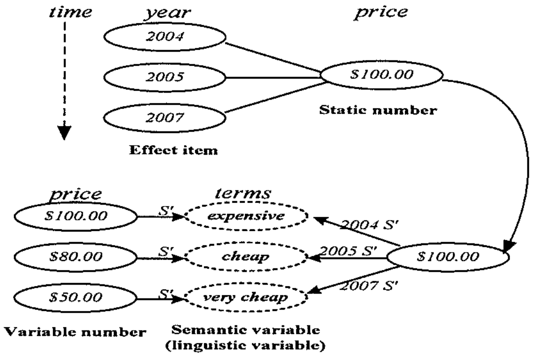


Figure 2. Concept of the Change of Semantic Variable

A Semantic variable (or linguistic variable) is the word, in natural language, that expresses or identifies a fuzzy set that may or may not be formally defined. Thus, the membership function $\mu_A(x)$ of a fuzzy set *A* expresses the degree in which *x* verifies the category specified by *A*. With this definition, it is assured that in our everyday life we use several linguistic variables for expressing abstract concepts such as expensive, cheap, young, old, hot, cold, etc. The intuitive definition of these labels not only varies from one person to another and depends on the moment, but also varies with the context in which it applies. For example, an "*expensive*" person and an "*expensive*" price do not measure the same thing. Fuzzy techniques and XML database techniques, used for generating semantic information from metadata automatically, are followed in the next section.

Fuzzy Technique

Over the last decades, artificial intelligence (AI) has been actively studied and researched. Within AI area, computers can handle the vagueness of human expressions and numeric values of products. Fuzzy theory is the basic tool for dealing with this vagueness [7].

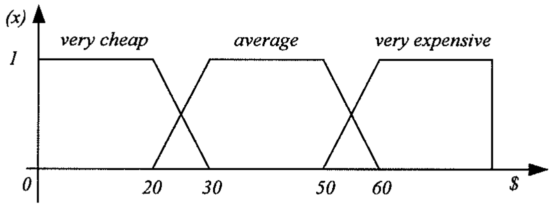


Figure 3. Fuzzy Membership Functions

For example, human being's expression such as "*expensive*", "*average*" or "*very expensive*" can be explained more clearly by something formal or numeric value. Other description of a product like "*light red*" or "*dark red*" should be treated as vague expression. As shown in Figure 3, a Fuzzy set can be designed to deal with human being's common expression of vagueness in the purchasing process of a desired product or service. There are several types of fuzzy set [8, 9, 10]. Suppose that an element ***e***, of a certain set ***B***, is mapped to a value, ***v***, which is from 0 to 1. ***e*** and ***v*** construct a pair which is an element of fuzzy set related to ***B***. ***e*** can be either discrete or continuous. In later case, the range of ***e*** is either limited or not limited. Look at Figure 4, which illustrates that well. Depending on the type of membership function, different types of fuzzy sets will be obtained. Zadeh proposed a series of membership functions that could be classified into two groups: one group is made up of straight or "linear" lines, and the other is "curved" lines [11].

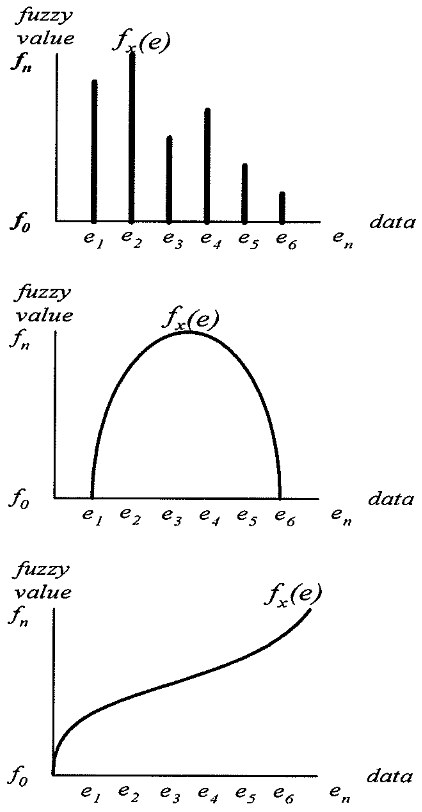


Figure 4. Types of Fuzzy Set

Say that the set of all documentation of the products which are being summated as ***I***. Then, the ***I*** can be defined as follows:

$$I = \{ i_1, i_2, i_3, i_4, i_5, i_{6,}, \ldots\ldots, i_n \}$$

In this case, fuzzy set A can be defined as formula 1:

$$A = \mu_A(i_1)/i_1 + \mu_A(i_2)/i_2 + \ldots + \mu_A(i_n)/i_n$$
$$= \sum_{k=1}^{n} \mu_A(i_k)/i_k \qquad \ldots\ \ldots\ \ldots\ \ldots \quad (1)$$

If the element of the set which is related to the given fuzzy set is continuously distributed, then it defines that fuzzy set A as formula 2, using the following fuzzy membership function:

$$\int \mu_A(i)/i \qquad \ldots\ \ldots\ \ldots\ \ldots \quad (2)$$

The fact that the theory of fuzzy sets generalizes the theory of classic sets means that the fuzzy sets allow operations of union, intersection, and complement. These and other operations can be found in Petry and Pedrycz and Gomide, such as concentration, dilatation, contrast intensification, and fuzzification, which are being used when linguistic hedges such as "*expensive*" or "*not expensive*" are used [12]. Operations on three fuzzy functions are defined by formula 3, as follows:

$$A \cap B = \int \mu_A(i)/i \cap \int \mu_B(i)/i = \int (\mu_A(i) \wedge \mu_B(i))/i$$
$$A \cup B = \int \mu_A(i)/i \cup \int \mu_B(i)/i = \int (\mu_A(i) \vee \mu_B(i))/i$$
$$A^c = \mu_{A^c}(i) = 1 - \mu_A(i) = \int (1 - \mu_A(i)) \ldots\ \ldots\ \ldots\ \ldots \quad (3)$$

XML Database

There are not a few researches have been conducted related to XML documents, for XML documents contain much information in diverse forms. In addition, researches on XML store-and-management and XML related query have been done a lot. In particular, the issues about how to store XML data in database and query about them have been widely studied; for instance, XML database system which is only for XML and the other known method uses existing database systems such as RDB or ORDB [13, 14, 15, 16]. Hence the difference of data model between XML's and existing databases can be challenge, abundant methodologies to store data in database have been proposed. In addition, many studies aim at transformation of data in existing database into XML documents. Database system which extracts and unifies distinct information on the web also has been studied. Queries related to XML are such as XPath, XML-QL, XQL, Quilt, and XQuery [17, 18]. These queries support structure-based and content-based searching technique while taking into account of a structural property in a XML document. They enable search algorithm for various XML document, such as content-based search and structure-based search. By using these queries, XML also can be transformed into data in various formats. CSS or XSL can be applied to transform XML documents into data in other format. It is provided as a document formatted by XSLT [19, 20]. Formatted XML document is the same XML document with different format.

THE ARCHITECTURE OF THE FXDB SYSTEM

Figure 5 shows the FXDB system, which is for semantic web services. FXDB saves and manages XML documents and dictionary of fuzzy data. It also includes the module for automatic generation of fuzzy data, which works with the triggers. Since FXDB extracts metadata from XML documents directly, it can generate semantic information in real time.

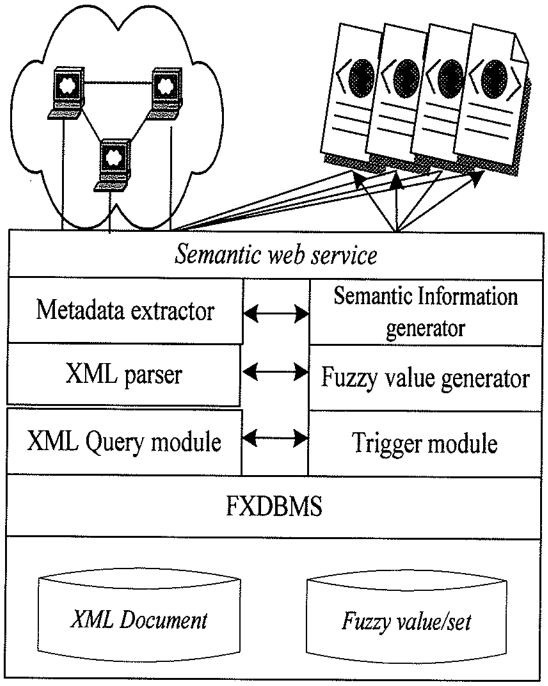
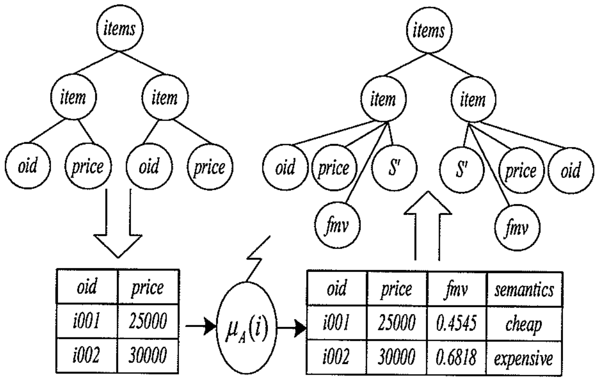


Figure 5. The Architecture of the FXDB System

Automatic Generation of Semantic Information

A methodology, enabling automatic generation of semantic information from extracted numerical data, was introduced. The methodology should extract metadata from XML documents, store it in database, and generate fuzzy data automatically. The fuzzy data generated should be stored in the dictionary of fuzzy data. Figure 6 shows these steps graphically. Information in fuzzy dictionary can be served in the form of XML documents.

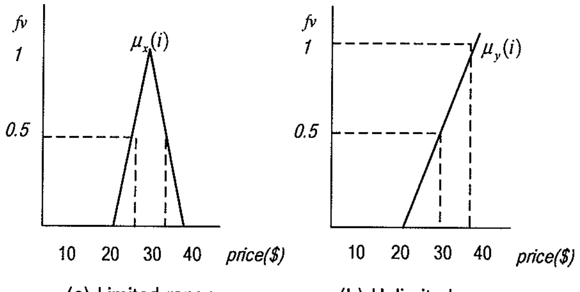


oid	price
i001	25000
i002	30000

oid	price	fmv	semantics
i001	25000	0.4545	cheap
i002	30000	0.6818	expensive

Figure 6. Automatic Generation of Fuzzy Data

Figure 7 shows that fuzzy set is applied to the price of the product; with an example of (a), the range of the price is limited. On the other hand, in the case of (b), the price is continuous with no upper bound.



(a) Limited range

(b) Unlimited range

Figure 7. Fuzzy Set of Price

Let ***A*** be a fuzzy set of ***I*** for this example. ***A*** is the set of pairs. The pairs consist of ***i***, which is an element of ***I***, which, in turn, is the set of all products and $u_A(i)$. Then, $u_A(i)$ is defined as follows:

$$\mu_A : I \rightarrow [0,1]$$

$u_A(i)$ is a membership function. If $u_A(i)$ is closer to 0 than 1, then, ***i*** is apt to belong to fuzzy set of ***A***. Set A is defined as formula 4:

$$A = (i, \mu_A(i)) \mid i \in I \qquad \dots\dots\dots\dots \quad (4)$$

To generate fuzzy data from information of the price of the product, an algorithm shown in Figure 8 is used. i_x represents the price of the random product. ***min_value*** is the price of the cheapest one of all. ***max_value*** is the highest price of all. This algorithm, first, extracts metadata about price from XML documents representing products. And then it computes their fuzzy values and stores those ***fuzzy_value*** in fuzzy dictionary.

```
Procedure 1. Generate_fuzzy_data (num:integer; row_count:integer; min_value:float; max_value:float, real_value:float, fuzzy_value: float)
BEGIN
num ← 1 ;
row_count ← COUNT($i_x$) ; /* number of record */
min_value ← MIN($i_x$) ;   /* minimum value */
max_value ← MAX($i_x$) ;   /* maximum value */
 WHILE num < (row_count+1)
   BEGIN
   real_value ← SELECT(num) ;
   front_value ← real_value - min_value ;
   rear_value ← max_value - min_value ;
   fuzzy_value ← (front_value / rear_value) ;
    IF fuzzy_value ≠ null DO
     BEGIN
     UPDATE(num);
     END
   num ← num+1 ;
   END
```

```
END
End of procedure 1
```

Figure 8. Auto Generation of Fuzzy Data Algorithm

This algorithm generates saving procedures and then stores these procedures in database. Saved procedures operate on metadata of the products and make fuzzy data as output. The algorithm shown in Figure 9 is executed when it is triggered. This algorithm also generates semantic information and saves it in database at the same time. If new metadata about the products comes up, the Procedure 2 will be triggered, and then it will be also executed in real time. In the algorithm, ***low_weight***, ***middle_weight***, and ***high_weight*** are weights for fuzzy data. Very cheap or cheap products have weight which varies from *0* to *0.25*. As a result, cheap one has weight in the range of ***{0.5 ≥ weight ≥ 0.25}***. For expensive products, the weight is in ***{0.75 ≥ weight ≥ 0.5}***. Very expensive products have weight in ***{1.0 ≥ weight ≥ 0.75}***. For convenience, let ***low_weight*** be *0.25*, ***middle_weight*** be 0.5, and ***high_weight*** be 0.75. Namely, unit weight, *0.25 (¼ = 0.25)* is applied to T_i, because T_i is categorized into "*very cheap*", "*cheap*", "*expensive*", and "*very expensive*".

```
Procedure 2. Generate_semantic_information
(get_price:integer; fuzzy_data:float)
BEGIN
fuzzy_data ← select_from_table(get_price)
  IF fuzzy_data < low_weight THEN
   { IF SI_data == null  INSERT($T_i$)  /* $T_i$ : Term */
     ELSE  UPDATE($T_i$) }
  ELSE IF fuzzy_data < middle_weight THEN
   { IF SI_data == null   INSERT($T_i$)
     ELSE  UPDATE($T_i$) }
  ELSE IF fuzzy_data < high_weight THEN
   { IF SI_data == null INSERT($T_i$)
    ELSE UPDATE($T_i$) }
  ELSE
   { IF SI_data == null INSERT($T_i$)
     ELSE UPDATE($T_i$) }
END
End of procedure 2.
```

Figure 9. Algorithm for Auto Generation of Semantic Information Based on the Triggers

The module generating semantic information also generates saving procedures and stores these procedures in database. These procedures make semantic information by using fuzzy data. If there was any insertion of new metadata, these procedures are invoked by a trigger and generate semantic information and new fuzzy data. Figure 10 shows the trigger module of the FXDB system. When there is any insertion of new metadata, *f(i)* will be executed. In the case of update of metadata, *f(u)* will be executed. And when deletion occurs, *f(d)* will be executed. *f(t)*, however, is triggered periodically, considering the performance of the system. These trigger modules automatically change a fuzzy data based on a change in a data and results in automatically converting web service with semantic information.

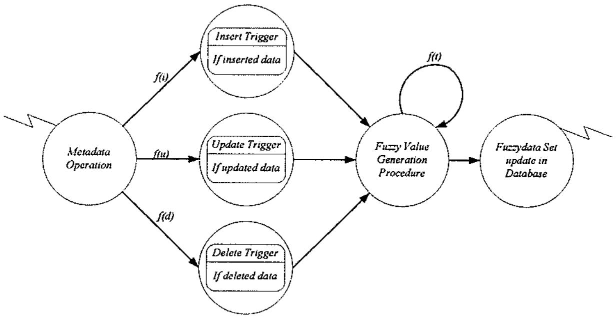
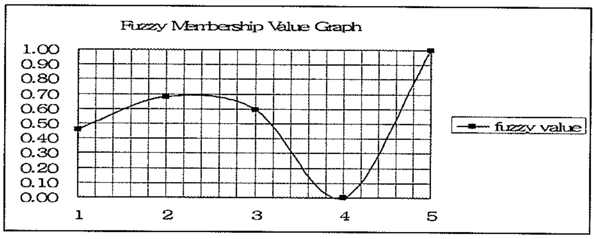


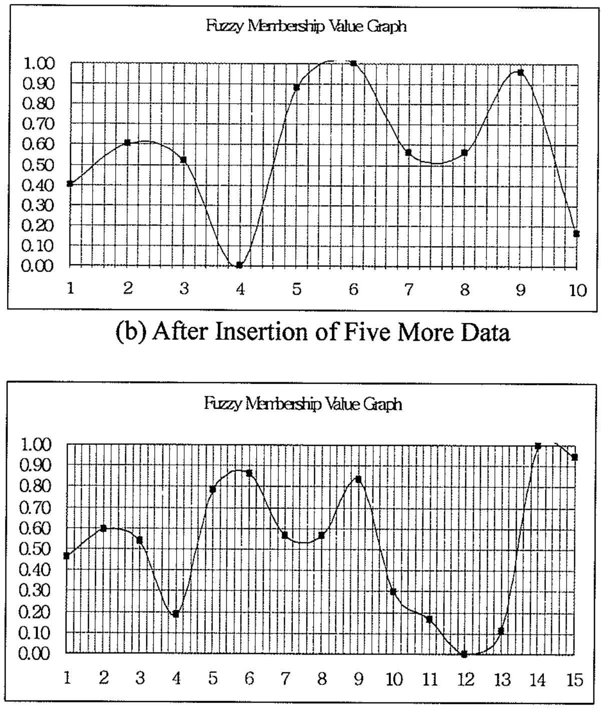
Figure 10. The Trigger Module of the FXDB System

If the trigger module processes metadata and generates semantic information in real time, it can enhance efficiency and accuracy of web services, since it uses required information in real time. However, if the amount of metadata increases rapidly, the level of performance becomes an issue. Thus, a way to process the module more efficiently needs to be studied. For example, after insertion of first five metadata, the resultant return values of fuzzy membership function of those five are displayed in (a) of Figure 11. Among those five, *i(5)* is the most expensive one and has *1.0* as its fuzzy value. In contrast, *i(4)* has *0* as its fuzzy value since it is the cheapest one. The last three prices, *i1*, *i2*, and *i3*, can be easily seen as medium price range, whose value is *0.45*, *0.7*, and *0.6*, respectively.

Figure 11 (b) shows change of return values of fuzzy membership function after insertion additional five items. And what (c) shows is the result that we insert five more data again.



(a) After Insertion of First Five Data



(b) After Insertion of Five More Data

(c) After Inserting Five More Data Again

Figure 11. Graph Representing Change of Return Values of Fuzzy Membership Function

Due to generation of semantic information from fuzzy data, web users may be given more meaningful information for their product purchasing decision. For example, it is sometimes hard for users to decide whether a product which costs *$100.00* is expensive or not. However, if web services provide users with a price of product and additional price information, saying the product is "*expensive*" or "*very cheap*", the web users can be better served in their products or services purchasing decisions. In Figure 12, the FXDB table which results from our algorithm for generation of semantic information (i.e., Procedure 2) is displayed.

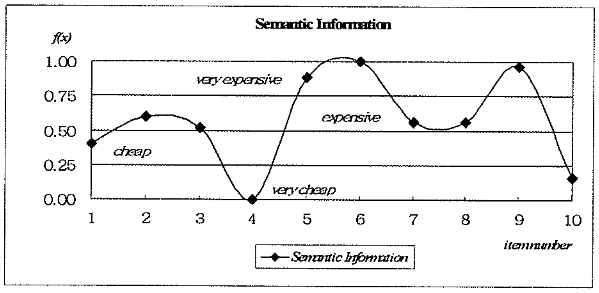


Figure 12. The Result of Generating Semantic Information

IMPLEMENTATION AND EXPERIMENT

The result of experiment, enabled by implementation of semantic web service system based on the FXDB system design, is being displayed. The proposed system can use XML documents containing information of products under real web environment. And this system can store and manage XML data that is mapped into relational database. For implementation, a computer system with Pentium 4 CPU with 2.80GHz and 1GB RAM is used. MS-SQL Server 2005 as database system is utilized too. Figure 13 shows the change of semantic information about the product price which is served to users in real time.

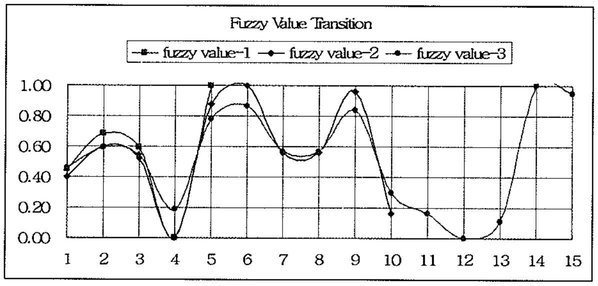


Figure13. The Change of Semantic Information, which Is Served Through the Web in Real Time

As it can be seen above, *i(2)* was announced to be "*very expensive*". However its estimation has been changed. *i(2)* is "*expensive*" now, after evaluating new metadata around the prices of products. Figure 14 shows the performance with varying amount of data. The Figure also shows whether the type of the triggering event affects the performance or not. It, however, can be concluded that the type of the triggering event does not really matter, but the amount of data makes the performance seriously slower. So improvement of the performance of trigger module is on demand.

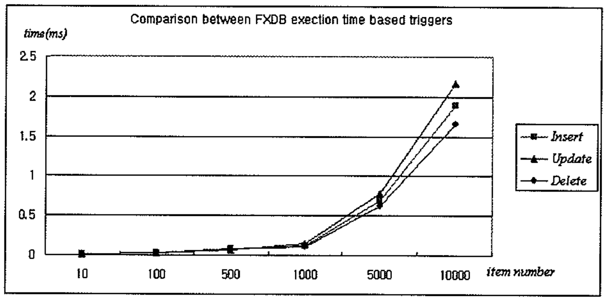


Figure 14. Comparison between FXDB Execution Time

CONCLUSION AND FUTURE WORKS

This research attempted to use the FXDB system for semantic web service. This system investigated how to adopt XML and fuzzy techniques; a way to use XML and fuzzy data to provide web users with semantic information in real time. In addition to that, the technique is experimented about how to intelligently automate analysis on a metadata for semantic web environment. Thus the system can be able to provide users with more useful and meaningful web services and semantic-based web search. These services may allow users to make a faster product purchasing decision making and can be used for effective searching for information of products. For the future research, the various metadata on the web and semantic web environment for intelligent business will be added into this experiment.

ACKNOWLEDGEMENTS

This research was supported by the R&E Program of the Korea Science and Engineering Foundation (KOSEF).

REFERENCES

1. World Wide Web Consortium Recommendation. (2004). Extensible Markup Language (XML) 1.1. Available: www.w3.org/TR/2004/REC-xml11-20040204/
2. World Wide Web Consortium. (2001). Semantic Web. Available: www.w3.org/2001/sw/
3. Tim Berners-Lee, James Hendler & Ora Lassila. (2001). The Semantic Web. Science American. Available: www.sciam.com
4. Lassila, O. & Webick, R.(1999). Resource Description Framework (RDF) model and syntax Specification. W3C Recommendation, Available: www.w3.org/TR/PR-rdf-syntax
5. Brickley, D. & Guha, R.V. (2004). Resource Description Framework (RDF) Schema Specification. W3C Recommendation. Available: www.w3.org/TR/rdf-schema
6. Mika, P.(2005). Flink: semantic web technology for the extraction and analysis of social networks. *Journal of Web Semantics, 3(2)*, 211-223.
7. Buckley, James J. & Eslami, Esfandiar.(2002). An Introduction to Fuzzy Logic and Fuzzy Sets. Springer publishers.
8. Chaudhry, N., Moyne, J. & Rundensteiner, E. (1994). Designing Databases with Fuzzy Data and Rules for Application to Discrete Control. *University of Michigan, CSE Division, Technical Report*, CSE-TR-224-94.
9. Chamorro-Mart´ınez, J., Medina, J.M., Barranco, C., Gal´an-Perales, E. & Soto-Hidalgo, J.M. (2005). An Approach to Image Retrieval on Fuzzy Object-Relational Database using Dominant Color Descriptors. *In Proc. of 4th Conference of the European Society for Fuzzy Logic and Technology*, 676–684.
10. Han, J. & Ma, K.-K.(2002). Fuzzy color histogram and its use in color image retrieval. *IEEE Transactions on Image Processing, vol. 11(8)*, 944-952.
11. Zadeh, L.A. (1965). Fuzzy sets, Information and Control 8(3), 338-353.
12. Gomide, F.e Pedrycz, W. (1998). An Introduction to Fzzy Sets-Analysis and Design. *MIT, New York.*
13. Daniela Florescu & Donald Kossmann.(1999). Storing and Querying XML Data using an RDBMS. *Bulletin of the Technical Committee on Data Engineering*, 22(3), 27-34.
14. Jayavel Shanmugasundaram, Kristin Tufte, Gang He, Chun Zhang, David J. DeWitt, & Jeffrey F. Naughton. (1999). Relational Databases for Querying XML Documents: Limitations and Opportunities. *VLDB*, 302-314.
15. Takeyuki Shimura, Masatoshi Yoshikawa, & Shunsuke Uemura.(1999). Storage and Retrieval of XML Documents using Object-Relational Databases. *Proceedings of the 10th International Conference on Database and Expert Systems Applications*, 206-217.
16. Shanmugasundaram, J., Eugene Shekita, Jerry Kiernan, Rajasekar Krishnamurthy, Efstratios Viglas, Jeffrey Naughton & Igor Tatarinov. (2001). A General Technique for Querying XML Documents using a Relational Database System. *ACM SIGMOD Record Vol. (30)*, 20-26.
17. World Wide Web Consortium Recommendation. (2007). XML Path Language (Xpath) 2.0. Available: www.w3.org/TR/xpath20/
18. World Wide Web Consortium Recommendation. (2007). An XML Query Language (XQuery) 1.0. Available: www.w3.org/TR/xquery/
19. World Wide Web Consortium Recommendation. (2006). Extensible Stylesheet Language (XSL)1.1 Available: www.w3.org/TR/xsl/
20. World Wide Web Consortium Recommendation. (2007). XSL Transformation (XSLT)2.0 Available: www.w3.org/TR/xslt20/