The Fuzzy Logic Based ECA Rule Processing for XML Databases

Thomson Fredrick[1]and Govindaraju Radhamani[2]
[1]R&D Centre, Bharathiar University, India
[2]School of IT and Science, Dr.G.R.D College of Science, India

Abstract: *Current needs of E-Commerce transactions require the development of XML database system like relational database systems. Fuzzy concepts are adapted to the field of XML Databases (DB) in order to deal with ambiguous and uncertain data. Incorporating fuzziness into Event Condition Action (ECA) rules would improve the effectiveness of XML DB as it provides much flexibility in defining rules for the supported application. An architecture that specifies how the fuzzy logic based rules are processed in the context of XML database transactions is presented in this paper. The algorithm for implementing fuzzy active rule based triggers for XML is proposed in this paper. The proposed architecture provides new forms of interaction, in support of fuzzy ECA rules between any application programs and the XML database. This paper presents a motivating example that illustrates the use of fuzzy trigger in stock market brokering agency. The testing has been done to compare the performance of fuzzy XML triggers and normal XML triggers. Our testing results show that Fuzzy ECA rule based triggers are providing better output than Normal ECA rule based triggers.*

Keywords: *XML DB, ECA rules, fuzzy eca rules, fuzzy xquery, fuzzy trigger.*

Received August 30, 2012; accepted March 20, 2014; published online August 16, 2015

1. Introduction

Traditional XML Databases (DB) were designed for efficient storage, safe modification and correct recuperation of large samples of precisely defined data. XML DB try to model real world data using few and precise structures, but unfortunately we are surrounded by uncertainty and imprecise information. Human beings manipulate very well that kind of information and in fact, frequently we take decisions based on these pieces of information. We reason with vague terms such as "The price is high" or "The car is going too fast". These facts are true (or false) only to some extent and computers may not process them. The fuzzy set theory introduced by Zadeh [24] has been utilized in many areas of research, from control to DB and expert systems [2, 3, 4]. Fuzzy rules in expert systems and control that make use of the concepts in fuzzy sets were proven to increase the flexibility and effectiveness of the systems. Special inference mechanisms have been developed for fuzzy control to be used to infer a fuzzy conclusion from a set of fuzzy rules [2]. So, we need fuzzy logic based XML database model to deal with uncertainty and vague information.

In XML database research, there are several proposals to develop models that support handling fuzziness, uncertainty and imprecision of real-world. The research area of fuzziness in XML Data Base Management Systems (DBMS) has resulted in a number of models aimed at the representation of imperfect information in DB and flexible fuzzy XQueries and fuzzy triggers on traditional XML DB. native XML DBSs are passive, in the sense that they only manipulate data in response to explicit requests from applications. An active XML DB is a DB that allows users to specify actions to be taken automatically, without user intervention, when certain conditions arise. Fuzzy Event Condition Action (ECA) rules provide active DB capabilities beyond what is found in a conventional, passive XML DB. This paper proposes architecture and algorithm for fuzzy logic based ECA rule processing in XML DB that implements our knowledge model and execution model.

2. Related Works

Applying fuzzy concepts to DBSs has been an active research topic over years. There are many research projects in the past to incorporate fuzzy concepts into relational DB. Bosc and Pivert [7] did the research on extending DB management systems functionalities with fuzzy querying called SQLf. Galindo *et al.* [11] developed a FSQL server that allows writing flexible conditions in queries. Limited research has been done so far on incorporating fuzziness into triggers. Bouaziz *et al*. [8] proposed a fuzzy trigger named C-Fuzzy Trigger to embed fuzzy rules within a Boolean Valued trigger condition. Montesi and Torlone [16] introduced a formal approach to active rule processing that relies on a method for rewriting user-defined transactions to reflect the behavior of a set of active rules. Anton and Tarik [2] did the research on fuzzy action and the reasoning mechanism with the purpose of selecting the actual action.

In [2, 8] the researchers focused their research on their special control systems, which cannot be easily

applied to other systems or application contexts. Bonifati *et al*. [5] presented Active XQuery an active language for XML repositories presented as an extension of XQuery. They proposed the syntax of Active XQuery and its semantics by describing an algorithm for support triggers and sketchy system architecture. Ferraz *et al*. [9] present a non-intrusive approach to store and manage Active XML documents. They also define a methodology to materialize Active XML documents at query time. Their storage approach is based on plain relational tables and user-defined functions of Object-Relational DBMS to trigger the service calls. Bonifati *et al*. [6] proposed the active XML rules for pushing reactive services to XML-enabled repositories. Their proposed active rules operate on XML documents and deliver information to interested remote E-Commerce users in reaction to update events occurring at the XML repository site. Their proposed active rule mechanism used an XQuery, standard DOM events and SOAP interchange standard for deliver information. Bailey *et al.* [3] presented a language for event-condition-action rules on XML repositories. They investigated the methods for analysing the behaviour of a set of ECA rules and examined more deeply the triggering and activation relationships in XML ECA rules. Prabha *et al*. [18] provided an efficient way of querying among many distributed and heterogeneous data sources. They introduced an XML oriented common data model and an XML Parser (XP) for accepting the SQL statement in distributed DB. The proposed XML based optimized query processor fastens the query processing time.

Liu and Goh [15], introduces an approach to define XML triggers through XML schema in order to realize various active functions in XML database. Bernauer *et al*. [4] explored composite event detection in XML documents. They explained composite event detection using event algebra snoop, as it is extensible and well defined for XML. In another research by Xu *et al*. [22] a XML-based composite event model that consists of the temporal logical model and the event composite pattern was presented. Swamynathan *et al*. [20] presented a system that handles complex events include simple, temporal, and composite events. The system they used a specially designed language called Generic Composite Event Rule Markup Language (GCERML) to implement a system. Their proposed system used an inference engine called Java Expert System Shell (JESS), to monitor the complex events in an easy and efficient manner. Jin and Bhavsar [12] did the research on investigating the use of fuzzy expressions in triggers in relational DBSs.

Anders *et al*. [1] introduce path-level granularity in combination with the novel concept of XML DB trigger to implement the validation of complex constraints. They propose XML DB trigger methodology to support both traditional data integrity constraints and advanced semantic constraints effectively and efficiently.

In our earlier work [21], we have introduced a fuzzy constraint-based framework for XML schema in our earlier work. Our approach is able to handle uncertainty in schema matching and data inconsistency in native XML DBSs by exploiting fuzzy constraints and it will restrict invalid XML data into XML DB by using fuzzy domain integrity constraints.

Jin and Bhavsar [13] presented a trigger language, named FZ-Trigger, to allow fuzziness in relational DB triggers. Their proposed FZ-Triggers will handle temporal events generated at a given time interval. They implemented temporal event triggers using oracle DB using Java as a front end. Ozgun *et al*. [17] proposed a system that is capable of automating the transformation of relational bitemporal DB into XML regardless of the underlying DB management system. They have implemented a fuzzy query model as part of the proposed framework in order to provide flexibility to a wide range of end users willing to access the database. Farnaz *et al.* [9] propose the fuzzy ECA rule-based negotiation agents in E-Commerce to negotiate between sellers and buyers in order to get the best deal. They use a fuzzy decision tree to understand and adapt other agents' behavior in order to produce the best contracts. In this paper, we propose an algorithm to implement fuzzy active rule based triggers and an architecture of fuzzy ECA rule processing system. The performance and output of fuzzy active rule based XML triggers and normal triggers is compared and analysed.

3. Technical Background

This section introduces basic concepts of fuzzy sets and fuzzy inference required to define fuzzy active rule based triggers. Informally, a fuzzy set is a set with imprecise boundaries in which the transition from membership to non-membership is gradual rather than crisp. In this way, a fuzzy set F in a universe of discourse U is characterized by a membership function μ_F, which associates each element $u \in U$ with a grade of membership $\mu_F(u) \in [0, 1]$ in the fuzzy set F. Note that, a classical set A in U is a special case of a fuzzy set with all membership values $\mu_A(u) \in \{0, 1\}$.

3.1. Linguistic Variable

The basic concept underlying fuzzy logic is a linguistic variable, which is a variable whose values are words rather than numbers. A linguistic variable is characterized by a quintuple (x, $T(x)$, U, G, M) in which x is the name of the linguistic variable; $T(x)$ is the term set of x, that is, the set of names of linguistic values of x defined on U; G is a syntactic rule for generating the names of values of x; and M is a semantic rule for associating with each value its meaning.

Let us consider the linguistic variable Temperature. Its term set T(Temperature) could be T (Temperature)

= {low, normal, hot} where each term is characterized by a fuzzy set in a universe of discourse U= [0, 300]. We might interpret "low" as "a temperature below 100°C," "normal" as "a temperature close to 120°C," and "hot" as "a temperature above about 130°C". These terms can be characterized as fuzzy sets whose membership functions are formulated. For example, if the current temperature is 90°C then the membership degree to the fuzzy subset low is equal to 0.6.

$$\mu_{low} = Trapezoidal\left(0,0,80,100\right)$$
$$\mu_{normal} = Trapezoidal\left(90,120,120,140\right)$$
$$\mu_{hot} = Trapezoidal\left(130,160,300,300\right)$$

3.2 Fuzzy Inference

A fuzzy implication is viewed as describing a fuzzy relation between fuzzy sets forming the implication. A fuzzy rule, such as "if *X* is A then *Y* is B" is a fuzzy implication which has a membership function $\mu \to B\left(x,y\right) \in \left[0,1\right]$.

Note that $\mu \to B\left(x,y\right)$ measures the degree of truth of the implication relation between *x* and *y*. The if part of an implication is called the antecedent (premise), where as the then part is called the consequent. Using the Mamdani's (minimum) implication, the membership function of the fuzzy implication is defined as:

$$\mu \to B\left(x,y\right) = \min\left[\mu_A\left(x\right), \mu_B\left(x\right)\right] \quad (1)$$

It is easy to see this is not a correct extension of a traditional propositional logic implication, because $0 \to 0$ yields zero. However, this interpretation of the fuzzy implication is more useful for some applications. In fuzzy logic, Modus Ponens is extended to generalized modus ponens in the following manner: given the input "*X* is *A**" and the fuzzy rule "if *X* is *A* then *Y* is *B*" then the consequence is "*Y* is *B**". The membership function of the conclusion, the fuzzy set *B**, is defined in [14, 16] as follows:

$$\mu_{B^*}\left(y\right) = \max_{xeA^*}\left[\mu_{A^*}\left(x\right) \wedge \mu_{A\to B}\left(x,y\right)\right] \quad (2)$$

Generalized modus ponens has been adapted and used widely in control applications; the mechanism is called interpolative reasoning. This mechanism is needed for applications for which the input-output relationship is described by a collection of fuzzy if-then rules. A fuzzy logic system, using the interpolative reasoning, is characterized by the following steps:

3.2.1 Fuzzification

The process of converting a crisp input data $x' = x_0 \in U$, to a fuzzy set *A*, is called fuzzification. It maps the inputs into their membership functions and truth values, these mappings are then fed into the rules. The most widely used fuzzifier is a fuzzy singleton defined by:

$$\mu_{A^*}\left(x\right) = 1 \ \ if \ x = x', \forall x \in U \quad (3)$$

$$\mu_{A^*}\left(x\right) = 0 \ \ if \ x \neq x' \quad (4)$$

The fuzzy input set *A** only contains a crisp element x'. In this case, the formula (2) becomes a fuzzy implication:

$$\mu_{B^*}\left(y\right) = 1 \wedge \mu_{A\to B}\left(x',y\right) = \mu_{A\to B}\left(x',y\right) \quad (5)$$

Let us now consider a rule base (where *X*, *Y* and *Z* are linguistic variables defined on the universe of discourse *U*, *V* and *W* respectively):
R_i : *if* X is A_i *and Y is* B_i *then Z is* C_i where i=1, ..., n and given the input crisp fact (x_0, y_0), the goal is to determine the output "*Z* is *C**".

The second step is to find the output, C^*_i of each of the rules using the inference:

$$\mu_{C^*_i}\left(w\right) = \mu_{\left(Ai \ \ and \ \ B_i\right)} \to C_i\left(x_0, y_0, w\right) \forall w \in W \quad (6)$$

In the Min inferencing, which uses the Mamdani's implication rule, the implication is interpreted as a fuzzy and operator:

$$\begin{aligned}\mu_{c^*i}\left(w\right) &= \mu_{Ai \ \ and \ Bi}\left(x_0, y_0\right) \ \ and \ \mu_{ci}\left(w\right) \\ &= \min\left(\mu_{Ai \ \ and \ Bi} \ \ \left(x_0, y_0\right), \mu_{ci}\left(w\right)\right)\end{aligned} \quad (7)$$

3.2.2. Composition

All fuzzy subsets assigned to each output variable are combined together to form a single fuzzy subset for each output variable. The purpose is to aggregate all the individual rule outputs to obtain the overall system output. In the Max composition, the combined output fuzzy subset *C** is constructed by taking the maximum over all of the fuzzy subsets assigned to the output variable by the inference rule, yields a crisp value which can be used in a regular comparison predicate evaluating, in turn, to true or false.

$$\mu_{c^*}\left(w\right) = \max\begin{pmatrix}\mu_{c^*1}\left(w\right) \\ \mu_{c^*2}\left(w\right), ..., \mu_{c^*n}\left(w\right)\end{pmatrix} \forall w \in W \quad (8)$$

3.2.3. Defuzzification

The result of the fuzzy inference system is a fuzzy set. The defuzzification step produces a representative crisp value as the final output of the system. There are several defuzzification methods. The most commonly used is the Centroid (Center-of-gravity) defuzzifier which provides a crisp value based on the center-of-gravity of the result (the output fuzzy set graph).

4. Architecture of a Fuzzy ECA Rule Processing System

User will give input of fuzzy active rule in the user interface module. Then, fuzzy parser module parses and checks the syntactic validity of a new rule. The JavaCClexer-parser generator is used for the

construction of the fuzzy parser. Fuzzy parser module identifies the linguistic variable using "%" symbol. It identifies the binding element between the event and condition action using the symbol "#". Fuzzy integrity constraints module will provide the interval values for the linguistic variables. All the fuzzy ECA rules are stored in the fuzzy rule base repository. All the valid fuzzy ECA rules are converted to crisp rules. All the crisp rules are stored in the crisp rule base repository module. After storage of crisp rules, even handler module will be invoked. Event handler module will identify the temporal event, composite event or primitive event for specific crisp rule. After identifying the event for the specific rule, Condition evaluator module will execute the condition for the specific crisp rule. After evaluating condition, Action executor module will perform the suitable DB action.

Temporal event module will handle all temporal events using execution schedule module. Temporal event module adds all temporal event rules into the temporal rules XML repository. Moreover, the temporal event module assigns a status of the rules. It will assign three types of status "new", "cancel", and "active". All the event actions are executed one by one by the execution schedule module. This execution schedule module arranges an execution order of scheduling the fuzzy ECA rules. The entire schedule action of ECA rule processing is carried out by execution schedule module. This module runs in a time interval to identify rules which have been newly added to the crisp rules xml repository and performs the needed action. If the status of the temporal event rule is "new", the timer is scheduled based on the rule parameters and its status is made active. If the temporal event rule is with status "cancel", then this module will cancel the temporal event rule from the execution module. The temporal rule with a status "active" runs uninterrupted. If there are many rules for to be executed, the execution module checks the priority for the rules from crisp rules repository. The highest priority for the rule is 1. Depends upon the priority, the execution module will execute all the rules one by one.

Composite event module will identify all the composite event rules and stores all the composite event rules in its repository. Since, composite event can trigger multiple rules composite event processing is more complicated than primitive event processing. composite event rules are not executed immediately like other events. One composite event rule depends on the occurrence of other events. This module uses an event log file to keep track of event occurrence. Based on the event history provided by the event log, the rules of the composite event can be triggered. The output of all the ECA rules will be updated and stored in the XML DB.

5. Algorithm for implementing Fuzzy Logic based Active XML DB

Algorithm 1 for executing fuzzy ECA rule deals with execution steps of fuzzy ECA XQuery statement. First of all, it will check fuzzy ECA XQuery rule or normal XQuery rule. If it is a fuzzy ECA XQuery rule, it will find out the corresponding fuzzy constraints from XML schema file. Then, it will invoke fuzzy parser algorithm for converting linguistic expression to crisp XQuery Boolean expression.

The fuzzy parser Algorithm 2 will take the input of fuzzy linguistic variables. Fuzzy parser will break the input into series of tokens. If '#' symbol is found, it identifies as binding character. It will bind the event and condition using the XML files following '#' character. If '%' symbol is found, it will check whether '~' symbol is available in the linguistic expression. If '~' symbol is found, it will assign triangular distribution for linguistic variable. If '~' symbol is not found, it will assign trapezoidal distribution for linguistic variable. After that this algorithm will calculate Min and Max values for triangular and trapezoidal distribution using the respective formulae. Finally this algorithm converts fuzzy linguistic expressions into XQuery Boolean expressions.

Then, the event handler Algorithm 3 will be invoked to find out whether the event is primitive, temporal or composite. If the event is temporal, the timer event scheduler will execute the timer XML rule in the corresponding time intervals. If the event is composite, the sequence of events to be executed is followed. Finally the output of fuzzy ECA XQuery rule is stored in the XML DB. If it is an ECA XQuery rule, then it will find out the corresponding integrity constraints from XML schema file. Then, it will invoke the event handler algorithm to find out whether the event is primitive, temporal or composite. Finally it will execute the specific condition for the XQuery expression. Finally, the output of ECA XQuery rule is stored in the XML DB.

Algorithm 1: Executing fuzzy ECA rule.

Input: *Fuzzy ECA XQuery.*
Output: *XML Document*.

Begin
Step 1: *Execute XQuery Statement*
Step 2: *If Fuzzy Trigger Exists Then*
Step 2.1.*a*: *Find out XML Schema based Fuzzy Constraints*
Step 2.2.*a*: *Execute Fuzzy Active Rule*
Step 2.3.*a*: *Invoke Fuzzy parser to convert Fuzzy rule to Crisp Rule*
Step 2.4.*a*: *Invoke Event Handler process to find out the type of event*
Step 2.5.*a*: *Execute the condition and perform the action*
Step 2.6.*a*: *Store the result in XML Database*
Step 2.7.*a*: *Generate XQuery Output*
Else
Step 2.1.*b*: *Find out XML Schema based Constraints*
Step 2.2.*b*: *Execute Active Rule*
Step 2.3.*b*: *Invoke Event Handler process to find out the type of event*
Step 2.5.*b*: *Execute the condition and perform the action*
Step 2.6.*b*: *Store the result in XML Database*
Step 2.7.*b*: *Generate XQuery Output*
End If
End

Algorithm 2: Fuzzy parser.

Input: *Fuzzy Linguistic variables*.

Output: *Crisp Condition.*
Begin
Step 1: *Break the Input into series of tokens*
Step 2: *If '#' character is found then*
Step 2.1.*a*: *Bind the event and condition using the XML document following '#' character*
End If
Step 3: *If '%' character is found then*
Step 3:*1a:If '~' Symbol is found then*
Identify Triangular Distribution
else
Identify Trapezoidal Distribution
End If
Step 4: *Identify the linguistic terms on LHS and RHS*
Step 5: *Calculate MIN value and MAX value*
Step 5.1.*a*: *If Distribution is Trapezoidal then*
Calculate Min and Max Value using the following formula
*Min=(β-α)*THOLD+α*
Max=[(δ-γ)(1-THOLD)]+γ*
End If
Step 5.1.*b*: *If Distribution is Triangular then*
Calculate Min and Max Value using the following formula
Min=d-(margin(1-THOLD))*
Max= d+(margin(1-THOLD))*
End If
End

Algorithm 3: Event handler.

Input : Fuzzy ECA XQuery
Output: Invoke the corresponding Routine for Fuzzy ECA Rule
Begin
Step 1. *Check whether the event is Primitive, Composite Or Temporal*
Step 1.1: *If Event Is Primitive Then*
Invoke The Primitive Event Handler Process
End If
Step 1.2: *If Event is Temporal Event Then*
Invoke The Condition In The Corresponding Time Intervals
End If
Step 1.3: *If Event is Composite Event*
Find out the sequence of Events to be executed
End If
End

6. Results and Discussion

Testing is carried out to compare the performance and output of Fuzzy triggers and normal triggers.

6.1. Performance Analysis of Normal Trigger and Fuzzy Trigger

In order to analyse the performance of a fuzzy trigger and a normal trigger, a temporal fuzzy trigger coding and the normal trigger coding is taken. The fuzzy trigger coding to find out the good performing company stocks every day is given below.

```
<rule>
 <rulename>Listing Good Performed Companies in Stock
Market</rulename>
<event_type>temporal_event</event_type> <event_frequency
>daily
</event_frequency>
<start_time>18:00</start_time>  <condaction>
 let $a := collection('StockMarket.dbxml')/StockBroker return
for $b in ($a/Company_Share) return
  if ($b/%(Close_NAV-Yesterday_NAV)/text()=VERY HIGH with
threshold 0.8 and $b/%Stock_Performance/text()=~good with
threshold 0.8) then
 insert nodes
<Top Performance Companies Stocks>
<Company_ID>{$b/StockMarket/Company_ID/text()}</Company
_ID>
<CompanyName>{$b/StockMarket/CompanyName/text()}
</CompanyName>
<CompanyType>{$b/StockMarket/CompanyType/ text()}
</CompanyType>
<NAV>{$b/StockMarket/NAV/ text()}</NAV>
 else()
  </condaction>
</rule>
```

The Normal Trigger coding to find out good performing company stocks every day is given below.

```
<rule>
 <rulename>Listing Good Performed Companies in Stock
Market</rulename>
<event_type>temporal_event</event_type> <event_frequency
>daily</event_frequency>
<start_time>18:00</start_time>  <condaction>
 let $a := collection('StockMarket.dbxml')/StockBroker return
 for $b in ($a/Company_Share) return
    if ($b/(Close_NAV-Yesterday_NAV)/text() >=20 and
$b/(Close_NAV- Yesterday_NAV)/text() <=40) and
($b/Stock_Performance_min/text()>=72 and
$b/Stock_Performance_min/text() <=86 ) then
  insert nodes
<Top Performance Companies Stocks>
<Company_ID>{$b/StockMarket/Company_ID/text()}</Company
_ID>
<CompanyName>{$b/StockMarket/CompanyName/text()}
</CompanyName>
<CompanyType>{$b/StockMarket/CompanyType/ text()}
</CompanyType>
<NAV>{$b/StockMarket/NAV/ text()}</NAV>
 else()
  </condaction>
 </rule>
```

If the closing NAV of the company share is very high and STOCK PERFORMANCE of the company share is good and then insert the company details into the top performance company stocks XML file. The NAV values of 500 Companies are updated for 10 days. The output for the Normal Trigger and Fuzzy Trigger is graphically represented in the Figure 2.

The tests were run in a 2.20 GHz I3 Processor PC with 2 GB RAM memory. The operating system was Windows 7 Ultimate. In the first test, the output produced by the Fuzzy Trigger and Normal Trigger are evaluated continuously for 10 days. The number of relevant records returned by both the Fuzzy Trigger and Normal Trigger are considered for output performance evaluation. The result for this testing is graphically represented in the Figure 1. It is clearly understood from the Figure 1 that Fuzzy triggers are returning more XML records than Normal triggers. We can conclude that the Fuzzy triggers are producing 20% to 30% better output than Normal Triggers. In the second test, the execution time for Fuzzy Triggers and Normal Triggers are evaluated. In the first time, the execution time of Normal Trigger with one rule and

Fuzzy Trigger with one rule is compared. In the second time, the execution time of Normal Trigger with two rules and fuzzy trigger with two rules is compared. This test is carried out until the normal trigger with 10 rules and fuzzy trigger with 10 rules. Java program is used to calculate the execution time in Milliseconds of both the triggers. The execution time comparison is graphically represented in the Figure 2. Normal triggers are 10% to 20% performing faster than fuzzy triggers. It is clearly understood that the average overhead involved in fuzzy trigger translation is only 10-20% when compared to normal triggers.

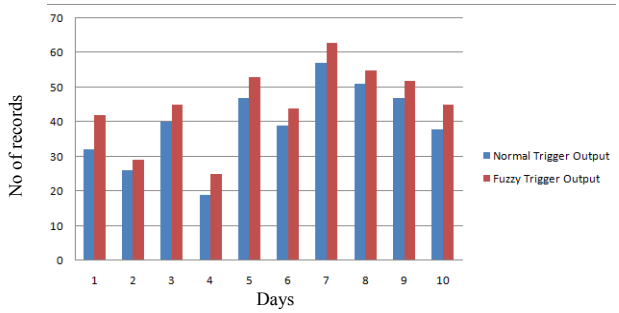

Figure 1. The output performance of fuzzy trigger and normal trigger.

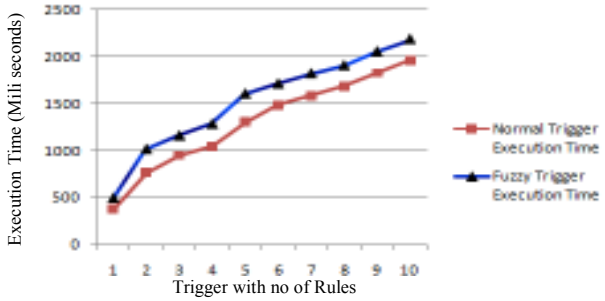

Figure 2. Execution time analysis of fuzzy trigger and normal trigger.

7. Conclusions

Fuzzy trigger for XML DB which focuses on combining two important areas: Fuzzy reasoning and active rules is proposed. In this paper, we extend the basic semantics of event-condition action rules with fuzzy rules and fuzzy inference. An algorithm to implement fuzzy active rule based triggers is presented in this paper. The architecture for a fuzzy ECA rule processing is proposed in this paper. The proposed fuzzy trigger system was implemented using Java and Oracle Berkeley DB XML. Testing is carried out to analyze the performance and output of normal XML triggers and fuzzy XML trigger. Our test results show that fuzzy XML triggers are providing better output than normal XML triggers. But the performance wise, fuzzy triggers are 10% to 25% slower than normal triggers. The proposed system lacks intelligent error handling functionality and GUI based user friendly interface. We will do research to introduce the intelligent error handling mechanism in fuzzy trigger system.

References

[1] Anders H., Wenny R., and Eric P., "XTrigger: XML Database Trigger," *Springer Journal: Computer Science-Research and Development*, vol. 25, no. 1, pp. 1-19, 2010.

[2] Anton W. and Tarik B., "Fuzzy Triggers: Incorporating Imprecise Reasoning into Active Databases," *in Proceedings of the 14th International Conference on Data Engineering*, Orlando, Florida, pp. 108-115, 1998.

[3] Bailey J., Poulovassilis A., and Peter W., "An Event-Condition-Action Language for XML," avalible at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.127.301&rep=rep1&type=pdf, last visited 2002.

[4] Bernauer M., Bonifati A., Braga D., Campi A., and Ceri S., "Active XQuery," *in Proceedings of the 18th IEEE International Conference on Data Engineering*, 2002.

[5] Bonifati A., Braga D., Campi A., and Ceri S., "Active XQuery," *in Proceedings of the 18th International Conference on Data Engineering*, San Jose, California, pp. 403-412, 2002.

[6] Bonifati A., Ceri S., and Paraboschi S., "Pushing Reactive Services to XML Repositories using Active Rules," *Computer Networks*, vol. 39, no. 5, pp. 645-660, 2002.

[7] Bosc P. and Pivert O., "SQLf: A Relational Database Language for Fuzzy Querying," *Fuzzy Systems*, vol. 3, no. 1, pp 1-17, 1995.

[8] Bouaziz T., Karvonen J., Pesonen A., and Wolski A., "Design and Implementation of TEMPO Fuzzy Triggers," *in Proceedings of 8th International Conference on Database and Expert Systems Applications*, Toulouse, France, pp. 91-100, 1997.

[9] Farnaz M., Ayaz I., and Leili M., "Using an Active Fuzzy ECA Rule-Based Negotiation Agent IN E-Commerce," *International Journal of Electronic Commerce Studies*, vol. 2, no. 2, pp. 127-148, 2011

[10] Ferraz C., Braganholo V., and Mattoso M., "ARAXA: Storing and Managing Active XML Documents," *Web Semantics*, vol. 8, no. 2-3, pp. 209-224, 2010.

[11] Galindo J., Medina M., Pons O., and Cubero C., "A Server for Fuzzy SQL Queries," *in Proceedings of the 3rd International Conference on Flexible Query Answering Systems*, Roskilde, Denmark, pp. 164-174, 1998.

[12] Jin Y. and Bhavsar T., "A Fuzzy Trigger Language for Relational Database Systems," *in Proceedings of the 20th International Conference on Software Engineering and Knowledge Engineering*, USA, pp 367-370, 2008.

[13] Jin Y. and Bhavsar T., "Incorporating Fuzziness into Timer Triggers for Temporal Event

Handling," *in Proceedings of the IEEE International Conference on Information Reuse and Integration*, USA, pp. 325-329, 2008.
[14] Kappel G. and Kramler G., "Composite Events for XML," *in Proceedings of the 13th World Wide Web Conference,* New York, USA, 2004.
[15] Liu T. and Goh A., "XML Schema-Based Triggers**,**" *in Proceedings of International Conference on World Wide Web*, pp. 937-940, 2003
[16] Montesi M. and Torlone R., "Analysis and Optimization of Active Databases," *Data and Knowledge Engineering*, vol. 40, no. 3, PP 1-39, 2002.
[17] Ozgun O., Ozyer T., Zarour O., Alhajj R., and Polat F., "TempoXML: Nested Bitemporal Relationship Modeling and Conversion Tool for Fuzzy XML," *Information Sciences: An International Journal*, vol. 193, pp. 247-274, 2012.
[18] Prabha S., Kannan A., and Kumar P., "An Optimizing Query Processor with an Efficient Caching Mechanism for Distributed Databases," *the International Arab Journal of Information Technology*, vol. 3, no. 3, pp. 231-236, 2006.
[19] Rodrigues R., Cruz A., Cavalcante R., "Alianca: A Proposal for Fuzzy Database Architecture Incorporating XML," *Fuzzy Sets and Systems*, vol. 160, no. 2, pp 269-279, 2009.
[20] Swamynathan S., Kannan A., and Geetha V., "Composite Event Monitoring in XML Repositories using Generic Rule Framework for Providing Reactive E-Services," *Decision Support Systems*, vol. 42, no. 1, pp. 79-88, 2006.
[21] Thomson J. and Radhamani G., "Fuzzy Integrity Constraints for Native XML Database," *International Journal of Computer Science Issues*, vol. 9, no. 3, pp. 466-471, 2012.
[22] Xu G., Ma J., and Huang T., "A XML-Based Composite Event Approach," *in Proceedings of the 4th International Conference*, Beijing, China, pp. 436-442, 2005.
[23] Zadeh A., "Knowledge Representation in Fuzzy Logic," *IEEE Transactioons on Knowledge and Data Engineering*, vol 1, no. 1, pp. 89-100, 1989.
[24] Zadeh A., "The Role of Fuzzy Logic in the Management of Uncertainty in Expert Systems," *ACM Journal of Fuzzy Sets and Systems*, vol. 11, no. 1-3, pp 197-198,1983.

Thomson Fredrick received his Bs and MS degrees from Bharathidasan University, India. Currently, he is a research scholor at R&D centre of Bharathiar University, India. He has published several papers in International Journal and Conferences. His research interests include: XML databases, web technology and artifical intelligence.

Govindaraju Radhamani is working as a Director in the School of IT and Science in Dr.G.R.Damodaran College of Science, Coimbatore. She received her PhD degree from Multimedia University, Malaysia and MSc, MPhil degrees from PSG College of Technology, India. She has published several papers in International Journal and Conferences. She is a Senior Member of IEEE and CSI. Her research interests are: Computer security, databases and mobile computing.