International journal of Computer Science & Network Solutions December.2013-Volume 1.No4
http://www.ijcsns.com ISSN 2345-3397

Presented a New Structure with Purpose Improvement Fuzzy Queries in Object Oriented Database

A. Fereshte Farazi, B. Ali Harounabadi, C. MashallahAbbasi Dezfouli

[1]*Department of computer engineering, Science and Research Branch, Islamic Azad University, khouzestan-Iran*
FERESHTE.FARAZI@GMAIL.COM
[2]*Department of computer engineering, Islamic Azad University, Central Tehran Branch, Iran*
[3]*Department of computer engineering, Science and Research Branch, Islamic Azad University, khouzestan-Iran*

Abstract

As more and more information is stored in XML, exchanged in XML, or presented as XML through various interfaces, the ability to intelligently query our XML data sources becomes increasingly important. The main focus of this paper is on fuzzy XML queries and tries to analyze a complicated and complex query to get more meaningful and closer responses. The method permits the user to provide the possibility of allocating the weight to various parts of the query, which makes it easier to follow better goals and return the target objects.

Keywords: Query in XML, Weighted query, Fuzzy query, Fuzzy object oriented data base.

I. Introduction

The Entrance of object orienting concept in databases caused the relation database gradually to replace with object oriented database in various fields. Since object oriented database has not been able to support existing non-definite data for the real-world applications, fuzzy concepts have recently been also added to these databases to let us have a kind of contact with a set of fuzzy objects through object oriented fuzzy databases (Ma et al., 2004).
Ma (Ma, 2005; Ma, 2006) defined the classes in a fuzzy way and created the fuzzy class to save the non-definite data in the object oriented databases. On the other side, Harounabadi and Teshnelab (Harounabadi, &Teshnelab, 2008; Harounabadi&Teshnelab, 2009) laid out fuzzy-UML to model the existing non-definite (uncertainly) in the information systems. They also designed various relations graphs among the classes for fuzzy object oriented database, and Porbehzadi and Harounabadi (Porbehzadi&Harounabadi), present a new SQL gerammer
but none of them mentioned anything about XML grammar.
In this article, we propose a fuzzy grammar in XML to increase the accuracy in fuzzy queries.
Through applying this approach, a complicated fuzzy query can be handled easily and objects near to the user's request will be returned, more easily. Meanwhile, we can observe that through

allocating weight to the each part of the query, we can allow the user to define the importance and priority of each characteristic for the query.
The organization of this paper first explains the required primary concepts such as object oriented databases, fuzzy concepts and fuzzy object oriented database. We then review the carried out works in this regard. Next, we present the proposed method and we will analyze and apply this approach in the following section. Finally, the technique is applied to some test data and concluding remarks are given in the last to summarize the contribution of the paper.

II. Introducing the required primary concepts

A.Object oriented database

During the past few years, there have tremendous efforts on implementing fuzzy concept on object oriented programming. Afshani et al. (2012) presented a new template based on fuzzy-UML concept for some of C4ISR products such as Logical Data Model (OV-7), Operational Event/Trace Description (OV-6c) and Systems Event/Trace Description (SV-10c). To explain further, fictional Fast Pass system used at OilCo gas stations was used to demonstrate details of the proposed model. Designing a database in an object oriented technique, which helps presenting the mentioned base in the presented mechanisms frame by this model such as class, inheritance, grouping, etc. The data in this database becomes the objects in this model which will be connected to each other in various ways and various operations will be possible such as data entering, deleting, adding, updating and exploiting data through exchanging messages between the objects (Unamo et al., 1998; Ma, 2005).

B. Fuzzy Sets

In the real world, we normally understand and handle most of the concepts in a fuzzy way representing inaccurate, unobvious and ambiguous terms. Despite the fact that terms such as hot, cold, long, short, old, young and etc., do not represent any special number, human mind understands everything quickly and in a flexible way and manipulates them to make decisions and conclusions. Yet, the machines only understand the numbers, which are precise and accurate. There are many attempts to acquire the secret of these capabilities and implement them in computers as much as possible (Zadeh, 1975). Due to the fact that our mind acts in another way, in the beginning, we need to create new logics and multi-value innovations and fuzzy logic is one of them. Entering fuzzy logic in databases, especially object oriented database, opened a new horizon to database designers (Ma et al., 2004).

C. Fuzzy Object Oriented Database

These kinds of databases are created through allocating fuzzy concepts to object oriented databases. This allocation can happen in 3 levels of attribute, entity and class (Ma, 2005).
First level: fuzzy attribute, an attribute, which can get its amounts from a fuzzy set, such as age which can acquire being old, young, middle-age or height which can be tall, short, medium and etc.

Second level: in this level, classes can be fuzzy and a class can be recognized with a membership degree to a range [0,1] as a subclass of a super class. Third level: objects also can be assigned to one or some classes in a range of [0,1]. In order to define a fuzzy class, it is required to add some of the definitions to the existing definitions in the classic object oriented database and these concepts and class official definition are presented according to the defined concepts in (Ma, 2005). Indefinite systems are modeled through fuzzy-UML in (Haroonabadi, &Teshnelab, 2008-9) and UML concepts are mentioned. In a general compare, the relationship among a relational database, object oriented database and fuzzy object oriented database can be found in Fig. 1.

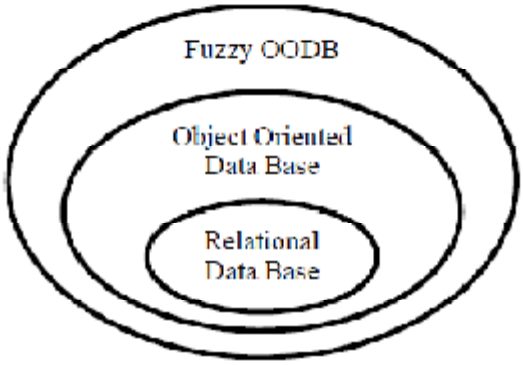

Fig. 1. Relation between Relational ,OO and FOODB

D. XML

XML *(extensible Markup Language)* is a markup metalanguage designed to enable semantics-aware tagging of World Wide Web information [32]. Generally speaking, an XML document is composed of a sequence of nested elements, each delimited by a pair of start and end tags (e. g., <tag> and </tag>).Using XML is possible to define a fuzzy query language with the required characteristics. The use of XML provides benefits such as data portability, allowing data interchange between different platforms and heterogeneous environments, data independence from representation, interoperability, scalability, flexibility and extensibility.

III. Related work in this field

The query process refers to a procedure in which the conditioned objects are met and selected. They will be handed over to the user according to the required frame. The classic databases lacked flexibility in queries. The query condition and database content were crisp. Yet, in the fuzzy object oriented database, both query conditions and database contents can be fuzzy, so we will have flexible queries. Besides, the process of the query in these kinds of databases refers to the procedure in which some objects of the classes are selected in order to achieve both the mentioned threshold and the mentioned condition at the condition threshold. Therefore a syntax principle of an XML query according to the fuzzy object oriented database model is in (2) (Carlos, 2005).

```
<xfsql> <query>
      <select>
            <columns>
                  <attribute name= "AName"/>
                  ...
            </columns >
      </select>
      <from>
            <table name= 'Tname"/>
                  ...
      </from>
      <where>
            <condition>
            <fuzzy-equal>
            <left> <attribute name= 'AName"/> </left>
            <right>
                  <trapezoid a="1" b=" 2" c=" 3" d=" 4">
            </right>
            </fuzzy-equal>
            ...
            </condition>
      </where>
</query> </xfsql>
```

(1)

These people have reviewed the fuzzy object oriented database but none of them has suggested the operation as a grammar to carry out the fuzzy queries on the database. The queries that our approach prepares will increase the user's accuracy to find the required output through allocating a weight to each attribute of a query.

IV. The proposed model

It was not possible for the user to prioritize the attributes in the queries in the Carlos's approach and only the level of membership of each object in the related class (μ) and the fuzzy degree of those attributes could be reviewed. The proposed model of this paper allocates weight (w) to the attribute and we leave prioritizing of each attribute in the query to the user. Meanwhile we consider a status that the query conditions to be some complicated conditions, which means our suggested syntax rule of the query is as follows:

```
<xfsql> <query>
    <select>
        <columns>
            <attribute name= "AName"/>
            ...
        </columns >
    </select>
    <from>
        <class name= "Cname" threshold="value"/>
        ...
    </from>
    <where>
    <condition>
        <fuzzy-equal>
        <left> <attribute name= "AName"/> </left>
        <right><linguistic label="Alabel"/></right>
        <left> <attribute threshold ="Thold"/> </left>
        <right> <numeric value ="t"/> </right>
        <left> <attribute outclass ="Weight"/> </left>
        <right> <numeric value ="w"/> </right>
        </fuzzy-equal>
        ...
    </condition>
    </where>
</query> </xfsql>
```

(2)

If we take a precise look at the above relation, we can see that the user should define three cases for each query condition:

1- The amount of an attribute (which can be fuzzy such as age attribute and amount ”teenager”).

2- Threshold of attribute membership degree (for example, with a threshold of 0.6 it can belong to teenager class) and

3- Attribute weight (this weight represents the level of importance or priority of each attribute for the user).

The user for each query conditions follows the objects where the attribute amount matches the required threshold (after user defined the three above mentioned cases). As soon as the user

carried out the query, the amount of final threshold will be calculated by the achieved amount out of the query as follows:

$$Threshold_{final} = \frac{\tau_1 \times w_1 + \tau_2 \times w_2 + \ldots + \tau_n \times w_n}{w_1 + w_2 + \ldots + w_n} \quad (3)$$

In the above mentioned relation, τ_i is the acceptable threshold for the attribute i and w_i is the required weight for the attribute i in the query. Now, we can calculate the required μ for each object separately and use the functions where we wrote in the database and class previously, then, multiply it in the related w of the same attribute mentioned in the query. Therefore, we will calculate the final μ through the Eq. (4).

$$\mu_{final}(obj.a) = \frac{\mu_i \times w_1 + \mu_j \times w_2 + \ldots + \mu_k \times w_n}{w_1 + w_2 + \ldots + w_n} \quad (4)$$

In Eq. (4), $\mu_i$is the degree of belonging object a to an attribute 1, μ_j is the degree of belonging object a to attribute 2 and $\mu_k$is the degree of belonging object a to attribute n and μ_{final}(obj. a) is the final belonging degree of object a to the set of answers. In this stage, for any object we have a degree of belonging and we should compare this degree of final belonging to the query required threshold and if the degree of final belonging is higher than the query threshold or equal to it, we can bring this object in the answer set. It means only the objects will be appeared in the answer set where they achieve the mentioned condition in Eq. (5) as follows,

$$\mu_{final} \geq Threshold_{final} \quad (5)$$

A. The purpose grammer for referencequery

The purpose grammer for referencequery as follows (6)

```
<xfsql> <query>
      <select>
            <columns>
                  <attribute name= "AName"/>
                        …
            </ columns >
      </select>
      <from>
            <class name= "Cname" threshold="value"/>
                  …
      </from>
      <where>
      <condition>
      <and>
      <fuzzy-equal>
            <left> <attribute name= "AName"/> </left>
            <right><linguistic label="Llabel"/></right>
            <left> <attribute threshold ="Thold"/> </left>
            <right> <numeric value ="τ"> </right>
            <left> <attribute  outclass ="Weight"/> </left>
            <right> <numeric value ="w"/> </right>
      </fuzzy-equal>
      …
       </and>
      </condition>
      </where>
</query> </xfsql>
```

(6)

B. The purpose grammer for seasonal querty

The purpose grammer for seasonal query as follows (7)

```
<xfsql> <query>
      <select>
            <columns>
                  <attribute name= "AName"/>
                        …
            </columns >
      </select>
      <from>
            <class name= "Cname" threshold="value"/>
                  …
      </from>
      <where>
      <condition>
       <or>
      <fuzzy-equal>
            <left> <attribute name= "AName"/> </left>
            <right><linguistic label="Alabel"/></right>
            <left> <attribute threshold ="Thold"/> </left>
            <right> <numeric value ="T"/> </right>
            <left> <attribute  outclass ="Weight"/> </left>
            <right> <numeric value ="w"/> </right>
      </fuzzy-equal>
      …
       </or>
      </condition>
      </where>
</query> </xfsql>
```

(7)

C. The purpose grammer for insert attribute and values

The purpose grammer for insert attribute and values as follows (8)

```
<xfsql> <insert>
      <insert_into>
            <tables>
                  <attribute name= "AName"/>
                        …
            </tables >
                  …
      </insert_into >
      <trapezoid a="α1" b=" α2" c=" α3" d=" α4"/>
                  …
</insert></xfsql>
```

(8)

D. The purpose grammer for update attribute and values

The purpose grammer for update attribute and values as follows (9)

```
<xfsql> <revision>
      <update>
            <tables>
                  <set >
                  <left> <attribute name= "AName"/> </left>
                  <right> <trapezoid a="α1" b=" α2" c=" α3" d=" α4"/></right>
                  …
                  </set >
            </tables >
                …
      </update >
</revision></xfsql>
```

(9)

E. The purpose grammer for insert a column

The purpose grammer for insert a column as follows (10)

```
<xfsql> <new_column>
       <alter_table>
             <table>
                   <add >
                   <column name= "AName"/>
                   <characteristic type="Atype"/>
                   <trapezoid a="α1" b=" α2" c=" α3" d=" α4"/>
                   </add >
             </table >
       </alter_table >
</new_column></xfsql>
```

(10)

F. The purpose grammer for update column

The purpose grammer for update a column as follows (11)

```
<xfsql> <revision _column>
      <alter_table>
            <table>
                  <modify >
                  <column name= "AName"/>
                  <characteristic type="Atype"/>
                  <trapezoid a="α1" b=" α2" c=" α3" d=" α4"/>
                  </modify >
            </table >
      </alter_table >
</revision _column></xfsql>
```

(11)

G. The purpose grammer for regimentation

The purpose grammer for regimentation as follows (12)

```
<xfsql> <group >
      <select>
            <columns>
                  <attribute name= "AName"/>
                        …
            </columns >
      </select>
      <from>
            <class name= "Cname" threshold="value"/>
                  …
      </from>
      <group_by>
            <fuzzy-atribute>
            <left> <attribute name= "AName"/> </left>
            <right><linguistic label="Alabel"/></right>
            <left> <attribute threshold ="Thold"/> </left>
            <right> <numeric value ="r"/> </right>
            <left> <attribute  outclass ="Weight"/> </left>
            <right> <numeric value ="w"/> </right>
            </fuzzy-atribute>
            …
      </group_by>
</group></xfsql>
```

(12)

V. Implementation

In this section, we explain the class's structure and finally discuss the query orders and the way to carry them out. We explain our method through an example. Consider a case where we wish to create a fuzzy class called "Persons" which keeps the specifications illustrated in Fig. 2.

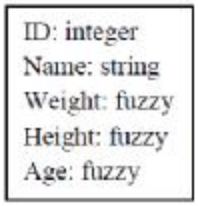

Figure. 2. Persons Class that is Fuzzy

Now, we create the illustrated instances out of Persons class

.

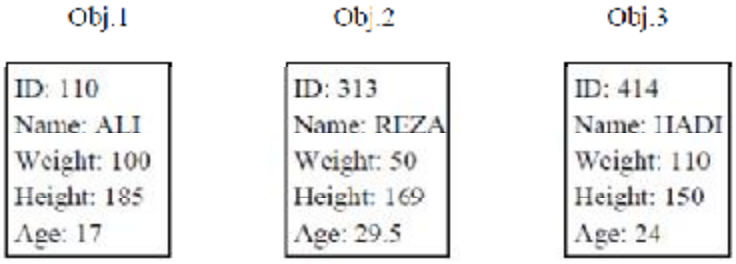

Figure. 3. Instances of Persons class

Suppose that the user asks the names of all the young people through a query, the user deals with the attribute (age) here. Due to the fact that this attribute accepts also fuzzy amounts, then it would be possible to use linguistic variables such as young, middle-aged, old, etc. instead of defining the accurate age number (Zadeh, 1975). In order to response to these queries, we should define functions called "membership functions". These functions can be designed as pre-assumptions by the designer of the database or let the user define them him/herself. The membership functions are created for the attributes amounts according to Eq. (13). For the trapezoidal number of T(a,b,α,β) which can present our required attributes, if the designer or the user of the database enters a, b, α and β, his/her membership function is created according to the relation 13.

$$\mu(x) = \begin{cases} \dfrac{x-(a-\alpha)}{\alpha} & a-\alpha \le x < a \\ 1 & a \le x < b \\ \dfrac{b+\beta-x}{\beta} & b \le x < b+\beta \\ 0 & o.w \end{cases} \qquad (13)$$

For instance, if we consider people aged between 18 to 28 absolutely young, and 16 to 18 and 28 to 30 rather young, then, we can present it through fuzzy trapezoidal numbers (18,28,2,2) as the young. So, we can define the concepts of middle-aged and old through the following fuzzy numbers of Mid(60,90,10,5) and Old(55,95,5,5). The membership function of each set accepts the person's age as an input and calculates the level of belonging of that person to the set. So, we can consider Fig. 4 for the age attribute.

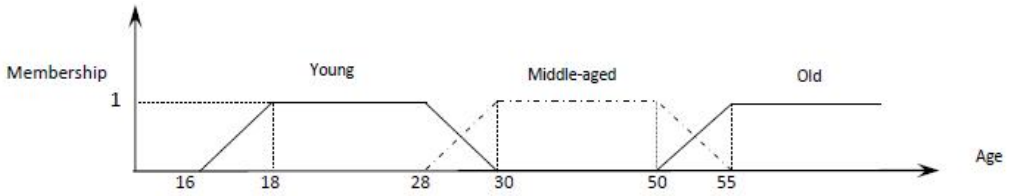

Figure. 4: Fuzzy value for Age Attribute

We can consider the above technique for weight, light, average and heavy weight as it is mentioned in Fig. 5.

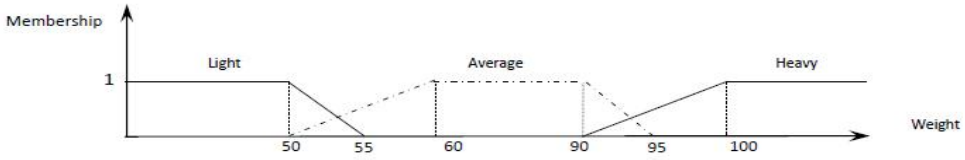

Figure. 5: Fuzzy Value for Weight Attribute

Finally, we consider the height as short, medium and tall as it is mentioned in Fig. 6.

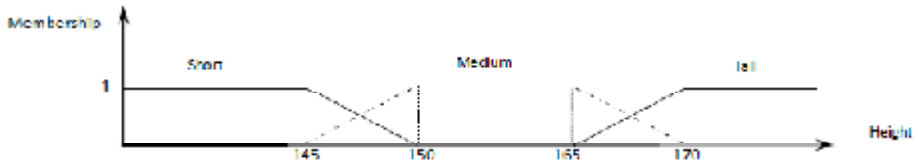

Figure. 6. Fuzzy Values for Height Attribute

Example 1: list the name of all the tall people (τ=0.7 and w=0.8) and heavy (τ=0.5 and w=0.2) and also middle-aged(τ=0.3 and w=0.1). According to Eq. (3), we should carry out the following query,

```
<xfsql> <query>
      <select>
            <columns>
                  <attribute name= "name"/>
            </columns >
      </select>
      <from>
            <table name= "persons"/>
      </from>
      <where>
<condition>
   <and>
      <fuzzy-equal_1>
            <left> <attribute name= "Height"/> </left>
            <right><linguistic label="tall"/></right>
            <left> <attribute threshold ="τ"/> </left>
            <right> <numeric value="0.7"/> </right>
            <left> <attribute  outclass ="W"/> </left>
            <right> <numeric value ="0.8"/> </right>
      </fuzzy-equal_1>
<fuzzy-equal_2>
            <left> <attribute name= "Weight"/> </left>
            <right><linguistic label="Heavy"/></right>
            <left> <attribute threshold ="τ"/> </left>
            <right> <numeric value=0.5 /> </right>
            <left> <attribute  outclass ="W"/> </left>
            <right> <numeric value ="0.2"/> </right>
      </fuzzy-equal_2>
<fuzzy-equal_3>
            <left> <attribute name= "Age"/> </left>
            <right><linguistic label="mid"/></right>
            <left> <attribute threshold ="τ"/> </left>
            <right> <numeric value=0.3 /> </right>
            <left> <attribute  outclass ="W"/> </left>
            <right> <numeric value ="0.1"/> </right>
      </fuzzy-equal_3>
```

(14)

Now, we should apply the following functions: μ-Age-Mid, μ-Weight-Heavy and μ-Height-Tall according to Eq. (13) for recalling all the objects and after achieving all the amounts of μ. Using Eq. (4) yields final μ for all the objects. Therefore, we have,

μ-Age-mid(obj 1) = 0 ,
μ-Weight-heavy(obj 1) = 1 ,

μ-Height-tall(obj 1) = 1 ,

$$\mu\text{-final(obj. 1)} - \frac{0 \times 0.1 + 1 \times 0.2 + 1 \times 0.8}{1.1} - 0.9 \quad (15)$$

We will calculate the amounts of the final membership degrees for obj2 and obj3, respectively. Eq. (16) shows the results.

μ-final (obj2) = 0.65 , μ-final (obj3) = 0.18 (16)

Now, we calculate the final-threshold according to the Eq. (4) to compare the object membership degree.

$$\tau\_final - \frac{0.7 \times 0.8 + 0.5 \times 0.2 + 0.3 \times 0.1}{0.8 + 0.2 + 0.1} = 0.627 \quad (17)$$

It would be obvious that obj.1 and obj.3 will be appeared as the output, because their membership degree is higher than the final threshold. Yet obj.2 cannot be considered at the out let response set. If we take a precise look at the example 1, we can find out about the importance of the attribute weight of the query. For instance, in this query, the obj.1 with zero membership degree belongs to the middle-aged people. Meanwhile, we know that the degree of 1 belongs to the tall people set. Due to the fact that in this query, we considered 0.1 weight for the middle-aged people and for the tall people 0.8 weight, so the importance of being tall is much more than the age and obj 1 will be presented as the output.

VI. Data samples

We can test the query through the test amounts in Fig. 7.

	ID	PNAME	AGE	HEIGHT	WEIGHT
▶ 1	1	hasan	25	189	56
2	2	Ali	45	155	67
3	3	karim	34	168	87
4	4	nedia	17	154	98
5	5	fateme	43	180	59
6	6	zehra	54	172	67
7	7	sina	16	167	72
8	8	ekram	23	162	52
9	9	kazem	18	174	89
10	10	Mahdi	17	185	100
11	11	nazenin	29.5	169	50
12	12	ahmad	24	150	110

Figure. 7.tblPersonstable that contains sample data

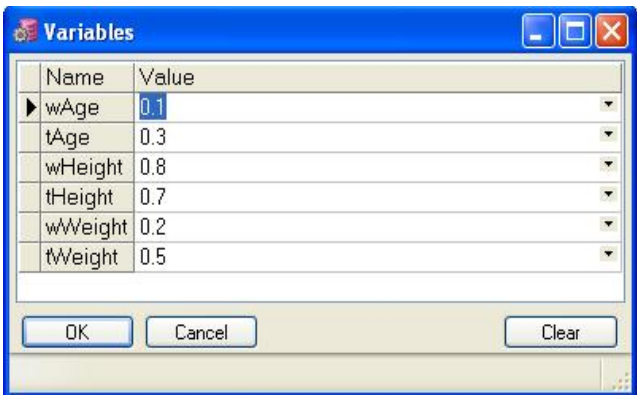

Figure. 8. Entering Attribute's Weights and Thresholds

SQL code is regarding the query example1 is presented in Eq. (18).

```
Select                                                    (18)
t.*,
t.muAge('mid')asMu_Age,
t.muHeight('tall')asMu_Height,
t.muWeight('heavy')asMu_Weight,
CalcTreshold(&wAge,&tAge,&wHeight,
&tHeight,&wWeight,&tWeight)
asTre
From
TBLPersons t
Where
((((t.muAge('mid')*&wAge)+
(t.muHeight('tall')*&wHeight)+
(t.muWeight('heavy')*&wWeight) ) / (&wAge+&wHeight+&wWeight) )(7)
1140
>= calctre (&wAge,&tAge,&wHeight,
&tHeight,&wWeight,&tWeight )
)
```

W amount (attribute weight) and t (attribute threshold) of each attribute would be entered by the user according to Fig. 8. The result of the relation 10 is illustrated in Fig. 9.

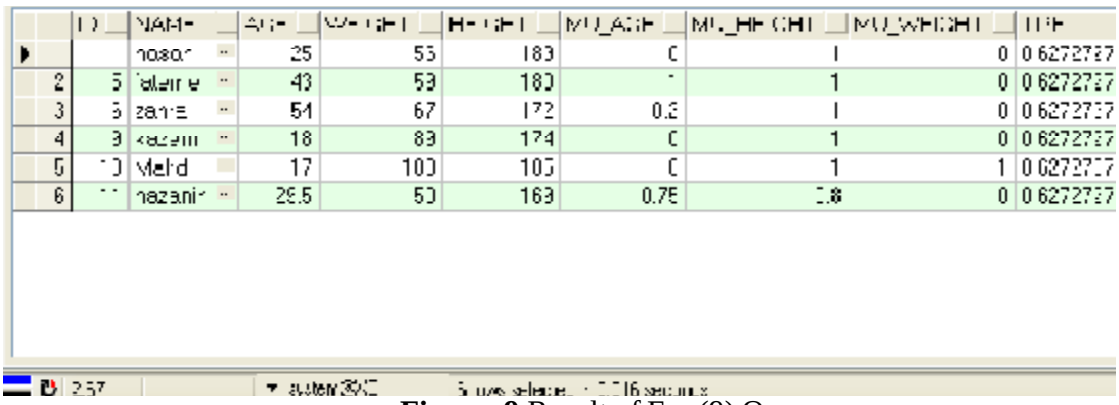

Figure. 9. Result of Eq. (9) Query

VII. Conclusion

In this paper, we have presented a new fuzzy grammar for fuzzy queries on the object oriented database. The proposed model of this paper has been able to make the necessary influence and the priority of each attribute in query result through allocating weight to the available attributes in the query. As we have observed, the user could find all the required objects through applying the linguistic variables without dealing with the numbers through the proposed model. We have used object and class concepts to define some functions in the class without any need to receive any digital parameters. We could establish the access to the required variables of the class and carried out the operation. Our suggested approach presents a better workability in huge and heavy databases and this grammar facilitates fuzzy queries.

References

i. Afshani, J., Harounabadi, A., AbbasiDezfouli, M. (2012). A new model for designing uncertain enterprise architecture. Management Science Letters, 2(2), P.P: 689-696.
ii. Buche,P., Dibie-Barth´elemy.J., Hignette,G. (2006). Fuzzy semantic taggin and flexible querying of XML documents extracted from the Web.JournalIntellInf System.
iii. Carlos D, B., Jesus R, C., Juan M, M. (2005).Towards a XML Fuzzy Structured Query Language, Eusflat– LFA.
iv. Damiani, E., Tanca, L., Fontana, F.A. (2000). Fuzzy XML queries via context-based choice of aggregations. Kybernetika_Volume 36,Number 6, Pages 635-655.
v. Galindo, J., Urrutia, A., Mario, P. (2006). Fuzzy Databasses: Modeling, Design and Implementation. Idea Group Publishing .Edition 2006.
vi. Ma, Z. (2005). Fuzzy Database Modeling With the UML. Idea Group Publishing, P.P:153-176. Edition 2005.
vii. Ma, Z.M., Yan, l. (2007). Fuzzy XML data modeling with the UML and relational data models. Data & Knowledge Engineering. Vol No:63 P.P: 972–996.
viii. Haroonabadi, A., &Teshnelab, M. (2008). A Novel Method for Behavior Modeling in Uncertain Information Systems. World Academy of Science, Engineering and Technolog.Vol No: 37

ix. Pourbehzadia,M., Haroonabadi,A., &Sadeghzadeh, M. (2012). A new weighted fuzzy grammar on object oriented database queries. Management Science Letters. Vol No: 2. P.P:1133–1140.
x. World Wide Web Council. Extensible Markup Language, Version 1.0. W3C Recommendation, www.w3.org/TR/1998/REC-xml-19980210.
xi. Zadeh, L. A. (1975). The Concept of a Linguistic Variable and Its Application to Approximate Reasoning. Information Sciences, 8, 199-248.