Handling of Fuzzy Queries using Relational DBMS

Nishant Agrawal
B. Tech(IT)
4th year

Anubhav Manan
B. Tech(IT)
4th year

Akash Aggarwal
B. Tech(IT)
4th year

Rashmi Sharma
Assoc.Professor

(ABES Engineering College, Ghaziabad)

ABSTRACT

Handling crisp and precise data in SQL is an easy process but classical data models often suffer from their incapability of representing and manipulating imprecise and uncertain information which is found in many real world applications. Since the early 1980's, Zadeh'sfuzzy logic has been used to improve and modify various data models. This introduction of fuzzy logic in databases enhances the capability of classical models so that uncertain and imprecise information could easily be represented and manipulated.This paper proposes an algorithm with the help of which crisp values are converted into fuzzy values by calculating their membership value at the database level. The paper then uses a GUI through which the result of fuzzy queries can be obtained from the database. With the help of proposed algorithm, the calculated membership value will be stored in the database for differentpredefined categories (e.g.-child, young, middle age and old in case of ages). These membership values helps in fetching the result of fuzzy queries from the database with the help of developed GUI (the database used here is oracle 10g but other databases can also be used).The fuzzy queries have a wider retrieved space and can be used to identify the characteristic of an individual (marks in this case).

1. INTRODUCTION

Complexity generally occurs due to uncertainty in the form of ambiguity. Computer System can address only simple or direct problems however humans have the capability of reasoning approximately.

In traditional database management systems, queries are intended to retrieve data which satisfies some specific crisp criteria's. This specific crisp criterion lacks flexibility which usually results in no result retrieval. Hence an extended version of these systems is required so that they could use and support imprecise querying capabilities.

In general SQL query systems, a twofold hypothesis have been maintained: data is assumed to be precisely known and queries are intended to retrieve elements that qualify for a given Boolean condition.

This paper concentrates on the second aspect of this hypothesis. In context to regular relational databases (where data is precisely known), the objective is to provide users with new querying capabilities based on conditions which involve preferences and describe more or less acceptable items, thus defining flexible queries. Since the problem is no longer to decide whether an element satisfies (or not) a particular condition but rather the extent to which it satisfies the condition. One of the advantages lies in the "natural" ordering of the answers (discrimination) which allows for calibration if desired.

In conventional DBMS systems, the query evaluation problem does not follow optimal pattern evaluation process. For fuzzy queries the process becomes more complex due to two reasons:

i) The available access paths cannot be used directly, and

ii) A larger number of tuples are selected by fuzzy conditions as compared to Boolean ones.

When humans interact with database they require vagueness or imprecision in the results. For removing this vagueness the proposed algorithm could be used to calculate the membership of the value to be stored in the database for different categories. The value is computed by the algorithm and then stored in the database. This algorithm can then give the result of the queries having linguistic variables which adds vagueness to the result.

Fuzzy data has multiple values between (0, 1). Fuzzy data is imprecise or has partial truth values. Therefore, fuzzy data is usually defined in terms of membership value. Fuzzy data is represented with linguistic variable or quantifier.

The truth-value of a variable "x" will be denoted as $\mu(x)$.

A fuzzy database is a database which is able to deal with uncertain and incomplete information. Uncertain, imprecise and vague type of data can be handled by fuzzy database easily. Membership value is calculated for every data input and according to that membership value data is fetched out from database.

2. LITERATURE REVIEW

Classical data models often suffer from their incapability of representing and manipulatingimprecise and uncertain information that may occur in many real world applications. Since the early 1980's Zadeh's fuzzy logic [1] has been used to extend various data models. The purpose of introducing fuzzy logic in databases is to enhance the classical models such that uncertain and imprecise information can be represented and manipulated.

A query is flexible if the following conditions are satisfied [3]:

3.1 A qualitative distinction between the selected tuples is allowed.

3.2 Imprecise conditions inside queries are introduced when the user cannot define his/her needs in a definite way, or when a pre-specified number of responses are desired and therefore a margin are allowed to interpret the query.

Here typically, the former case occurs when the queried relational databases contain incomplete information and the query conditions are crisp and the latter case occurs when the query conditions are imprecise even if the queried relational databases do not contain imperfect information [2]

The GEFRED model in [4, 5] generalized fuzzy domains, unknown, NULL values, is a possibility model. The GEFRED model is based on the generalized fuzzy domain (D) and generalized fuzzy relation (R), which include classic domains and classic relations, respectively. This model defines fuzzy comparators, which are general comparators based on any existing classical comparator (>, <, =, etc.). GEFRED redefines the relational algebraic operators in the so-called generalized fuzzy relational algebra: union, intersection, difference, Cartesian product, projection, selection, join, and division.

Takahashi presents a fuzzy query language for relational databases [6] and discusses the theoretical foundation of query languages to fuzzy databases in [7]. Based on matching strengths of answers in FRDBs, a method for fuzzy query processing is presented in Chaing et al [8]. Yang et al [9] discussed nested fuzzy SQL queries in a FRDB.

3. Methodology

3.1 The GUI has been developed for calculation of

Membership values for crisp data.

3.2 The user of GUI enters marks on front endusing theproposed algorithm and formulae, the membership value of marks for different categories is calculated and these values are stored in the database.

3.3 If user desires to see the data, he/she may select any category and the query is fired and data is fetched out using the GUI.

4. Proposed Algorithm

4.1 Terms and variables used in the algorithm

X – Value to be stored in the database.

µs (X) – membership of the value in its own category.

µs-next (X) – membership of value in next category.

µs-prev(X) – membership of value in previous category.

a - lowerboundary values of same category.

b - Upper boundary value of same category.

AF- Adjustmentfactor.

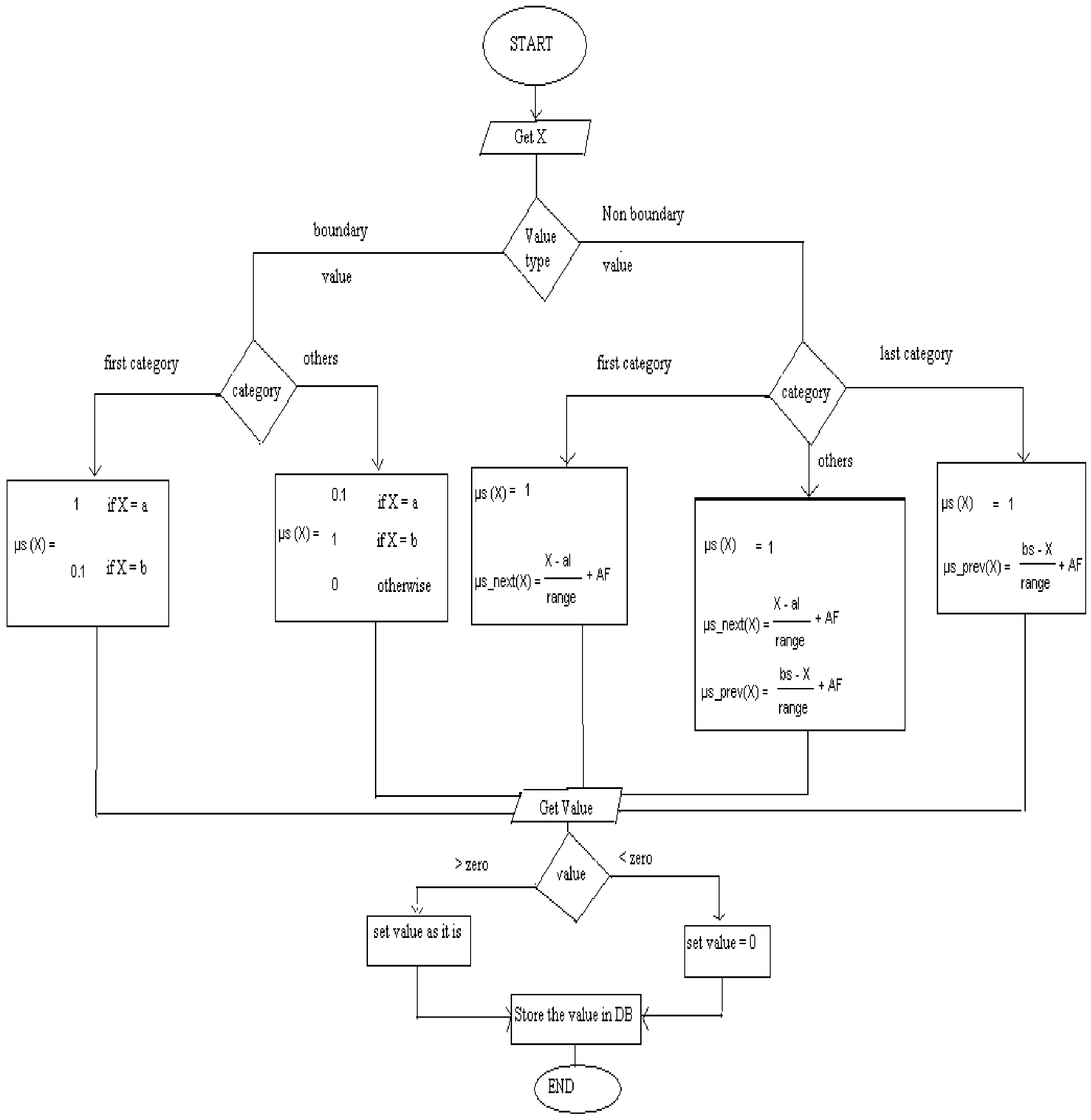
as - lower boundary value of previous category.

bs - upper boundary value of previous category.

al- lower boundary value of next category.

bl-upper boundary value of next category.

5.2 Flowchart for proposed algorithm (does not shows the front end part)



5.3 Stepwise Algorithm

1. Get the value of X [i.e. the value to be stored in the database].Also calculate the range as,
   Range = b – a.
2. Check whether the value to be stored is a boundary value or a non-boundary value,
   If X is not a boundary value,go to step 3
   else go to step 6.
3. If X lies in first category, Then μs(X) = 1
   μs_next (X)= ((X – al)/range) + AF
   go to step 7.
4. Else if X lies in last category then, μs (X)= 1
   μs_prev (X)= ((bs – X)/range) + AF
   go to step 7.
5. Else (when category is neither first nor last)μs (X) = 1
   μs_next (X) =((X – al)/range) + AF
   μs_prev (X)= ((bs – X)/range) + AF
   go to step 7.
6. When X is a boundary value, If X is in first category

$$\mu s\,(X) = \begin{cases} 1 & \text{if } X = a \\ 0.1 & \text{if } X = b \end{cases}$$

else

$$\mu s\,(X) = \begin{cases} 0.1 & \text{if} X = a \\ 1 & \text{if } X = b \\ 0 & \text{otherwise} \end{cases}$$

Note that if X is a boundary value then it should be checked for all the categories and its membership should be calculated for all the categories using above formulas.

7. If μs (X) < 0 [i.e. if any of the above calculated value of μs (X) is negative]
   Then set μs (X) =0.
8. Store the value of μs (X).
9. END

Note that the algorithm only calculates the membership of value in the immediate next and immediate previous categories along with the same category in which the value lies. The membership of value for all other categories can be directly set to zero or can be calculated according to above formulaswhich will result in a negative value. This negative value can then be set to zero in step 7 of the algorithm.

5.4 Calculation of membership value

Let us take the data of marks of students of a class in a subject out of 50. Based on the marks the students are divided into five categories i.e.

1 Below average [0 – 10]
2 Average [10 – 20]
3 Good [20 – 30]
4 Very good [30 – 40]
5 Excellent [40 – 50]

When working on crisp values, any student with 0 marks should be below average and any student with 9 marks should also be below average. I.e. there is no membership value of student with 9 marks in average category. Also student with 10 marks should be inclusive in only one of the category i.e. below average or average. This makes the representation discrete or non-continuous.

In crisp database, if we want to get the names of students in category below average, then the query passed should be-

Select roll_no, name, marks from marks where marks < 10;

When dealing with fuzzy data the student with 9 marks should have some membership in average category also. Also the student with 10 marks should have the membership in both below average and average categories. The above mentioned algorithm provides a way to store such a fuzzy data in the database by calculating the membership value.

The membership value of marks of a student for all the different categories is calculated. Let the value of adjustment factor, AF = 0.2

Note that any value of adjustment factor can be taken between (0,1) according to own choice.

The table 1 shows the stepwise calculation of values for different categories for student with marks 9 and 15:

Now, consider a student with 10 marks out of 50, since it is a boundary value, hence go to step 6 of the algorithm.According to step 6, since 10 is a boundary value of first category i.e. 10 = b,

μs (X) = 0.1 (for below average)

Since according to algorithm, boundary value of all the categories should be checked

Hence, μs (X) = 0.1 (for average)

Andμs (X) = 0 (for all other categories).

Table 1 Stepwise calculation of values for different categories

	MARKS = 9	**MARKS =15**
1.	Range= b-a = 10	Range = b-a = 10
2.	9 is not a boundary value, goto step 3.	15 is not a boundary value, goto step 3.
3.	9 lies in first category, Hence µs(X) = 1 Now, membership of student with 9 marks in average category, µs_next (X)= ((X – al)/range) + AF = ((9 – 10)/10) + 0.2 = - 0.1 + 0.2 = 0.1 Now, goto step 7	15 does not lie in first category, goto step 4. 15 does not lie in last category, goto step 5 µs (X) = 1 Now, membership of student with 15 marks in good category, µs_next (X) =((X – al)/range) + AF = ((15-20)/10) + 0.2 = - 0.3 µs_prev (X)= ((bs – X)/range) + AF =((10 – 15)/10) + 0.2 = - 0.3 Now, goto step 7
4.	According to step 7, check the above calculated values, µs(X) = 1 (positive value) µs_next (X) = 0.1(positive value)	According to step 7, check the above calculated values, µs (X) = 1(positive value) µs_next (X)= - 0.3 (negative value) hence set µs_next (X) = 0. µs_prev (X)= - 0.3 (negative value) hence set µs_prev (X) = 0.
5.	Since both the values are positive, store them in database.	Store the values in the database.

A table with name, marks and calculated membership values for all five categories is shown;

Za represents membership of marks in below average

Zb represents membership of marks in average category.

Zc represents membership of marks in good category.

Zd represents membership of marks in very good category.

Ze represents membership of marks in excellent category.

Suppose we want to find the names of all the students who are average, then we can select the linguistic variable as Average and the front end will generate a query to the database as follows,

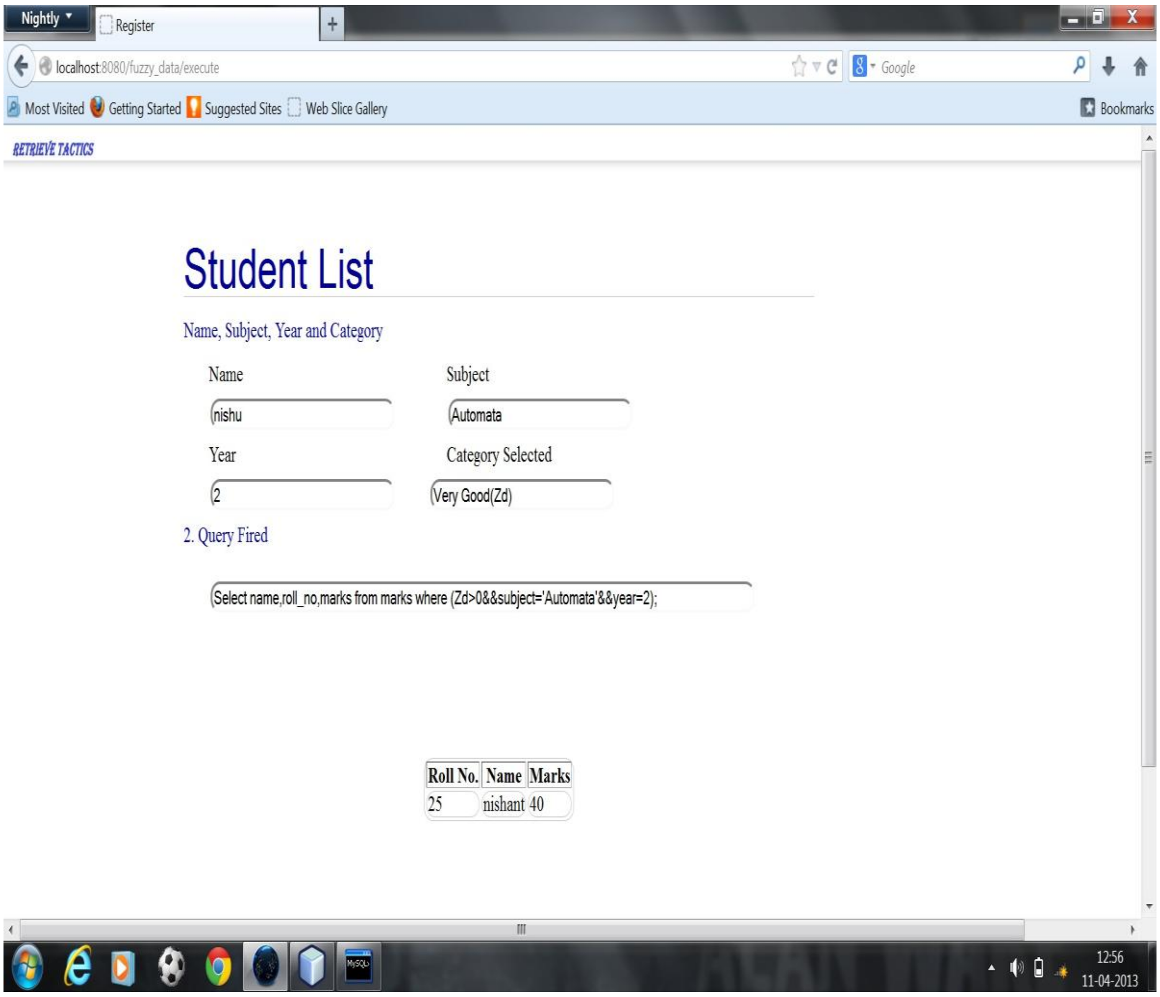
Select roll_no, name, marks from marks where Za> 0;

Hence the database will give the name of student with 11 marks also in the result of average students.

Table 2 shows storage of fuzzy data in crisp form in database

Roll No.	Name	Marks	Za	Zb	Zc	Zd	Ze
1	Aman	2	1	0	0	0	0
2	Ankit	9	1	0.1	0	0	0
3	Yugal	10	0.1	0.1	0	0	0
4	Neha	11	0.1	1	0	0	0
5	Tushar	20	0	1	0.1	0	0

6. Interface Design



Student List

Name, Subject, Year and Category

Name: nishu
Subject: Automata
Year: 2
Category Selected: Very Good(Zd)

2. Query Fired

```
Select name,roll_no,marks from marks where (Zd>0&&subject='Automata'&&year=2);
```

Roll No.	Name	Marks
25	nishant	40

Figure1 GUI of the application

Figure 1 shows the selection of linguistic variable in the front end of the application and the text area (Editor) shows the query passed in the database which gives the result shown in

.

the table student details. The front end has been developed in JSP and the database usedis MYSQL

7. Conclusion

Till now data used for the value used are crisp value. But now with the help of proposed algorithm in this paper fuzzy data can be stored in the form of crisp value in database. By calculating membership value of data, the result is fetched for any type of fuzzy input. It will store the membership value in the database and will return output for all positive values of membership values which will be the desired output for the given input.

8.Acknowledgement

The satisfaction that accompanies that the successful completion of any task would be incomplete without the mention of people whose ceaseless cooperation made it possible, whose constant guidance and encouragement crown all efforts with success. We are grateful to our paper guide **MrsRASHMI SHARMA** for the guidance, inspiration and constructive suggestions that helpful us in the preparation of this paper.

9.References

[1] Zadeh, L. A. (1971). Similarity relations and fuzzy orderings. *Information Sciences, 3*, 177-200.

[2] Umano, M., &Fukami, S. (1994). Fuzzy relational algebra for possibilitydistribution- fuzzy-relation model of fuzzy data. *Journal of Intelligent Information Systems, 3*, 7-28.

[3] Zemankova-Leech, M., &Kandel, A. (1984). *Fuzzy relational databases: A key to expertsystems.* Köln, Germany: Verlag TUV Rheinland.Fgf.

[4] Prade, H., &Testemale, C. (1987a). Fuzzy relational databases: Representational issues and reduction using similarity measures. *J. Am. Soc. Information Sciences, 38*(2), 118-126.

[5] Y. Takahashi, "A fuzzy query language for relational databases,"*IEEE Transactions onSystems, Man and Cybernetics*, Vol. 21, 1991, pp. 1576-1579.

[6] D. A. Chiang, N. P. Lin, and C. C. Shis, "Matching strengths of answers in fuzzy relational databases," *IEEE Transactions on Systems, Man, and Cybernetics-Part C:Applications and Reviews*, Vol. 28, 1998, pp. 476-481.

[7] V. Cross, "Defining fuzzy relationships in object models: Abstraction and interpretation,"*Fuzzy Sets and Systems*, Vol. 140, 2003, pp. 5-27.

[8] V. Cross, "Fuzzy extensions for relationships in a generalized object model," *InternationalJournal of Intelligent Systems*, Vol. 16, 2001, pp. 843-861.

[9] G. Q. Chen, E. E. Kerre, and J. Vandenbulcke, "The dependency-preserving decomposition and a testing algorithm in a fuzzy relational data model," *Fuzzy Sets and Systems*, Vol. 72, 1995, pp.27-37.

[10] L.A. Zadeh, The concept of a linguistic variable and its application to approximate reasoning. Information Sciences, Part 1: 8:199-249; Part 2:2:301-357; Part 3: 9:43-80, 1975.

[11] Galindo, J., Urrutia, A., Piattini, M., Fuzzy Databases: modelling, Design and Implementation, Idea Group Publishing, Hershey, USA. (2006).

[12] B. Bhuniya and P. Niyogi, "Lossless join property in fuzzy relational databases," *Data and Knowledge Engineering*, Vol. 11, 1993, pp. 109-124.

[13] T. K. Bhattacharjee and A. K. Mazumdar, "Axiomatisation of fuzzy multivalued dependencies in a fuzzy relational data model," *Fuzzy Sets and Systems*, Vol. 96, 1998, pp. 343-352.

[14] G. Q. Chen, *Fuzzy Logic in Data Modelling*; *Semantics, Constraints, and DatabaseDesign*, Kluwer Academic Publisher, 1999.

[15] Z. M. Ma, W. J. Zhang, and F. Mili, "Fuzzy data compression based on data dependencies," *International Journal of Intelligent Systems*, Vol. 17, 2002, pp. 409-426.

[16] S. Y. Liao, H. Q. Wang, and W. Y. Liu, "Functional dependencies with null values, fuzzy values, and crisp values," *IEEE Transactions on Fuzzy Systems*, Vol. 7, 1999, pp. 97-103.

[17] K. H. Lee, *First Course on Fuzzy Theory and Applications*, Springer, 2004.

[18] M. Kamel, B. Hadfield, and M. Ismail, "Fuzzy query processing using clustering techniques," *Information Processing and Management*, Vol. 26, 1990, pp. 279-293.

[19] R. Intan and M. Mukaidono, "Fuzzy functional dependency and its application to approximate data querying," in *Proceedings of International Database Engineering and Applications Symposium*, 2000, pp. 47-54.