Handling Fuzzy SQL on Crisp databases using Lex-YACC

Tejas Mehta
Nirma University
Ahmedabad, India

Pooja Shah
Nirma University
Ahmedabad, India

ABSTRACT
Conventional querying language presumes precision and certainty in user queries. But in factual situations the queries may be imprecise and may be desired to result in uncertain outcome. Introducing fuzziness in querying permits the realistic querying on the crisp data. In this paper, we propose the architecture for fuzzy querying along with an experimental implementation of the same. The implementation is using LEX and YACC that facilitate the lexical analysis of fuzzy terms and parsing the fuzzy query respectively. Fuzzy query is interpreted by the parser and the consequent semantic actions are carried out on MySQL database.

General Terms
Fuzzy SQL

Keywords
Fuzzy database, LEX, YACC,

1. INTRODUCTION
Most of today's database management system consists of the crisp information. The primary goal of the DBMS system is to provide the precise information at the time of retrieval. The conventional database management system does not handle imprecise, incomplete or vague information such as very high, approximately some values. To triumph over this problem, the fuzzy database system has been introduced.

In the real world scenario, every time the information we need can not be precisely stated but rather than that there is a need to deal with natural language and retrieving the pertinent information. The paper has been divided in four parts. The essential idea is to extend the SQL to incorporate fuzzy querying.

Section 1 and 2 presents introduction and the basics of fuzzy set correspondingly. Section 3 presents fuzzy querying concept. Section 4 describes our system architecture.. Section 5 gives the implementation details and 6 and 7 portray the future enhancements and conclusion respectively.

2. BASIC CONCEPTS: FUZZY SET
Normally sets are defined as the collection of objects having one or more common characteristics. The objects that belong to the set are called the member of the set.

Fuzzy sets [7] are sets whose boundaries are not precise and the membership in the fuzzy set is not the matter of whether object clearly belongs to the set or not that is the membership is not in the form of either true or false but rather a matter of degree. A fuzzy set extends the binary membership: {0, 1} of a conventional set to a spectrum in the interval of [0, 1]. Furthermore unlike conventional set all elements of the universal set are the member of given set. Thus for each element x U

$$0 \leq \mu(x) \leq 1$$

In theory, membership functions can take any form, but typical functions are γ function, s-function, L-function, triangle and Gaussian etc. [10] The various functions are described as. The γ function has two parameters α and β.

$$\gamma(u; \alpha, \beta) = 0 \quad u <= \alpha$$
$$= (u- \alpha) / (\beta - \alpha) \quad \alpha <= u <= \beta$$
$$= 1 \quad u > \beta$$

The s-function is a smooth version of γ function. L-function is the inverse of γ function. The triangle membership is defined as

$$T(u; \alpha, \beta, \gamma) = 0 \quad u <= \alpha$$
$$= (u- \alpha) / (\beta - \alpha) \quad \alpha <= u <= \beta$$
$$= (\alpha - u) / (\beta - \alpha) \quad \beta <= u <= \gamma$$
$$= 0 \quad u > \gamma$$

The Π-function can be formally described as follows:

$$\Pi(u; \alpha, \beta, \gamma, \delta) = 0 \quad u <= \alpha$$
$$= (u- \alpha) / (\beta - \alpha) \quad \alpha <= u <= \beta$$
$$= 1 \quad \beta <= u <= \gamma$$
$$= (\gamma - \delta) / (\delta - \gamma) \quad \gamma <= u <= \delta$$
$$= 0 \quad u > \delta$$

The Gaussian function is described as

$$G(u;m,\sigma) = \exp[-\{(u-m)/ \sqrt{2}\sigma\}^2]$$

3. NEED OF FUZZY QUERY
Fuzziness is introduced in database query language to allow imprecise querying on conventional data. If we aim for conventional crisp querying language it is precise and certain querying. Precision assumes that the effect will be exactly our perception and certainty assumes the structure and parameter are exactly identified.

But for factual database there may be the understated complications:

- Actual situations are very often not crisp and deterministic and cannot be described precisely i.e. Real situations are very often uncertain or vague in a number of ways.
- Complete description of a real system would entail far more detailed data than a human being could ever be acquainted with and process at the same time.

To get ride of these complications we need to be concerned about the notion of uncertainty. We have implemented fuzzy SQL using Lex and YACC to incorporate uncertainty in querying. Our experiment includes creation of meta knowledge, grammar, parser and enabling fuzzy querying on MySQL crisp database keeping the default SQL intact.

The constraint imposed on the system is that we have not changed the database model and thus fuzziness can be archived at the querying front end only.

As an example consider the employee record in the database system. Suppose we want to list out the employees whose salary is greater than 20000 and age is greater than 25. The crisp query can be specified as

Select emp_name from EMP where sale_amount > 20000 and age < 25

The conventional query like above produces the required result but the major drawback of above query is its rigid boundaries. Here the employee whose sal_amount is 19000 and having age less than 25 is not considered. Such employee should have been considered but it is not. In realistic situation retrieval of such rigid information is not useful in true sense. Instead we would be more interested in finding out the employees who are young and made good sale. The notion of fuzzy logic is helpful to produce such results. The various distribution functions are shown in the figure 2.

4. ARCHITECTURE

Our proposed architecture is shown in the figure 2. This simply shows that an actor/user can fire a fuzzy query which will then be parsed by our query parser and the relevant semantic action will be taken on MySQL crisp database. For incorporating fuzziness we use metadata as explained in section 5.1. The user can also directly enter the data into MySQL database as well as in the meta data tables. [1][6][9]

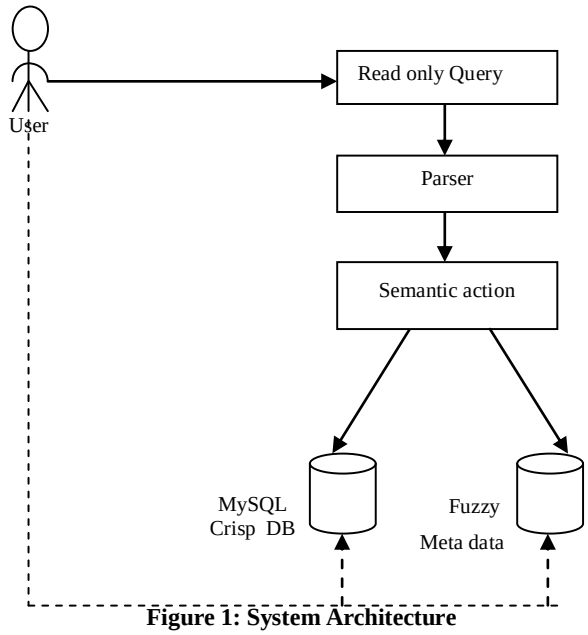

Figure 1: System Architecture

5. IMPLEMENTATION

The implementation has been carried out using lexical analyzer tool flex and parser generator tool YACC. Most of the terms given by users can be handled by the lexical analyzer implemented in the experiment and which in turn returns appropriate tokens to the parser. The parser accepts the tokens and parses the input query. Once the query is parsed successfully then the semantic actions are carried out. In this way the fuzzy query is taken as input and column name and terms are recognized. The rows are fetched and displayed after looking at the calculated value of membership. The membership of column and in turn row to be displayed is determined by seeing the fuzzy metadata which will be discussed later in this section. Once the appropriate membership function is applied, the final membership value is calculated as stated below:

Final membership value = membership value ^ n

Where n is the number of times the particular term repeats [10].

5.1 Meta Knowledge

We have implemented Fuzzy attribute type 1 Querying model [GEFRED Model] [1][2][9]. This means we are introducing fuzziness in querying leaving the database crisp. At the level of meta knowledge we need to add only a single table, with the following structure:

Level	CName	Alpha	Beta	Gamma	Delta

This generic table is used to store the information of all the fuzzy functions defined on all the attribute domains. A description of each column in this table is as follows:

Level: This column indicates the level of fuzziness for the database attribute specified in CName. Level stores tokens associated with the various linguistic terms such as high, low etc. . This facilitates the use of wide range of linguistic terms and its corresponding functions. Stores the linguistic variable associated with the given linguistic term. The rest of the attributes Alpha, Beta, Gamma, Delta provides required information to calculate the membership of the specified column and level [4].

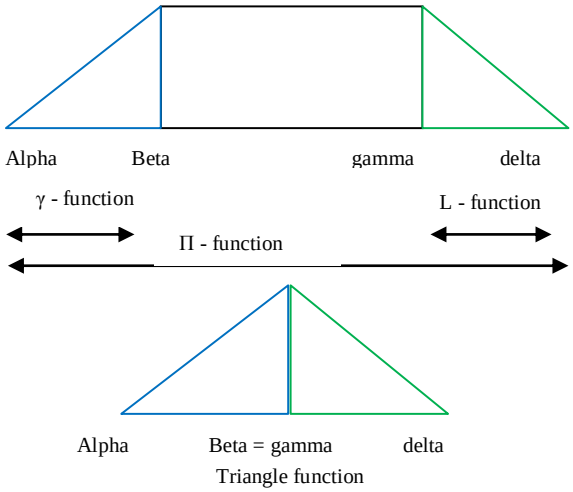

Figure 2. Membership functions

As an instance if the values of Alpha and Beta are zero then L function is considered. In the same way γ function is determined if the values of Gamma and Delta are zero. Whenever values of beta and gamma are same the triangular function is used. And when all the four values are non-zero Π-function is used. The formal specifications of all these functions are given in section II of this paper. The graphical representation of these functions is shown in figure 2

5.2 Grammar

The subsequent figure shows the YACC grammar for handling fuzzy queries on MySQL database. The grammar mentioned here handles read-only queries with single condition. The lexical analyzer employed handles almost all real world linguistic terms. The grammar is robust enough to handle uncertain query format such as any number of white spaces between lexemes.

```
sqllist :  sql sqllist
        |
  { printf("Bye ! Have a nice day"); }
 ;
sql : SELECT SPACE '*' SPACE
     FROM SPACE tablename SPACE wstart ';'
{
/* SEMENTIC ANALYSIS
// Get details from fuzzy meta data table
/* if(opt==1)
   if alpha=0 and beta =0
             calculate L- membership
          else
   if gemma=0 AND delta=0
             if col-value >= beta
       membership = 0
             else
              calculate γ - membership
   else if beta <>0 AND gemma <> 0
    if col-val > alpha AND col-val < beta
      calculation of pi membership
       if col-val >= beta AND col-val <= gemma
                membership = 1;
       if col-val > gemma
             calculate membership

 if any membership > THRESHOLD
                      Print current row
else  // option2 used for handling approximate query
  get the details from approximate table
  calculate fuzziness
  and display the results
end
 flag=0  //used in error handling
 count=1 //used to reset the count value

*/ }
;
tablename : NAME
         { strcpy(tname,name);};
wstart : WHERE SPACE columnname
               SPACE IS SPACE st { opt =1; }
          | WHERE SPACE columnname
               SPACE IS SPACE APPR SPACE VAL
        { opt = 2; }
;
columnname : NAME
        { strcpy(colname,name); } ;
st : adj level
   | level       ;
level :      HIGH { tok = HIGH; }
         | LOW  { tok = LOW; }
         | MED  { tok = MED; }
         | MODERATE { tok = MODERATE; }
         | LESS { tok = LESS; }
            ;
adj : adj SPACE adj
            | VERY {   count = count+1; }
            | APPR
            | EXTR { count = count + 4; }
      |
    ;
```

It should be noted that the care is taken for the use of the words such as extremely where the count value is incremented to 4 when EXTR token for the word extremely is matched. This is equivalent to specifying the word "very" four times. Whenever the valid syntax is found the count value is reset so as to handle the newer query. Whenever the wrong query is given the error message should be given only once as parser calls yyerror ( ) function multiple times for each wrong term. To achieve this value of flag is used [8].

5.3 Results

The output of the FSQL implementation is shown in the figure 3. As after compilation the output file is executed the FSQL prompt appears.

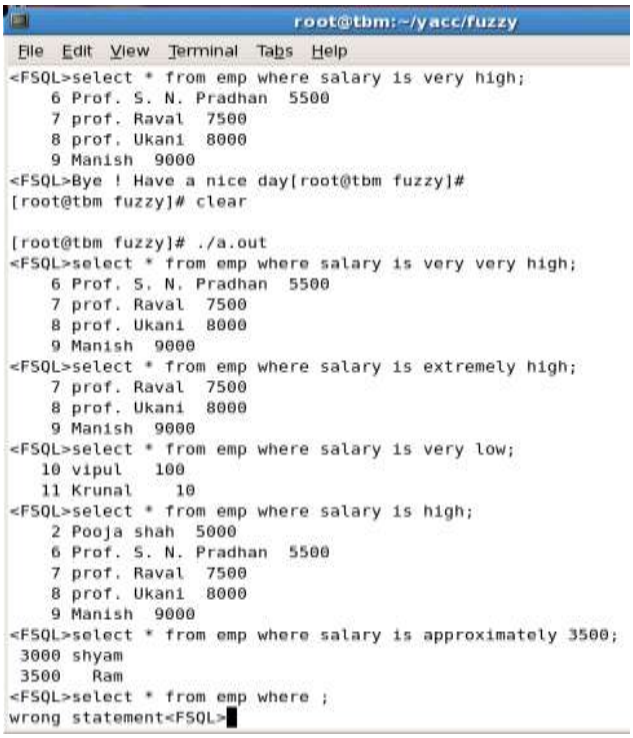

Figure 3. Implementation Outcome

One can give any selection query with any condition leading to uncertain outcome. As it can be seen that for the queries select * from emp where salary is very high or very very high or extremely high or very low, high etc. These queries may generate different or similar outputs depending on the application of membership function [5][8]. Moreover user can specify his own fuzzy function to be incorporated. In such situation, it is required to specify the required meta information which can be used to calculate membership function.

5.4 Constraints

There are a few constraints along with the experiment that has been carried out. As mentioned earlier the concentration was on having fuzziness in querying, the database involved is still crisp. The implementation works for read only queries for retrieval of data. The queries with simple conditions can only be parsed.

6. FUTURE ENHANCEMENTS

With this concept and its implementations we can look forward for the following future enhancements [5][8].

- Processing of complex as well as manipulation queries.
- Implementation of fuzzy database along with the fuzzy queries.
- Automatic mapping of crisp and fuzzy database.

7. CONCLUSION

It is found that lexical analyzer tool such as flex and parser generator tool such as YACC are suitable for writing down entirely new grammar for handling the real world scenario of querying. The grammar can be enhanced to as per the future needs of the user. The same way more terms can be incorporated in Lexical analyzer. This approach facilitates users to fire the fuzzy queries in natural way. In near future we may work upon making the implementation fully equipped with all sorts of queries rather than just read only ones and that too upon fuzzy database.

8. REFERNCES

[1] Jose Golindo, A. Urrutia, M. Piattini 2004. Representation of Fuzzy Knowledge in Relational Databases. In Proceedings of the 15th International Workshop on Database and Expert Systems Applications.

[2] Amel Grissa Touzi and Mohamed Ali Ben Hassine 2009. "New architecture of fuzzy database management system". The International Arab Journal of Information Technology, Vol. 6, No. 3.

[3] A.H.M. Sajedul Hoque, Md. Sadek Ali, Md. Aktaruzzaman, Sujit Kumer Mondol, and Dr. Babul Islam 2008. Performance Comparison of Fuzzy Queries on Fuzzy Database and Classical Database. In 5th International Conference on Electrical and Computer Engineering.

[4] Hrudaya Ku. Tripathy, B.K.Tripathy, Pradip K Das and Saraju Pr. Khadanga 2008. Application of Parallelism SQL in Fuzzy Relational Databases International Conference on Computer Science and Information Technology.

[5] T.C. Ling, Mashkuri Hj . Yaacob, K.K. Phang 1997. Fuzzy Database Framework - Relational Versus Object-oriented. In Proceedings of the 1997 IASTED International Conference on Intelligent Information Systems.

[6] Claudia González, Marlene Goncalves and Leonid Tineo 2009. A New Upgrade to SQLf: Towards a Standard in Fuzzy Databases. In Proceedings of 20th International Workshop on Database and Expert Systems Application.

[7] Zhu Yanqin, Li Fanzhang, and Hu Yuemei, Soochow University 2005. The Design and Application of Dynamic Fuzzy Expert Database System.

[8] Qi Yang', Chengwen Liu2, Jing Wu', Clement Yu', Son Dao3, Hiroshi Nakajima4 1995. Efficient Processing of Nested Fuzzy SQL Queries*

[9] Angélica Urrutia, José Galindo, Mario Piattini 2002. Modeling Data Using Fuzzy Attributes*. In Proceedings of the XXII International Conference of the Chilean Computer Science Society

[10] Amit Konar. Computational Intelligence. ISBN 3-540-20898-4. Springer Berlin Heidelberg New York.