Mem. Fac. Sci. Shimane Univ.,
27, pp. 73–80 Dec. 27, 1993

Fuzzy Databases: Structures and Queries

Takashi Kayano and Khin M. Yin

Department of Information Science, Shimane University,
Matsue 690, JAPAN
School of Technology
Kent State University, Kent, Ohio, U.S.A.

(Received September, 6, 1993)

In this paper, we give a brief review of approaches to fuzzy databases and present our view on a specific structure of fuzzy database systems. In doing so, we examine difficulties in dealing with imprecise knowledge and incomplete knowledge. Fuzzy logic and first order predicate logic form a basis in implementing fuzzy databases.

Key Index Words: fuzzy databases, fuzzy logic, logic programming

Introduction

The use of databases for data storage and retrieval operations has been in existance for over two decades. Among the three database models–the *network* model, the *hierarchical* model and the *relational* model, the latter appears to be more in use nowadays. As defined in the papers of Codd [1], a *relational* database is essentially a table consisting of rows and columns. Each row is like a record. The colums in a row correspond to fields in a record. A row is called a *tuple* and a field is called an *attribute.* The total number of *attributes* in a *tuple* is called *arity*. The total number of *tuples* in a database is called *cardinality.* The data element in an *attribute* must be atomic. That is, the value can not be further divided nor has structures. In spite of the restrictive atomic nature of the database values, the use of *relational* databases appear to have wide spread applications in data processing. One of the requirements of a *relational* database is that all the values be present in a database. If a particular value is not known, a dummy data value is usually inserted. Creating databases, inserting and deleting *tuples,* and updating desired *attributes* are major functions in using *relational* databases. Such databases are primarily good for storing and retrieving precise data values. An example database is shown in *Figure 1.* Notice that all the values in the database relation are precise. In the fifth row, the dummy value no-name is intended as not knowing the child's name, although the child exists. In the sixth row, the dummy value no-child means that Glen has no child.

relation FRIENDS

Name	Age	Child
Thomas	21	Bill
John	27	Jack
Tanaka	40	Fujie
Victor	37	Mary
Yamada	60	no-name
Glen	51	no-child

Figure 1. A crisp relational database.

In real data processing applications, precise values are not always known. Therefore, a database creator may not be able to put the imprecise values into the system. The knowledge of imprecise values may have usefulness for the users and they should be stored as such. Consider the database *relation* shown in *Figure 2*. The first row and the third row have precise values. The second row has precise values except the value of Child, which is a list [Jack, Joan]. Since a list has a structure, the database is not conforming to the strict requirements of a relational database. However, the information is useful to the users. The foruth row has the value *young* which has a linguistic structure in terms of fuzzy logic. It is interpreted by a possibility distribution function.

The value *unknown* is not atomic. It is interpreted as a possibility distribution in which all child names are equally possible. The fifth row has the Age value [60, 61] which is a list. The list may mean that Yamada's age is 60 or 61, or around these values. The Child's name is *undefined.* It is interpreted as a possibility distribution in which all names have a possibility of zero. The sixth row has the Age value of *middle-aged*, which has a linguistic structure in terms of a possibility distribution. The value *null* is interpreted as a special value. The value implies that we do not know if Glen has any children or not. The values: *unknown, undefined* and *null* are examples of incomplete information. A complete database is an ideal database.

Relation PERSONS

Name	Age	Child
Thomas	21	Bill
John	27	[Jack, Joan]
Tanaka	40	Fujie
Victor	*joung*	*unknown*
Yamada	[60, 61]	*undefined*
Glen	*middle-aged*	*null*

Figure 2. A fuzzy relational database.

There is a wealth of articles describing ways to extend the relational database to include imprecise data using fuzzy logic [3–5]. Such databases are called fuzzy databases. These models attempt to treat imprecise data and to some extend to treat incomplete data. Ours is an approach to representing imprecise data using simple fuzzy logic membership functions. We describe a fuzzy database system structure that has interpreters for three cases: (1) PROLOG-like queries, (2) SQL-like queries, (3) Natural Language Processing (NPL) queries. In the three modes of query processing, fuzzy interpreters are to be implemented. We show a few example of fuzzy queries in PROLOG. The discussion includes representations of imprecise data, expressiveness of query processing and difficulties in efficient implementations.

Fuzzy Logic

Fuzzy sets are a mathematical concept proposed by L. A. Zadeh in 1965. Since then, much progress has been made in developing mathematical possibility theory and the applications of the possibility theory to various systems of practical uses. See the references at the end of this paper. From communication point of view, human language expressions contain fair amount of ambiguity and subjectivity. The expressions *low temperature, high temperature* and *tall person* are examples of ambiguity. They are subject to interpretations. These expressions are said to be fuzzy. Consider a classification of age and assigning adjectives (linguistic variables) to the ranges of age values:

Variable	range of ages	possibility value
young	range [1, 33)	1
middle-aged	range [33, 66)	1
old	range [66, 99]	1

In this table, a person is *young* if he is of the age between 1 and less than 33. The possibility value is 1 for all the possible values between the two limits. Similarly, a person is *middle-aged* if his age is between 33 and less than 66 with the possibility value of 1 for all the age values between the two limits. Similarly, a person is *old* if his age is between 66 and 99 with the possibility value of 1 for all the values of ages between the two limits. This representation is said to be crisp, and it corresponds to the two-valued logic. Now, condider the situation where the possibility values are different for different age values.

variable	fuzzy set representation
young	{1/10, 1/20, .8/30, .5/33}
middle-aged	{.1/30, .5/40, 1/50, .5/60, .1/66}
old	{.2/50, .4/60, .6/70, .8/80, .9/99}

In representing the variable *young*, different values of the possibility are used for different ages. For example, the possibility value of 1 for the age value 10, written as 1/10, the possibility value of .8 for the age value 30, written as .8/30, and so on. The variable is thus represented by a finite sequence of numbers, the fuzzy set. The possibility values is said to constitute the membership function for the variable *young*. Similarly, the variables *middle-aged* and *old* are represented by the finite fuzzy sets shown above.

There are infinitely many sets that can be used to represent the variables. The crisp set and fuzzy set representations of the fuzzy variables are shown in *Figure 3*. The variable *young* may be represented by a monotonically decreasing function, the variable *old* may be represented by a monotonically increasing function, and the *middle-aged* by a gaussian function. For practical purposes, the monotonic functions may be chosen to be polygonal function, and the gaussian function is approximated by a triangular function.

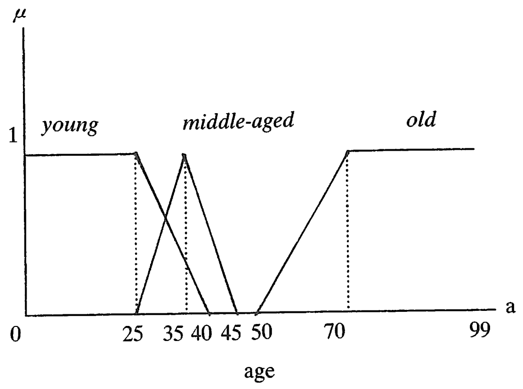

Figure 3. Representative membership functions.

In cases where the results are represented by two or more variables, the desired results are manipulated using the fuzzy set operations. To illustrate, consider a person's age is said to be *young* and *middle-aged.* The result is obtained by applying the logical AND operation of the two fuzzy sets that represent the two variables. Similarly, the result of *young* or *middle-aged* is obtained by applying the logical OR operation of the two fuzzy sets that represent the two variables.

Let X be a domain of a fuzzy set and A be a fuzzy subset of X. A membership function for a variable is written as μ_A (x) for values of x that are in X. Let B be a fuzzy subset of X, and μ_B (x) be a membership function for another variable. The union of the two fuzzy sets A and B is defined by the membership function $\mu_{A\cup B}$ (x):

$$\mu_{A\cup B}(x)=\mu_A(x) \vee \mu_B(x). \qquad \text{(AND operation)}$$

The intersection of the two sets is defined by the membership function $\mu_{A\cap B}$ (x)

$$\mu_{A\cap B}(x)=\mu_A(x) \wedge \mu_B(x). \qquad \text{(OR operation)}$$

There are ways of performing these operations using the Extension Principle of Fuzzy Set Theory [3, 4]. The resultant membership functions are used to get appropriate fuzzy variables.

Fuzzy-Relational Database Structures

In standard relational databases, the data values are of crisp type and the operations performed on the relevent numerical values are those of the ordinary statistics. In fuzzy databases, the data values may be fuzzy variables and non-atomic functions such as lists and unknowns. So, it is appropriate to use fuzzy operations and logic. In practice, fuzziness enter from several sources: (1) the fuzziness of the data providers, (2) the language of the providers, (3) the fuzziness of the data acquistition individuals, (4) the way the data is represented in the database, (5) the way the fuzzy interpreter treat the data, (6) the way the querying processes are treated. The first three sources are in the areas of data acquisitions. The processes of data acquistion are rather complicated.

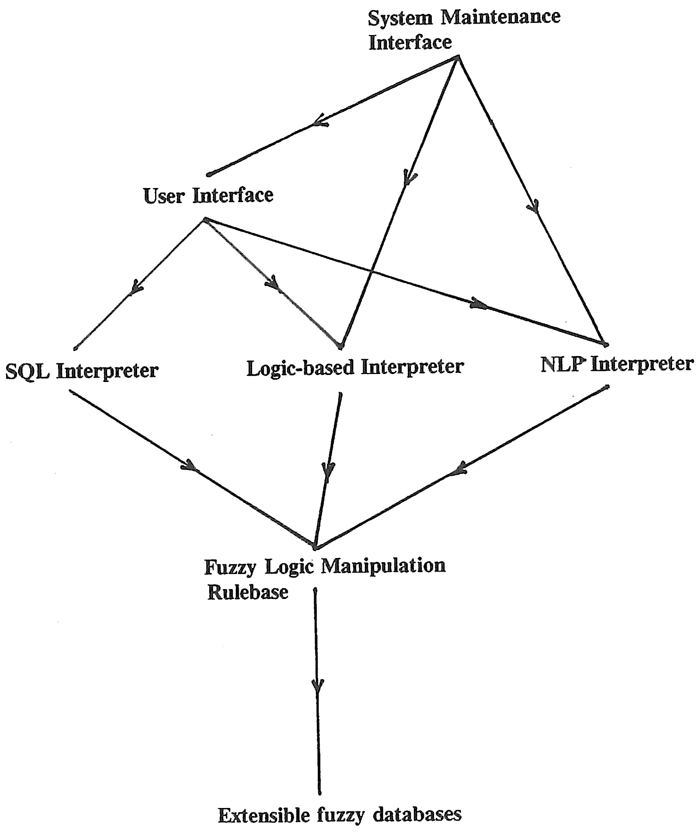
System Maintenance Interface
User Interface
SQL Interpreter
Logic-based Interpreter
NLP Interpreter
Fuzzy Logic Manipulation Rulebase
Extensible fuzzy databases

Figure 4. Fuzzy-Relational Database System.

The forms of imprecision and incompleteness are of five types: (1) non-determinism, (2) multiple meanings, (3) uncertainty, (4) incompleteness, (5) fuzziness. The focus of this paper is on (5), the treatment of fuzziness.

A fuzzy database system should have a structure that accomodates more than one way of manipulating data. We follow the basic ideas of Possibilistic Relational Universal Fuzzy (PRUF), as proposed by Zedah [2]. The system that we visualize has these components:

(1) The User Interface
(2) The System Maintenance Interface
(3) The Interpreters
(4) The Extensible Fuzzy Database

The system is shown in *Figure 4.* The User Interface interacts with the users of the database system. The Interface provides the use of SQL language, logic-based language, and Natural Language Processing (NLP) language. All the three languages can have fuzzy variables and values. Each language interpreter uses the fuzzy data manipulation rules that are organized as a rule base. This is equivalent to the PRUF's translation rules. The extensible fuzzy database is a collection of fuzzy database relations accessable by all the language interpreters. The System Maintenance Interface is primarily for maintaining the integrity of the database and it is used by the developers and maintenance groups.

Some Examples of Implementation

The prototype work that has been done are in PROLOG language. The choice of PROLOG language is due to its simplicity in creating fuzzy relational database. PROLOG language allows database values to have structures, such as finite sets and lists, fuzzy variables, undefined, unknown, and null. The values are restricted to first order logic expressions. The sample fuzzy databases are similar to those described in reference [6]. However, the fuzzification and defuzzification calculations are done using simple fuzzy representations: monotonically increasing and decreasing functions, and triagular functions. The first type of function is appropriate for fuzzification of the variables *small, low, short, less* and their appropriate hedges. The second type of function is appropriate for fuzzification of the variables *large, tall, more* and their appropriate hedges. Triangular functions are used to fuzzify *middle-aged, average, most, overall, etc.* Test databases are described in reference [7].

Calculation dealing with *young*: Consider the fuzzy variable represented by a function:

$$\mu(a)=\begin{cases} 1.0 & (0\leqq a<25) \\ -\frac{1}{15}a+\frac{8}{3} & (25\leqq a<40) \\ 0.0 & (40\leqq a\leqq 99) \end{cases}$$

A PROLOG fuzzification rule to calculate all possible values of age is rather simple:

```
fuzzify(Input_string, Mu,
              Nage, Delta, Amin, Amax) :-
          Input_string = '$Young',
          Delta is 15 * ( 1 - Mu ),
          Nage is Delta + 25,
          Amin = 0,
          Amax is Nage + 3.
```

The outputs of this PROLOG rule are the numerical values of Amin and Amax. The fuzzy database access using these two values of age as boundry values would produce relevent data from tuples that have age values within the two limits, and the value *young*.

Calculation dealing with *old*: Consider this representation:

$$\mu(a) = \begin{cases} 0.0 & (0 \leqq a < 50) \\ -\frac{1}{20}a + \frac{5}{2} & (50 \leqq a < 70) \\ 1.0 & (70 \leqq a \leqq 99) \end{cases}$$

A PROLOG fuzzification rule to calculate all possible values of age is also simple:

```
fuzzify(Input_string, Mu,
              Nage, Delta, Amin, Amax) :-
          Input_string = '$Old',
          Delta is 20 * (1 - Mu),
          Nage is 70 - Delta,
          Amin is Nage - 3,
          Amax is 99.
```

Accessing the fuzzy database would produce desired data values within the two boundry values of Amin and Amax.

Calculation dealing with *middle-aged*: This variable may be represented by this function:

$$\mu(a) = \begin{cases} 0.0 & (0 \leqq a < 25) \\ \frac{1}{10}a + \frac{5}{2} & (25 \leqq a < 35) \\ -\frac{1}{10}a + \frac{9}{2} & (35 \leqq a < 45) \\ 0.0 & (45 \leqq a \leqq 99) \end{cases}$$

A PROLOG fuzzification rule to calculate all possible values of age is:

```
fuzzify(Input_string, Mu,
                  Delta, Amin, Amax) :-
             Input_string = '$Moddle-aged',
             Delta is 10 * (1 - Mu),
             Amin is 32 - Delta,
             Amax is 48 + Delta.
```

Similarly, the outputs of this rule are numerical values of Amin and Amax. A database access using these boundry values produce the desired data values.

In the above examples, constant numerical values are used. In the fuzzy manipuation database, variables are used to select the desired numerical range values through the User Interface. Thus, there is no loss of generality in the PROLOG fuzzification rules. PROLOG implementation, such as ARITY/PROLOG, was used for testing purposes.

Discussion

The structure and organization of fuzzy database system described above is straight forward. The extensible databases are disk-based and accessing them as required by the query process. In accessing low cardinality disk-based relational database, complicated indexing schemes are not necessary. The speed of fuzzy calculation and database access are acceptable. For large cardinality databases, efficient indexing schemes are needed. Our future work includes developing fast SQL and NLP interpreters, which have capability of dealing with fuzziness.

References

[1] C. J. Date. *An Introduction to Database Systems.* Addison-Wesley, Volume 1, 1981.

[2] L. A. Zaheh, "PRUF—A Meaning Representation Language for Natural Language" in *Fuzzy Sets and Applications: Selected Papers* by L. A. Zadeh., R. Yager, S. Ovchinnikov, R. M. Tong, H. T. Nguyen. John Wiley and Sons, Inc., 1987.

[3] T. Terano, K. Asai and M. Sugeno. *Fuzzy Systems Theory and Applications.* Academic Press, Inc. 1992.

[4] A. Kandel. *Fuzzy Mathematical Techniques with Applications.* Addison-Wesley Publishing Co., 1986.

[5] J. Kacprzyk and A. Ziolkowski. "Retrieval from Data Bases using Queries with Fuzzy Linguistic quantifiers" in *Fuzzy Logic in Knowledge Engineering.* H. Prade and C. V. Negoita. Koln: Verlag TUV Rheinland, 1986.

[6] D. Li and D. Liu. *A Fuzzy Prolog Database System.* John Wiley and Sons, Inc., 1990.

[7] Y. Yamada. *Fuzzy Relational Databases: Structures and Queries*: A Senior Thesis. Department of Information Science, Shimane University, Matsue 690, Japan 1993. 3. 23.