Control and Cybernetics

vol. **25** (1996) No. 2

A user interface prototype for fuzzy query definition

by

Waldemar Dobrzyński

Systems Research Institute, Polish Academy of Sciences,
ul. Newelska 6, 01-447 Warsaw, Poland
E-mail: dobrzyns@ibspan.waw.pl

Abstract: This paper presents a prototype of the intelligent user interface which allows for imprecise database information retrieval. It provides graphical tools for defining queries containing special elements such as fuzzy terms, e.g. "high", "small" etc. and facilitates the overall process of creation and application, imprecise components of the query. It uses Microsoft Windows graphical environment and typical relational databases.

Keywords: user interface, imprecise query, fuzzy logic, database querying.

1. Introduction

A great progress in the area of database management systems can be shortly characterized by creation of more and more sophisticated, quicker and user friendly systems. Looking at this process one can say that it aims at systems which will be more or less intelligent – will more or less imitate the human way of thinking. Unfortunately, among actually available database systems it is not easy to find one which would be able to provide means for including some degree of vagueness or imprecision – feature omnipresent in our everyday life and most complex decision problems.

Recently, more and more popular become the approaches based on fuzzy set theory and fuzzy logic developed by Zadeh (1972). In the context of database systems fuzzy sets can be used in two basic ways. The first one assumes conventional database and tries to make it more "human-consistent" mainly by the help of "add-on" systems which make queries more "human-friendly", see Bosc, Galibourg and Hamon (1989), Bosc and Pivert (1991a;b;1992;1994), Yager (1980;1988), Kacprzyk and Ziółkowski (1986), Kacprzyk, Zadrożny and Ziółkowski (1989). According to the second one not only the queries but the database itself can also contain some imprecise data, Buckles and Petry (1982; 1985), Zemankowa-Leech and Kandel (1984). In this paper the former approach

is followed. So the classical relational database is used and "ontop" of it special user interface is built.

In general the problem is how to effectively assist the user during formulation of fuzzy query. It generates two basic subproblems – fuzzy terms definition and fuzzy terms manipulation (we deal with them in sections 3 and 4). Another problem is how to process a fuzzy query (see section 5). Both of the problems are very important and are of course also very closely related but here we concentrate on the former one.

It is possible in the literature to find different approaches to dealing with imprecision in DBMS. There are some which not based on the fuzzy set theory. Motro in his system VAGUE, see Motro (1988), employs the concept of distance and includes the system of special metrics to its definition and measurement. Ichikawa and Hirakawa (1986) in the system ARES apply a similar idea but create special construct called the similarity graph for sophisticated distance evaluation. Another approach is based on the fuzzy logic and the fuzzy set theory. Yager in his system SUMMARIZER performs intelligent database information summarization using linguistic terms, Yager (1980;1988). Bosc's team extends Information Warehouse system, Bosc and Pivert (1994). It allows for fuzzy querying using well-known query language QBE (Query By Example) enriched with the special fuzzy elements. Another achievement of the Bosc's team which should be mentioned here is the SQL-f, fuzzy query language – extended fuzzy version of the most commonly used query language. The subsequent proposal is FQUERY III+ system, "add-on" to DBase III+ database system developed by Kacprzyk, Zadrożny and Ziółkowski (1989). It allows for constructing very advanced and complicated queries including specific fuzzy elements such as fuzzy terms and especially linguistic quantifiers. The system provides also special tools for fuzzy query element definition. The newest approach of Kacprzyk and Zadrożny (1994;1995) is the fuzzy querying system for Microsoft Access DBMS. It is an "add-on" to Microsoft Access v.2. On the base of the ACCESS – proper QBE query language and its very flexible architecture it allows for queries including fuzzy terms and linguistic quantifiers.

We propose here the prototype of the user interface system for the fuzzy query definition. It was implemented using FoxPro for Windows v.2.5, a well known and efficient database management system. We present theoretical background of a new method of the fuzzy term elicitation together with the practical algorithm and examples of the implementation. The mechanisms of the fuzzy terms manipulation, including the common library concept is also proposed.

In section 2 we shortly present basic information concerning the fuzzy set theory. In section 3 we introduce the problem of fuzzy terms definition. Section 4 contains description of the fuzzy terms manipulation mechanisms and especially the common library creation method. In section 5 we deal with the problem of information retrieval and in section 6 we give detailed description of the proposed system.

2. Basics of fuzzy sets theory

In this paper we apply fuzzy sets theory using standard notation. We present it shortly below.

A fuzzy set A in X, written $A \subseteq F(x)$, is defined as a set of pairs $A = \{(\mu_A(x), x)\}$, for each $x \in X$, where μ_A: $X \to [0,1]$ is so-called membership function of the fuzzy set A; $\mu_A(x) \in [0,1]$ is a degree of membership of x in A: from 0 (for definite non-membership) to 1 (for definite membership) through all intermediate values (for intermediate degrees of membership). For non-fuzzy sets this degree of membership can be of course only 0 or 1 (element may be or not a member of the set).

The pair $(\mu_A(x), x)$ is usually denoted by a symbol $\mu_A(x)/x$. A fuzzy set $A \subseteq X$ can be described in the form given below:

$$A = \sum_{x \in X} \mu_A(x)/x,$$

A shape of the membership function is a subjective matter and a key problem in practical applications consists in its adequate definition.

The intersection of two fuzzy sets $A, B \subseteq F(x)$, $A \cap B \subset X$ is defined generally as

$$\mu_{A \cap B}(x) = \mu_A(x) t \, \mu_B(x) \; \forall x \in X. \tag{1}$$

where t is a t-norm – a function such that:

1. $at1 = a$;
2. $atb = bta$;
3. $atb \geq ctd$ if $a \geq c$, $b \geq d$;
4. $atbtc = at(btc) = (atb)tc$.

Examples of t-norm are:

1. $a \wedge b = \min(a, b)$,
2. ab,
3. $1 - (1 \wedge ((1-a)^p + (1-b)^p)^{1/p}, p \geq 1$.

T-norm 1. is the most commonly used one.

The union of two fuzzy sets A, $B \subset X$, $A \cup B \subset X$, is defined as:

$$\mu_{A \cup B}(x) = \mu_A(x) s \mu_B(x), \; \forall X \in x. \tag{2}$$

where s is a s-norm – a function such that:

1. $as0 = a$;
2. $atb = bta$;
3. $atb \geq ctd$ if $a \geq c$, $b \geq d$;
4. $atbtc = at(btc) = (atb)tc$.

Examples of s-norm are:

1. $a \vee b = \max(a, b)$,

2. $a + b - ab$,
3. $1 \wedge (a^p + b^p)^{1/p}\ p \geq 1$.

Like before the s-norm 1. is the most commonly used one.

The intersection and union operations correspond to the logical connectives AND/OR respectively.

3. The definition of fuzzy terms

Fuzzy terms are one of the most important components of the fuzzy query. The problem here is at first how to elicit them from the user and then how to store and use in the database information retrieval. A fuzzy term is represented as a membership function of a certain fuzzy set so the task can be more precisely defined as the membership function acquisition.

In the literature, Turksen (1991), the following methods for the membership function acquisition are mentioned:

- direct rating,
- polling,
- reverse rating.

Assuming a membership function $\mu_V(v)$ of a fuzzy set corresponding to a fuzzy term V is to be determined, the following algorithms are used:

1. Direct rating. In this method membership function is constructed on the basis of the user's answers to the following question:

   "How V is v?",

   where V is fuzzy term for which membership function is to be defined and v is random value from the interval $[v_{\min}, v_{\max}]$ which is given by the user as a range of values for the domain of the fuzzy term V. A user gives his answers usually by sliding a kind of indicator on the sliding scale. The experiment is repeated a reasonable number of times with randomly generated value of v. Then the shape of the membership function is approximated using the data gathered.
2. Polling. According to this approach the values of the membership function are derived by repeatedly and randomly presenting to the user a question:

   "Do you agree that v is V?",

   where like above v is a random value and V is a fuzzy term. On the base of the 'yes' and 'no' responses the membership function is constructed according to the following formula:

$$\mu_V(v) = \frac{Total\ number\ of\ \mathbf{yes}\ responses\ for\ v}{Total\ number\ of \mathbf{yes} + \mathbf{no}\ responses\ for\ v}.$$

3. Reverse rating. Here randomly selected membership values from the interval $[\mu_{\min}, \mu_{\max}]$ are presented to a user in a random manner. He is asked to answer the question:

   "Identify v which is of the y-th degree (grade) of membership in the fuzzy set V"

The questions of such a type are then presented to the user a reasonable number of times. The final shape of the mambership function is obtained through an approximation, assuming some general form of this function.

In our approach we use an original method which is special combination of the direct rating and polling methods. It can be called here as *indirect rating*. Membership function is constructed on the basis of the user's answers to the following question:

"Is v your opinion V?",

where v are values from the interval $[v_{\min}, v_{\max}]$ which is defined by the user and V is the fuzzy term.

The most important modification in the proposed method is the mechanism of the indirect membership function rating by the help of the special fuzzy set. This set can be called v_is_V and its membership function may look like below:

$$\begin{aligned} v\_is\_V? \quad = \quad & Yes/1 + Rather\ yes/0.8 + Hard\ to\ say/0.5 \\ & + Rather\ not/0.2 + Not/0. \end{aligned} \tag{3}$$

The actual form of the above function is defined on the basis of experts' opinions. Choosing as an answer one of the fuzzy set elements the user can easily define membership function for a given fuzzy term.

From the practical point of view the idea of the *indirect rating method* consists in allowing the user to select one from the several proposed answers. Each of them points to the certain user acceptance level. It makes overall process more natural and close to the human way of thinking. The user is not bound to make sometimes difficult and uncertain selection between Yes/Not answers as in polling method but can also select such answers as: *rather yes*, *hard to say* or *rather not*. The final shape of the membership function is obtained through the approximation illustrated on the example given in section 6.

In practice the fuzzy terms are usually defined using the trapezoidal membership function (see Fig.1).

In our approach we propose a modified shape of the membership function which can be called as *rounded trapezium* (see Fig.2).

Notice that the transition around the point B in Fig.1 (i.e. from B_1 to B_2 or vice versa) and around C (i.e. form C_1 to C_2 and vice versa) are now much less abrupt that in the case of the trapezoidal membership function (Fig.1). This may be of relevance in practice. Evidently the rounding of the membership at A and D makes little sense as such low values (ca. 0) are not interesting.

4. Manipulation of fuzzy terms

Fuzzy terms are naturally "context dependent". For example such fuzzy term as "high" will have probably quite different interpretation in the context "high temperature" than in the context "high salary".

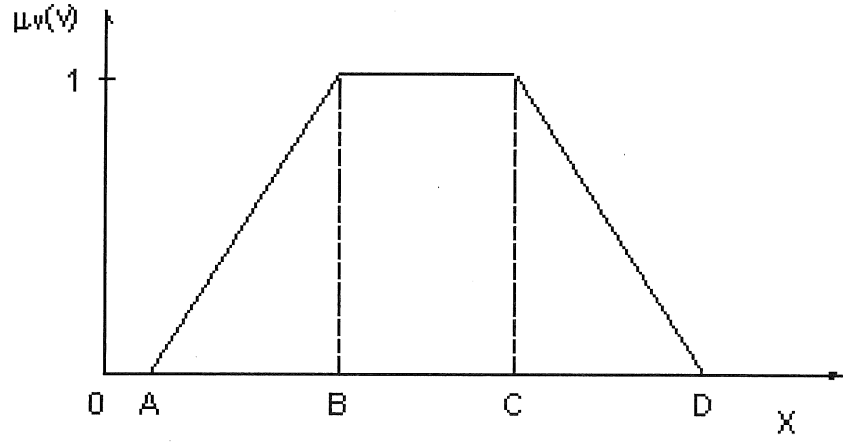

Figure 1. A trapezoidal membership function

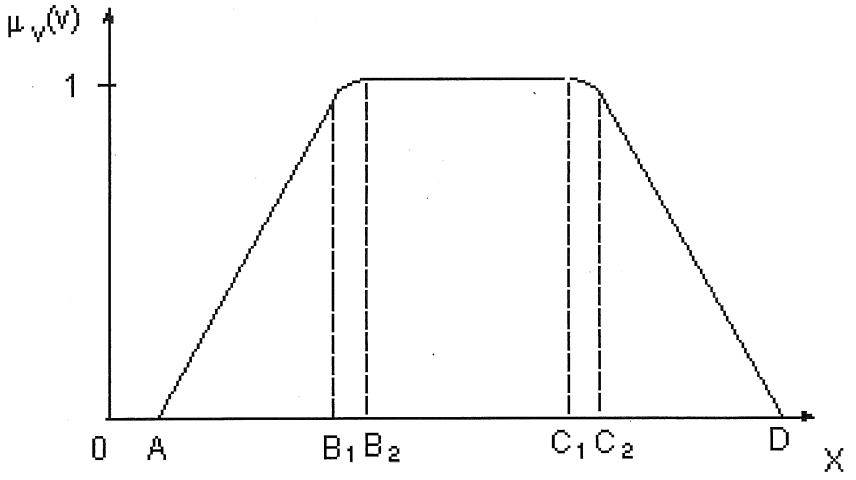

Figure 2. A rounded trapezium membership function

In this paper it is assumed that a given fuzzy term should be defined and used always in the given context. When it is used out of the right context it may cause many erroneous results.

In the proposed system fuzzy terms are grouped together in the special libraries. Each library is created for a given system user and for a given context. In the process of the fuzzy query construction the user can retrieve necessary fuzzy term from his personal or from the common library.

At this point the problem of the common library construction arises. It was not mentioned in the practical implementations so far. The common library contains fuzzy terms defined by particular users. The problem arises when two or more users have fuzzy terms of identical meaning in their personal libraries. Here we present our approach to solve this question.

To create the common library of the fuzzy terms we must find a method for membership function aggregation. Here we propose calculations according to the three alternative algorithms.

$$\overline{\mu}(x) = \mu_1(x) \vee \mu_2(x) \tag{4}$$

where $\vee$ denotes the maximum operation,

$$\overline{\mu}(x) = \frac{1}{2}\{\mu_1(x) + \mu_2(x)\} \tag{5}$$

$$\overline{\mu}(x) = \frac{1}{2} \odot \{\mu_1(x) \oplus \mu_2(x)\} \tag{6}$$

where $\odot$ and $\oplus$ denote the intersection and union of fuzzy sets (1), (2), which correspond to $\odot$ as t-norm and to $\oplus$ as s-norm respectively.

In Figs. 3, 4 and 5 we describe graphically the results of the above algorithms. The dotted lines represent fuzzy terms being aggregated and the solid line the aggregate obtained.

Here in the process of common library creation we use the last method. It seems to be the most intuitive and in distinction to the other presented methods it allows to preserve the form of the membership function.

5. Retrieval from a database

In this work we deal with sequential database information retrieval. All database is scanned and for each record the matching degree which is real number from 0 to 1 is calculated. Due to the presence of the fuzzy terms in the query it would be unreasonable to preserve the traditional approach where a record may be only accepted (matching = 1) or rejected (matching = 0). Information retrieval process is described in Fig.6.

The fundamental problem is how to determine the matching degree. We give a general formula for determining matching degree first for *simple* and next for *compound* query.

General form of the *simple* fuzzy query looks like below:

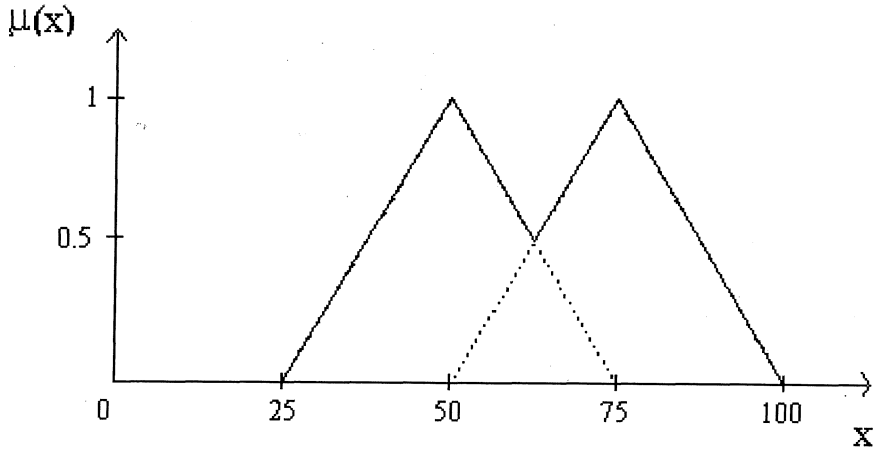

Figure 3. The first method of aggregation of the membership functions

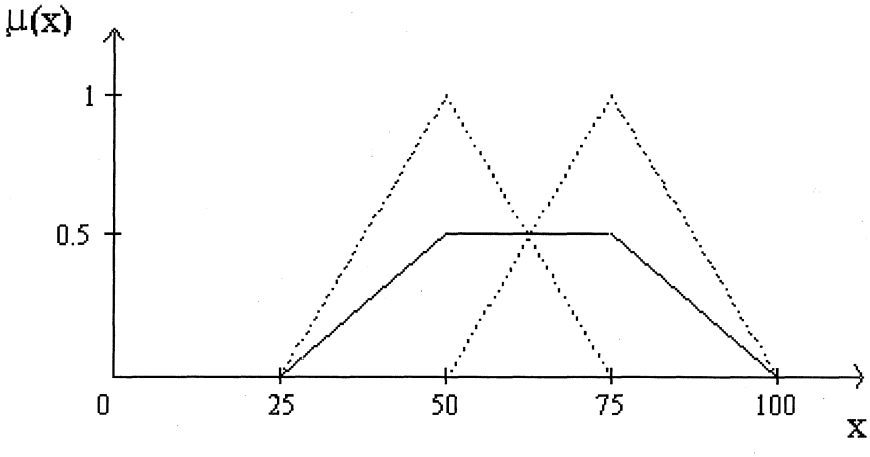

Figure 4. The second method of aggregation of the membership functions

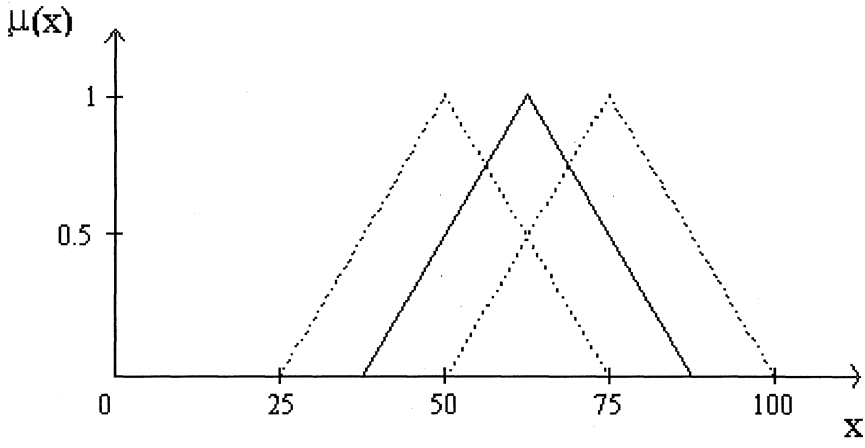

Figure 5. The third method of aggregation of the membership functions

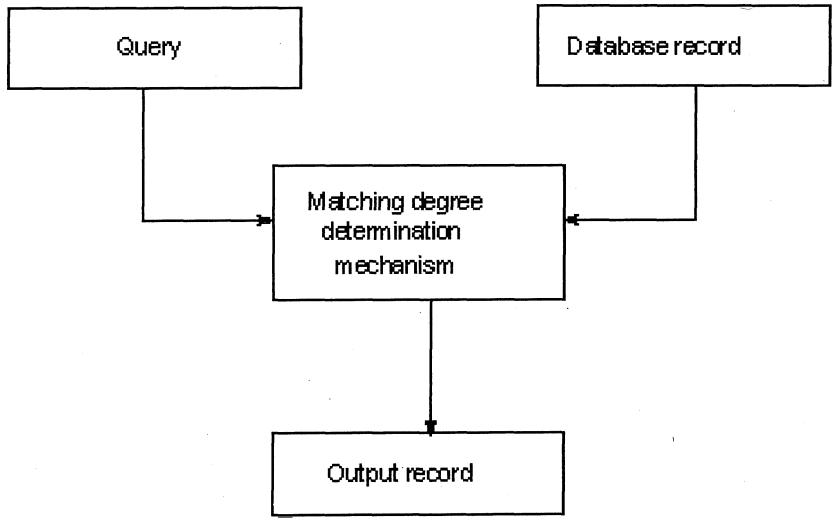

Figure 6. The scheme of the information retrieval process

SELECT *
FROM **table**
WHERE
  attribute is $fuzzy\_term$.

For this situation the matching degree (MD) is calculated for attribute (AT) according to the classical formula (see Kacprzyk, Zadrożny and Ziółkowski, 1989):

$$MD = \mu_{fuzzy\_term}(AT_i). \tag{7}$$

where AT denotes the value of attribute AT for a given record.

General form of the *compound* fuzzy query looks like below:

SELECT *
FROM **table**
WHERE
  attribute$_1$ is $fuzzy\_term_1$
AND/OR
  attribute$_2$ is $fuzzy\_term_2$

For the compound query we have the following formulas:

- for AND connector
$$MD = \mu_{fuzzy\_term1}(AT_1) t \mu_{fuzzy\_term2}(AT_2), \tag{8}$$
- for OR connector
$$MD = \mu_{fuzzy\_term1}(AT_1) s \mu_{fuzzy\_term2}(AT_2). \tag{9}$$

Since in this article emphasis is on interface, we will not consider here more complicated query forms.

6. A user interface prototype presentation

User interface system for fuzzy query definition should satisfy some special requirements. Here we mention two most important points which appear in the literature (see Zemankowa-Leech and Kandel, 1984):

- individualization,
- user's convenience.

The system presented here offers a wide range of individualization. It is realized on several levels. First each user can define his or her private library of the fuzzy query elements. Next, depending on the user's knowledge level, different ranges of tools during the fuzzy query definition are available. Novice users can create only a simply query whereas intermediate users can build compound queries including AND/OR connectives.

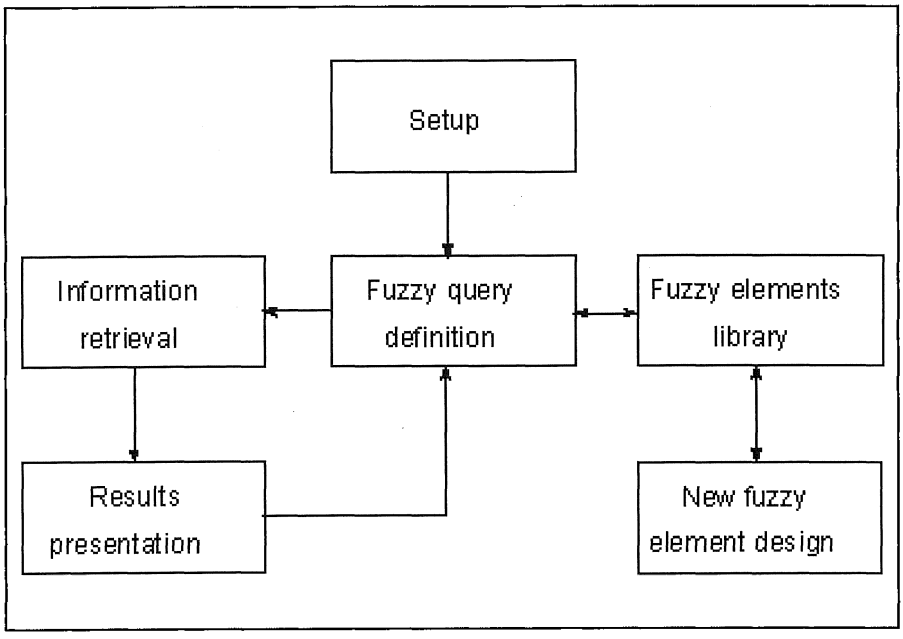

Figure 7. Overall system scheme

For the user's convenience the user interface system based on MS Windows was built. It contains special graphical tools for easy and natural fuzzy query managing. It constitutes the subject of special interest in this paper.

The proposed system is implemented in the FoxPro for Windows v.2.5 relational database system for MS Windows. Using a graphical user interface just makes it more "user friendly" and easy to operate. The prototype system is able to cooperate with each database accepted by the FoxPro system ("*.DBF" type database). A general structure of the interface is presented in Fig.7.

Particular modules have the following functions:

1. The central point consists of the *fuzzy query definition* module which is responsible for maintaining the overall structure of the fuzzy query.
2. *Setup* allows for definition of system parameters.
3. Elements of a fuzzy query are managed in the special module called *Fuzzy elements library.*
4. Whenever it is necessary to define new element of the query the module *New fuzzy element design* is activated.
5. After query completion the *information retrieval* process begins.
6. Results of the query are in turn presented to the user in the module *result presenter.*

Let us assume here that there is a classical non-fuzzy relational database of lake water pollution. It is required to find information concerning lakes where

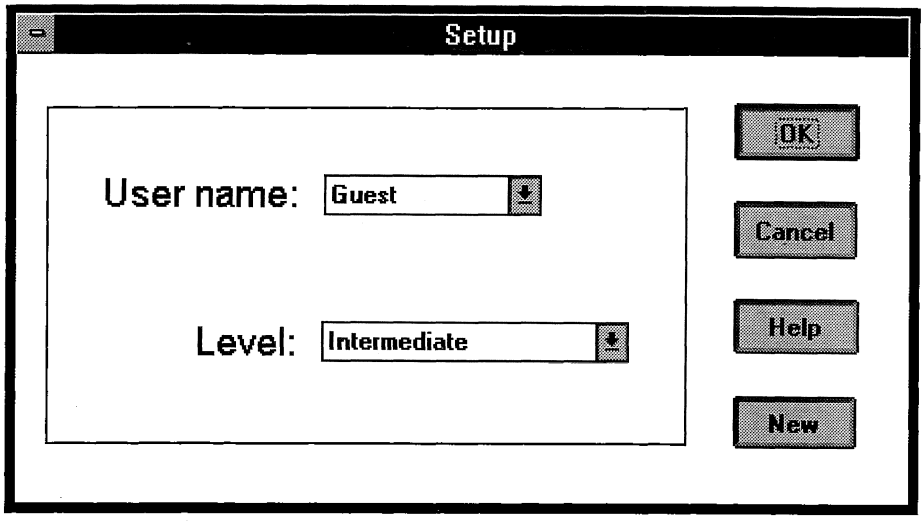

Figure 8. Setup screen

water pollution in terms of overall phosphorus is *high* and water pollution in terms of overall nitrogen is *middle*. Often, as in this example, it is unnecessary or impossible to define precise data retrieval conditions. Then, the possibility to use such imprecise terms as "high" or "middle" directly in a query may be very helpful.

At first we will formally express the requirements from our example using standard SQL-like query language notation. It may look like below:

SELECT information_concerning_lakes
FROM table_of_lakes_water_pollution
WHERE
  overall_phosphorus_pollution = *high*
AND
  overall_nitrogen_pollution = *middle*

On the example of the above problem we will present the typical working session and we will also describe in more detail the system structure.

In the first step the user should define system parameters which are user name and knowledge level. Fig. 8 presents the example of the setup screen.

The first parameter is called *user name*. It is predefined as "Guest". User can easily change his system name using typical for the MS Windows "pop-up menu". He can also enter new name after pushing the button called "New".

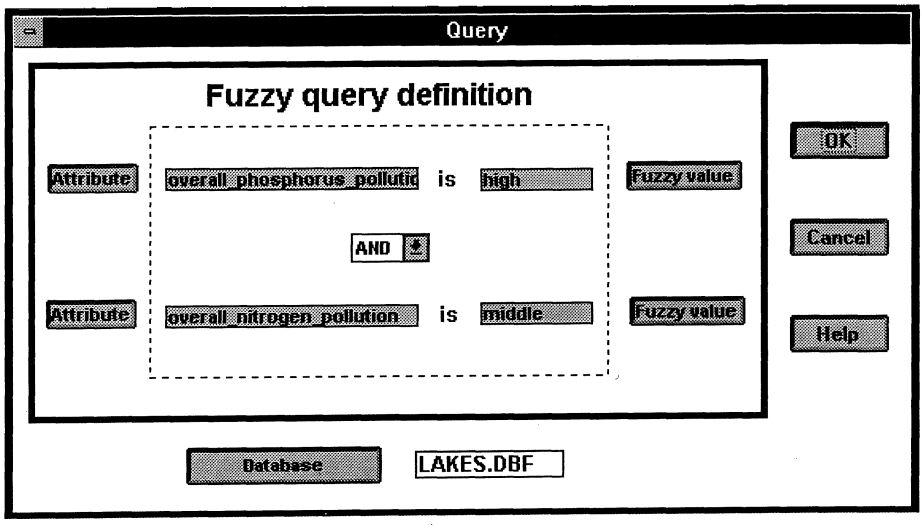

Figure 9. Main screen of the fuzzy query definition

User name parameter influences the fuzzy value selection and definition process. The default fuzzy values are stored in the user personal library and are accessible during the query definition. In the presented system it is also possible to open common fuzzy elements library which will be described in more detail later.

The second parameter is called *level*. Its value can be selected as *Beginner* or *Intermediate* also with the help of typical for MS Windows "pop-up menu". This parameter is responsible for the system flexibility. User interface will look different and will present different options depending on the value of the *level* parameter. When it is initiated as *Beginner* it becomes possible to create only simple, one block fuzzy queries without any logical connectives. When the *level* parameter is established as *Intermediate*, user interface is a little more sophisticated and allows for compound fuzzy query definition.

Next the query screen is displayed. The example of such a system screen corresponding to our example query is presented in the Fig.9.

Using the graphical user interface tools like lists, "pop-up menus" or push buttons typical for Windows the user can easily build appropriate query following the steps given below.

In the first step the database file which will be the subject of the query should be selected. After pushing the button called "Database" the user can choose one from the list of the available file names. Then the database name will appear next to the push button. In Fig.9 the database name is "LAKES.DBF".

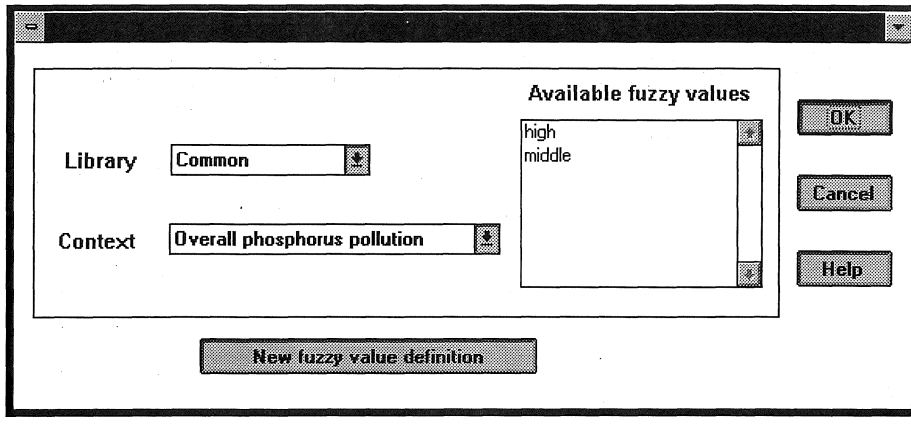

Figure 10. Fuzzy value selection screen

After pushing the button called "Attribute" it becomes possible to select one attribute which belongs to the selected database structure. User can join database attributes with their longer description stored in the special data dictionary. Then the attribute description will appear in the box near the button. In the query example of Fig.9 attribute descriptions are *overall_phosphorus_pollution* and *overall_nitrogen_pollution*.

Using "pop-up lists" user can select the logical connective – AND or OR.

In the system presented two standard connectives are applied: the conjunction (AND) and the disjunction (OR) along with a special interpretation required by the fuzziness of their arguments. We define them in terms of fuzzy sets intersection (1) and union (2).

For the intersection we use the following t-norm:

$$a \wedge b = \min(a, b),$$

and for the union the following s-norm:

$$a \vee b = \max(a, b).$$

Fuzzy value selection is possible after pushing the button called "Fuzzy value". Then, the subsequent screen appears.

It is possible here to select the needed fuzzy value from a given library (Common or Personal) and for a given context. The system enables user to select fuzzy values defined by himself and stored in his personal library and also fuzzy values which were defined by all system users which are collected in the common library. Common fuzzy values are calculated according to the method described in section 4.

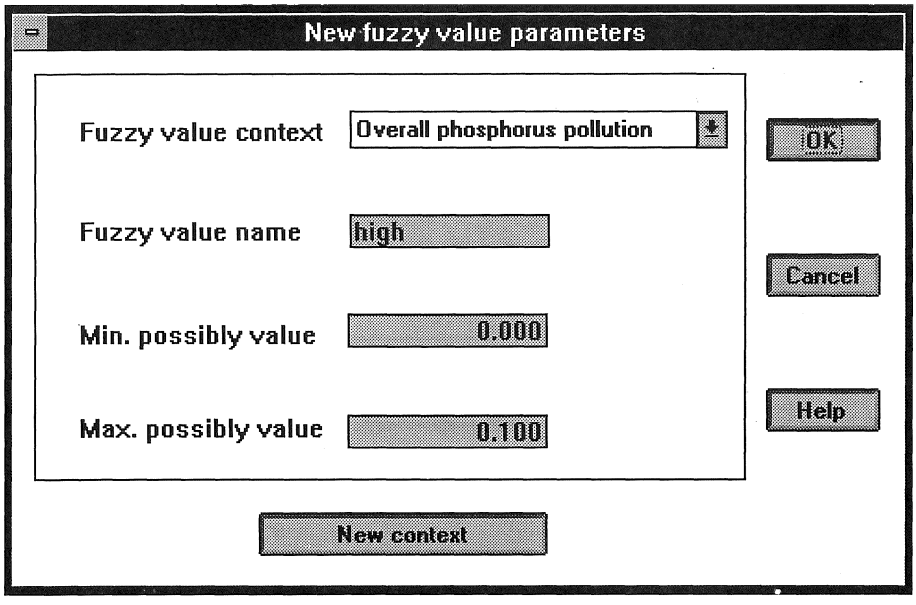

Figure 11. The first phase of the new fuzzy value definition

One of the most important and the most interesting problems is the fuzzy value definition mechanism. This process in our prototype system begins after pushing the button called "New fuzzy value definition". It includes two general phases. During the first one the user defines new fuzzy value parameters. The Fig.11 shows the appropriate system screen.

There are four main parameters defining fuzzy value. These are context, fuzzy value name, its minimum and maximum possible values. These values are then used in the second phase of the fuzzy value definition. In practice, in the above example of the definition of the fuzzy value "high" for the context *overall phosphorus pollution* the minimum and maximum values are arbitrarily established by the user and in fact should be also connected to the water pollution standards. By pushing the "New context" button the user can easily select here a new context and then define for it new fuzzy values.

After setting all parameters the second part of the process begins – the membership function elicitation. The corresponging screen is shown in Fig.12.

Here the user is asked to answer several questions, as mentioned in section 3. He can give his answer simply by pushing appropriate button which in the best way reflects his opinion. The buttons are called: "Yes", "Rather yes", "Hard to say", "Rather not" and "Not" and they correspond to the fuzzy set called *v_is_V?* presented in section 3 and its membership function:

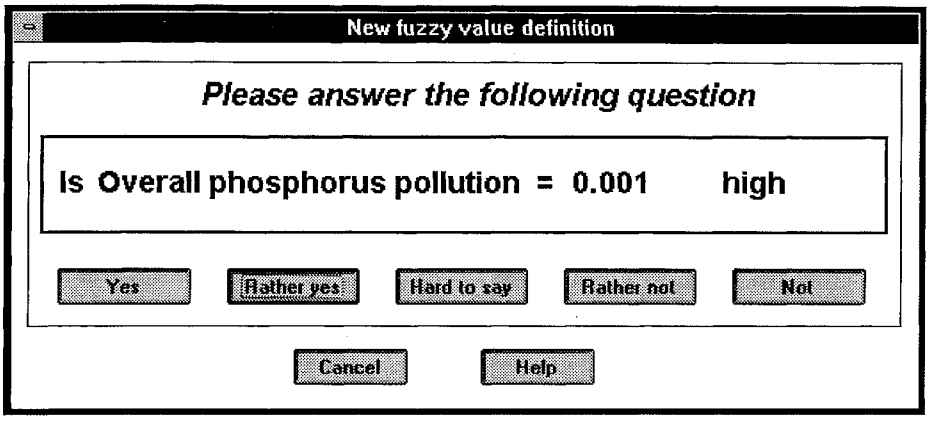

Figure 12. The second phase of the new fuzzy value definition

$$v\_is\_V? \quad = \quad Yes/1 + Rather\ yes/0.8 + Hard\ to\ say/0.5 + Rather\ not/0.2 + Not/0.$$

Each time the question refers to a different value of the attribute. In the example in Fig.12 the user should simply answer several times the following question:

$$\text{Is the } Overall\ phosphorus\ pollution = 0.001\ high, \tag{10}$$

by simply pushing one of the buttons. Each button represents specific rank which corresponds to a given membership function value.

The first step of the proposed method is typical for such systems (see articles of Kacprzyk and Bosc in the literature) but the second phase is quite original. The key problem here is the method for generation of the values which are presented to the user for evaluation. Most of the propositions are based on the randomly generated numbers from a given interval. In our approach we develop a special algorithm which at first allows for a more precise membership function evaluation and also will optimize the number of the interactions (which naturally will also influence the quality of the evaluation).

Our method consists of two general steps:

1. Function type selection.
2. Function form evaluation.

The aim of the first step of the method is to select the membership function type from among the three main types – unimodal, non-decreasing and non-increasing. On the basis of answers received from the user to the questions of the type (10), referring to the minimun and maximum possible value from the domain, it is easy to categorize the membership function according to Table 1.

The answer for the minimum possible value (button in Fig.12)	The answer for the minimum possible value (button in Fig.12)	The type of the membership function
0 (Not)	0 (Not)	unimodal
1 (Yes)	0 (Not)	non-increasing
0 (Not)	1 (Yes)	non-decreasing

Table 1.

When both answers are 1, the system interrupts the process and the error message is generated.

In the second step of the proposed method the form of the membership function is evaluated according to the following algorithm (let us assume that we have the non-decreasing membership function type):

1. $\Delta := (val\_max - val\_min)/step$
2. $point := val\_max - \Delta$
3. **WHILE** $memb\_degree(point) = 1.0$
4. $\quad point := point - \Delta$
5. $memb\_degree(point + \Delta/2)$
6. $point := val\_min + \Delta$
7. **WHILE** $memb\_degree(point) = 0$
8. $\quad point := point + \Delta$
9. $memb\_degree(point - \Delta/2)$

Here, *step* is the parameter which influences the precision of the membership function evaluation. The greater the value of that parameter, the function is more precisely evaluated but on the other hand the more interactions with the user there are.

The function $memb\_degree(point)$ represents the process of the interaction with the user and it relates one of the values from the set of linguistic values {"Yes", "Rather yes", "Hard to say", "Rather not", "Not"} (see Fig.12) to the following set of numbers {1.0, 0.8, 0.5, 0.2, 0} which correspond to the membership function values.

After finishing the second phase of the proposed method the form of the membership function is reconstructed (see Fig.2 and rounded trapezium membership function).

When the process of the fuzzy query definition is completed the relevant records stored in the database should be found. At first fuzzy query is transformed from the very "user friendly" form to the form which can be processed by the database management system. Next for each record in the database special value is calculated, see (7), (8) and (9). It corresponds to the degree in

Results

Level	Lake_name	Surface	Time	Phosphorus	Nitrogen
1.00	Pozezdrze	122.5	08/04/86	0.560000	1.04000
1.00	Kruklin	356.4	08/06/86	0.610000	1.05000
1.00	Krępsko	377.3	08/29/89	0.219000	1.27000
1.00	Szczytno	645.2	08/24/89	0.209000	1.29000
1.00	Szczytno Małe	33.2	08/28/89	0.261000	1.42000
1.00	Końskie	52.5	08/30/89	0.260000	1.27000
1.00	Linowskie	163.4	08/28/89	0.490000	1.32000
1.00	Linowskie	163.4	08/28/89	0.550000	1.17000
1.00	Linowskie	163.4	08/28/89	0.340000	1.61000
1.00	Linowskie	163.4	04/11/89	0.805000	1.37000
1.00	Linowskie	163.4	08/28/89	0.496000	1.32000
1.00	Klebarskie	261.9	04/10/89	0.270000	1.53000
1.00	Klebarskie	261.9	08/21/89	0.530000	1.71000
1.00	Klebarskie	261.9	08/21/89	0.450000	1.74000
1.00	Klebarskie	261.9	08/21/89	0.710000	1.35000
1.00	Klebarskie	261.9	08/28/89	0.574000	1.68000
1.00	Wadąg	494.5	08/17/89	0.437000	1.77000
1.00	Rańskie	291.3	08/27/90	0.270000	1.50000
1.00	Rańskie	291.3	08/27/90	0.330000	1.45000
1.00	Ruda Woda	654.1	08/23/90	0.870000	1.01000
1.00	Ruda Woda	654.1	08/23/90	0.370000	1.88000

Figure 13. Results of the fuzzy query processing

which a given record fulfils fuzzy query. At the end results are presented to the user. An example of the result screen is presented in Fig.13.

The first column tells the user to the what degree a given record fulfils the query. The subsequent columns correspond to the database attributes. Records are sorted for the value of the field "Level" from 1 to 0. User can easily move through the result table and see all attributes for all records.

7. Conclusions

The system presented in this paper includes several original ideas. At first, in the point which is the special area of our interest, a new user interface for the fuzzy query definition is developed. It uses graphical facilities of the MS Windows system. The subsequent point includes methods applied in the process of the interaction with the user during the fuzzy element definition. Fuzzy values are from their nature very "context-dependent". The use of the given fuzzy value out of its right context will probably lead to many errors and misunderstandings during fuzzy query processing. In the literature of the subject and in the systems constructed so far the problem of context is not adequately mentioned. In our approach we always consequently consider fuzzy value with its original context

(that is, the context for which given fuzzy value was defined). It makes the system more reliable and at last leads to another original feature of our proposal – organization of the fuzzy values in libraries.

There are some promising directions for future research. At first it would be useful to allow for more complicated queries including more than two blocks connected by different type of connectives. Next, application of various aggregation operators can also be very interesting. At last one of the most unexplored areas is the use of the linguistic quantifiers and modifiers in fuzzy queries.

References

BOSC, P., GALIBOURG, M. and HAMON, G. (1989) Fuzzy querying with SQL: extensions and implementation aspects. *Fuzzy Sets and Systems*, **28**, 333-349.

BOSC, P. and PIVERT, O. (1991A) About equivalents in SQLf, a relational language supporting imprecise querying. *Proc. International Fuzzy Engineering Symposium*, Yokohama (Japan), 309-320.

BOSC, P. and PIVERT, O. (1991B) Fuzzy querying in conventional databases. In: *Fuzzy logic for the management of uncertainty*, J.Kacprzyk and L.Zadeh eds., John Wiley.

BOSC, P. and PIVERT, O. (1992) Some approaches for relational databases flexible querying. *International Journal of Intelligent Systems*, **1**.

BOSC, P. and PIVERT, O. (1994) Integrating fuzzy queries into an existing database management system: An example. *International Journal of Intelligent Systems*, **9**, 475-492.

BUCKLES, B.P. and PETRY, F.E. (1982) A fuzzy representation of data for relational databases. *Fuzzy Sets and Systems*, **7**, 213-226.

BUCKLES, B.P. and PETRY, F.E. (1985) Query languages for fuzzy databases. In: *Management Decisions Support Systems Using Fuzzy Sets and Possibility Theory*, J.Kacprzyk and R.R.Yager, eds., Verlag TÜV Rheinland, Cologne, 241-252.

ICHIKAWA, T. and HIRAKAWA, M. (1986) ARES: A relational database with the capability of performing flexible interpretation of queries. *IEEE Transactions on Software Engineering*, **12**, 5.

KACPRZYK J. (1986) *Zbiory rozmyte w analizie systemowej.* PWN, Warszawa.

KACPRZYK, J. and ZIÓŁKOWSKI, A. (1986) Retrieval from databases using queries with fuzzy linguistic quantifiers. In: *Fuzzy logic in Knowledge Engineering*, H.Prade and C.V.Negoita, eds., Verlag TÜV Rheinland, Cologne, 46-57.

KACPRZYK, J., ZADROŻNY, S. and ZIÓŁKOWSKI, A. (1989) FQUERY III+: A "human-consistent" database querying system based on fuzzy logic with linguistic quantifiers. *Information Systems*, **14**, 6, 443-453.

KACPRZYK, J. and ZADROŻNY, S. (1994) Fuzzy querying for Microsoft Access. *Proceedings of Third IEEE International Conference on Fuzzy Sys-*

tems – FUZZ – IEEE'94 (Orlando, FL, USA), **1**, 167-171.

KACPRZYK, J. and ZADROŻNY, S. (1995) Fuzzy querying for Microsoft Access v2. *Proceedings of the FUZZ – IFES'95 Workshop on Fuzzy Database Systems and Information Retrieval* (Yokohama, Japan), 61-66.

MOTRO, A. (1988) VAGUE: A user interface to relational database that permits vague queries. *ACM Transaction on Office Information Systems*, **6**, 3, 187-214.

SHNEIDERMAN, B. (1987) *Designing the user interface: strategies for effective human-computer interaction.* Addison-Wesley Publishing Company.

TURKSEN, B.I. (1991) Measurement of membership functions and their acquisition. *Fuzzy Sets and Systems*, **40**, 5-38.

YAGER, R.R. (1980) A logical on-line bibliographical search: an application of fuzzy sets. *IEEE Transactions on Systems, Man and Cybernetics*, **10**, 51-53.

YAGER, R.R. (1988) On ordered weighted averaging aggregation in multicriteria decisionmaking. *IEEE Transactions on Systems, Man and Cybernetics*, **18**, 1, 183-190.

ZADEH, L.A. (1972) A fuzzy-set theoretic interpretation of linguistic hedges. *J. of Cybernetics*, **2**, 2, 4-34.

ZEMANKOWA-LEECH, M. and KANDEL, A. (1984) *Fuzzy Relational Databases – A Key to Expert Systems.* Verlag TÜV Rheinland, Cologne, 46-57.